ALGORITHMS FOR GENE CLUSTERING ANALYSIS ON GENOMES

A Dissertation

by

GANG MAN YI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2011

Major Subject: Computer Science

ALGORITHMS FOR GENE CLUSTERING ANALYSIS ON GENOMES

A Dissertation

by

GANG MAN YI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Sing-Hoi Sze
Committee Members,	Yoonsuck Choe
	Clare Gill
	Tiffani L. Williams
Head of Department,	Valerie E. Taylor

May 2011

Major Subject: Computer Science

ABSTRACT

Algorithms for Gene Clustering Analysis on Genomes. (May 2011)

Gang Man Yi, B.S., Kangnung National University, South Korea;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Sing-Hoi Sze

The increased availability of data in biological databases provides many opportunities for understanding biological processes through these data. As recent attention has shifted from sequence analysis to higher-level analysis of genes across multiple genomes, there is a need to develop efficient algorithms for these large-scale applications that can help us understand the functions of genes.

The overall objective of my research was to develop improved methods which can automatically assign groups of functionally related genes in large-scale data sets by applying new gene clustering algorithms. Proposed gene clustering algorithms that can help us understand gene function and genome evolution include new algorithms for protein family classification, a window-based strategy for gene clustering on chromosomes, and an exhaustive strategy that allows all clusters of small size to be enumerated. I investigate the problems of gene clustering in multiple genomes, and define gene clustering problems using mathematical methodology and solve the problems by developing efficient and effective algorithms.

For protein family classification, I developed two supervised classification algorithms that can assign proteins to existing protein families in public databases and, by taking into account similarities between the unclassified proteins, allows for progressive construction of new families from proteins that cannot be assigned. This approach is useful for rapid assignment of protein sequences from genome sequencing projects to protein families. A comparative analysis of the method to other previously developed methods shows that the algorithm has a higher accuracy rate and lower mis-classification rate when compared to algorithms that are based on the use of multiple sequence alignments and hidden Markov models. The proposed algorithm performs well even on families with very few proteins and on families with low sequence similarity.

Apart from the analysis of individual sequences, identifying genomic regions that descended from a common ancestor helps us study gene function and genome evolution. In distantly related genomes, clusters of homologous gene pairs serve as evidence used in function prediction, operon detection, etc. Thus, reliable identification of gene clusters is critical to functional annotation and analysis of genes. I developed an efficient gene clustering algorithm that can be applied on hundreds of genomes at the same time. This approach allows for large-scale study of evolutionary relationships of gene clusters and study of operon formation and destruction. By placing a stricter limit on the maximum cluster size, I developed another algorithm that uses a different formulation based on constraining the overall size of a cluster and statistical estimates that allow direct comparisons of clusters of different size. A comparative analysis of proposed algorithms shows that more biological insight can be obtained by analyzing gene clusters across hundreds of genomes, which can help us understand operon occurrences, gene orientations and gene rearrangements.

To my wife, daughter and parents for their patience, encouragement and love

ACKNOWLEDGMENTS

I would like to express my thanks and gratitude to Dr. Sing-Hoi Sze. His valuable advice and guidance allowed me to persevere and finish my doctoral degree program. Without his patience and caring for me, I could not have completed my doctoral degree. I would also like to thank Dr. Michael R. Thon for sharing his knowledge and experience.

I am grateful to the members of my committee, Dr. Yoonsuck Choe, Dr. Tiffani L. Williams and Dr. Clare Gill for their interests and guidances in my research.

This dissertation would not have been possible without the support, understanding, encouragement and sacrifice of my wife, Jaehee, who has shared so many difficulties, uncertainties and challenges for completing this dissertation. I would like to thank my parents and parents-in-law for their constant encouragement. My thanks also go to my daughter who has brought numerous joy to me while I was working on this dissertation.

TABLE OF CONTENTS

Page

ABSTRACT iii

DEDICATION v

ACKNOWLEDGMENTS vi

TABLE OF CONTENTS vii

LIST OF TABLES ix

LIST OF FIGURES xi

CHAPTER

I INTRODUCTION 1

A. Motivation 1
B. Objectives 4

II BACKGROUND 11

A. Gene cluster 11
B. Protein family 13
C. Operon 15

III SUPERVISED PROTEIN FAMILY CLASSIFICATION AND NEW FAMILY CONSTRUCTION 17

A. Introduction 17
B. Methods 19
1. Classifying proteins to existing families 19
2. Constructing new families from unclassified proteins . 21
3. Availability 22
C. Experiments 24
1. Data sets 24
2. Choice of parameters 27

CHAPTER Page

3. Comparison with other supervised algorithms 30
4. Comparison with unsupervised algorithms 35
5. Classifying a set of proteins 39

D. Discussion 40

IV LARGE-SCALE CLUSTERING ACROSS MULTIPLE GENOMES 49

A. Introduction 49
B. Method 52
1. Window-based approach 52
2. Large-scale clustering across multiple genomes 60

C. Experiments 63
1. Data sets 63
2. Implementation 64
3. Gene clusters on *E. coli K12* 67
4. Comparative analysis on bacteria groups 71
5. Gene rearrangement 76
6. Comparative analysis on operon occurrences 78

D. Discussion 83

V CONCLUSION 86

A. Summary 86
B. Future work 89

REFERENCES 91

VITA 101

LIST OF TABLES

TABLE Page

I UniProt 3-oxoacid CoA-transferase family. 13

II Data sets for performance evaluation. For each set, family is the number of families, protein is the total number of proteins in all families, avg_pro is the average number of proteins within a family, and avg_len is the average length of proteins in amino acids. 25

III Accuracy rate comparison of SClassify (using either SSEARCH or BLAST to compute pairwise scores), LIBSVM and SMT on individual species from Pfam and on SCOP. 35

IV Result comparison with GeneTeam. 59

V Distribution of the number of homologs and its average size over different e-value cutoffs. 64

VI Intersection & Union of subregions to *E. coli* operons from the RegulonDB database. 69

VII Intersection result of subregion overlaps with the *E. coli* operon database. (a) showing the distribution of the number of subregions among all clusters, (b) showing the distribution of the size of subregions among all clusters, and (c) showing the distribution of the best overlap of subregions with *E. coli* operons among the ones that has overlap (S_{min}, S_{avg} and S_{max}) with some *E. coli* operon, which is defined to be the largest ratio of the number of shared genes between a subregion and an operon to the average number of genes of the subregion and the operon. 72

TABLE Page

VIII Union result of subregion overlaps with the *E. coli* operon database. (a) showing the distribution of the number of subregions among all clusters, (b) showing the distribution of the size of subregions among all clusters, and (c) showing the distribution of the best overlap of subregions with *E. coli* operons among the ones that has overlap (S_{min}, S_{avg} and S_{max}) with some *E. coli* operon, which is defined to be the largest ratio of the number of shared genes between a subregion and an operon to the average number of genes of the subregion and the operon. 73

IX Bacterial groups. 74

X Pearson correlation coefficient of subregions and gene clusters. 78

LIST OF FIGURES

FIGURE Page

1 Illustration of gene cluster. 12

2 The example of the lac operon structure. 15

3 Algorithm SClassify when one unclassified protein is given. 20

4 Algorithm SClassify when a set of more than one unclassified protein is given. 23

5 Illustration of the procedure of the proposed method. 26

6 Accuracy rate and mis-classification rate of SClassify over different SSEARCH *e*-value cutoffs by using SSEARCH to compute pairwise scores. Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. More tests are performed for *e*-value cutoffs between 1 and 1e–5. 27

7 Accuracy rate and mis-classification rate of SClassify over different BLAST *e*-value cutoffs by using BLAST to compute pairwise scores. Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. More tests are performed for *e*-value cutoffs between 1 and 1e–5. 28

8 Accuracy rate and mis-classification rate of HMMER over different *e*-value cutoffs. Light bars and solid lines represent accuracy rates, while dark bars and dotted lines represent mis-classification rates. Bars denote performance on Pfam by applying HMMER on the profile hidden Markov model that was created for each curated family from a subset of curated seed sequences in Bateman *et al.* (2000). Lines denote performance by applying HMMER on the profile hidden Markov model that is constructed for each family in the training set from a multiple sequence alignment obtained by ClustalW. These alignments are constructed from all sequences in each family. More tests are performed for *e*-value cutoffs between 1 and 1e–5. 32

FIGURE Page

9 Accuracy rate and mis-classification rate of HMMER over different e-value cutoffs. Light bars and solid lines represent accuracy rates, while dark bars and dotted lines represent mis-classification rates. Bars denote performance on Pfam by applying HMMER on the profile hidden Markov model that was created for each curated family from a subset of curated seed sequences in Bateman *et al.* (2000). Lines denote performance by applying HMMER on the profile hidden Markov model that is constructed for each family in the training set from a multiple sequence alignment obtained by ClustalW. These alignments are constructed from a subset of distinct sequences in each family with pairwise BLAST e-values above 0.1 (a BLAST e-value is considered to be above the cutoff if no BLAST hits are obtained). 33

10 Performance comparison of SClassify with unsupervised clustering algorithms MCL and BLASTClust by using BLAST to compute pairwise scores. For each data set, PC is the precision, RC is the recall, and the lines denote the F-measure. 38

11 Accuracy rate comparison of different variants of SClassify when a set of unclassified proteins is given while using BLAST to compute pairwise scores. Given a set U of unclassified proteins, Seq corresponds to the original algorithm for classifying one protein on each protein sequentially in U according to a random order, while Set corresponds to the algorithm for classifying the proteins in U simultaneously by also taking into account similarities between unclassified proteins. 39

12 Mis-classification rate comparison of different variants of SClassify when a set of unclassified proteins is given while using BLAST to compute pairwise scores. Given a set U of unclassified proteins, Seq corresponds to the original algorithm for classifying one protein on each protein sequentially in U according to a random order, while Set corresponds to the algorithm for classifying the proteins in U simultaneously by also taking into account similarities between unclassified proteins. 41

FIGURE Page

13 Accuracy rate and mis-classification rate of SClassify by grouping together all families that contain the same number of proteins (each point denotes the performance on proteins within such families). Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. Missing points correspond to no proteins being assigned to the category. 42

14 Accuracy rate and mis-classification rate of SClassify by grouping together all families with average sequence identity between proteins within a specified range (each point denotes the performance on proteins within such families with average sequence identity within the range from l to u, where l is the label to the left of the point on the x-axis and u is the label to the right of the point on the x-axis), while using BLAST to compute pairwise scores. Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. Missing points correspond to no proteins being assigned to the category. 43

15 Running time of SClassify on one processor while including the time to compute pairwise scores by BLAST. In each case, the average running time is given. 44

16 Running time of SClassify on one processor while excluding the time to compute pairwise scores by BLAST. In each case, the average running time is given. 45

17 Running time of SClassify on one processor while including the time to compute pairwise scores by SEARCH. In each case, the average running time is given. 46

18 Memory requirement of SClassify on a single processor while excluding to compute pairwise scores by SSEARCH or BLAST. 47

19 Algorithm to identify gene clusters that include same gene number at least once within a window of length l. Paralogous genes are allowed while requiring that each cluster appears in each of the k given chromosomes. $\mathcal{G}$ is the set of clusters returned with each cluster G' represented by a list of gene numbers. l, n, e is the window length, the average length of chromosomes, and e-value cutoff respectively. 54

FIGURE Page

20 Illustration of the window-based algorithm when window length $l = 3$. The window under consideration on chromosome c_1 is enclosed in a box. The occurrences of the gene numbers in chromosomes c_2 and c_3 are enclosed in boxes. The set of returned clusters is $\mathcal{G} = \{G'_1, \ldots, G'_8\}$. 56

21 The biological process overview of the glutathione metabolic process (GO:0006749) in the GO structure. 58

22 Algorithm to identify gene clusters that include same gene numbers at least once within a window of length l by the combination of unique gene numbers not to lose potential gene clusters. Paralogous genes are allowed while requiring that each cluster appears in each of the k given chromosomes. $\mathcal{G}$ is the set of clusters returned with each cluster B represented by a list of gene numbers. l, n, m, e is the window length, the average length of chromosomes, the minimum number of chromosomes to appear, and e-value cutoff respectively. 61

23 Illustration of sample clusters of size greater than one. l, l', k and k' represent the window size, the number of genes in the window, the total number of chromosomes and the number of chromosomes that appears gene clusters, respectively. 62

24 Illustration of constructing groups of homologous genes. The first graph is constructed by BLAST e-value. Two graphs below the first original graph represent different homologous groups constructed by the different e-value cutoff. The stringent e-value cutoff makes homologous groups smaller in size, but produces a larger number of homologous group. 65

25 Illustration of removal of clusters that are exact subsets of a larger cluster that has a lower e-value. 66

26 Illustration of clusters that overlap by a user specified threshold. Clusters are labeled as a member of a group of overlapping clusters. Threshold of overlapping clusters = 50%. 67

27 The distribution of *E. coli* operon size in the database. 70

FIGURE Page

28 The distribution of the number of bacterial groups in gene cluster. . . 75

29 Distribution of the percentage of conserved neighboring gene pairs. . 77

30 Illustration of trp operon in Gammaproteobacteria bacteria group. . 79

31 Illustration of trp operon in Firmicutes bacteria group. 81

32 Illustration of trp operon in Betaproteobacteria bacteria group. . . . 83

CHAPTER I

INTRODUCTION

A. Motivation

One of the major technological advances in biology in the last few years is the development of high-throughput genome sequencing technology that produces gigabases of data in a single run. Thus, the increased availability of data in biological databases provides many opportunities for understanding biological processes through these data. To investigate the value of sequence data, scientists need to identify proteins encoded by genomes and understand how these proteins are functioning, but the functions of these genes are still unknown. The traditional way of predicting the gene function is laborious and expensive, because biologists deduce gene functions through experimentation. The general approach for functional annotations of unknown sequences is to predict functions based on sequence similarity to known sequences in databases, but the very weak similarities between sequences that have a common function often gives improper results. An alternative way of constructing ontological frameworks that describe gene function (e.g., Gene Ontology (Ashburner *et al.*, 2000)) still relies on manual curation (Fraser and Marcotte, 2004). As recent attention shifts from sequence analysis to higher-level analysis of genes across multiple genomes, automated prediction methods of gene function and annotation in computational bioinformatics also have created many challenging computational problems,

This dissertation follows the style of *Bioinformatics*.

such as identification of functionally related gene clusters and their functional annotations in large-scale genome data. Thus, it is a challenge for computational biologists to devise the best computational model that can accurately predict gene functions from available data.

One important gene clustering problem is the protein family classification problem, which groups a given set of genes into families so that genes within the same family have a common function. After a genome is sequenced, the first steps are to annotate protein coding genes and assign the genes to families using automated and manual methods. Proteins within gene families (or protein families) are usually homologous and have similar structure of conserved functional domains. The classification of proteins to protein families is an intrinsic part of comparative and evolutionary genomics. The pace of genome sequencing continues to increase, and thus, the need for sensitive and automated protein family classification methods also continues to increase.

Whereas supervised classification algorithms are available for this purpose (Bejerano and Yona, 2001; Eskin *et al.*, 2003; Liao and Noble, 2003), they do not allow for progressive construction of new families and most of them are not efficient enough to handle large data sets. Although it is possible to use unsupervised classification algorithms (Altschul *et al.*, 1990; Enright *et al.*, 2002; Chen *et al.*, 2006; Kim and Lee, 2006), these algorithms do not make use of knowledge from known protein families and do not work well for large data sets. By precomputing sequence similarities between proteins in existing families, I observe that it is possible to develop efficient algorithms that achieve much improved accuracy over previous approaches while allowing for automatic construction of new families. This approach will be useful for rapid assignment of protein sequences from genome sequencing projects to protein families.

Another key factor is that identifying genomic regions that descended from a common ancestor helps us study gene function and genome evolution. In distantly related genomes, clusters of homologous gene pairs are evidence used in function prediction, operon detection, etc. (Raghupathy and Durand, 2009). Thus, reliable identification of gene clusters is critical to functional annotation and analysis of genes. Many kinds of computational methods have been proposed for defining gene clusters in order to study the genome organization, evolution and function of individual gene groups. But there are not many efficient algorithms for identifying gene families and operons in the large-scale data set.

In gene clustering on chromosomes, while the most popular approaches require that the distance between adjacent genes in a cluster be small (A. Bergeron and Raffinot, 2002; Luc *et al.*, 2003; He and Goldwasser, 2005; Parida, 2007), it has been shown that there are considerable difficulties in developing efficient algorithms when paralogous genes are allowed in the model (He and Goldwasser, 2005). By using a different formulation of gene clusters, which requires that the distance between any pair of genes within a cluster be small (Rose Hoberman and Durand, 2005), I observe that it is possible to use a window-based strategy that can identify significantly more functionally-enriched gene clusters than previous approaches.

While many of these algorithms can be used to perform gene clustering across two or more genomes, very few algorithms are efficient enough to analyze a large number of genomes (Kim *et al.*, 2005). By placing a stricter limit on the maximum cluster size, I observed that efficient algorithms can be developed to perform gene clustering on hundreds of genomes at the same time. This strategy is sufficient for analyzing gene clustering in bacterial genomes, and allows for large-scale study of evolutionary relationships of gene clusters and study of operon formation and destruction. It will also allow for the study of whether gene clusters in bacteria occur only at the operon level or whether there are higher-level structures, and the functional relationships between them.

B. Objectives

The overall objective of the proposed research is to improve our ability to automatically make groups of functionally related genes. I propose new methods which automatically assign groups of genes by applying new gene clustering algorithms. The proposed gene clustering algorithms that can help us better understand genome evolution include new algorithms for protein family classification, a window-based strategy for gene clustering on chromosomes, and an exhaustive strategy that allows all clusters of small size to be enumerated. I investigate the problems of gene clustering in multiple genomes, and I define gene clustering problems using mathematical methodology and solve the problems by developing efficient and effective algorithms.

Supervised protein family classification and new family construction – The goal of the protein family classification problem is to group genes into families so that genes within the same family have common function. It is an important problem in computational biology, since it suggests possible functions and structures for proteins

(Smith, 2004).

The general approach for functional annotations of unknown proteins is to predict protein functions based on sequence similarity to known protein sequences in databases (Wu *et al.*, 2003). Existing methods that rely on sequence similarity comparisons utilize BLAST and pattern/profile search methods (Attwood *et al.*, 2002; Gribskov *et al.*, 1990; Krogh *et al.*, 1994). Predicting protein function by sequence similarity is a powerful approach, but one of the problems in identifying protein families only based on sequence similarity is that there is significant difficulty in classifying proteins where the similarity between proteins is very weak within a family. Numerous genome annotation errors have also been reported (Brenner, 1999; Henikoff *et al.*, 1997; Smith and Zhang, 1997; Doolittle, 1995).

Whereas supervised classification algorithms are available for this purpose, most of them focus on assigning unclassified proteins to the correct families and do not allow for progressive construction of new families. Although unsupervised classification algorithms are also available, they do not make use of information from known protein families.

I propose a new method that assigns unclassified proteins to families based on sequence alignments from the sequence alignment algorithm (Altschul *et al.*, 1990; Smith and Waterman, 1981). I developed supervised classification algorithms that overcome the problems of existing supervised and unsupervised algorithms and achieve much improved accuracy. The proposed approach can assign proteins to existing protein families in public databases and, by taking into account similarities between the unclassified proteins, can assign them to new families. This approach will be useful for rapid assignment of protein sequences from genome sequencing projects to protein families.

- Classifying proteins to existing families

The simpler problem is first considered when only one unclassified protein is given. We assume that each protein belongs to at most one family, which consists of either individual protein domain or single domain proteins. The goal is to determine whether the unclassified protein belongs to one of the existing families. I assume that optimal alignment scores between any two proteins in these families have already been precomputed by using a pairwise sequence comparison algorithm such as BLAST (Altschul *et al.*, 1990) or the Smith-Waterman local alignment algorithm SSEARCH (Smith and Waterman, 1981), and assign the unclassified protein to the family with the highest average minus log e-value.

- Constructing new families from unclassified proteins

To further improve accuracy, I consider the generalized problem when more than one unclassified protein is given, and the goal is either to assign each unclassified protein to an existing family or to automatically construct new families if necessary, while also making use of the similarities among the unclassified proteins. Each unclassified protein is initially treated as a family by itself, and a family that contains unclassified proteins is iteratively merged into either an existing family or with another family that contains unclassified proteins. During each iteration, we make sure that the average minus log e-value between the two merged families is the highest among all such pairs of families, where the average is taken over pairs of proteins that have e-value below a cutoff. The procedure terminates when it is no longer possible to satisfy the e-value cutoff or the sequence coverage requirement, with the families that have not been merged with existing families becoming new families.

Large-scale analysis of gene clusters in multiple genomes – The goal of the proposed method is to define an efficient gene clustering algorithm that can investigate related gene clusters across multiple genomes and study operons and their evolution. In bacteria, one of the main mechanisms to facilitate control of gene expression is the organization of genes into operons. The most popular approaches require only that the distance between adjacent genes in a cluster be small (A. Bergeron and Raffinot, 2002; Luc *et al.*, 2003; He and Goldwasser, 2005; Parida, 2007), because homologous regions are straightforward when genomic regions are closely related (Raghupathy and Durand, 2009). Although current algorithms are available to identify gene clusters across multiple genomes, very few algorithms are efficient enough to study gene clusters across multiple genomes. I propose a different formulation based on constraining the overall size of a cluster and develop statistical estimations that allow direct comparisons of clusters of different sizes. I use this algorithm by starting from a well-characterized list of operons in *Escherichia coli K12* and perform comparative analysis of operon occurrences, gene orientations, and rearrangements both within and across clusters.

- Window-based approach

I investigate a different formulation which requires that the distance between any pair of genes within a cluster be small (Rose Hoberman and Durand, 2005), thus placing a constraint on the overall cluster size.

Each gene (while ignoring its orientation) is represented by a number so that the same number represents a set of orthologous genes in different genomes and each chromosome is represented by a sequence of gene numbers. The same gene number is allowed to appear more than once within each chromosome, which represents paralogous copies of a gene. To ensure that all genes in genomes are clustered within a region of size at most length l on each chromosome, we require that each gene number appear at least once in each cluster.

To obtain statistical significance estimates, the probability of a gene cluster appearing in a given chromosome can be obtained by assuming a random background distribution based on the average length of the chromosomes, the cluster size and the average size of the non-empty lists (Durand and Sankoff, 2003). I estimate the p-value of gene cluster by the probability of the gene cluster appearing in at least k' out of k chromosomes using the binomial distribution, and obtain an e-value from this p-value. To consider the algorithm that guarantees that all clusters with e-value below a cutoff are found, I start with each window of size l on chromosome c_1 and find the locations of all occurrences of its gene numbers in the other chromosomes. I apply the algorithm over $1 \leq l \leq 50$ on four yeast genomes.

- Exhaustive approach

I investigate a modified version of the window-based strategy to analyze hundreds of genomes simultaneously by placing a stricter limit on the maximum cluster size.

This makes it possible to avoid the combinatorial explosion of intersecting all combinations at one time with one from each chromosome, and allows the use of a different strategy for finding the clusters. I consider each window within each chromosome and enumerate each of the subsets of genes within the window. I think of each subset as a potential gene cluster and identify the subset of chromosomes in which all genes that appear within a window.

I estimate the p-value of the subset by its probability of appearing in at least these many chromosomes and obtain an e-value. Since all possible combinations of included genes and intervening genes are included within each window, this algorithm will not lose any clusters that satisfy the definition. One important advantage of the algorithm is that its time complexity grows linearly with the input size and the base of 2 in the exponential part of the time complexity is small, thus a large number of genomes can be considered at the same time. To investigate whether these gene clusters correspond mostly to one operon or many operons, I partition each cluster into maximal subregions so that all genes within each subregion have the same orientation and there are no intervening genes between these genes within the subregion. I evaluate the agreement between these subregions with experimentally validated *E. coli* operons from the RegulonDB database (Salgado *et al.*, 1999). I apply the algorithm over $1 \leq l \leq 12$ on 700 bacterial genomes.

With these objectives, I organize the dissertation as follows. In Chapter II, I describe the background that includes gene clusters, protein families and operons. In Chapter III, I describe two algorithms for grouping proteins into families so that proteins within the same family have a common function or are related by ancestry. In addition, I demonstrate that the proposed method uses a large-scale dataset such as the protein cluster DB (Klimke *et al.*, 2009) and the entire pfam-a database (Bateman *et al.*, 2000) that has at least two proteins in one family. In Chapter IV, I review the related work on gene clustering on multiple genomes and propose two efficient approaches for investigating clustering of related genes across multiple genomes. I use this algorithm to study gene clustering in 700 bacterial genomes. In Chapter V, I summarize my contributions to developing new gene clustering algorithms and conclude the dissertation by discussing the importance of gene clustering to studying genome organization, evolution and function of individual gene groups, and future work.

CHAPTER II

BACKGROUND

A. Gene cluster

A gene cluster is a set of genes that have a common function. There are several ways to identify gene clusters that belong to a common function. One of the ways is to investigate a set of homologous genes or similar structure of conserved functional domains that can belong to different genomes but have a common function. We identify these gene clusters by comparisons between sequence alignments or functional annotations of genes. Another way is to identify sets of genes that are spatially co-located. The gene cluster is constrained by the maximum proximity between adjacent genes or the maximum size of the cluster, and considered as a set of orthologous and paralogous gene pairs. Orthologs are homologous genes in different species that evolved from a common ancestral gene by speciation, and paralogs are genes related by duplication within a genome (Hunter *et al.*, 2008). The set of orthologous gene pairs is a gene cluster that includes genes with one gene from g_1 genome and another gene from g_2 genome, Those gene clusters within genomic regions descended from a common ancestor are important for understanding the function and evolution of genomes (See Figure 1).

Computational approaches to identifying gene clusters are usually aimed at identifying specific cluster types, such as those that correspond to metabolic pathways or that represent sets of co-expressed genes, which are part of a single transcriptional unit or operon. A generalized approach that can identify all clusters in a genome

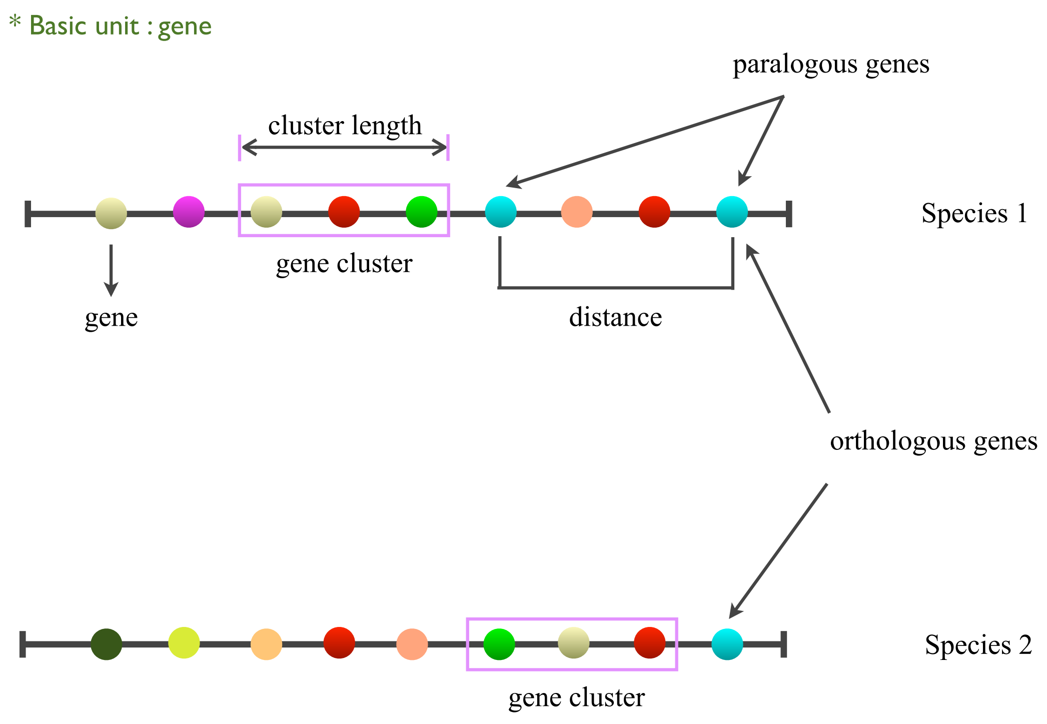


Fig. 1. Illustration of gene cluster.

would be of great value for the study of eukaryotic genome organization and evolution. In addition, identification of gene clusters may help to identify functional relationships among genes, and aide in the discovery of metabolic pathways and protein interactions. Currently, many kinds of computational approaches to identify gene clusters have been applied to identify protein families in which a set of genes in the same family has a common function, or operons in which a set of genes facilitates control of gene expressions.

In this dissertation, a gene cluster is defined as a group of genes that are annotated with the same function or are evolutionarily related from a common ancestor, and are also found within close proximity to each other on a chromosome. Cluster

Table I. UniProt 3-oxoacid CoA-transferase family.

ID	Organism Gene Ontology
SCO2A_MOUSE (Q9JJN4)	Mus musculus (Mouse) GO:0046952, GO:0005739, GO:0008260
SCO2A_RAT (Q5XIJ9)	Rattus norvegicus (Rat) GO:0046952, GO:0005739, GO:0008260
SCOT1_HUMAN (P55809)	Homo sapiens (Human) GO:0046952, GO:0005739, GO:0008260 GO:0042803
SCOT1_PIG (Q29551)	Sus scrofa (Pig) GO:0046952, GO:0005739, GO:0008260
SCOT_CAEEL (Q09450)	Caenorhabditis elegans GO:0046952, GO:0005739, GO:0008260
SCOT_DICDI (Q54JD9)	Dictyostelium discoideum (Slime mold) GO:0046952, GO:0005739 GO:0045335, GO:0008260

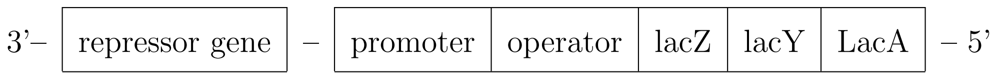
size refers to the number of genes in the cluster having the same function. Cluster length refers to the chromosomal length occupied by the cluster, including intervening genes that are not members of the cluster.

B. Protein family

A protein family is the group of proteins that has a common function or is related by ancestry, so proteins in the same protein family have similar functions, threedimensional structures, and sequences, because proteins descended from a common ancestor are produced as the result of gene duplication, divergence and combination (Vogel and Chothia, 2006).

Currently, many computational approaches have been researched to organize proteins into families and to describe their component domains and motifs, because reliable identification of protein families is important for biologists to study evolutionary analysis, localization analysis, functional annotation, and the exploration of diversity of protein function. There are several protein family databases such as ProtClustDB (Klimke *et al.*, 2009), UniProt (Apweiler *et al.*, 2001), pfam (Bateman *et al.*, 2000) and SCOP (Murzin *et al.*, 1995). ProtClustDB and UniProt support curated protein families with full-length sequences that include all amino acids that begin with N-terminus and end with C-terminus in the amino acids (Table I). Apart from full-length sequences, pfam and SCOP are based on protein domains that are functional regions in proteins. pfam defines a large collection of protein domain families.

One of the reasons that protein family has been needed is that high-throughput genome sequencing has resulted in the availability of large sets of genes, but the functions of many of these genes are still unknown. The traditional way of predicting the gene function is laborious and expensive, since biologists deduce gene functions through experimentation. Thus, the construction of protein families enables identifying functions of unknown sequences.



Repressor Gene - Produces a repressor protein that fits in the operator to turn the operon off

Promoter - RNA polymerase initiates transcription of the genes

Operator - The active repressor physically blocks RNA polymerase and turns off transcription

Structural Genes - The genes that are co-regulated by the operon.

Fig. 2. The example of the lac operon structure.

C. Operon

One of the main mechanisms of facilitating control of a single regulatory signal or promoter in bacteria is organization of the set of adjacent genes into operons (See Figure 2). An operon is a functional unit that organizes groups of transcriptionally linked genes. Operon prediction is based on finding gene clusters in which gene order and orientation is conserved in localized regions of more than two genomes (Ermolaeva *et al.*, 2001), and that belong to the same pathways. Identifying clusters of related genes within localized regions across multiple bacterial genomes helps us study operons and their evolution.

An operon includes three major components that are located each other on the same chromosome. Basically, the operon is working with two regions. The promoter region is the site where RNA polymerase binds to initiate transcription of genes, the operator region is the site where the repressor protein binds to block RNA transcription, and the structural genes are transcribed into mRNA and include information to translate proteins (See Figure 2). An operon is off under normal circumstances when the repressor protein is bound to the operator region, and RNA polymerase is blocked and transcription can not occur. An operon can be turned on only when the repressor protein leaves the operator region. However, the repressor protein will not leave the operator region if substrate is not present. When RNA polymerase is not blocked, it can transcribe the structural genes. The operon will return to the off position and the repressor protein returns to the original shape and bind to the operator region.

CHAPTER III

SUPERVISED PROTEIN FAMILY CLASSIFICATION AND NEW FAMILY CONSTRUCTION

The goal of protein family classification is to group proteins into families so that proteins within the same family have common function or are related by ancestry. While supervised classification algorithms are available for this purpose, most of these approaches focus on assigning unclassified proteins to known families but do not allow for progressive construction of new families from proteins that cannot be assigned. Although unsupervised clustering algorithms are also available, they do not make use of information from known families. By computing similarities between proteins based on pairwise sequence comparisons, we develop supervised classification algorithms that achieve improved accuracy over previous approaches while allowing for construction of new families. We show that our algorithm has higher accuracy rate and lower mis-classification rate when compared to algorithms that are based on the use of multiple sequence alignments and hidden Markov models, and our algorithm performs well even on families with very few proteins and on families with low sequence similarity.

A. Introduction

After a genome is sequenced, the first steps are to annotate protein coding genes and assign the genes to families. Depending on the definition of a family, proteins within gene families (or protein families) are usually homologous and have similar structure of conserved functional domains. The classification of proteins to families is an intrinsic part of comparative and evolutionary genomics. As the pace of genome sequencing continues to increase, the need for sensitive and automated protein family classification methods also continues to increase.

One strategy for protein family classification is through the use of unsupervised clustering algorithms, which take as input a large set of proteins, and perform all-against-all comparisons of sequence similarity using BLAST (Altschul *et al.*, 1990), the Smith-Waterman local alignment algorithm (Smith and Waterman, 1981), or some other pairwise comparison algorithm. The pairwise scores are then used as a basis to apply a variety of clustering algorithms such as single linkage clustering, as implemented in the popular BLAST package. BLASTClust (Dondoshansky, 2002) is based on a hierarchical clustering algorithm by utilizing the BLAST package from NCBI. It yields fairly good results, however, a large single cluster that contains huge number of sequences may be generated due to multi-domain proteins (Chen *et al.*, 2006). Other unsupervised approaches include the Markov clustering strategy (Enright *et al.*, 2002), the density-based ordering method (Chen *et al.*, 2006), and clustering algorithms based on the use of graph-theoretic properties (Kim and Lee, 2006). Unsupervised methods are useful for clustering a set of proteins, but they have two shortcomings: they do not assign proteins to existing families that can be found in databases such as Pfam (Bateman *et al.*, 2000) and InterPro (Mulder *et al.*, 2003), and they do not make use of properties of known families, such as inter- and intra-family sequence diversity, to aide in the formation of new families.

Supervised classification algorithms can overcome some of these shortcomings.

These methods include algorithms based on the use of profile hidden Markov models (Eddy, 1998), algorithms based on probabilistic suffix trees (Bejerano and Yona, 2001), algorithms based on the use of discriminative strategies such as support vector machines (Jaakkola *et al.*, 2000; Liao and Noble, 2003), and algorithms based on sparse Markov transducers (Eskin *et al.*, 2003). Whereas most of these supervised techniques focus on the assignment of unclassified proteins to known families, they do not allow for progressive construction of new families from proteins that cannot be assigned.

We develop supervised classification algorithms that overcome the problems of existing supervised and unsupervised algorithms and achieve improved accuracy. By utilizing sequence similarity from pairwise comparisons, we show that our algorithm has higher accuracy rate and lower mis-classification rate when compared to algorithms that are based on the use of multiple sequence alignments and hidden Markov models. Our approach can assign proteins to existing families in databases, and by taking into account similarities between the unclassified proteins, can assign them to new families. Our algorithm will be useful for rapid assignment of sequences from genome sequencing projects to families.

B. Methods

1. Classifying proteins to existing families

We first consider the problem of assigning an unclassified protein to known families. We assume that each protein belongs to at most one family, which consists of either individual protein domains or single domain proteins. The goal is to determine

```
Algorithm SClassify ($\mathcal{F}$,$p$,$t$)
input: a set $\mathcal{F}$ of protein families, an unclassified protein $p$, $e$-value cutoff $t$;
output: either a family $F \in \mathcal{F}$ to which $p$ is classified, or a new family for $p$; {
        for each family $F \in \mathcal{F}$ do {
            if there exists a protein $p' \in F$ with $e$-value $e(p, p') \leq t$ then {
                $s_F \leftarrow$ average value of $-\log e(p, p')$ over those $p'$ with $e(p, p') \leq t$;
            }
        }
        if $s_F$ is defined for some $F$ then {
            return the family $F$ with the largest $s_F$; }
        else {
            return a new family for $p$;
        }}
```

Fig. 3. Algorithm SClassify when one unclassified protein is given.

whether the unclassified protein belongs to one of the existing families. We consider the following algorithm: compute an e-value score from the given unclassified protein to each protein within the existing families by using a pairwise sequence comparison algorithm such as BLAST (Altschul *et al.*, 1990) or the Smith-Waterman local alignment algorithm SSEARCH (Smith and Waterman, 1981), and assign the unclassified protein to the family with the highest average minus log e-value, where the average is taken over proteins in the family that have e-value below a cutoff (see Figure 3 for the detailed algorithm.)

Because sequences within the same family can have low similarity, we only consider proteins within a family that are of sufficiently high similarity to the unclassified protein during the computation of average scores. This avoids the problem of getting consistently low average scores, and thus not being able to assign any new proteins to these families. When the unclassified protein is not sufficiently similar to any of the existing families, it is not assigned to any family. The worst case time complexity of the algorithm is $O(fs)$, where f is the total number of proteins within the existing families, and s is the time to perform one pairwise comparison. The memory requirement of the algorithm is proportional to the total size of the existing families.

2. Constructing new families from unclassified proteins

We consider the problem when a set of more than one unclassified protein is given, and the goal is either to assign each unclassified protein to an existing family or to automatically construct new families if necessary. We treat each unclassified protein initially as a family by itself, and iteratively merge a family that contains unclassified proteins either into an existing family or with another family that contains unclassified proteins. During each iteration, we make sure that the average minus log e-value between the two merged families is the highest among all pairs of families considered, where the average is taken over pairs of proteins that have e-value below a cutoff. The algorithm terminates when no pairs of proteins have e-value below the cutoff within all pairs of families considered and thus it is no longer possible to merge, with the families that have not been merged into existing families becoming new families (see Figure 4 for the detailed algorithm).

The algorithm also takes advantage of similarity between unclassified proteins but will not merge existing families together, thus there is no need to perform pairwise comparisons between proteins in existing families. There are $u(f+u)$ pairwise comparisons to perform, where f is the total number of proteins within the existing families, and u is the number of unclassified proteins. There are a total of $O(u)$ iterations, with $O(u(n+u))$ average scores to compare within each iteration, and $O(n+u)$ average scores to update after each merge, where n is the total number of existing families. Because each update takes constant time, the worst case time complexity of the algorithm is $O(u(f+u)s+u^2(n+u))$, where s is the time to perform one pairwise comparison. The memory requirement of the algorithm is proportional to the number of scores that need to be stored, which is $u(f+u)$.

3. Availability

The SClassify software is available for download at

http://faculty.cse.tamu.edu/shsze/sclassify.

```
Algorithm SClassify ($\mathcal{F}$,$U$,$t$)
input: a set $\mathcal{F}$ of protein families, a set $U$ of unclassified proteins, $e$-value cutoff $t$;
output: a set of families that include all the proteins; {
$\mathcal{N} \leftarrow \{\{p\} \mid p \in U\}$;
loop {
        for each pair of families $(F_1, F_2)$ where
        $F_1 \in \mathcal{N}$ and $F_2 \in \mathcal{F}$ or $F_2 \in \mathcal{N}$ with $F_1 \neq F_2$ do {
            if there exist proteins $p_1 \in F_1$ and $p_2 \in F_2$ with $e$-value $e(p_1, p_2) \leq t$ then {
                $s_{(F_1,F_2)} \leftarrow$ average value of $-\log e(p_1, p_2)$ over
                those $(p_1, p_2)$ with $e(p_1, p_2) \leq t$;
            }
        }
        if $s_{(F_1,F_2)}$ is defined for some $(F_1, F_2)$ then {
            $(F_1, F_2) \leftarrow$ pair of families with the largest $s_{(F_1,F_2)}$;
            $\mathcal{N} \leftarrow \mathcal{N} - \{F_1\}$;
            if $F_2 \in \mathcal{F}$ then {
                $\mathcal{F} \leftarrow \mathcal{F} - \{F_2\} \cup \{F_1 \cup F_2\}$; }
            else {
                $\mathcal{N} \leftarrow \mathcal{N} - \{F_2\} \cup \{F_1 \cup F_2\}$; } }
        else {
            return the set of families $\mathcal{F} \cup \mathcal{N}$; } } } }
```

Fig. 4. Algorithm SClassify when a set of more than one unclassified protein is given.

C. Experiments

1. Data sets

We apply our algorithm to a few large-scale data sets, including curated families from the Pfam database (Bateman *et al.*, 2000), protein families from the SCOP database (Murzin *et al.*, 1995), full length prokaryotic sequences from the ProtClustDB database (Klimke *et al.*, 2009), and curated proteins from the Swiss-Prot subset of the UniProt database (Apweiler *et al.*, 2001). To compare the performance of our algorithm to slower algorithms, we use families within individual species from Pfam, including *Arabidopsis thaliana*, *Caenorhabditis elegans*, *Drosophila melanogaster*, *Escherichia coli*, *Homo sapiens*, *Mus musculus* and *Saccharomyces cerevisiae*, with the proteins that are within each species forming a data set.

Whereas the sequences from Pfam and SCOP are short sequences that correspond to protein domains, the sequences from ProtClustDB and UniProt are full length sequences that correspond to entire proteins (see Table II). We remove individual proteins that do not belong to any family within each of the data sets, while allowing proteins from different species to be within the same family in the data sets that contain multiple species. Except for UniProt, each remaining protein domain or full length sequence belongs to one family. We remove the very small percentage of proteins (less than 1%) that belong to more than one family from UniProt.

Table II. Data sets for performance evaluation. For each set, family is the number of families, protein is the total number of proteins in all families, avg_pro is the average number of proteins within a family, and avg_len is the average length of proteins in amino acids.

Data set	family	protein	avg_pro	avg_len
Pfam	9318	2286710	245.4	151.1
A. thaliana	1962	36387	18.6	114.7
C. elegans	1164	15971	13.7	137.4
D. melanogaster	1294	13664	10.6	128.6
E. coli	404	2177	5.4	159.7
H. sapiens	1596	23051	14.4	102.2
M. musculus	1516	21891	14.4	110.8
S. cerevisiae	814	4526	5.6	157.2
SCOP	2234	15045	6.7	169.8
ProtClustDB	6521	356615	54.7	348.1
UniProt	8213	448469	54.6	343.7

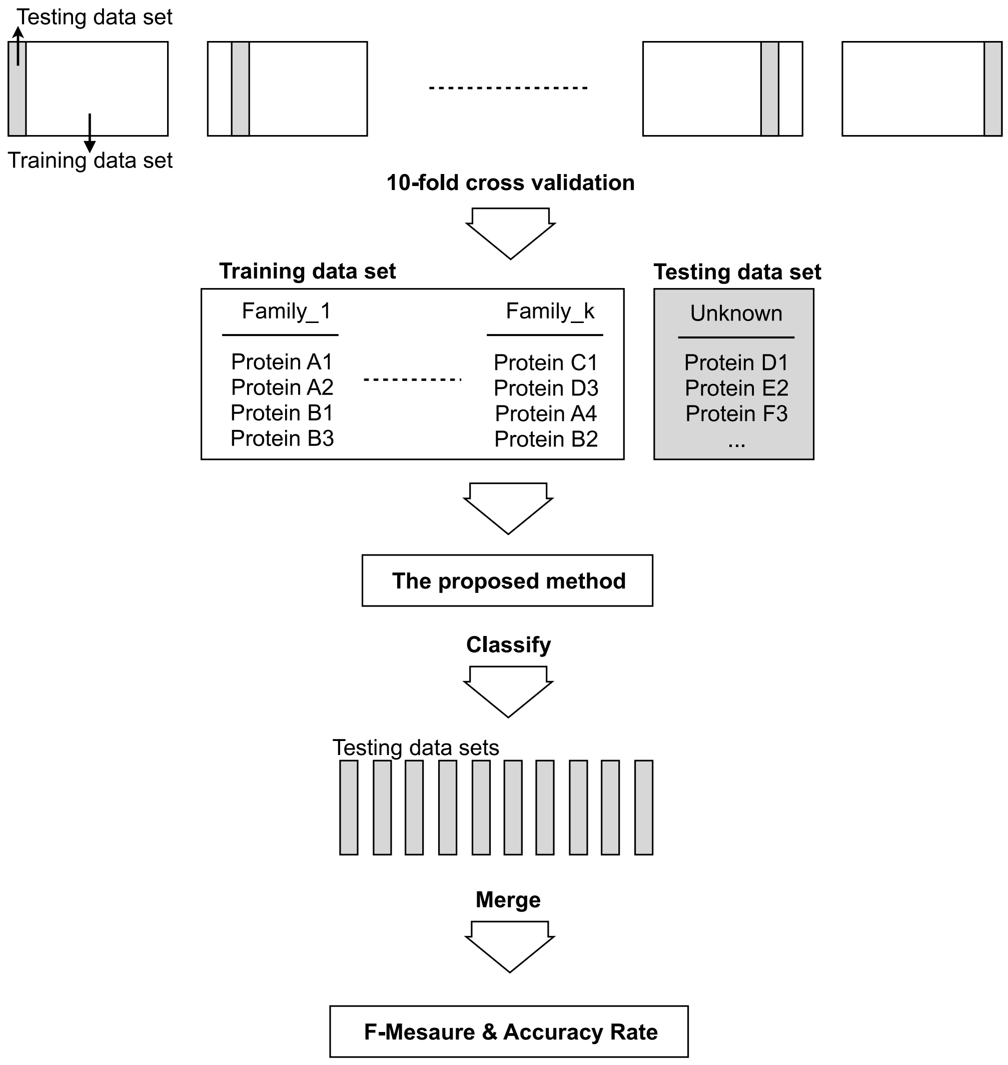


Fig. 5. Illustration of the procedure of the proposed method.

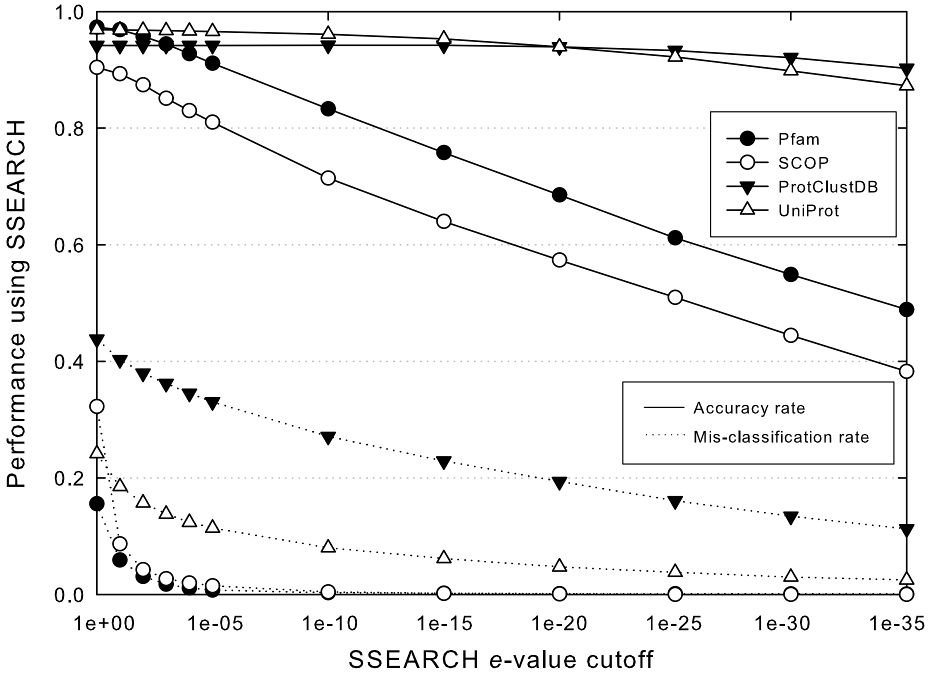


Fig. 6. Accuracy rate and mis-classification rate of SClassify over different SSEARCH e-value cutoffs by using SSEARCH to compute pairwise scores. Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. More tests are performed for e-value cutoffs between 1 and 1e–5.

2. Choice of parameters

We evaluate the accuracy of our algorithm by employing the 10-fold cross validation procedure over different e-value cutoffs (see Figure 5). We randomly subdivide a given set of proteins into ten subsets, and take each subset as a testing data set while using the remaining nine subsets as a training data set. For each testing data set, we take each protein as an unclassified protein and apply our algorithm for classifying one protein against families that include proteins in the training data set. We define the

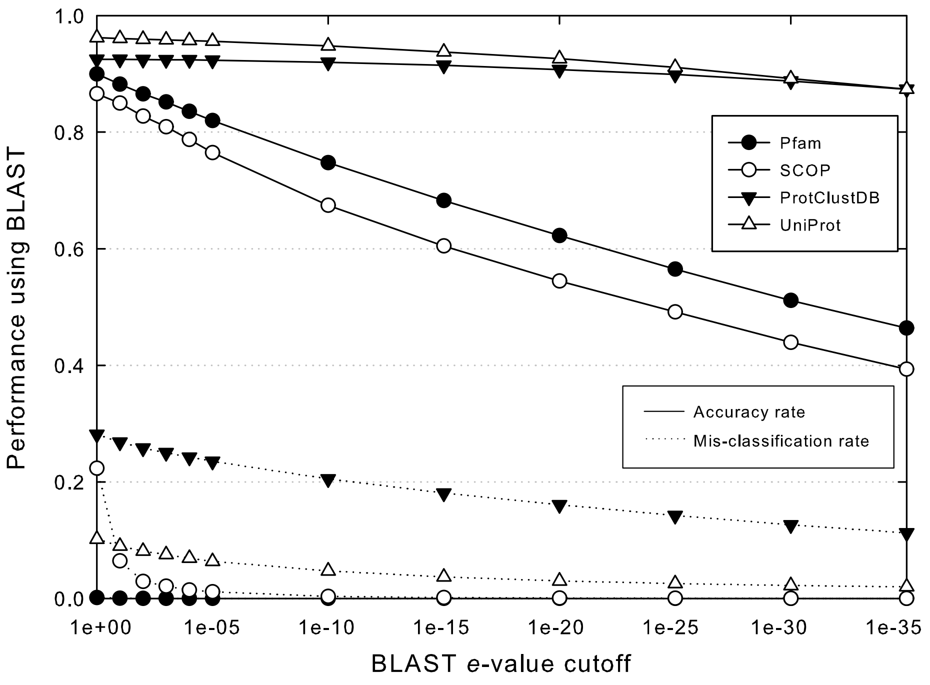


Fig. 7. Accuracy rate and mis-classification rate of SClassify over different BLAST *e*-value cutoffs by using BLAST to compute pairwise scores. Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. More tests are performed for *e*-value cutoffs between 1 and 1e–5.

accuracy rate to be the fraction of proteins that are classified to the correct family.

Since it is not likely that the 10-fold cross validation procedure groups many complete families into individual subsets, we further evaluate the mis-classification of our algorithm by removing proteins from one entire family at a time, and taking each protein as an unclassified protein while applying our algorithm for classifying one protein against all the other families. We define the mis-classification rate to be the fraction of proteins that are incorrectly classified to some other family. Since this procedure retains the largest number of already classified families during each test, it corresponds to the most difficult scenario that one can use for evaluating mis-classification. This procedure complements the above 10-fold cross validation procedure by evaluating the reliability of our algorithm on unclassified proteins that should not belong to any existing families.

Figures 6 and 7 shows the performance of our algorithm SClassify over different e-value cutoffs by using SSEARCH and BLAST to compute pairwise scores. There is a large performance difference between data sets that contain sequences corresponding to protein domains, including Pfam and SCOP, and data sets that contain full length sequences, including ProtClustDB and UniProt. To simultaneously achieve high accuracy rate and low mis-classification rate, a high e-value cutoff is needed for data sets that contain sequences corresponding to protein domains, and a low e-value cutoff is needed for data sets that contain full length sequences to avoid mis-classification.

When SSEARCH is used to compute pairwise scores, we choose an e-value cutoff of 0.1 for data sets that contain sequences corresponding to protein domains, including Pfam and SCOP, which achieves an accuracy rate of at least 89% and a mis-classification rate of at most 9%. We choose a SSEARCH e-value cutoff of 1e–30 for data sets that contain full length sequences, including ProtClustDB and UniProt, which achieves an accuracy rate of at least 90% and a mis-classification rate of at most 13%. For these choices, the minimum accuracy rate is about the same as one minus the maximum mis-classification rate.

We use a similar strategy to obtain appropriate e-value cutoffs for BLAST, except that we consider a BLAST e-value to be above the cutoff if no hits are obtained between a protein pair. We choose a BLAST *e*-value cutoff of 0.1 for data sets that contain sequences corresponding to protein domains, including Pfam and SCOP, which achieves an accuracy rate of at least 85% and a mis-classification rate of at most 6%, resulting in a decrease of the minimum accuracy rate by 4% when using BLAST instead of SSEARCH. Such a decrease in performance is expected since SSEARCH computes optimal alignments while BLAST employs a heuristic, and the similarity scores from SSEARCH are more accurate than the ones from BLAST. We choose a BLAST *e*-value cutoff of 1e–30 for data sets that contain full length sequences, including ProtClustDB and UniProt, which achieves an accuracy rate of at least 89% and a mis-classification rate of at most 13%, resulting in a similar minimum accuracy rate and maximum mis-classification rate as before, although the actual accuracy rate on the ProtClustDB data set will decrease by 3% when using BLAST instead of SSEARCH. A caution is that although the same *e*-value cutoffs are chosen for SSEARCH and BLAST, the SSEARCH *e*-value and the BLAST *e*-value use different formulas and are not directly comparable.

3. Comparison with other supervised algorithms

We compare the performance of SClassify to HMMER (Eddy, 1998), which classifies proteins against existing families according to profile hidden Markov models, to LIB-SVM (Fan *et al.*, 2005), which performs supervised classification based on the use of support vector machines, and to SMT (Eskin *et al.*, 2003), which is a supervised classification algorithm that uses sparse Markov transducers to train a given set of known families. We employ the 10-fold cross validation procedure as before, and use the same training and testing sets in each case.

For SClassify, we use the variant for classifying one protein. For HMMER, we construct a profile hidden Markov model for each family in the training set from a multiple sequence alignment obtained by ClustalW (Thompson *et al.*, 1994). These alignments are constructed either from a subset of distinct sequences in each family with pairwise BLAST e-values above a cutoff, or from all sequences in each family. For each family, a subset of distinct sequences is obtained by starting from an empty subset and iteratively adding a sequence that has the largest possible minimum BLAST e-value against sequences in the current subset as long as this largest minimum is above the cutoff. For the Pfam database, we also use the profile hidden Markov model that was created for each curated family from a subset of curated seed sequences in Bateman *et al.* (2000). For a given e-value cutoff, we assign an unclassified protein to the family with the lowest e-value if such a family exists, otherwise the unclassified protein is not assigned to any family. For LIBSVM, we follow a similar strategy as in Liao and Noble (2003), and define a feature vector for a protein based on the set of minus log e-values from BLAST to each training protein, while allowing missing values that correspond to pairwise scores above the BLAST e-value cutoff 1. Each attribute of the feature vector is normalized by using the same scaling factor for the training and testing data, and the radial basis kernel is used. Various gamma and cost parameters are tested, and the result with the best performance is selected for each data set. For SMT, we use the classifier variant that allows wild cards in the sequence motifs, and assign an unclassified protein to the family with the highest average log probability of the proteins within the family without using a cutoff.

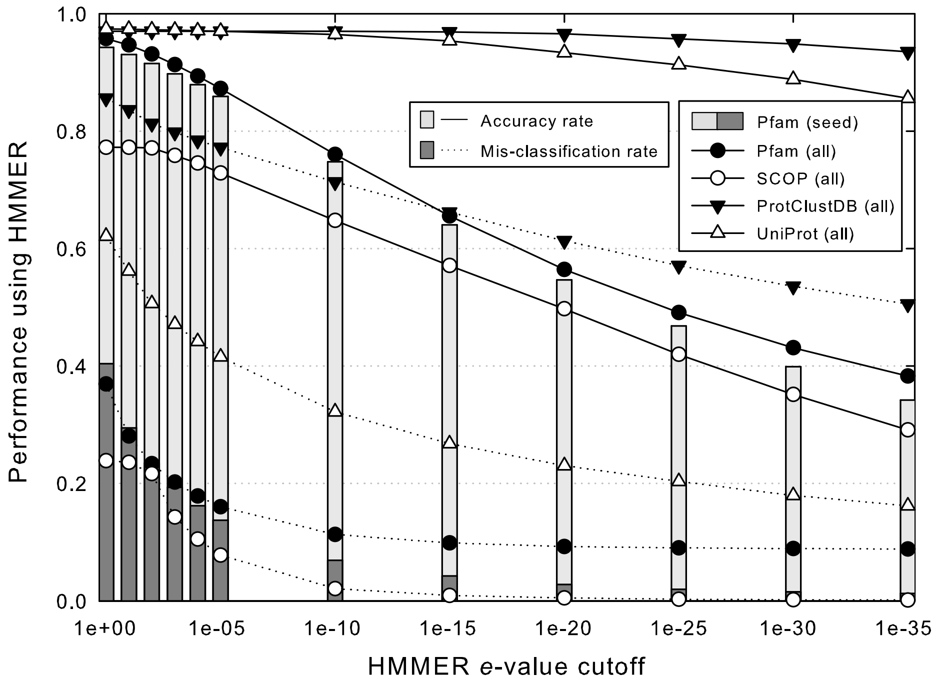


Fig. 8. Accuracy rate and mis-classification rate of HMMER over different *e*-value cutoffs. Light bars and solid lines represent accuracy rates, while dark bars and dotted lines represent mis-classification rates. Bars denote performance on Pfam by applying HMMER on the profile hidden Markov model that was created for each curated family from a subset of curated seed sequences in Bateman *et al.* (2000). Lines denote performance by applying HMMER on the profile hidden Markov model that is constructed for each family in the training set from a multiple sequence alignment obtained by ClustalW. These alignments are constructed from all sequences in each family. More tests are performed for *e*-value cutoffs between 1 and 1e–5.

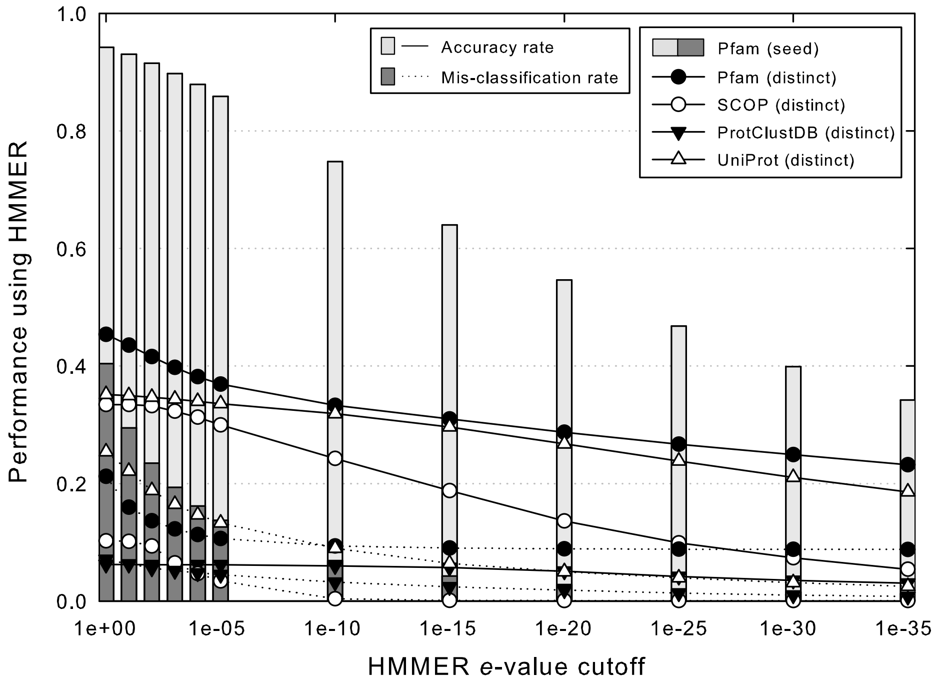


Fig. 9. Accuracy rate and mis-classification rate of HMMER over different *e*-value cutoffs. Light bars and solid lines represent accuracy rates, while dark bars and dotted lines represent mis-classification rates. Bars denote performance on Pfam by applying HMMER on the profile hidden Markov model that was created for each curated family from a subset of curated seed sequences in Bateman *et al.* (2000). Lines denote performance by applying HMMER on the profile hidden Markov model that is constructed for each family in the training set from a multiple sequence alignment obtained by ClustalW. These alignments are constructed from a subset of distinct sequences in each family with pairwise BLAST *e*-values above 0.1 (a BLAST *e*-value is considered to be above the cutoff if no BLAST hits are obtained).

Figures 8 and 9 shows two boundary cases for HMMER in which the pairwise BLAST *e*-value cutoff to obtain a subset of distinct sequences in each family is set to a very high value 0.1, and in which all sequences in each family are used to obtain a multiple alignment. While the minimum mis-classification rate of HMMER is achieved in the first case, the accuracy rate is low. While the maximum accuracy rate of HMMER is achieved in the second case, the mis-classification rate is high. The performance of using other pairwise BLAST *e*-value cutoffs is intermediate between the two. For Pfam, using the profile hidden Markov model that was created for each curated family from a subset of curated seed sequences in Bateman *et al.* (2000) simultaneously achieves high accuracy rate and low mis-classification rate. However, the accuracy rate of HMMER is significantly lower than the accuracy rate of SClassify at a fixed mis-classification rate, while the mis-classification rate of HMMER is significantly higher than the mis-classification rate of SClassify at a fixed accuracy rate (compare to Figure 6 and 7). A caution is that the HMMER *e*-value cutoffs are not directly comparable to the SSEARCH or BLAST *e*-value cutoffs.

Table III shows that SClassify has better accuracy than LIBSVM and much better accuracy than SMT on individual species from Pfam and on SCOP. When compared to our previous results, SClassify performs better on families that contain multiple species from Pfam, which may be due to the increased amount of information from the much larger number of proteins in multiple species.

Table III. Accuracy rate comparison of SClassify (using either SSEARCH or BLAST to compute pairwise scores), LIBSVM and SMT on individual species from Pfam and on SCOP.

Data set	SClassify (SSEARCH)	SClassify (BLAST)	LIBSVM	SMT
Pfam				
A. thaliana	0.96	0.90	0.82	0.11
C. elegans	0.91	0.80	0.71	0.46
D. melanogaster	0.90	0.77	0.68	0.28
E. coli	0.90	0.82	0.61	0.40
H. sapiens	0.92	0.84	0.74	0.22
M. musculus	0.92	0.83	0.76	0.18
S. cerevisiae	0.81	0.72	0.52	0.38
SCOP	0.89	0.85	0.63	0.00

4. Comparison with unsupervised algorithms

We compare the performance of SClassify to unsupervised clustering algorithms MCL (Enright *et al.*, 2002), which uses the Markov cluster algorithm to classify a given set of proteins into families, and to BLASTClust (Altschul *et al.*, 1990), which groups a given set of proteins into clusters based on computing pairwise scores from BLAST. In order to make the results from SClassify comparable to the clusters obtained from these unsupervised classification algorithms, we use the same 10-fold cross validation procedure as before and merge the family assignments of each protein from SClassify to obtain a set of clusters, with each protein that is not assigned to any existing family being in a new family by itself. This procedure roughly models the subdivision of a given set of proteins into clusters of proteins.

We evaluate the performance of each algorithm by checking whether each pair of proteins are correctly classified either to the same family or to different families. For each set of true clusters that are predefined in the original data set and each set of predicted clusters that are obtained from each algorithm, we compute the statistics true positives(TP), false positives(FP) and false negatives(FN), which are the number of protein pairs that are within the same true cluster and within the same predicted cluster, the number of proteins pairs that are within the same predicted cluster but in different true clusters, and the number of protein pairs that are within the same true cluster but in different predicted clusters, respectively (Halkidi *et al.*, 2001). True clusters are predefined by the original data set, thus we know all proteins and their families. Predicted clusters are results from each clustering method. For each protein pair(i, j), I define $T(i, j)$=1 if they are within the same true cluster, $T(i, j)$=0 otherwise. I define $S(i, j)$=1 if they are within the same predicted cluster, $S(i, j)$=0 otherwise. I define TP, TN, FP and FN as follows.

- $TP = \{\ (i, j) \mid T(i, j) = 1 \text{ and } S(i, j) = 1, \text{ where } i < j\ \}$
- $FN = \{\ (i, j) \mid T(i, j) = 1 \text{ and } S(i, j) = 0, \text{ where } i < j\ \}$
- $FP = \{\ (i, j) \mid T(i, j) = 0 \text{ and } S(i, j) = 1, \text{ where } i < j\ \}$
- $TN = \{\ (i, j) \mid T(i, j) = 0 \text{ and } S(i, j) = 0, \text{ where } i < j\ \}$

From these statistics, we compute the precision, the recall and the f-measure as follows:

- $\text{PC} = \dfrac{\text{TP}}{(\text{TP} + \text{FP})}$
- $\text{RC} = \dfrac{\text{TP}}{(\text{TP} + \text{FN})}$
- $\text{F-measure} = 2 \times \dfrac{(\text{PC} \times \text{RC})}{(\text{PC} + \text{RC})}$

Note that the F-measure from SClassify roughly corresponds to the accuracy rate in our previous tests but not the mis-classification rate, and is highly correlated to the accuracy rate.

For SClassify, we use the variant for classifying one protein while using BLAST to compute pairwise scores. For MCL, we use the minus log *e*-value from BLAST as the edge weight, and test various inflation values while selecting the result with the best F-measure for each data set. For BLASTClust, we test various similarity thresholds and minimum length coverages while selecting the result with the best F-measure for each data set.

Figure 10 shows that while MCL and BLASTClust are able to obtain high precision or high recall for some data sets, this is often at the expense of low recall or low precision respectively, and the overall F-measure is not high. By using information from known families, SClassify is able to obtain very good performance with respect to both precision and recall, which results in consistently high F-measure across all data sets.

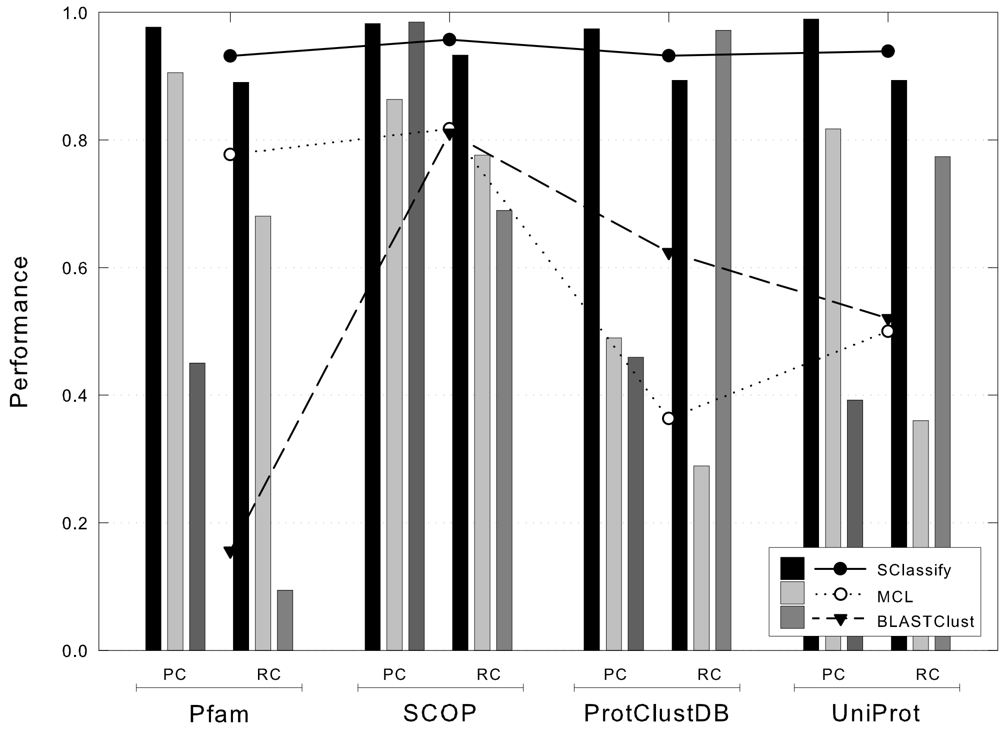


Fig. 10. Performance comparison of SClassify with unsupervised clustering algorithms MCL and BLASTClust by using BLAST to compute pairwise scores. For each data set, PC is the precision, RC is the recall, and the lines denote the F-measure.

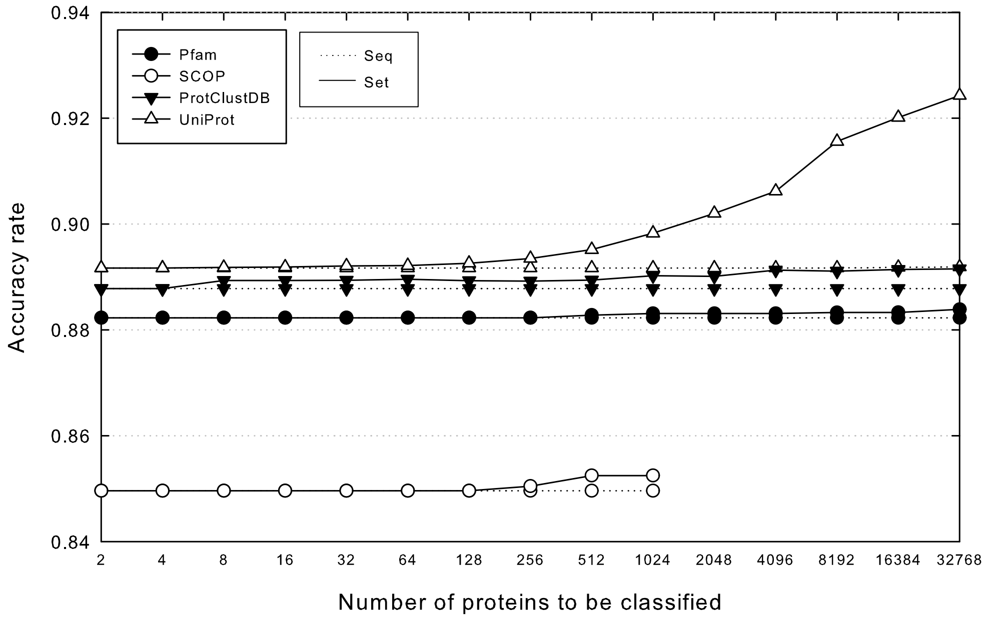


Fig. 11. Accuracy rate comparison of different variants of SClassify when a set of unclassified proteins is given while using BLAST to compute pairwise scores. Given a set U of unclassified proteins, Seq corresponds to the original algorithm for classifying one protein on each protein sequentially in U according to a random order, while Set corresponds to the algorithm for classifying the proteins in U simultaneously by also taking into account similarities between unclassified proteins.

5. Classifying a set of proteins

To evaluate the performance of SClassify when a set of unclassified proteins is given, we perform the 10-fold cross validation procedure as before. Instead of taking each protein within the testing data set as an unclassified protein individually, given a set size u, we subdivide the testing data set further into subsets U that are roughly of the same size u and as evenly as possible. We apply the SClassify variant for classifying a set of proteins on each subset U independently against families that include proteins in the training data set. We compare the performance of this algorithm to the alternative strategy of applying the original algorithm for classifying one protein on each protein sequentially within each subset U according to a random order, in which each newly classified protein in U is retained in its assigned family before the next one in U is classified. To determine whether there are significant improvements in performance between these algorithms, we evaluate their accuracy rates over increasing values of u.

We further evaluate the mis-classification by removing proteins from one entire family at a time and setting U to be the removed family. Since it is possible that proteins in U are classified to different families, we take the largest new family that contain proteins in U to be the correct family for U after classification.

Figure 11 and 12 shows that significantly better accuracy rate can be obtained when u becomes large enough, which is especially evident on UniProt. Significantly lower mis-classification rates are also obtained, which is evident on SCOP and ProtClustDB.

D. Discussion

We have developed an algorithm SClassify that allows both accurate classification of proteins to existing families and progressive construction of new families. We have shown that there is a tradeoff between achieving high accuracy rates and low mis-classification rates. While we have used the default parameters in BLAST that limit the output to the top 250 alignments, removing this constraint simultaneously

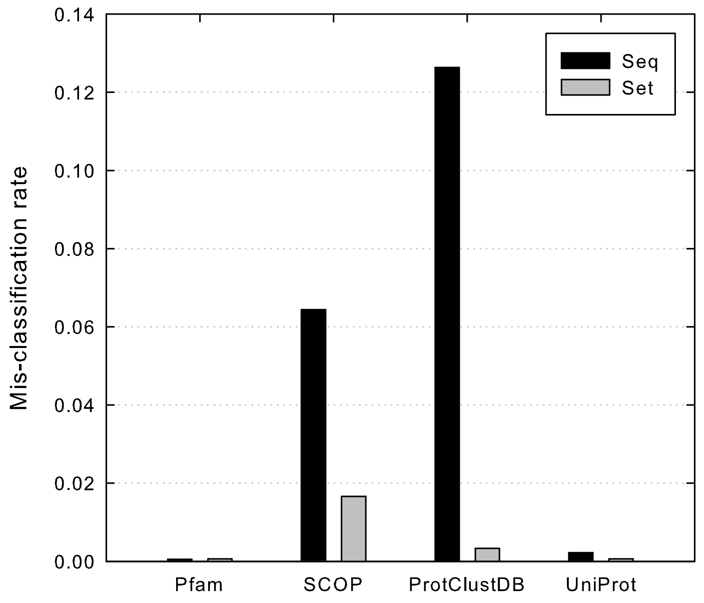


Fig. 12. Mis-classification rate comparison of different variants of SClassify when a set of unclassified proteins is given while using BLAST to compute pairwise scores. Given a set U of unclassified proteins, Seq corresponds to the original algorithm for classifying one protein on each protein sequentially in U according to a random order, while Set corresponds to the algorithm for classifying the proteins in U simultaneously by also taking into account similarities between unclassified proteins.

increases the accuracy rate and the mis-classification rate, and does not necessarily give better results.

To investigate whether SClassify performs well on families with very few proteins and on families with low sequence similarity, we subdivide our results on the variant for classifying one protein into categories by grouping together all families that contain the same number of proteins and grouping together all families with average sequence

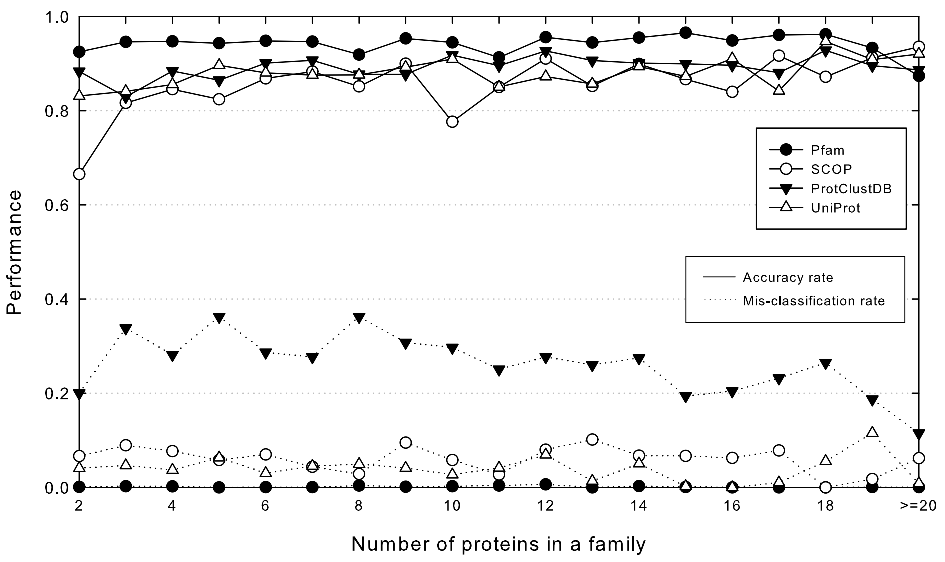


Fig. 13. Accuracy rate and mis-classification rate of SClassify by grouping together all families that contain the same number of proteins (each point denotes the performance on proteins within such families). Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. Missing points correspond to no proteins being assigned to the category.

identity between proteins within a specified range, where the sequence identity is obtained by dividing the number of exact matches by the average length of two aligned proteins. The performance of our algorithm remains high on families with very few proteins or on families with low sequence identity (see Figures 13 and 14).

To evaluate the running time of SClassify on each data set, we apply SClassify against all existing protein families in each data set by using the variant for classifying a set of proteins. Although the worst case time complexity of SClassify is not linear, Figures 15, 16 and 17 shows that the running time is roughly linear after the number of proteins to be classified becomes large enough and the computational overhead

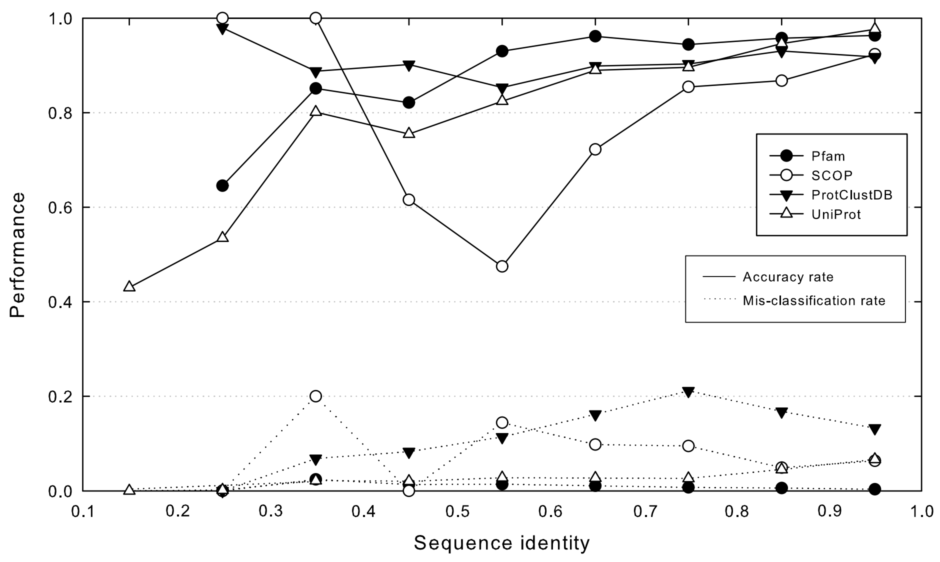


Fig. 14. Accuracy rate and mis-classification rate of SClassify by grouping together all families with average sequence identity between proteins within a specified range (each point denotes the performance on proteins within such families with average sequence identity within the range from l to u, where l is the label to the left of the point on the x-axis and u is the label to the right of the point on the x-axis), while using BLAST to compute pairwise scores. Solid lines represent accuracy rates, while dotted lines represent mis-classification rates. Missing points correspond to no proteins being assigned to the category.

becomes relatively low. When BLAST is used to compute pairwise scores, it takes less than a day on one processor for the largest case with 32768 proteins to classify. When the time to compute pairwise scores is excluded, SClassify is very fast and takes at most an hour in all the tests. If SSEARCH is used instead of BLAST, the total time to compute SSEARCH optimal alignments dominates, and the running time increases by about a factor of ten. The memory requirement is less than two gigabytes in all the tests with at most 32768 proteins to classify (Figure 18).

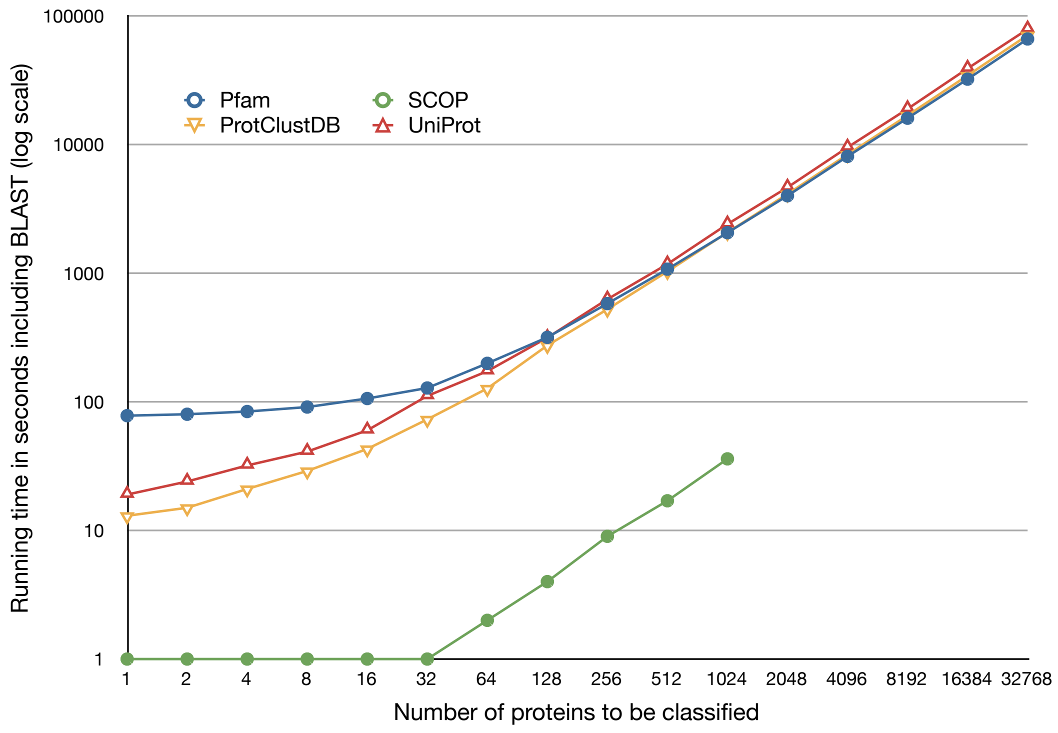


Fig. 15. Running time of SClassify on one processor while including the time to compute pairwise scores by BLAST. In each case, the average running time is given.

We have shown that for the purpose of protein family classification, it may not be necessary to consider more complicated models such as multiple sequence alignments and hidden Markov models, as it is possible to obtain better performance from the use of pairwise sequence comparisons alone. Since pairwise alignments may conflict with each other within a family, such multiple sequence representations will still be needed to define a consistent model for a family. These representations are especially important for determining conserved or critical residue positions within a family. In order to handle multiple domain proteins, domain prediction algorithms such as

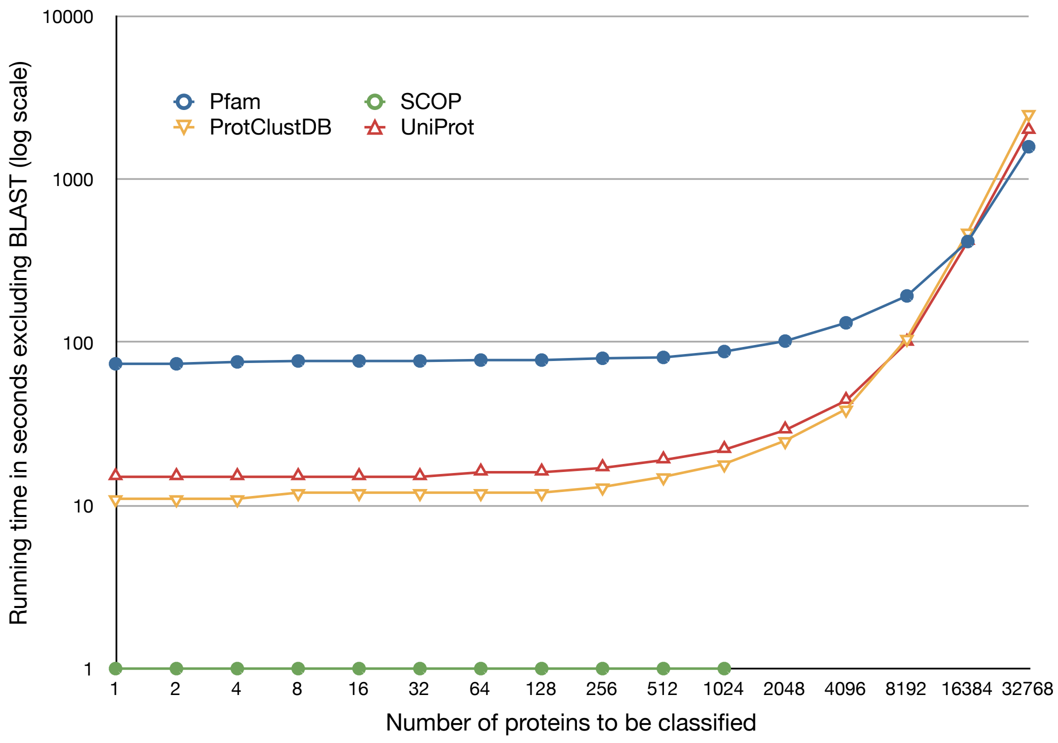


Fig. 16. Running time of SClassify on one processor while excluding the time to compute pairwise scores by BLAST. In each case, the average running time is given.

ADDA (Heger and Holm, 2003) can be used to determine domain boundaries before SClassify is applied.

To illustrate the use of SClassify in real life applications, we consider all 688172 computationally predicted domains in Pfam, and apply SClassify to these domains against all existing curated families in Pfam (containing a total of over two million proteins, see Table II), by using BLAST to compute pairwise scores. It takes about 60 processor-days to obtain all the BLAST scores, and about two weeks on one processor when the variant for classifying a set of proteins is used after all the BLAST scores

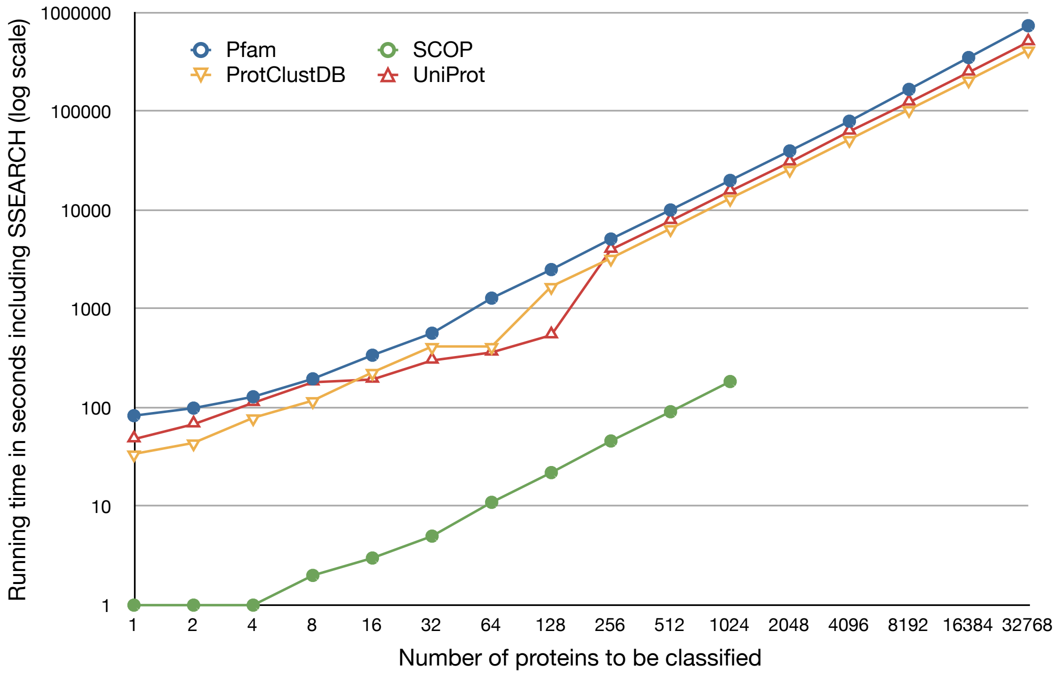


Fig. 17. Running time of SClassify on one processor while including the time to compute pairwise scores by SEARCH. In each case, the average running time is given.

are obtained.

When the variant for classifying a set of proteins is used on all the predicted domains, about 31% of these domains are assigned to some curated family. When the variant for classifying one protein is used sequentially according to a random order of predicted domains, about 27% of these domains are assigned to some curated family. These results are in contrast to the smaller number of predicted domains that are assigned (about 20%) when HMMER is applied with an *e*-value cutoff of 0.1 while using the profile hidden Markov model that was created for each curated family from a subset of curated seed sequences in Bateman *et al.* (2000). A caution is that the

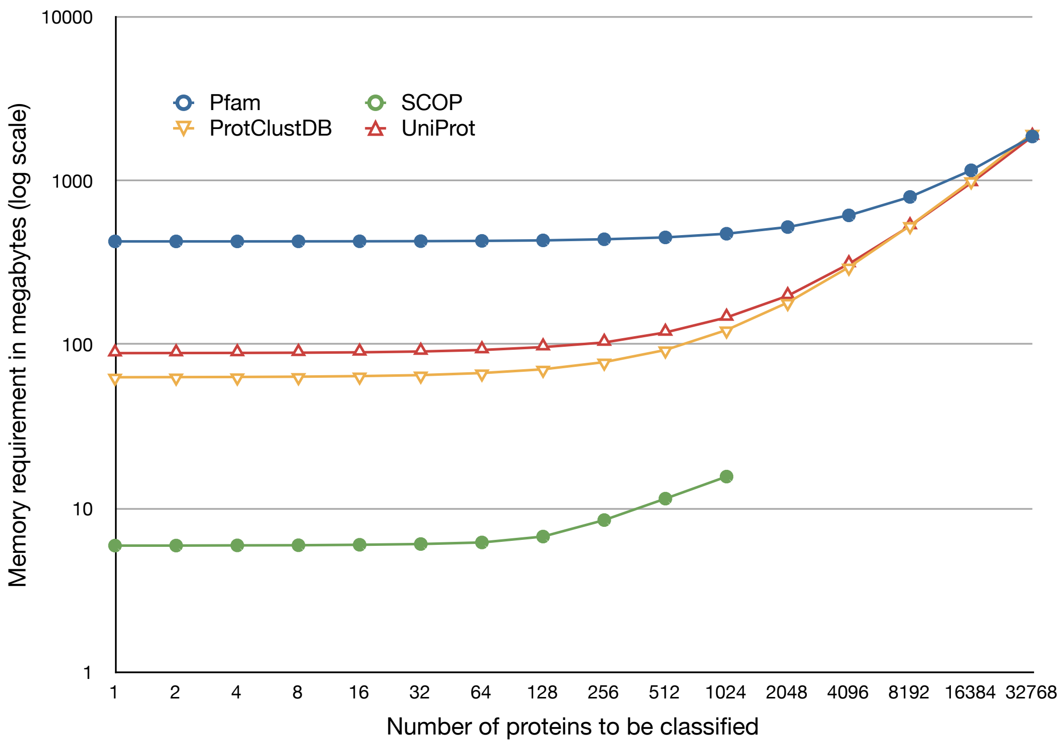


Fig. 18. Memory requirement of SClassify on a single processor while excluding to compute pairwise scores by SSEARCH or BLAST.

domain boundaries in the predictions may not be accurate and the assignments from different algorithms can be quite different from each other.

For the remaining domains that are not assigned to a curated family, both variants of SClassify are able to create new families. In both cases, about 60% of these new families contain only one domain. For the remaining families that contain more than one domain, about 90% of them are completely contained within a computationally generated family in Pfam that contains these predicted domains. These results indicate that the new families generated by SClassify are largely consistent with the families generated by Pfam, except that a large number of them are split into smaller families by SClassify due to insufficient sequence similarities.

While most existing classification algorithms are based on sequence similarity information from alignments, alignment-free approaches are also available (Ma and Chan, 2008). One future direction is to investigate whether it is possible to use these techniques to improve accuracy.

CHAPTER IV

LARGE-SCALE CLUSTERING ACROSS MULTIPLE GENOMES

Identifying genomic regions that descended from a common ancestor helps us study the gene function and genome evolution. In distantly related genomes, clusters of homologous gene pairs are evidence used in function prediction, operon detection, etc. (Raghupathy and Durand, 2009). Thus, reliable identification of gene clusters is critical to functional annotation and analysis of genes. Many kinds of computational methods have been proposed defining gene clusters in order to study the genome organization, evolution and function of individual gene groups. But there are not many efficient algorithms for identifying gene families and operons in the large-scale data set. Thus, the goal of the proposed method is to define an efficient gene clustering algorithm that can investigate related gene clusters across multiple genomes and study operons and their evolutions.

A. Introduction

In bacteria, one of the main mechanisms to facilitate control of gene expression is the organization of gens into operons (Che *et al.*, 2006; Price *et al.*, 2005; Salgado *et al.*, 2000; Yang and Sze, 2008). In gene clustering on chromosomes, the most popular approaches require that the distance between adjacent genes in a cluster to be small (A. Bergeron and Raffinot, 2002; He and Goldwasser, 2005; Luc *et al.*, 2003; Parida, 2007), because homologous regions are straightforward when genomic regions are closely related (Raghupathy and Durand, 2009). However, it has been shown that there are considerable difficulties to develop efficient algorithms when paralogous genes are allowed in the model (He and Goldwasser, 2005). By using a different formulation of gene clusters which requires that the distance between any pair of genes within a cluster to be small (Rose Hoberman and Durand, 2005), we observe that it is possible to use a window-based strategy that can identify significantly more functionally enriched gene clusters than previous approaches.

Several clustering methods have been developed to find gene clusters that have common functions in different genomes. In order to identify gene clusters, GeneTeams (A. Bergeron and Raffinot, 2002; Luc *et al.*, 2003) require that the distance between adjacent genes in a cluster to be small. It allows intervening genes that appear consecutively, possibly in different orders, between genes in a cluster so that various cluster sizes can be adopted. But it includes a restriction that each gene has at most one occurrence in each chromosome. This limits the model as many genomes contain multigene families. To overcome the problem, HomologyTeams (He and Goldwasser, 2005) is a generalized version of GeneTeams that does not require a cluster to appear in every chromosome. It provides statistical significant estimates of identified gene clusters and a fast way to identify sets of gene clusters. Within the gene clusters, neither the order of the genes nor their orientation need to be conserved, but a fixed threshold of the distance between adjacent genes limits the model (Pertea *et al.*, 2009). DomainTeams (Pasek *et al.*, 2005) is a modified version of GeneTeams that considers chromosomal regions of conserved protein domains as domain teams rather than each gene as basic unit, while other algorithms allows multiple genomes and use a gene proximity parameter that restricts the number of intervening genes between adjacent genes in a cluster.

Didier (2003); Heber and Stoye (2001); Schmidt and Stoye (2004) generalized the common interval formulation to allow for the inclusion of paralogous genes within a genome by representing sets of genes as sequences rather than permutations and proposed efficient algorithms to find conserved clusters. It finds all common intervals of two sequences in $O(n^2 \log n)$ time, using $O(n)$ space. Schmidt and Stoye presented an algorithm that finds all common intervals of two sequences and extended the algorithm to find all common intervals in more than two genomes.

Since it is difficult to find gene clusters from sequence comparisons, stringent alignment criteria and statistical validation are required to evaluate accurately whether the association between two or more genes found in the same order on two chromosomal segments in different genomes occurs by chance or reflects true colinearity (Salse *et al.*, 2008). FISH (Calabrese *et al.*, 2003) models the probability of observing putative segmental homologies and establishes correspondences between segments on two chromosomes that may not simply be genes, but imposes an almost colinear ordering of pairwise homologous regions.

Lee and Sonnhammer (Lee and Sonnhammer, 2003) examines genes linked to the same pathway described in the Kyoto Encyclopedia of Genes and Genomes (Kanehisa and Goto, 2000). The average distance of gene pairs within a pathway is compared to the distance calculated with a randomized gene order. Each genome is analyzed separately by identifying clusters of genes belonging to the same metabolic pathway and comparing the result across a large number of genomes. This method is one of the first genome-wide analyses of metabolic pathway clustering in eukaryotes which revealed that gene clusters may span large segments of the genome. One drawback of the strategy is that it is impossible to utilize comparative data during the initial analysis.

While many of these algorithms can be used to perform gene clustering across two or more genomes, very few algorithms are efficient enough to analyze a large number of genomes (Kim *et al.*, 2005). By placing a stricter limit on the maximum cluster size, we observe that efficient algorithms can be developed to perform gene clustering on hundreds of genomes at the same time. This strategy is sufficient for analyzing gene clustering in bacterial genomes, and allows for large-scale study of evolutionary relationships of gene clusters and study of operon formation and destruction. We develop a different formulation based on constraining the overall size of a cluster and develop statistical estimations that allow direct comparisons of clusters of different sizes. It also allow the study of whether gene clusters in bacteria occur only at the operon level or whether there are higher-level structures, and the functional relationships between them. This algorithm performs seven hundred bacteria data sets which contain a well-characterized list of operons such as *Escherichia coli K12* and perform comparative analysis of operon occurrences, gene orientations, and rearrangements both within and across clusters.

B. Method

1. Window-based approach

We investigate a different formulation which requires that the distance between any pair of genes within a cluster to be small (Rose Hoberman and Durand, 2005), by placing a constraint on the overall cluster size. We allow genes in a same paralogous group to appear in more than one gene on a same chromosome. This model is suitable for identifying gene clusters in localized regions with a small number of gene rearrangements.

Let $C = \{c_1, \ldots, c_k\}$ be a set of k chromosomes, one from each genome under consideration. We represent each gene by a number (while ignoring its orientation) so that the same number represents a set of orthologous genes in different genomes and each chromosome c_i is represented by a sequence of gene numbers. The same gene number is allowed to appear more than once within each c_i which represents paralogous copies of a gene.

A gene cluster G' that appears in k' chromosomes is represented by k lists $P'_1, \ldots, P'_k$, with k' of these lists non-empty and each list P'_i containing a set of positions on c_i, which together specify all the positions of genes that are in G'. To ensure that all genes in G' are clustered within a region of size at most l on each chromosome c_i for which P'_i is non-empty, we require that each gene number that appears in G' must appear at least once in each of the k' non-empty lists P'_i, and within each non-empty P'_i, $|r - s| < l$ for any pair of positions $r, s \in P'_i$ (Figure 19).

To obtain statistical significance estimates, let n be the average length of the chromosomes c_i and l' be the average size of the non-empty lists P'_i. The probability of G' appearing in a given chromosome can be obtained by assuming a random background distribution based on n, l and l' (Durand and Sankoff, 2003). We estimate the p-value of G' by the probability of G' appearing in at least k' out of k chromosomes using the binomial distribution, and obtain an e-value from this p-value as follows.

```
Algorithm window_based_clustering($\{c_1, ..., c_k\}$,$l$,$n$,$e$) {
  $\mathcal{G} \leftarrow$ empty;
  for $s \leftarrow 1$ to $k$ do {
    for each windows $w_1$ of length $l$ on chromosome $c_s$ do {
      $G' \leftarrow$ set of gene numbers in $w_1$;
      $G'' \leftarrow$ empty;
      for $i \leftarrow 1$ to $k$, where $i \neq s$, do {
        $P_i \leftarrow$ set of positions on chromosome $c_i$ in which gene numbers in $G'$ appear;
        for $j \leftarrow 1$ to $|P_i|$ do {
          $w_j \leftarrow$ window of length $l$ starting from $j$th position in $P_i$ on $c_i$;
          $G''_{i,j} \leftarrow$ set of gene numbers in $w_j$ that must appear at least once in $G'$; }}
      for each tuple($j_1$,...,$j_k$) with 1$\leq j_i \leq |P_i|$ do {
        $G' = G_s \bigcap (\bigcap_{i=1}^{k} G_{ij_i}$, where $i \neq s)$;
        if $e$-value($G'$) that appears in $G''$ with $n, l, l' \leq e$ then {
          add $G'$ to $\mathcal{G}$;
        }}}}}
```

Fig. 19. Algorithm to identify gene clusters that include same gene number at least once within a window of length l. Paralogous genes are allowed while requiring that each cluster appears in each of the k given chromosomes. $\mathcal{G}$ is the set of clusters returned with each cluster G' represented by a list of gene numbers. l, n, e is the window length, the average length of chromosomes, and e-value cutoff respectively.

$$p\left(n,l,l'\right)=\left(\left(n-l\right)\begin{pmatrix} l-1 \\ l'-1 \end{pmatrix}+\begin{pmatrix} l \\ l' \end{pmatrix}\right)/\begin{pmatrix} n \\ l' \end{pmatrix}$$

$p(n,l,l')$ is a simple definition of a gene cluster that calculates the probability of observing a cluster in a chromosome. If genes in l' are required to appear in a given order, the probability of observing the gene cluster is $p(n,l,l')/l'$, but we consider the probability of observing such a gene cluster in a chromosome with a random gene order, so $p(n,l,l')$ performs a significant test that shows how a gene cluster were found.

In case of G' appearing in k' chromosomes, we estimate the p-value of G' by the probability of G' appearing in at least k' out of k chromosomes using the binomial distribution, thus the p-value can be modeled as

$$p\left(k,k',n,l,l'\right)=\sum_{i=k'}^{k}\begin{pmatrix} n \\ l' \end{pmatrix}p\left(n,l,l'\right)^{i}\left(1-p\left(n,l,l'\right)\right)^{k-i}$$

If a gene cluster is found in more chromosomes, the probability of this gene cluster decreases and its statistical significance increases. The e-value of G' from the p-value can be estimated by

$$e\left(k,k',n,l,l'\right)=\begin{pmatrix} n \\ l' \end{pmatrix}p\left(k,k',n,l,l'\right)$$

$$\begin{array}{lllllllllll}
c_1 = (& \boxed{g_1} & \boxed{g_2} & \boxed{g_3} & g_4 & g_5 & g_6 & g_6 & g_7 & &) \\
c_2 = (& g_7 & \boxed{g_2} & g_5 & \boxed{g_3} & g_6 & \boxed{g_1} & g_4 & \boxed{g_2} &) & \\
c_3 = (& g_8 & g_5 & g_4 & g_9 & \boxed{g_1} & g_7 & \boxed{g_2} & g_7 & g_8 &)
\end{array}$$

$$G_1 = \{g_1, g_2, g_3\}$$
$$G_{21} = \{g_2, g_3\}, G_{22} = \{g_1, g_3\}, G_{23} = \{g_1, g_2\}, G_{24} = \{g_2\}$$
$$G_{31} = \{g_1, g_2\}, G_{32} = \{g_2\}$$

$$\begin{array}{ll}
G'_1 = G_1 \cap G_{21} \cap G_{31} = \{g_2\} & G'_2 = G_1 \cap G_{21} \cap G_{32} = \{g_2\} \\
G'_3 = G_1 \cap G_{22} \cap G_{31} = \{g_1\} & G'_4 = G_1 \cap G_{22} \cap G_{32} = \{\} \\
G'_5 = G_1 \cap G_{23} \cap G_{31} = \{g_1, g_2\} & G'_6 = G_1 \cap G_{23} \cap G_{32} = \{g_2\} \\
G'_7 = G_1 \cap G_{24} \cap G_{31} = \{g_2\} & G'_8 = G_1 \cap G_{24} \cap G_{32} = \{g_2\}
\end{array}$$

Fig. 20. Illustration of the window-based algorithm when window length $l = 3$. The window under consideration on chromosome c_1 is enclosed in a box. The occurrences of the gene numbers in chromosomes c_2 and c_3 are enclosed in boxes. The set of returned clusters is $\mathcal{G} = \{G'_1, \ldots, G'_8\}$.

We consider the following algorithm that guarantees that all clusters with e-value below a cutoff are found. We start with each window of size l on chromosome c_1 and find the locations of all occurrences of its gene numbers in the other chromosomes. We form windows of size l on each chromosome that contain these occurrences and intersect them, up to k windows at a time with one from each chromosome, to obtain a list of candidate gene clusters G' each containing a list of gene numbers. The positions of the genes in each cluster G' is recovered by investigating how the intersection was originally obtained (See Figure 20).

We apply the algorithm over $1 \leq l \leq 50$ on four yeast genomes, including *Saccharomyces cerevisiae*, *Saccharomyces paradoxus*, *Saccharomyces mikatae* and *Saccharomyces bayanus*, with ortholog groups from Kellis *et al.* (2003). We retain gene clusters that appear in all the four genomes with e-value below 1e–5 while removing clusters that have their position lists of gene occurrences completely contained within another cluster with a better e-value.

We compare results to distinct clusters obtained from GeneTeams (A. Bergeron and Raffinot, 2002; Luc *et al.*, 2003), which is among the most popular algorithms that require the different gene distance parameter that restricts number of intervening genes between adjacent genes in a cluster to be small. We evaluate the functional enrichment of each cluster by applying the GO Term Finder (Boyle *et al.*, 2004) on the *Saccharomyces cerevisiae* genes in the gene cluster. The Gene Ontology (GO) is a common controlled vocabulary of terms and phrases describing the function of genes and gene products (Ashburner *et al.*, 2000). Each GO term is assigned to one of the three categories of molecular function, biological process or cellular component. The terms and relationships are organized into a directed acyclic graph (DAG) in which vertices represent GO terms and edges represent relationships among similar terms. Genes can be annotated with GO terms creating gene associations that can be used for whole genome analyses. The Gene Ontology provides a rich framework for identifying gene clusters, regardless of the evolutionary mechanisms responsible for their formation (See Figure 21). We define a cluster to have a significant GO term if its Bonferroni corrected p-value is below 0.05 within any of the three ontologies that are biological process, molecular function and celluar component.

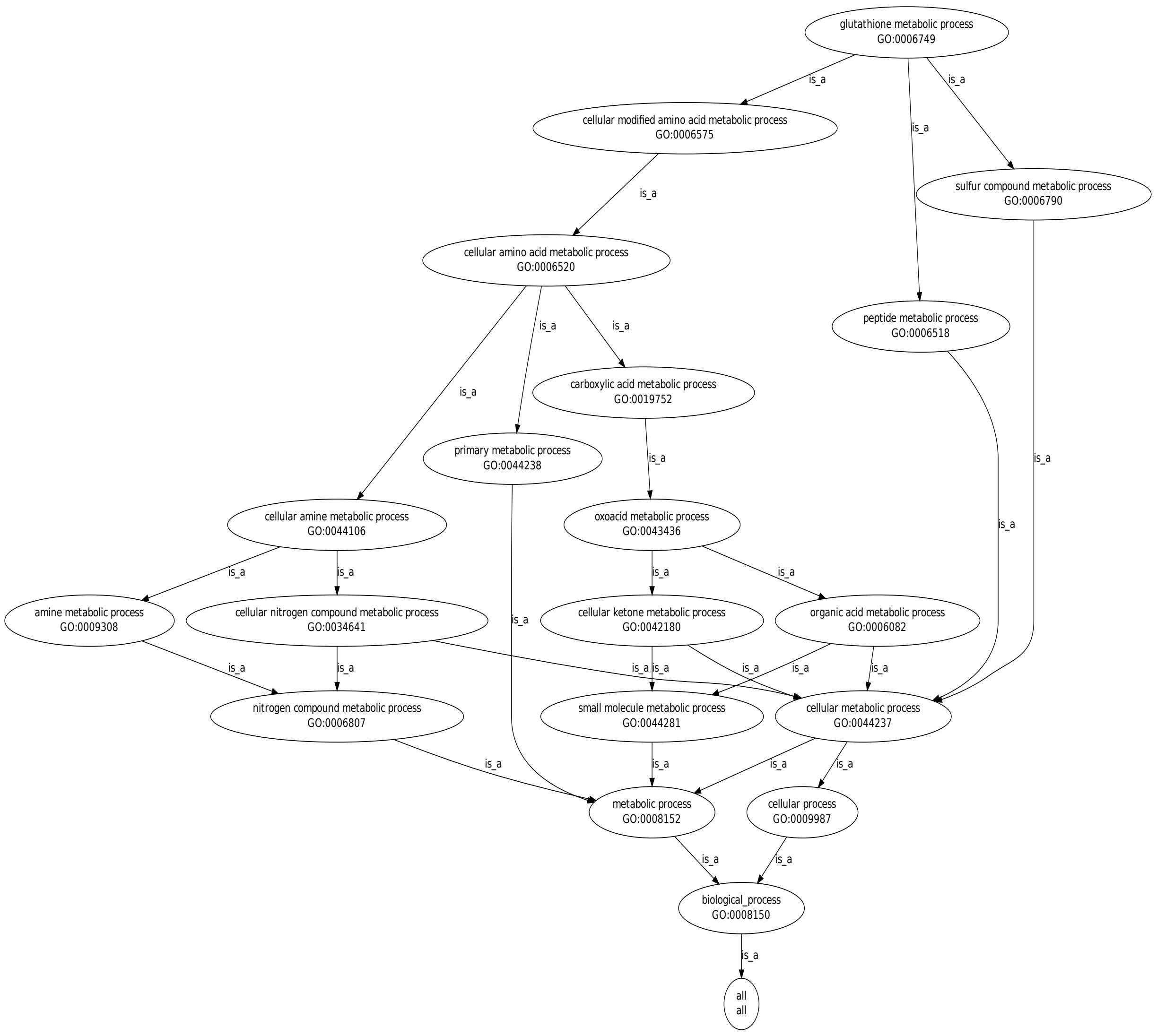

glutathione metabolic process GO:0006749
cellular modified amino acid metabolic process GO:0006575
sulfur compound metabolic process GO:0006790
cellular amino acid metabolic process GO:0006520
peptide metabolic process GO:0006518
carboxylic acid metabolic process GO:0019752
primary metabolic process GO:0044238
cellular amine metabolic process GO:0044106
oxoacid metabolic process GO:0043436
amine metabolic process GO:0009308
cellular nitrogen compound metabolic process GO:0034641
cellular ketone metabolic process GO:0042180
organic acid metabolic process GO:0006082
nitrogen compound metabolic process GO:0006807
small molecule metabolic process GO:0044281
cellular metabolic process GO:0044237
metabolic process GO:0008152
cellular process GO:0009987
biological_process GO:0008150
all all
is_a

Fig. 21. The biological process overview of the glutathione metabolic process (GO:0006749) in the GO structure.

Table IV. Result comparison with GeneTeam.

	#cluster	#cluster (GO)	#gene	#gene (GO)	#extra	#extras (GO)
GeneTeams	2534	606	4510	2240	316	587
Proposed algorithm	2633	1203	4197	3239	3	1586

Table IV shows results where #cluster is the total number of gene clusters, #cluster(GO) is the number of clusters that have significant GO terms, #gene is the total number of *S. cerevisiae* genes covered by the gene clusters, #gene(GO) is the number of genes covered by gene clusters that have significant GO terms, #extra is the total number of genes that are covered by gene clusters from one algorithm but not by gene clusters from the other, and #extra(GO) is the number of genes that are covered by gene clusters with significant GO terms from one algorithm but not by gene clusters with significant GO terms from the other.

While the total number of gene clusters and the total number of genes covered by these gene clusters are similar for both algorithms, the window-based algorithm found almost twice as many gene clusters that are significantly functionally enriched. A similar trend was observed both for the number of genes covered by the gene clusters and the number of genes that are covered by gene clusters from one algorithm but not by gene clusters from the other.

2. Large-scale clustering across multiple genomes

We investigate a modified version of the window-based strategy to analyze hundreds of genomes simultaneously by placing a stricter limit on the maximum cluster size. This makes it possible to avoid the combinatorial explosion of intersecting all combinations of up to k windows at a time with one from each chromosome, and allow the use of a different strategy to find the clusters.

Let $C = \{c_1, \ldots, c_k\}$ be a set of k chromosomes that is same as Window-based approach, one from each genome under consideration. We represent each gene by a number, while ignoring its orientation, so that the same number represents a set of orthologous genes in different genomes and each chromosome c_i is represented by a sequence of gene numbers. The same gene number is allowed to appear more than once within each c_i which represents paralogous copies of a gene.

We consider each window of length l within each chromosome and enumerate each of the 2^l subsets of genes within the window. We think of each subset S as a potential gene cluster and identify the subset of chromosomes in which all genes in S appear within a window of length l (See Figures 22 and 23). A subset S that appears in k' chromosomes is represented by k lists $P'_1, \ldots, P'_k$, with k' of these lists non-empty and each list P'_i containing a set of positions on c_i, which together specify all the positions of genes that are in S. To ensure that all genes in S are clustered within a region of size at most l on each chromosome c_i for which P'_i is non-empty, we require that each gene number that appears in G' must appear at least once in each of the k' non-empty lists P'_i, and within each non-empty P'_i, $|r - s| < l$ for any pair of positions $r, s \in P'_i$.

```
Algorithm largescale_clustering($\{c_1, ..., c_k\}$,$l$,$n$,$m$,$e$) {
 $\mathcal{G} \leftarrow$ empty;
 for $s \leftarrow 1$ to $k$ do {
   for each windows $w_1$ of length $l$ on chromosome $c_s$ do {
     $G' \leftarrow$ set of gene numbers in $w_1$;
     for each combination $B$ of $G'$, where $|B| \geq 2$, do {
       $G'' \leftarrow$ empty;
       for $i \leftarrow 1$ to $k$, where $i \neq s$, do {
         $P_i \leftarrow$ set of positions on chromosome $c_i$ in which gene numbers in $B$ appear;
         for $j \leftarrow 1$ to $|P_i|$ do {
           $w_j \leftarrow$ window of length $l$ starting from $j$th position in $P_i$ on $c_i$;
           $G''_{i,j} \leftarrow$ set of gene numbers in $w_j$ that must appear at least once in $B$; }}
       if $e$-value($B$) that appears in $G''$ with $n, l, l' \leq e$ then {
         add $B$ to $\mathcal{G}$;
       }}}
```

Fig. 22. Algorithm to identify gene clusters that include same gene numbers at least once within a window of length l by the combination of unique gene numbers not to lose potential gene clusters. Paralogous genes are allowed while requiring that each cluster appears in each of the k given chromosomes. $\mathcal{G}$ is the set of clusters returned with each cluster B represented by a list of gene numbers. l, n, m, e is the window length, the average length of chromosomes, the minimum number of chromosomes to appear, and e-value cutoff respectively.

$$\begin{array}{lccccccccc}
c_1 = & g_1 & * & g_2 & * & \boxed{g_3} & \boxed{g_4} & & & \\
c_2 = & g_3 & g_5 & g_{10} & * & g_1 & g_2 & g_4 & l = 2 & l' = 2 \\
c_3 = & g_2 & g_5 & \boxed{g_4} & \boxed{g_3} & * & * & g_1 & k = 3 & k' = 2
\end{array}$$

$$\begin{array}{lccccccccc}
c_1 = & \boxed{g_1} & * & \boxed{g_2} & * & g_3 & g_4 & & & \\
c_2 = & g_3 & g_5 & g_{10} & * & \boxed{g_1} & \boxed{g_2} & g_4 & l = 3 & l' = 2 \\
c_3 = & g_2 & g_5 & g_4 & g_3 & * & * & g_1 & k = 3 & k' = 2
\end{array}$$

$$\begin{array}{lccccccccc}
c_1 = & g_1 & * & g_2 & * & g_3 & g_4 & & & \\
c_2 = & \boxed{g_3} & \boxed{g_5} & g_{10} & * & g_1 & g_2 & g_4 & l = 3 & l' = 2 \\
c_3 = & g_2 & \boxed{g_5} & g_4 & \boxed{g_3} & * & * & g_1 & k = 3 & k' = 2
\end{array}$$

$$\begin{array}{lccccccccc}
c_1 = & g_1 & * & \boxed{g_2} & * & g_3 & \boxed{g_4} & & & \\
c_2 = & g_3 & g_5 & g_{10} & * & g_1 & \boxed{g_2} & \boxed{g_4} & l = 4 & l' = 2 \\
c_3 = & \boxed{g_2} & g_5 & \boxed{g_4} & g_3 & * & * & g_1 & k = 3 & k' = 3
\end{array}$$

Fig. 23. Illustration of sample clusters of size greater than one. l, l', k and k' represent the window size, the number of genes in the window, the total number of chromosomes and the number of chromosomes that appears gene clusters, respectively.

To obtain statistical significance estimates, the p-value of S is estimated by the probability of S appearing in at least this many chromosomes. Let n be the average length of the chromosomes c_i and l' be the average size of the non-empty lists P'_i. The probability of S appearing in a given chromosome can be obtained by assuming a random background distribution based on n, l and l' where n is the average number of genes in a genome, l is the window size, and l' is the number of genes selected by subregion S. This calculation is same as Window-based approach. We estimate the p-value of S by the probability of S appearing in at least k' out of k chromosomes spanning at most l in each case using the binomial distribution, and obtain an e-value from this p-value, where l' genes are observed in windows of length at most l in k' out of k chromosomes.

Since all possible combinations of included genes and intervening genes are included within each window, this algorithm will not lose any clusters that satisfy the definition. One important advantage of the algorithm is that its time complexity grows linearly with the input size and the base of 2 in the exponential part of the time complexity is small, thus a large number of genomes can be considered at the same time.

C. Experiments

1. Data sets

The data set used in the experiment is comprised of 700 bacterial genomes from NCBI (ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria). The total number of genes used in 700 bacterial genomes is 2214301 genes, and the average number of genes per a genome

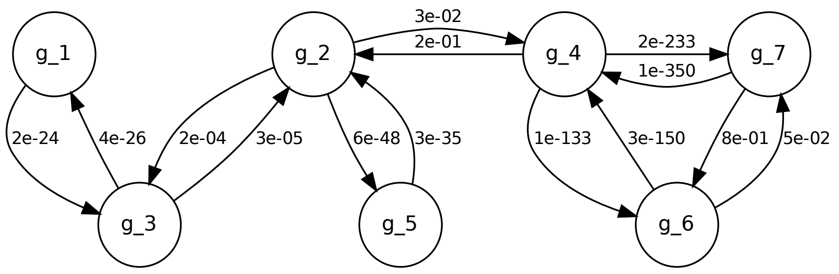
Table V. Distribution of the number of homologs and its average size over different *e*-value cutoffs.

BLAST *e*-value cutoff	1e–05	1e–10	1e–15	1e–20
#Homologs	64340	76873	87329	97220
Avg. Homolog Size	31.80	25.99	22.49	19.66

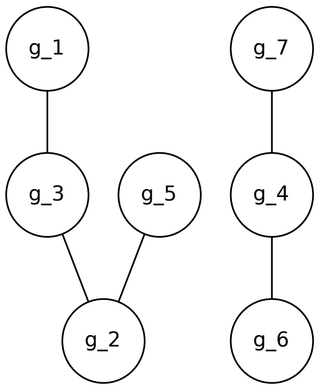
is 3163 genes. We apply the algorithm over $1 \leq l \leq 12$ on these bacterial genomes, with homolog groups constructed by finding bidirectional best hits using protein-protein BLAST (Altschul *et al.*, 1990) with various *e*-value cutoffs, and performing single linkage clustering. The stringent *e*-value cutoff to construct homologous groups generates homologs of the smaller size and the larger number of homologs (See Table V and Figure 24). Since we allow paralogous and orthologous genes, genes that belong to the same homologous group identified by each *e*-value cutoff can appear more than once on the same chromosome and can also appear more than once in the different genomes. We do not assign a gene into new homologous group if only one gene is within the homologous group, since this enables the combination of homolgous group within a windows to bre reduced. We construct the gene map for each chromosome with the gene location information defined by NCBI.

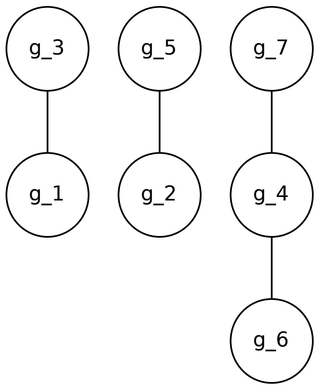
2. Implementation

We implements the above described algorithm and provides output of the clusters and statistical test in human readable format. The algorithm finds all gene clusters that



The graph by the BLAST hits





e-value cutoff : 1e–01 *e*-value cutoff : 1e–10

Fig. 24. Illustration of constructing groups of homologous genes. The first graph is constructed by BLAST *e*-value. Two graphs below the first original graph represent different homologous groups constructed by the different *e*-value cutoff. The stringent *e*-value cutoff makes homologous groups smaller in size, but produces a larger number of homologous group.

cluster 1		$\boxed{h_2}$		$\boxed{h_4}$	$\boxed{h_5}$		*e-value* : 2.32*e*-08
removed				$\boxed{h_4}$	$\boxed{h_5}$		*e-value* : 1.27*e*-05
removed		$\boxed{h_2}$		$\boxed{h_4}$			*e-value* : 3.05*e*-05
cluster 2	$\boxed{h_1}$	$\boxed{h_2}$	$\boxed{h_3}$	$\boxed{h_4}$	$\boxed{h_5}$	$\boxed{h_6}$	e-*value* : 4.32e-04
removed			$\boxed{h_3}$	$\boxed{h_4}$		$\boxed{h_6}$	e-*value* : 1.32e-04
removed	$\boxed{h_1}$	$\boxed{h_2}$					e-*value* : 4.32e-03
removed			$\boxed{h_3}$			$\boxed{h_6}$	e-*value* : 2.02e-02
removed	$\boxed{h_1}$	$\boxed{h_2}$	$\boxed{h_3}$				e-*value* : 3.12e-02
removed				$\boxed{h_4}$		$\boxed{h_6}$	e-*value* : 3.33e-01
removed		$\boxed{h_2}$	$\boxed{h_3}$				e-*value* : 4.82e-01

Fig. 25. Illustration of removal of clusters that are exact subsets of a larger cluster that has a lower *e*-value.

have an *e*-value below a user specified cutoff and as such, numerous overlapping gene clusters are often reported. To facilitate comparative analyses of multiple genomes and improve readability of the output, I also apply two steps: filtering subsets and grouping subsets (Figure 25 and 26). The standard filtering step consists of the removal of clusters that are exact subsets of a larger cluster that has a lower *e*-value. I also implement the grouping step that either masks or removes highly similar overlapping clusters. In the grouping process, the clusters are first sorted by *e*-value. Then, starting with the cluster with the lowest *e*-value, all other clusters that overlap by a user specified threshold are labeled as members of a group of overlapping clusters. This process is repeated for each cluster that has not yet been labeled as a member of another group. A user supplied parameter defines whether the labeled groups are

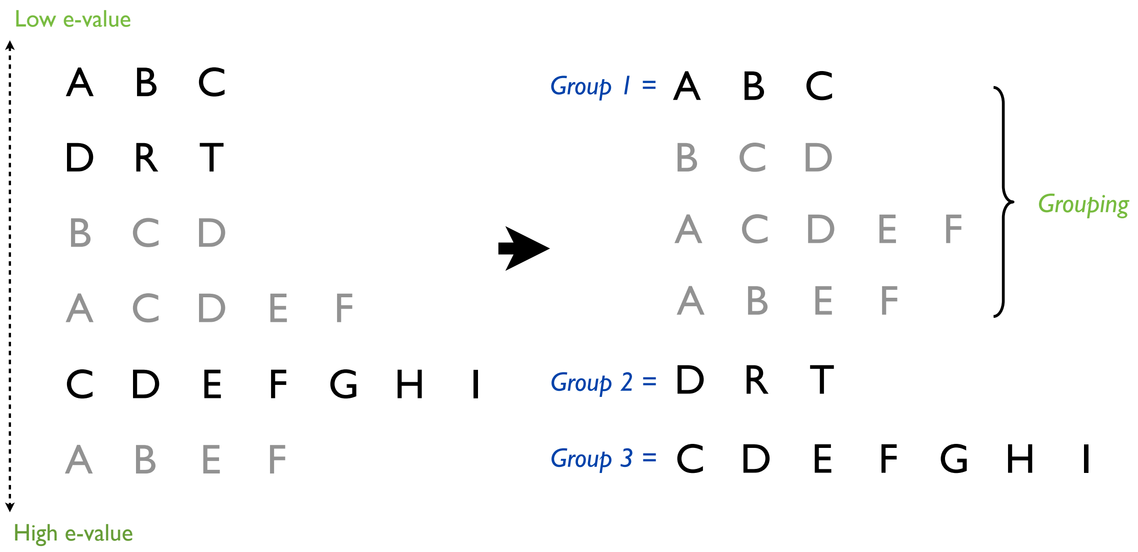


Fig. 26. Illustration of clusters that overlap by a user specified threshold. Clusters are labeled as a member of a group of overlapping clusters. Threshold of overlapping clusters = 50%.

reported in the output file or are ignored.

3. Gene clusters on *E. coli K12*

To validate our algorithm, we compare the results of gene clusters that include *E. coli K12* operons which are experimentally confirmed by the RegulonDB database (Salgado *et al.*, 1999). For each predicted gene cluster, we retain gene clusters with an *e*-value below a certain cutoff, while allowing gene clusters that are completely contained within another gene cluster with a better *e*-value so that we do not lose any potential gene clusters that satisfy the definition. We investigate whether these gene clusters correspond mostly to one operon or many operons with different *e*-value cutoffs, and we partition each gene cluster into maximal subregions so that all genes within each subregion have the same orientation and there are no intervening genes between these genes within the subregion. Maximal subregions are constructed by intersection and union. Intersection subdivides subregion by the start and end position of overlap, and union makes a group of overlapping subregions into one region. We evaluate predicted subregions with respect to a given operon from the database by computing the S_{min} score that divides the number of genes that overlap in both an operon and the gene cluster by the minimum size of the operon and the gene cluster. We compute the S_{max} score that divides the number of genes that overlaps in both the operon and the gene cluster by the maximum size of the operon and the gene cluster. A predicted gene cluster will have a high S_{min} and low S_{max} score if the size of either operon from the database or subregion is very small, then the S_{min} score will be high, although the significant overlaps with genes is small. Thus, we compute the S_{avg} score which divides the number of genes that overlap in both the operon and the gene cluster by the average size of the operon and the gene cluster.

For each subregion, we only consider predicted gene clusters that include *E. coli*. We investigate subregions with different *e*-value cutoffs, and evaluate the agreement between these subregions with experimentally validated *E. coli* operons from the database. Given a subregion, we compute it with respect to the entire RegulonDB by finding the maximum scores of S_{min}, S_{avg} and S_{max} over all operons from the database. Since these evaluations are only approximations, predicted subregions may correspond to higher level organizations such as super-operons or more than one operon, and real

Table VI. Intersection & Union of subregions to *E. coli* operons from the RegulonDB database.

BLAST	Intersection				Union			
e-value	gclust *e*-value							
S_{min}	1e-05	1e-10	1e-15	1e-20	1e-05	1e-10	1e-15	1e-20
1e-01	0.9937	0.9969	0.9985	0.999	0.9692	0.9673	0.9676	0.9673
1e-10	0.9934	0.9964	0.9984	0.999	0.9687	0.9673	0.9676	0.9673
1e-20	0.9932	0.9964	0.9982	0.9987	0.9689	0.9676	0.9682	0.9676
1e-30	0.9928	0.9964	0.9982	0.9989	0.9695	0.968	0.9692	0.9673
1e-40	0.993	0.996	0.9975	0.9983	0.9716	0.9686	0.9688	0.9677
1e-50	0.9919	0.9954	0.997	0.998	0.9713	0.9699	0.9699	0.967
S_{avg}	1e-05	1e-10	1e-15	1e-20	1e-05	1e-10	1e-15	1e-20
1e-01	0.6884	0.6799	0.6778	0.6757	0.7684	0.7661	0.7661	0.7648
1e-10	0.6885	0.6802	0.678	0.6759	0.7669	0.7661	0.7661	0.7649
1e-20	0.6891	0.6799	0.6777	0.6762	0.7697	0.7672	0.7668	0.7658
1e-30	0.6893	0.6798	0.6779	0.6762	0.7729	0.7696	0.769	0.7661
1e-40	0.6888	0.6796	0.6784	0.6763	0.7733	0.771	0.7718	0.7677
1e-50	0.6897	0.6799	0.6791	0.6771	0.775	0.7744	0.7727	0.771
S_{max}	1e-05	1e-10	1e-15	1e-20	1e-05	1e-10	1e-15	1e-20
1e-01	0.6042	0.5942	0.5919	0.5898	0.6995	0.697	0.6966	0.6952
1e-10	0.6044	0.5946	0.5921	0.5899	0.6974	0.697	0.6967	0.6952
1e-20	0.605	0.5942	0.5918	0.5903	0.7007	0.698	0.6973	0.6965
1e-30	0.6052	0.5942	0.5919	0.5904	0.7043	0.7004	0.6999	0.697
1e-40	0.6045	0.5939	0.5926	0.5906	0.7045	0.7024	0.7032	0.6988
1e-50	0.6058	0.5944	0.5939	0.5919	0.7072	0.7068	0.7041	0.7026

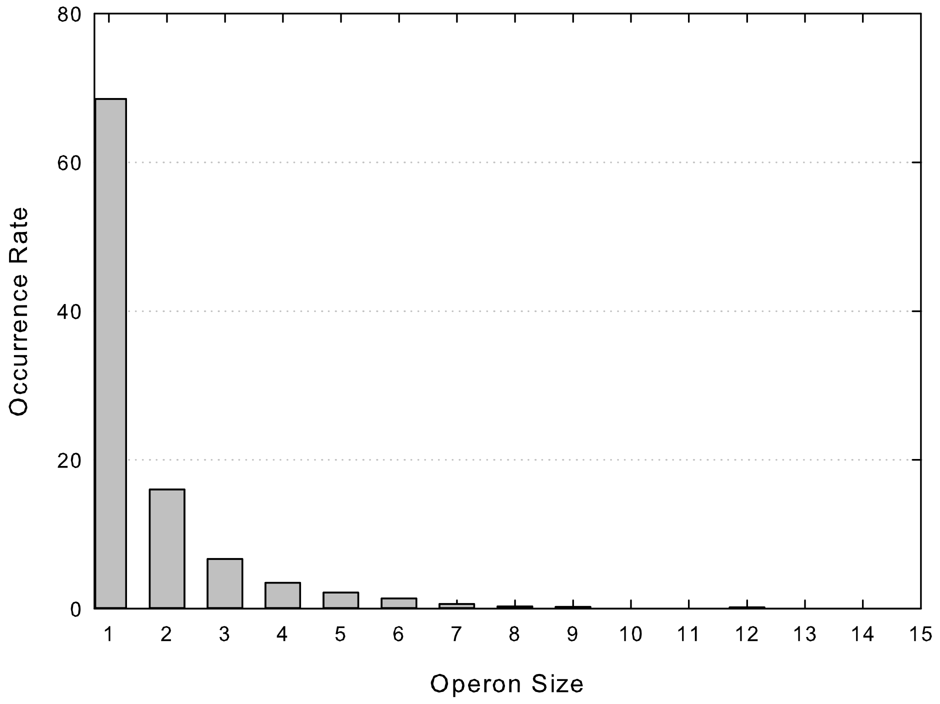


Fig. 27. The distribution of *E. coli* operon size in the database.

operons that are not in the database or are not experimentally confirmed yet.

To obtain appropriate e-value cutoffs for BLAST and gene cluster, we perform our algorithm by employing different e-value cutoffs in both BLAST and gene cluster (Table VI). The goal of choosing the e-value is to ensure that the number of predicted subregions is not too low while having a high average S_{min} score and an acceptable average S_{max} score, since the result may be different in other bacteria. We choose a BLAST e-value cutoff of 1e-10 for constructing homlogous groups, and our algorithm e-value cutoff of 1e-10 for retaining gene clusters below the cutoff. The overlapping subregion by intersection decreases in S_{min} and increases in S_{max}, while union increases in both S_{min} and S_{max}. We choose the optimal cutoff which does not drastically effect performance and achieves 99% and 96% in both intersection and union rates respectively. S_{avg} and S_{max} in intersection outperforms other cutoffs, although it does not achieve the best rate in union: the rate only decreases 1% in overall cutoffs. We will use these *e*-value cutoffs to define significant clusters in our later analysis.

To show that the choice of *e*-value gives high accuracy, we investigate whether subregions correspond mostly to one operon or many operons with the *e*-value cutoffs. Figure 27 shows the size of the *E. coli* operon, in which 68% of operons in the database are the size of one and the maximum operon size in the database is 15 genes (0.08%) in two operons. While most *E. coli* operons contain a small number of genes, most of the clusters contain at least a few subregions (see Table VII and VIII). About half of these subregions have perfect overlap with an *E. coli* operon by intersection and union, thus the gene clustering results reveal a significant amount of spatial conservation that is at a higher level than operons, and some of these clusters are likely to correspond to uber-operons (Lathe *et al.*, 2000; Che *et al.*, 2006).

4. Comparative analysis on bacteria groups

We study the distribution of occurrences of operons in 23 bacterial groups (Table IX). We define the occurrence rate of each bacterial group that appears in gene cluster to be a number of each bacterial group that divides by the total of genomes that appears in gene cluster. We define the overall occurrence rate of a unique bacteria group in all gene cluster to be a number of gene clusters that divides by the total number of gene

Table VII. Intersection result of subregion overlaps with the *E. coli* operon database. (a) showing the distribution of the number of subregions among all clusters, (b) showing the distribution of the size of subregions among all clusters, and (c) showing the distribution of the best overlap of subregions with *E. coli* operons among the ones that has overlap (S_{min}, S_{avg} and S_{max}) with some *E. coli* operon, which is defined to be the largest ratio of the number of shared genes between a subregion and an operon to the average number of genes of the subregion and the operon.

(a)		(b)		(c)			
#subregion	%cluster	size	%subregion	%overlap	%subregion		
					min,	avg,	max
1	0.26	1	96.68	0-9	0.00	0.00	3.45
2	5.10	2	2.26	10-19	0.00	3.61	10.54
3	21.70	3	0.65	20-29	0.05	10.49	14.24
4	33.32	4	0.22	30-39	0.05	5.72	10.70
5	25.12	5	0.08	40-49	0.03	8.52	0.24
6	10.80	6	0.05	50-59	0.43	11.02	19.98
7	3.04	7	0.03	60-69	0.13	19.82	0.32
8	0.59	8	0.03	70-79	0.00	0.05	0.05
9	0.06			80-89	0.00	0.32	0.05
10	0.00			90-100	99.30	40.44	40.41
11	0.00						

Table VIII. Union result of subregion overlaps with the *E. coli* operon database. (a) showing the distribution of the number of subregions among all clusters, (b) showing the distribution of the size of subregions among all clusters, and (c) showing the distribution of the best overlap of subregions with *E. coli* operons among the ones that has overlap (S_{min}, S_{avg} and S_{max}) with some *E. coli* operon, which is defined to be the largest ratio of the number of shared genes between a subregion and an operon to the average number of genes of the subregion and the operon.

(a)		(b)		(c)			
#subregion	%cluster	size	%subregion	%overlap	%subregion		
					min,	avg,	max
1	0.26	1	36.87	0-9	0.00	0.08	0.08
2	5.10	2	20.34	10-19	0.49	0.89	3.00
3	21.70	3	11.43	20-29	1.30	4.05	9.16
4	33.32	4	8.59	30-39	0.97	4.70	8.02
5	25.12	5	6.32	40-49	0.41	5.59	4.70
6	10.80	6	4.54	50-59	0.97	10.94	18.56
7	3.04	7	2.92	60-69	1.10	18.15	7.70
8	0.59	8	2.43	70-79	0.24	1.70	1.95
9	0.06	9	1.38	80-89	0.08	7.37	0.73
10	0.00	10	1.54	90-100	94.49	46.52	46.11
11	0.00	11,12,13	0.97,0.57,0.41				
		14,15,16	0.49,0.49,0.16				
		17,18,20	0.08,0.08,0.16				
		23,26,35	0.08,0.08,0.08				

Table IX. Bacterial groups.

Bac Group	%Appear in original data	%Appear in gene cluster
Gammaproteobacteria	23.86	29.15
Firmicutes	20.86	22.73
Alphaproteobacteria	11.86	10.17
Betaproteobacteria	8.29	8.72
Actinobacteria	7.43	6.71
Euryarchaeota	4.71	3.40
Cyanobacteria	4.57	5.38
Epsilonproteobacteria	2.86	2.86
Deltaproteobacteria	2.57	1.32
Bacteroidetes/Chlorobi	2.57	1.50
Crenarchaeota	2.29	1.81
Chlamydiae/Verrucomicrobia	1.86	3.63
Spirochaetes	1.57	2.78
Chloroflexi	1.00	0.99
Other Bacteria	1.00	0.35
Thermotogae	1.00	1.04
Deinococcus-Thermus	0.57	0.47
Aquificae	0.29	0.06
Acidobacteria	0.29	0.09
Other Archaea	0.14	0.01
Planctomycetes	0.14	0.03
Fusobacteria	0.14	0.01
Nanoarchaeota	0.14	0.0

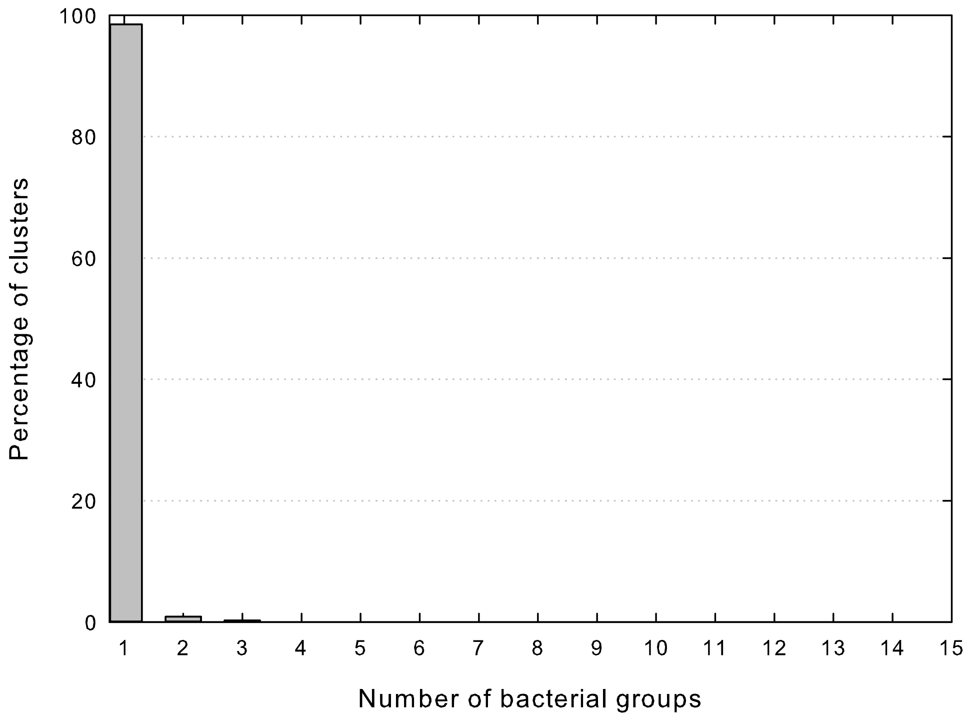


Fig. 28. The distribution of the number of bacterial groups in gene cluster.

clusters. Because the algorithm finds gene clusters by homologs that have a similar gene structure and evolutionary origin to a gene in another species, Figure 28 shows that 98% of gene clusters belong to one bacterial group, which is not surprising. In five gene clusters, all 23 bacterial groups appeared. These gene clusters consist of two homologs with (h_0,h_{251}), (h_0, h_{385}), (h_0,h_{300}), (h_0,h_{717}) and (h_0,h_{920}), where h_0 was found in common. Since h_0 appears in 30% of the total number of homologs in all genomes, and the rest of homologs appear in at least 600 genomes.

5. Gene rearrangement

The gene rearrangement between adjacent genes within bacterial operons is important for function, expression and regulation of these genes (Itoh *et al.*, 1999; Tamames, 2001). We study the distribution of gene order with subregions. For a given pair of subregions in a genome g_1 and a set of correspondences with each of them linking a gene in a subregion to a related gene in subregion of another genome g_2, we obtain a subset of one-to-one corresponding pairs of link as follows: if there is more than one link for a gene in subregion s_1 to another gene in subregion s_2, we retain the one with the lowest BLAST e-value in both s_1 and s_2. In the remaining set of k genes in s_1 and k related genes in s_2, we assign a number from 1 to k to each gene according to the order of genes in subregion, and assign a direction of genes to the number from 1 to k by its gene strand, such as forward and reverse. k genes in subregion s_1 that correspond to a signed permutation, in which each pair of neighboring genes in s_1 is with number n_1 and n_2, are considered to be a breakpoint if n_1 and n_2 are not consecutive ($|n_1 - n_2| \neq 1$) (Kececioglu and Sankoff, 1995). This iterates $|g|(|g|-1)/2$ times, where $|g|$ is the total number of genomes. We define the percentage of conserved neighboring gene pairs to be the total number of neighboring gene pairs that are not breakpoints, which is divided by the total number of neighboring gene pairs by k-1, and use it to evaluate the degree of conservation of gene order. 99.92% and 94.6% had perfectly conserved neighboring gene pairs in the intersection and union test, respectively, which means that gene order within subregions are the same either both the forward and reverse directions. While comparing all subregions in g_1 and g_2, we also evaluate subregion in g_1 that corresponds to one subregion in g_2 with the

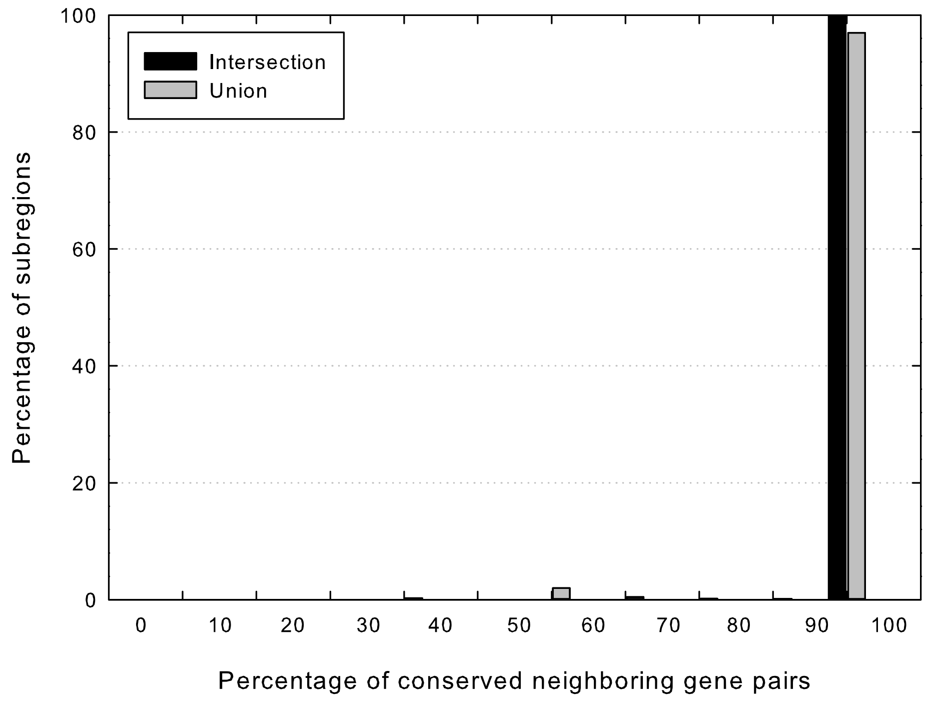


Fig. 29. Distribution of the percentage of conserved neighboring gene pairs.

best percentage of conserved neighboring gene pairs (Figure 29). 99.91% and 96.93% of them are perfectly conserved neighboring gene pairs in intersection and union, respectively. We also performed this rearrangement test within gene clusters. 39.74% are perfectly conserved. This is not surprising since it is possible to locate more than one operon within a gene cluster. Although gene order within operons can be unstable (Itoh *et al.*, 1999), our results on gene orientation and gene order indicate that predicted subregions tend to contain only one orientation and the gene order tends to be conserved.

To investigate stronger correlations between the frequency of gene duplication

Table X. Pearson correlation coefficient of subregions and gene clusters.

	intersection		union		within cluster
	all	best	all	best	
CorrCoef	0.63	0.58	0.51	0.63	0.10
p-value	0.0	4.4e-16	2.2e-16	1.1e-16	5.6e-16

that contains more than two duplicated genes in a subregion and the frequency of gene rearrangement that contains breakpoints between neighboring gene pairs, we compute the Pearson correlation coefficient between them (See Table X). While there were significant positive correlations among subregions by intersection and union, there were no significant correlations within gene clusters with a value of 0.1. Since there can be more than two operons in a gene cluster, the relationship between gene rearrangement and gene duplication is decreased.

6. Comparative analysis on operon occurrences

We study occurrences of the trp operon in 700 bacterial genomes. The trp operon is present in many bacteria and promotes the production of a chemical (tryptophan) when tryptophan, which is one of the standard amino acids and an essential amino acid in the human diet, is not present in the environment (Morse *et al.*, 1969). It consists of *trpE, trpD, trpC, trpB* and *trpA* that encodes tryptophan synthetase. Since the trp operon is an important experiment system in gene regulation, we investigate the occurrences of trp operon in multiple genomes in detail.

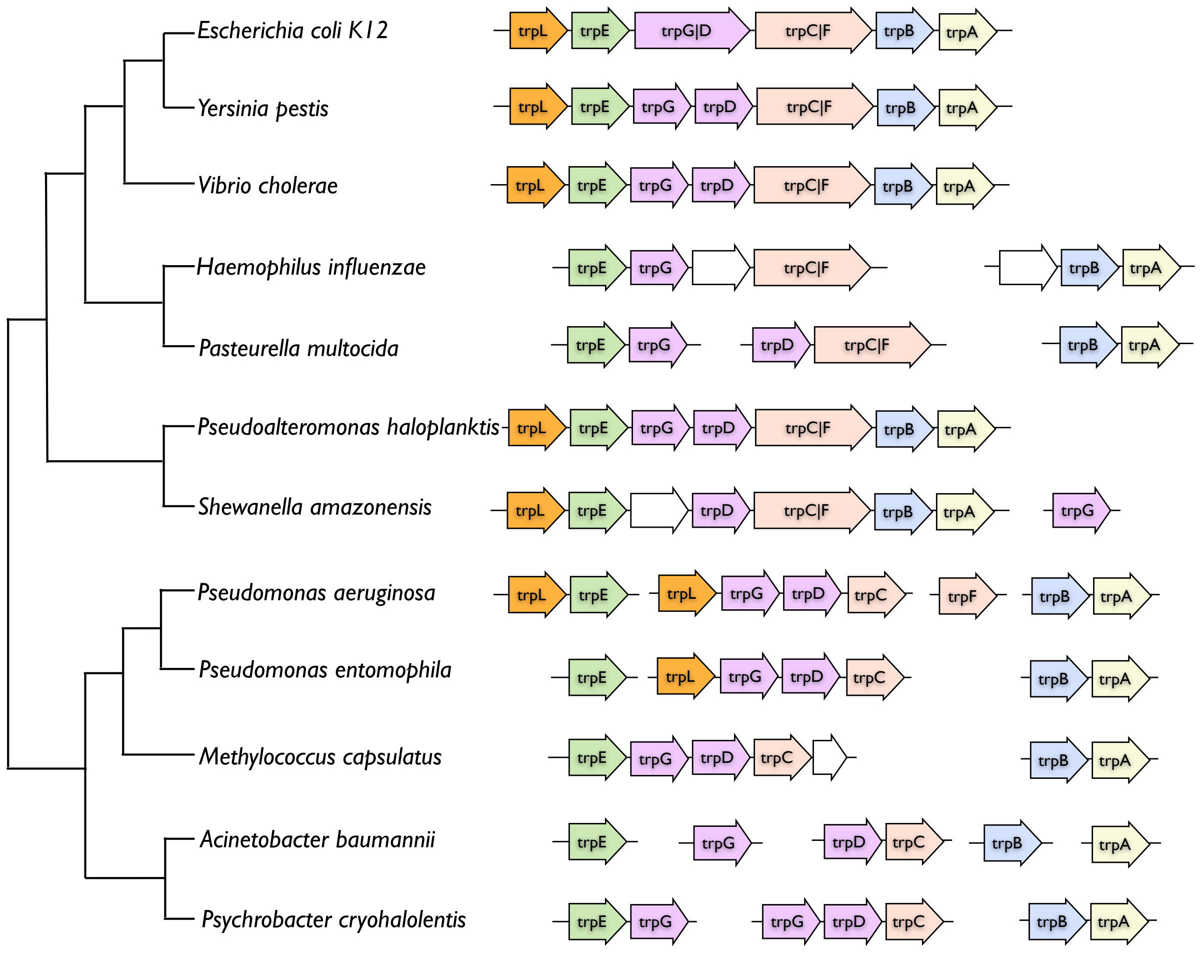


Fig. 30. Illustration of trp operon in Gammaproteobacteria bacteria group.

Merino *et al.* (2008) showed the phylogenetic tree of trp gene organization and associated regulatory factors, elements and mechanisms. The trp operon in Merino *et al.* (2008) appears in 16 bacterial groups that are covered by 700 bacterial genomes that we use as input genomes. While seven trp biosynthetic genes are organized within a single transcriptional unit that includes a *trpCF* fusion and a *trpGD* fusion as well, we only consider *trpE, trpD, trpC, trpB* and *trpA* that encode tryptophan synthetase in the investigation of operon occurrences in gene clusters.

The total number of gene clusters that include the trp operon in all genomes within a gene cluster is 55202 clusters. Within those 55202 gene clusters, we consider 16 bacteria groups that appear in the distribution of trp gene organization. The 16 bacteria groups include *Gammaproteobacteria, Betaproteobacteria, Alphaproteobacteria, Spirochaetales, Epsilonproteobacteria, Bacillasles, Lactobacillales, Clostridia, Actinobacteria, Chroococcales, Thermotogales, Bacteroidetes, Chiamydiales* and *Deinococci*.

In the *Gammaproteobacteria* bacteria group which is the largest group (23.86%) among the 700 bacteria genomes, two gene clusters that only consist of trp operons are found with 14 genomes that perfectly match with genomes identified by Merino *et al.* (2008). One of the two gene clusters that has the lowest *e*-value is the most significant gene cluster out of 55202 gene clusters (See Figure 30).

In the *Alphaproteobacteria* bacteria group, all genomes that appeared in two gene clusters that are the same as the cluster that appeared in the *Gammaproteobacteria* bacteria group, supporting the theory that *Alphaproteobacteria* is evolutionarily related with *Gammaproteobacteria*, and it validates the transcription attenuation

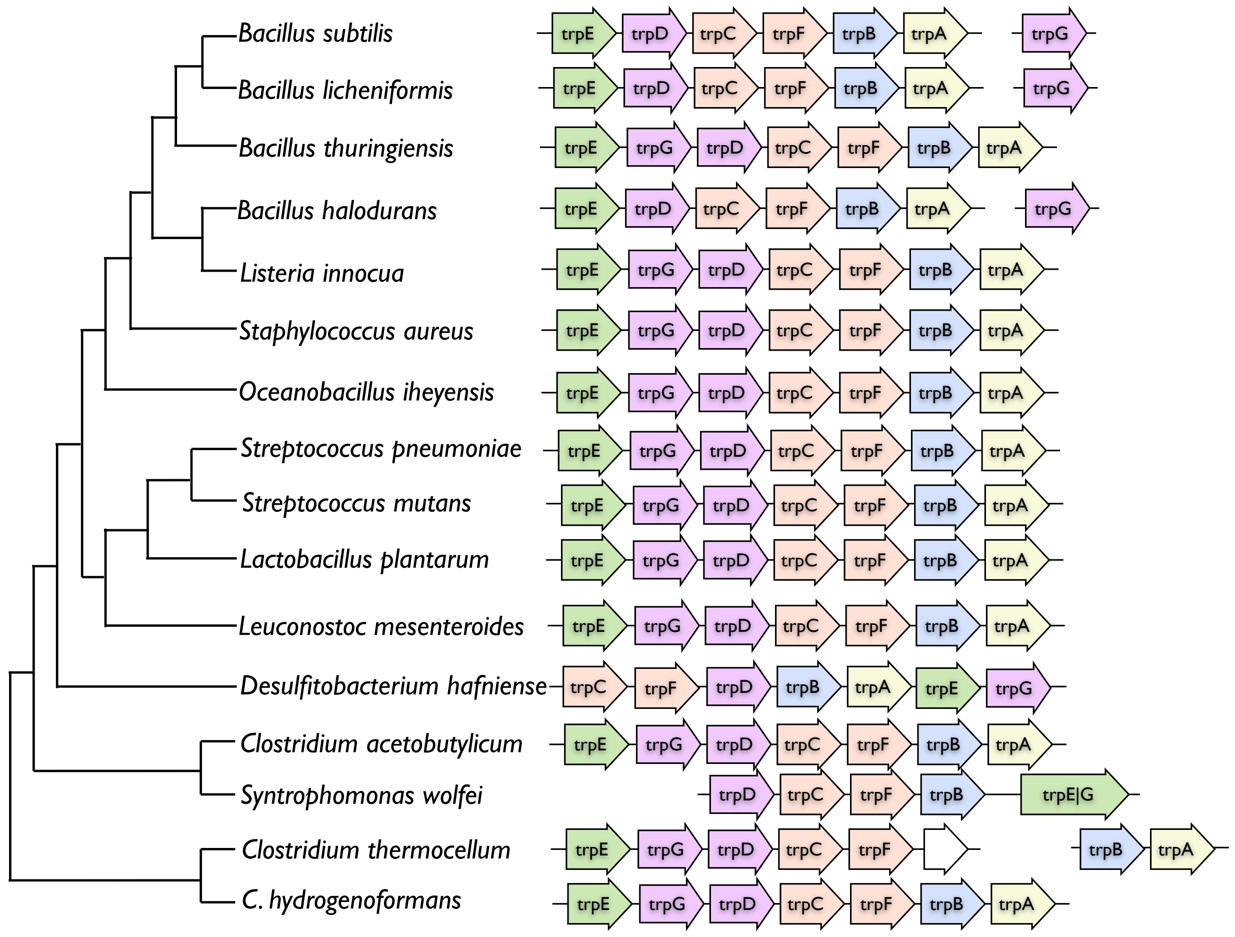

Bacillus subtilis
trpE trpD trpC trpF trpB trpA trpG
Bacillus licheniformis
trpE trpD trpC trpF trpB trpA trpG
Bacillus thuringiensis
trpE trpG trpD trpC trpF trpB trpA
Bacillus halodurans
trpE trpD trpC trpF trpB trpA trpG
Listeria innocua
trpE trpG trpD trpC trpF trpB trpA
Staphylococcus aureus
trpE trpG trpD trpC trpF trpB trpA
Oceanobacillus iheyensis
trpE trpG trpD trpC trpF trpB trpA
Streptococcus pneumoniae
trpE trpG trpD trpC trpF trpB trpA
Streptococcus mutans
trpE trpG trpD trpC trpF trpB trpA
Lactobacillus plantarum
trpE trpG trpD trpC trpF trpB trpA
Leuconostoc mesenteroides
trpE trpG trpD trpC trpF trpB trpA
Desulfitobacterium hafniense
trpC trpF trpD trpB trpA trpE trpG
Clostridium acetobutylicum
trpE trpG trpD trpC trpF trpB trpA
Syntrophomonas wolfei
trpD trpC trpF trpB trpE|G
Clostridium thermocellum
trpE trpG trpD trpC trpF trpB trpA
C. hydrogenoformans
trpE trpG trpD trpC trpF trpB trpA

Fig. 31. Illustration of trp operon in Firmicutes bacteria group.

mechanism used to regulate the *trpE* in both *Gammaproteobacteria* (e.g *Pseudomonas aeruginosa*) and *Alphaproteobacteria* (e.g *Rhizobium etli*) (Merino *et al.*, 2008).

We separately evaluate *Bacillasles, Lactobacillales* and *Clostridia* that are within the *Firmicutes* phylum, and also perform all genomes in those three bacteria groups that are within *Firmicutes. Bacillasles, Lactobacillales* and *Clostridia* consist of 7, 4 and 5 genomes, respectively. *Bacillales* and *Lactobacillales* are an order of Gram-positive bacteria in which *Bacillales* includes *Bacillus, Listeria* and *Staphylococcus*, and *Lactobacillales* includes *Streptococcus, Lactobacillus* and *Leuconostoc. Clostridia* are in the group of *Firmicutes* that includes *Clostridium*, but is different from *Bacillales* by the lack of aerobic respiration. 20 gene clusters in *Bacillales* and *Lactobacillales* perfectly match with the graph. Those of 20 gene clusters are the same in both bacteria groups, since they are in the same order of Gram-positive bacteria within *Firmicutes.* In *Clostridium* group, there were only 4 gene clusters that perfectly matched with 5 genomes, since the gene proximity from *trpB* and *trpA* to other genes in *Syntrophomonas wolfei* is too far to be within a gene cluster. We found 4 gene clusters that are covered by all genomes in *Firmicutes*, since *Clostridium* consists of the smallest number of gene clusters that are perfectly matched (See Figure 31).

In the *Betaproteobacteria* bacteria group, there were no gene clusters that include trp operon in all genomes in the group, while *Bordetella parapertussis, Neisseria meningitidis* and *Acidovorax sp.* were found separately in 252, 1426 and 301 gene clusters, respectively. There are no gene clusters that have trp operon between *Bordetella parapertussis* and *Neisseria meningitidis*, but 10 and 4 gene clusters were found in *Bordetella parapertussis* & *Acidovorax sp.* and *Neisseria meningitidis* & *Aci-*

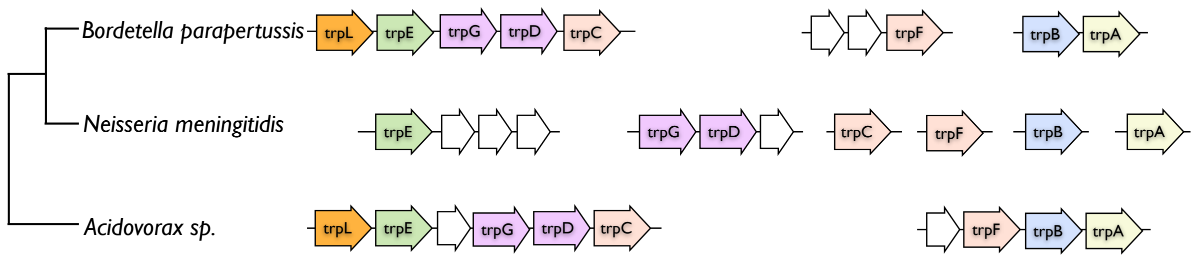


Fig. 32. Illustration of trp operon in Betaproteobacteria bacteria group.

dovorax sp., since all genes of the trp operon in *Neisseria meningitidis* are sparsely located in different regions. This makes it difficult to find clusters with other genomes that include all trp genes within localized regions (See Figure 32).

The rest of the bacteria groups that include *Spirochaetales, Epsilonproteobacteria, Chroococcales, Thermotogales, Bacteroidetes, Chiamydiales* and *Deinococci* are composed of at least two genomes in a group: thus, the perfect matching rate was significantly high. The result so far shows that although very few trp operons are shared by 700 bacterial genomes, most trp operons can be found in many bacteria groups defined by Merino *et al.* (2008). The proposed algorithm allows the evaluation of various hypotheses regarding evolution and conservation of gene clusters.

D. Discussion

We have developed a gene clustering algorithm that allows the analysis of gene clusters across a large number of genomes and important biological insights to be obtained from this analysis. We validated the results of gene clusters that include *Escherichia coli K12* operons, which are experimentally confirmed by RegulonDB. The gene clustering result reveals a significant amounts of spatial conservation that is at a higher level than operon and some of these clusters are likely to correspond to uber-operon. The study for the distribution of gene order within a subregion showed the spatial arrangement of genes within bacterial subregion that is important for function, expression and regulation. Although gene order within operons can be unstable (Itoh *et al.*, 1999), our results implied that the gene order tends to be conserved and gene orientation appears to face the same direction in the subregion. Because our algorithm does not impose constraints on gene orientations, we can conclude that there is a strong force to preserve gene orientations in bacterial clusters, which is a prerequisite for functioning as operons. To validate subregions that are predicted from clusters, we investigated a relationship between gene duplication and rearrangement. As a result, a strong positive correlation was predictable between gene duplication and rearrangement considering gene rearrangements include a gene duplication. According to our comparative analysis, it was confirmed that trp operon was shared among bacterial genomes within the same group and the ortholog of trp operon was found as well (Merino *et al.*, 2008).

The first algorithm allows paralogous genes and clusters that appear in input chromosomes, and estimates the statistical significance of each gene cluster. The second algorithm placed a strict limit on the maximum cluster size to allow analyzing of hundreds of genomes simultaneously with the modified version of the window-based strategy. We have a maximum size of gene clusters to avoid the combinatorial explosion of intersecting all combinations, since the overall maximum size of operons is not large. There are only three *Escherichia coli K12* operons in RegulonDB which are larger than other operons that include at least 13 genes. The window of small size is efficient enough to identify gene clusters across hundreds of genomes simultaneously in terms of the speed and the performance.

The proposed algorithm is able to identify gene clusters that are not found by other algorithms, since we developed a different formulation. One of the advantages in the proposed algorithm is that it allows paralogous genes in a same chromosome and orthologous genes in different genomes so that it considers the more biologically accurate model, and the statistical significance estimate while allowing gene cluster that may not appear in every genome is important for biologists in identifying important gene clusters. We used BLAST and the method in Kellis *et al.* (2003) to construct homologous gene groups, but it is also possible to use existing database on homology relationships such as COG (Tatusov *et al.*, 1997) and other ways that establish orthologous and paralogous correspondences (Calabrese *et al.*, 2003; Fujibuchi *et al.*, 2000; Luc *et al.*, 2003; Remm *et al.*, 2001; Vandepoele *et al.*, 2002).

CHAPTER V

CONCLUSION

A. Summary

One of the major technological advances in biology in the last few years is the development of sequencing technology that produces gigabases of data in a single run. Thus, automated prediction of gene function has been an important issue for biologists, since the development of high-throughput genome sequencing has produced large amounts of sequence data, rendering biologists unable to deduce their gene functions through experimentation. In addition, automated prediction of gene functions and gene annotation methods in computational bioinformatics have created many challenging computational problems such as functionally related gene clusters and their functional annotations in large-scale genome data. Thus, I have developed and improved algorithms and applications to identify clusters of functionally related genes across multiple genomes that can help us understand how gene clusters are functioning. In this dissertation, I have developed four algorithms for predicting groups of functionally related genes.

For the classification of protein family, I developed two algorithms that classify proteins to existing families and construct new families from unclassified proteins. By utilizing sequence similarity from pairwise comparisons using either BLAST or the Smith-Waterman algorithm SSEARCH, I developed a supervised classification algorithm based on an iterative merging strategy. The major difference of the proposed approach from previous supervised techniques is that the simplicity of the algorithm results in a low time complexity, and the new approach automatically constructs new families if necessary.

The first algorithm enables us to determine whether the unclassified protein belongs to one of the existing families. Since sequences within the same family can have low similarity, I only considered proteins within a family that are of sufficiently high similarity to the unclassified protein during the computation of scores. This avoids the problem of getting consistently low average scores, and enables us to assign any new proteins to existing families. In the second algorithm, I considered the problem when a set of more than one unclassified protein is given. This method is not only able to assign each unclassified protein to an existing family, but also able to automatically construct new families if necessary.

I applied the algorithm to a few large-scale data sets that contain sequences corresponding to protein domains, and data sets that contain full length sequences. I showed that very good performance can be obtained by choosing appropriate e-value cutoffs in these two cases. I compared the performance of the proposed algorithm to a few supervised algorithms, and showed that the proposed algorithm has significantly higher accuracy rate and lower mis-classification rate than other algorithms. Through evaluations of the proposed algorithm, I have also shown that for the purpose of protein family classification, it may not be necessary to consider more complicated models such as multiple sequence alignments and hidden markov models.

In the gene clustering across multiple genomes, I defined two algorithms that can investigate related gene clusters and study operons and their evolutions. The first algorithm uses a different formulation based on constraining the overall size of a cluster and statistical estimations that allow direct comparisons among clusters of different sizes. I compared the performance of the proposed algorithm to a few gene clustering algorithms, and showed that the functional enrichment of each cluster from the proposed algorithm outperforms clusters obtained from GeneTeams which is the most popular algorithm that requires the different gene distance parameter to restrict number of intervening genes between adjacent genes in a cluster. I used GO term finder to evaluate the functional enrichment of gene clusters. In the second algorithm, I developed a modified version of the window-based strategy to analyze hundreds of genomes simultaneously by placing a stricter limit on the maximum cluster size. Due to the different formulation, the proposed algorithm can identify gene clusters that are not found by other gene clustering algorithms, and it also provides statistical significance estimates that may be important for biologists to identify the most important clusters. I demonstrated proposed algorithms that will be useful for biological insight by analyzing gene clusters across a large number of genomes that can help us understand operon occurrences, gene orientations and distributions of gene rearrangements.

I constructed gene maps on different chromosomes with gene location information defined by NCBI, and homologous gene group by finding bidirectional genes with best hits using protein–protein BLAST. In addition, to reduce the computational time for identifying homologous gene groups using BLAST, it is also possible to use existing database such as COG (Tatusov *et al.*, 1997) or INPARANOID (Remm *et al.*, 2001), to find groups of genes that are homologous (Fujibuchi *et al.*, 2000; Vandepoele *et al.*, 2002).

B. Future work

While the proposed algorithm for classifying proteins into families and most existing classification algorithms are based on sequence similarity information from alignments, alignment-free approaches are also available (Ma and Chan, 2008; Vinga and Almeida, 2003; Pham and Zuegg, 2004; Kantorovitz *et al.*, 2007). The alignment-free approach may be applied to improve the similarity in which two sequences to be compared are not orthologous but are functionally related or they are highly diverged evolutionarily. This new sequence similarity score that can be used for detecting functional and evolutionary similarities may be helpful in increasing the accuracy of the proposed method. One future direction is to investigate whether it is possible to use these techniques to improve accuracy with this new sequence similarity information. Another future direction is to incorporate more features into the classification algorithm such as bio-chemical properties, protein structure information, etc., so that the accurate classification can be increased and the mis-classification can be decreased. These features that help in identifying evolutionary similarities can be essential values to improving the performance of the proposed algorithm in the future direction.

I developed a modified version of the window-based strategy to analyze hundreds of genomes simultaneously by placing a stricter limit on the maximum cluster size. This makes it possible to avoid the combinatorial explosion of intersecting all combinations of windows at one time with one from each chromosome, and allow the use of a different strategy to find the gene clusters. However, the most computationally heavy part in the algorithm is the combination of genes within windows when it becomes larger and larger, because all possible combinations of genes and intervening genes are included within each window, while we do not allow duplicated gene numbers in the combination of genes. One future direction is to investigate a novel approach that can overcome computations of combinations and will not lose any clusters that satisfy the definition.

It is well known that genes in bacterial genomes are usually not distributed randomly in the genome, and genes are organized into groups of operons. Unlike genes in bacterial genomes, genes in eukaryotic genomes are traditionally thought of as being randomly distributed among the chromosomes (Yi *et al.*, 2007). Another possible future direction for gene clustering algorithm is to investigate a new possibility that can identify larger clusters in higher organisms by combining overlapping seed clusters that are located in sparse regions. Seed clusters may be utilized to identify clusters of larger size that commonly occure in higher organisms.

REFERENCES

A. Bergeron,S.C. and Raffinot,R. (2002) The algorithmic of gene teams. *Lecture Notes in Computer Science,* **2452**, 464–476.

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.,* **215** (3), 403–410.

Apweiler,R., Attwood,T., Bairoch,A., Bateman,A., Birney,E., Biswas,M., Bucher,P., Cerutti,L., Corpet,F., Croning,M. *et al.* (2001) The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Res.,* **29** (1), 37.

Ashburner,M., Ball,C.A., Blake,J.A., Botstein,D., Butler,H., Cherry,J.M., Davis,A.P., Dolinski,K., Dwight,S.S., Eppig,J.T., Harris,M.A., Hill,D.P., Issel-Tarver,L., Kasarskis,A., Lewis,S., Matese,J.C., Richardson,J.E., Ringwald,M., Rubin,G.M. and Sherlock,G. (2000) Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat. Genet.,* **25** (1), 25–9.

Attwood,T.K., Blythe,M.J., Flower,D.R., Gaulton,A., Mabey,J.E., Maudling,N., McGregor,L., Mitchell,A.L., Moulton,G., Paine,K. and Scordis,P. (2002) Prints and prints-s shed light on protein ancestry. *Nucleic Acids Res.,* **30** (1), 239–41.

Bateman,A., Birney,E., Durbin,R., Eddy,S.R., Howe,K.L. and Sonnhammer,E.L. (2000) The pfam protein families database. *Nucleic Acids Res.,* **28** (1), 263–6.

Bejerano,G. and Yona,G. (2001) Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics,* **17** (1), 23–43.

Boyle,E.I., Weng,S., Gollub,J., Jin,H., Botstein,D., Cherry,J.M. and Sherlock,G. (2004) Go::termfinder–open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics,* **20** (18), 3710–5.

Brenner,S.E. (1999) Errors in genome annotation. *Trends Genet.,* **15** (4), 132–3.

Calabrese,P.P., Chakravarty,S. and Vision,T.J. (2003) Fast identification and statistical evaluation of segmental homologies in comparative maps. *Bioinformatics,* **19**, 74–80.

Che,D., Li,G., Mao,F., Wu,H. and Xu,Y. (2006) Detecting uber-operons in prokaryotic genomes. *Nucleic Acids Res.,* **34** (8), 2418–27.

Chen,Y., Reilly,K.D., Sprague,A.P. and Guan,Z. (2006) Seqoptics: a protein sequence clustering system. *BMC Bioinformatics,* **7 Suppl 4**, S10.

Didier,G. (2003) Common intervals of two sequences. *Lecture Notes in Computer Science,* **2812**, 17–24.

Dondoshansky,I. (2002) *Blastclust (NCBI Software Development Toolkit).* NCBI, Bethesda, Md.

Doolittle,R.F. (1995) The multiplicity of domains in proteins. *Annu. Rev. Biochem.,* **64**, 287–314.

Durand,D. and Sankoff,D. (2003) Tests for gene clustering. *J. Comput. Biol.,* **10** (3-4), 453–82.

Eddy,S. (1998) Profile hidden Markov models. *Bioinformatics,* **14** (9), 755.

Enright,A.J., Dongen,S.V. and Ouzounis,C.A. (2002) An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.,* **30** (7), 1575–84.

Ermolaeva,M., White,O. and Salzberg,S. (2001) Prediction of operons in microbial genomes. *Nucleic Acids Res.,* **29** (5), 1216.

Eskin,E., Noble,W.S. and Singer,Y. (2003) Protein family classification using sparse markov transducers. *J. Comput. Biol.,* **10** (2), 187–213.

Fan,R., Chen,P. and Lin,C. (2005) Working set selection using second order information for training support vector machines. *J. Machine Learning Res.,* **6**, 1889–1918.

Fraser,A.G. and Marcotte,E.M. (2004) A probabilistic view of gene function. *Nat. Genet.,* **36** (6), 559–64.

Fujibuchi,W., Ogata,H., Matsuda,H. and Kanehisa,M. (2000) Automatic detection of conserved gene clusters in multiple genomes by graph comparison and P-quasi grouping. *Nucleic Acids Res.,* **28** (20), 4029.

Gribskov,M., Lüthy,R. and Eisenberg,D. (1990) Profile analysis. *Meth Enzymol.,* **183**, 146–59.

Halkidi,M., Batistakis,Y. and Vazirgiannis,M. (2001) On clustering validation techniques. *J. Intell. Inform. Syst.,* **17** (2), 107–145.

He,X. and Goldwasser,M.H. (2005) Identifying conserved gene clusters in the presence of homology families. *J. Comput. Biol.,* **12** (6), 638–56.

Heber,S. and Stoye,J. (2001) Finding all common intervals of k permutations. *Lecture Notes in Computer Science,* **2089**, 207–218.

Heger,A. and Holm,L. (2003) Exhaustive enumeration of protein domain families. *J. Mol. Biol.,* **328** (3), 749–767.

Henikoff,S., Greene,E.A., Pietrokovski,S., Bork,P., Attwood,T.K. and Hood,L. (1997) Gene families: the taxonomy of protein paralogs and chimeras. *Science,* **278** (5338), 609–14.

Hunter,C.N., Daldal,F. and Thurnauer,M.C. (2008) *The Purple Phototrophic Bacteria.* Springer-Verlag, The Netherlands.

Itoh,T., Takemoto,K., Mori,H. and Gojobori,T. (1999) Evolutionary instability of operon structures disclosed by sequence comparisons of complete microbial genomes. *Mol. Biol. Evol.,* **16** (3), 332–46.

Jaakkola,T., Diekhans,M. and Haussler,D. (2000) A discriminative framework for detecting remote protein homologies. *J. Comput. Biol.,* **7** (1-2), 95–114.

Kanehisa,M. and Goto,S. (2000) Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.,* **28** (1), 27–30.

Kantorovitz,M., Robinson,G. and Sinha,S. (2007) A statistical method for alignment-free comparison of regulatory sequences. *Bioinformatics,* **23** (13), i249.

Kececioglu,J. and Sankoff,D. (1995) Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica,* **13**, 180–210.

10.1007/BF01188586.

Kellis,M., Patterson,N., Endrizzi,M., Birren,B. and Lander,E.S. (2003) Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature,* **423** (6937), 241–54.

Kim,S., Choi,J.H. and Yang,J. (2005) Gene teams with relaxed proximity constraint. *Proc. IEEE Comput. Syst. Bioinformatics Conf.,* **5**, 44–55.

Kim,S. and Lee,J. (2006) Bag: a graph theoretic sequence clustering algorithm. *Int. J. Data Mining and Bioinformatics,* **1** (2), 178–200.

Klimke,W., Agarwala,R., Badretdin,A., Chetvernin,S., Ciufo,S., Fedorov,B., Kiryutin,B., O'Neill,K., Resch,W., Resenchuk,S., Schafer,S., Tolstoy,I. and Tatusova,T. (2009) The national center for biotechnology information's protein clusters database. *Nucleic Acids Res.,* **37** (Database issue), D216–23.

Krogh,A., Brown,M., Mian,I.S., Sjölander,K. and Haussler,D. (1994) Hidden markov models in computational biology. applications to protein modeling. *J. Mol. Biol.,* **235** (5), 1501–31.

Lathe,W.C., Snel,B. and Bork,P. (2000) Gene context conservation of a higher order than operons. *Trends. Biochem. Sci.,* **25** (10), 474–9.

Lee,J.M. and Sonnhammer,E.L.L. (2003) Genomic gene clustering analysis of pathways in eukaryotes. *Genome Res.,* **13** (5), 875–82.

Liao,L. and Noble,W.S. (2003) Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *J. Comput. Biol.,* **10** (6), 857–68.

Luc,N., Risler,J.L., Bergeron,A. and Raffinot,M. (2003) Gene teams: a new formalization of gene clusters for comparative genomics. *Comput. Biol. Chem.,* **27** (1), 59–67.

Ma,P.C.H. and Chan,K.C.C. (2008) Upsec: an algorithm for classifying unaligned protein sequences into functional families. *J. Comput. Biol.,* **15** (4), 431–43.

Merino,E., Jensen,R. and Yanofsky,C. (2008) Evolution of bacterial trp operons and their regulation. *Curr. Opin. Microbiol.,* **11** (2), 78–86.

Morse,D., Mosteller,R. and Yanofsky,C. (1969) Dynamics of synthesis, translation, and degradation of trp operon messenger RNA in E. coli. *Cold Spring Harb. Symp. Quant. Biol.,* **34**, 725–40.

Mulder,N.J., Apweiler,R., Attwood,T.K., Bairoch,A., Barrell,D., Bateman,A., Binns,D., Biswas,M., Bradley,P., Bork,P., Bucher,P., Copley,R.R., Courcelle,E., Das,U., Durbin,R., Falquet,L., Fleischmann,W., Griffiths-Jones,S., Haft,D., Harte,N., Hulo,N., Kahn,D., Kanapin,A., Krestyaninova,M., Lopez,R., Letunic,I., Lonsdale,D., Silventoinen,V., Orchard,S.E., Pagni,M., Peyruc,D., Ponting,C.P., Selengut,J.D., Servant,F., Sigrist,C.J.A., Vaughan,R. and Zdobnov,E.M. (2003) The interpro database, 2003 brings increased coverage and new features. *Nucleic Acids Res.,* **31** (1), 315–318.

Murzin,A.G., Brenner,S.E., Hubbard,T. and Chothia,C. (1995) Scop: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.,* **247**, 536–540.

Parida,L. (2007) Gapped permutation pattern discovery for gene order comparisons. *J. Comput. Biol.,* **14** (1), 45–55.

Pasek,S., Bergeron,A., Risler,J., Louis,A., Ollivier,E. and Raffinot,M. (2005) Identification of genomic features using microsyntenies of domains: domain teams. *Genome Res.,* **15** (6), 867.

Pertea,M., Ayanbule,K., Smedinghoff,M. and Salzberg,S.L. (2009) Operondb: a comprehensive database of predicted operons in microbial genomes. *Nucleic Acids Res.,* **37** (Database issue), D479–82.

Pham,T. and Zuegg,J. (2004) A probabilistic measure for alignment-free sequence comparison. *Bioinformatics,* **20** (18), 3455.

Price,M., Huang,K., Alm,E. and Arkin,A. (2005) A novel method for accurate operon predictions in all sequenced prokaryotes. *Nucleic Acids Res.,* **33** (3), 880.

Raghupathy,N. and Durand,D. (2009) Gene cluster statistics with gene families. *Mol. Biol. Evol.,* **26** (5), 957–68.

Remm,M., Storm,C. and Sonnhammer,E. (2001) Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J. Mol. Biol.,* **314** (5), 1041–1052.

Rose Hoberman,D.S. and Durand,D. (2005) The statistical significance of max-gap clusters. *Lecture Notes in Computer Science,* **3388** (1), 55–71.

Salgado,H., Moreno-Hagelsieb,G., Smith,T. and Collado-Vides,J. (2000) Operons in Escherichia coli: genomic analyses and predictions. *Proc. Natl. Acad. Sci.,* **97** (12), 6652.

Salgado,H., Santos,A., Garza-Ramos,U., van Helden,J., Díaz,E. and Collado-Vides,J. (1999) Regulondb (version 2.0): a database on transcriptional regulation in escherichia coli. *Nucleic Acids Res.,* **27** (1), 59–60.

Salse,J., Bolot,S., Throude,M., Jouffe,V., Piegu,B., Quraishi,U.M., Calcagno,T., Cooke,R., Delseny,M. and Feuillet,C. (2008) Identification and characterization of shared duplications between rice and wheat provide new insight into grass genome evolution. *Plant Cell,* **20** (1), 11–24.

Schmidt,T. and Stoye,J. (2004) Quadratic time algorithms for finding common intervals in two and more sequences. *Lecture Notes in Computer Science,* **3109**, 347–358.

Smith,S. (2004) Protein family classification using structural and sequence information. *Comput. Intell. Bioinformatics and Comput. Biol.,* **1**, 168 – 174.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.,* **147** (1), 195–7.

Smith,T.F. and Zhang,X. (1997) The challenges of genome sequence annotation or "the devil is in the details". *Nat. Biotechnol.,* **15** (12), 1222–3.

Tamames,J. (2001) Evolution of gene order conservation in prokaryotes. *Genome Biol.,* **2** (6).

Tatusov,R., Koonin,E. and Lipman,D. (1997) A genomic perspective on protein families. *Science,* **278** (5338), 631.

Thompson,J., Higgins,D. and Gibson,T. (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** (22), 4673.

Vandepoele,K., Saeys,Y., Simillion,C., Raes,J. and Van de Peer,Y. (2002) The automatic detection of homologous regions (ADHoRe) and its application to microcolinearity between Arabidopsis and rice. *Genome Res.,* **12** (11), 1792.

Vinga,S. and Almeida,J. (2003) Alignment-free sequence comparison–review. *Bioinformatics,* **19** (4), 513.

Vogel,C. and Chothia,C. (2006) Protein family expansions and biological complexity. *PLoS Comput. Biol.,* **2** (5), e48.

Wu,C.H., Huang,H., Yeh,L.S.L. and Barker,W.C. (2003) Protein family classification and functional annotation. *Comput. Biol. Chem.,* **27** (1), 37–47.

Yang,Q. and Sze,S. (2008) Large-scale analysis of gene clustering in bacteria. *Genome Res.,* **18** (6), 949.

Yi,G., Sze,S.H. and Thon,M.R. (2007) Identifying clusters of functionally related genes in genomes. *Bioinformatics,* **23** (9), 1053–60.

VITA

Gang Man Yi received his B.S. in Computer Science from Kangnung National University, Korea in 2002. He received the M.S. degree in Computer Science at Texas A&M University in 2006. He was a Ph.D. student in the Department of Computer Science at Texas A&M University since January 2007, and he received his Ph.D. in May 2011. His research interests are computational biology and bioinformatics.

He can be reached at Department of Computer Science, Texas A&M University TAMU 3112 College Station, TX 77843. His email is gangmanyi@tamu.edu.