CONSERVED GENE CLUSTER DISCOVERY AND APPLICATIONS IN COMPARATIVE GENOMICS

MELVIN ZHANG
(B. Comp (Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2011

Acknowledgement

I would like to take this opportunity to express my gratitude to my advisor, Associate Professor Hon Wai Leong. Hon Wai not only gave me valuable advice on research directions and methodology, he also exposed me to the other facets of academia, such as teaching, peer review, and networking. In particular, I'm very grateful for the opportunity to visit and work with researchers from the CAS-MPG Partner Institute of Computation Biology (PICB) in Shanghai.

I am also grateful to Dr Guillaume Bourque, Professor Lim Soon Wong, and Associate Professor Ken Sung. Guillaume and Hon Wai jointly proposed a project on genome rearrangements which became my final year project. Working on this project sparked my interest in research and lead me to purse graduate studies at NUS. During my candidature, my thesis advisory committee members, Lim Soon and Ken, provided invaluable feedback on how to improve the strength and impact of my research.

In the course of my candidature, I had the wonderful opportunity to work with a number of students and researchers. I would like to thank my collaborators: Dr Xingguang Zhu, Dr Axel Mosig, Zhu Liang, Xiao Hang, Fan Chang, Cao Fan, Trong Dao Le, and Zhou Zhong.

Lastly, I would like to thank my family, friends, and members of NUS RAS Group (Ket Fah Chong, Francis Ng, Ning Kang, Max Tan, and Sriganesh) for their continual encouragement and support.

Contents

Summary

We share the vast majority of our genes with the great apes, our closest living relative. However, how the genes are arranged is quite different. We have 23 pairs of chromosomes, whereas other great apes have 24 pairs; our chromosome 2 was formed by the fusion of two ancestral chromosomes. We have at least nine chromosomal regions that are inverted in chimpanzees. Fusions, inversions and other rearrangements result in a "shuffling" of the genes. *Conserved gene clusters* are sets of genes that can be found near one another in several species despite these rearrangements. They may result from functional pressure to keep these genes close together or a lack of rearrangements. In either case, conserved gene clusters provide information for inferring gene function and better understanding of genome evolution.

In the first part of this thesis, we propose new gene cluster models that make use of biological constraints or structural properties to reduce the number of parameters. We then develop efficient algorithms to identify gene clusters based on our models. The second part of this thesis, studies the conservation of individual genes, also known as the Ortholog Assignment problem. For this problem, many sophisticated methods have been proposed. Our contribution is a simple yet effective method that integrates sequence and gene context similarity in a single framework.

Max-gap clusters (aka gene teams) is a popular model of conserved gene clusters. This model uses a max-gap parameter δ to restrict the maximum distance

between adjacent genes in a cluster. In practice, determining an ideal value of δ is a matter of trial and error. We proposed the *Gene Team Tree* (GTT) structure as a compact representation of gene teams for all possible values of δ. Surprisingly, we were able to extend algorithms for finding gene teams, based on a specific value of δ, to compute the GTT without increasing the time/space complexity. We applied our model to compute the GTT for *E. coli* K-12 and *B. subtilis* and confirmed that known *E. coli* K-12 operons corresponds to gene teams with different values of δ

Max-length clusters (aka r-window clusters) is a different gene cluster model where a cluster has length at most r and contains at least k genes. The bidirectional best hit (BBH) heuristic is widely used in sequence analysis to identify putative homologous genes. As conserved gene clusters are a generalization of homologous genes, we proposed to use the BBH heuristic to identify conserved r-window clusters. We name this new model *bidirectional best hit r-window model* (BBHRW) and designed a sub-quadratic time algorithm to find all clusters. We investigated how well the gene clusters modelled by the two models corresponds to known *E. coli* K-12 operons. We found that the two model are complementary; the gene team model has more clusters that corresponds to operons, while the BBHRW model has fewer clusters that do not correspond to any operon.

We also studied the problem of identifying individual conserved genes, the so called Ortholog Assignment problem. Several sophisticated methods exists for this problem. Our contribution is a simple yet effective method (BBH-LS) to identify positional homologs. BBH-LS applies the bidirectional best hit heuristic to a combination of sequence similarity and gene context similarity scores. We applied BBH-LS to the human, mouse, and rat genomes and found that the best results are obtained when using both sequence and gene context information equally. In our comparisons, BBH-LS reported the largest number of true positives and a medium number of false positives.

List of Tables

List of Figures

Chapter 1

Introduction

The genome of an organism is the combined hereditary information that is found in every cell of the organism. To a large extent, this information represents the "nature" in the classical nature versus nurture debate. In our case, our genome is stored in 23 pairs of chromosomes. Each chromosome is a long chain composed of four different types of deoxyribonucleic acid (DNA) molecules, giving us (and most other species) a four letter genetic code. One of the early successes of Computational Biology (the application of computational techniques for solving biological problems) is the early completion of the Human Genome Project [Collins et al., 1998]. The initial goal of the project is to determine every letter of our genome. This is also commonly known as the sequencing of the human genome. Sequencing machines can only sequence short fragments reliably, scaling them up to handle the over three billion letters in our genome was an insurmountable task. Sophisticated algorithms that are able to computationally assemble small fragments of overlapping DNA sequences enabled bold new strategies based on randomly breaking many copies of the genome into overlapping short fragments and using existing machines to sequence these short fragments in parallel [Venter et al., 1998].

Since then, enhancements to the computational algorithms for assembly and improvements to the underlying sequencing hardware has enabled us to sequence

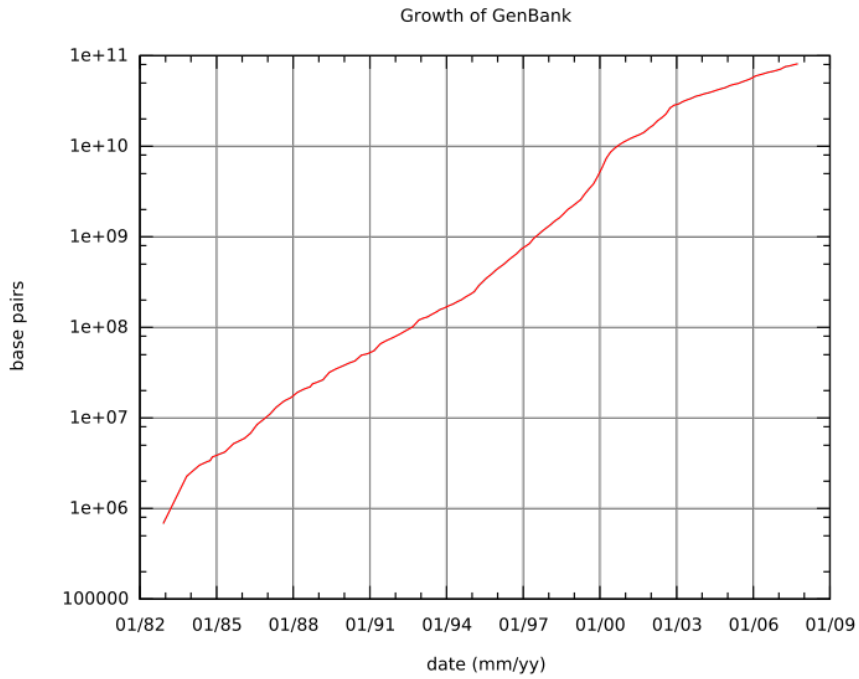

Fig. 1.1: Number of base pairs stored in NCBI's GenBank database as a function of time. Created by user 121a0012 on Wikipedia and released into public domain.

more and more species. The rate at which new sequences are being produced follows an exponential growth reminiscent of Moore's Law (see Figure 1.1). As more and more complete genomes have been sequenced, the emphasis in Computational Biology is shifting toward understanding and interpreting the information encoded in these genomes.

Traditional wet lab techniques cannot keep up with the deluge of genome sequences. A promising approach to gain some initial understanding of a newly sequenced genome is to compare it with well studied genomes such as the human genome. This comparative approach to genomics exemplifies the principle behind the field of *Comparative Genomics*. Such a strategy requires us to be able to identify identify conserved elements across species boundaries [Koonin, 2005]. In this thesis, we consider two classes of conserved elements: individual genes and sets of genes.

1.1 Motivation

A gene is a segment of our genome that gets translated into proteins. Proteins are long polymers that can fold into intricate three dimensional structures to act as nano machines. Thus, we can think of genes as the blueprint for making these biological nano machines.

A central problem in Comparative Genomics is to identify homologous genes. These are genes that are descended from the single ancestral gene in the most recent common ancestor [Fitch, 2000]. This is an important task because homologous genes typically perform similar functions and any whole genome comparison would first need to establish the set of homologous genes [Koonin, 2005].

This is a non-trivial problem because the DNA sequence of genes are altered by mutations of the genome that changes the letters in the sequence or inserts/deletes letters from the sequence of the gene. Genes can also be duplicated, therefore different copies of a single ancestral gene may exists in different species. Finally genes may be lost if it accumulates so many mutations that it is no longer able to perform its function.

Homologous genes can be further divided into orthologs and paralogs. Orthologs are genes separated by a speciation event, while paralogs are genes separated by a duplication event. Figure 1.2 shows the family tree of three homologous genes superimposed on top of the species tree.

Ideally, we would like to establish one-to-one correspondences between genes in different species. This greatly simplify certain tasks such as transfer of function annotation [Friedberg, 2006] and genome rearrangement studies [Sankoff, 1999]. Unfortunately, orthologs are not necessarily one-to-one due to gene duplication.

The Ortholog Assignment problem was proposed in Fu et al. [2007] to identify for each ancestral gene, a single descendant gene in each species that best reflects the position of the ancestral gene. We call these genes positional homologs, following the terminology of Burgetz et al. [2006]. A similar problem called the Exemplar problem was proposed earlier in Sankoff [1999] in the context

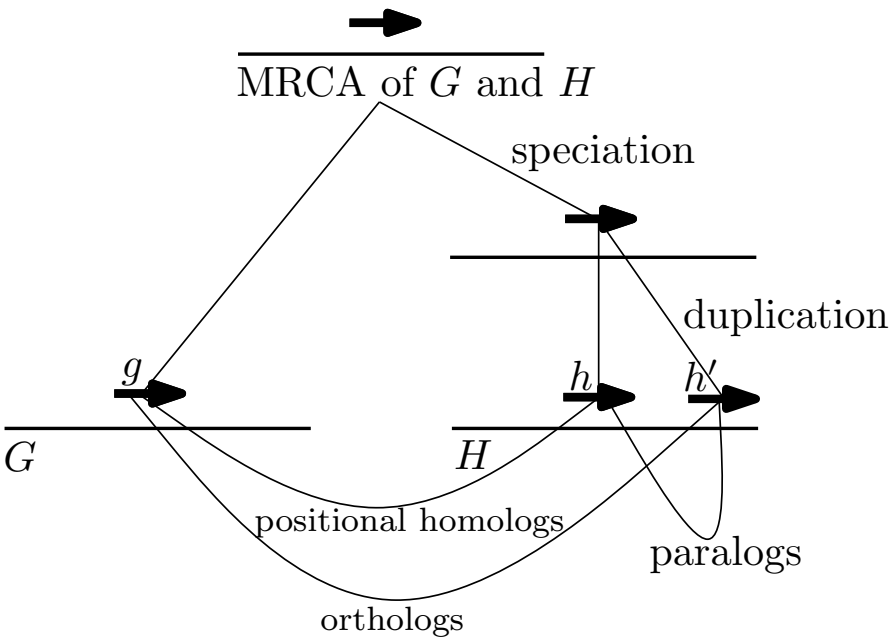

Fig. 1.2: The gene tree for three genes g, h, and h' that descended from a single ancestral gene in the most recent common ancestor (MRCA) of genome G and H. Gene g is orthologous to both h and h', but only g and h are positional homologs because h is the original gene that was duplicated to get h'. Genes h and h' are paralogs as they are separated by a duplication event.

of computing the genomic distance between gene orders. The Exemplar problem can be considered to be a approach for solving the Ortholog Assignment problem based on minimizing the genomic distance.

Assuming we have identified which are the homologous genes, we can start to consider conservation on a larger scale. A natural generalization is to consider sets of genes. What does it mean for a set of genes to be conserved? First, we have to understand the mutation events that affect whole segment of genes at once.

Table 1.1 show how the order of genes in a chromosome (gene order) is affected by different kinds of large scale mutations, also known as rearrangements. We represent genes by letters.

These large-scale mutations or rearrangements are relatively rare but they affect the content and order of the genomes, thereby obscuring the relationship between species [Sankoff, 2003]. These rearrangements are not entirely arbitrary as selective pressure removes those rearrangements which are fatal to the organism.

Type of rearrangement	**Effect on gene order**		
Reversal	a b c d e	⇒	a b e d c
Transposition	a b c d e	⇒	a c d e b
Inverted transposition	a b c d e	⇒	a e d c b
Insertion	a b _ c d e	⇒	a b f c d e
Duplication	a b _ c d e	⇒	a b a c d e
Deletion	a b c d e	⇒	a b d e

Table 1.1: Effect of rearrangements on gene order and gene content

As a result, over time regions of the genome which are not functionally related tend to accumulate more rearrangements as compared to regions which contains functionally dependent genes.

When comparing the genomes of several species, we can identify relatively compact regions in different species that have the same set of homologous genes. These genes managed to stay in close proximity to one another despite the rearrangements. As they are found in several species (conserved) and located in a compact region (clustered), we call them *conserved gene clusters*.

One possible reason for the existence of these clusters is that any rearrangement that disrupts the cluster is deleterious to the organism. This implies some kind of functional dependency among the genes in a cluster. In fact, Overbeek et al. [1999] showed that it is possible to infer functional coupling between genes based on the fact that they are part of some conserved clusters. In the study of prokaryotic genomes such as that of bacteria, conserved gene clusters is used in predicting operons [Ermolaeva et al., 2001] and detecting horizontal transfers [Lawrence, 1999].

Another reason why such clusters are observed is simply because not enough rearrangements have occurred since the species diverged. In either case, the clusters reflect the organization of these genes in the most recent common ancestor. Thus, conserved gene clusters are used to infer the gene order of the ancestral genome [Bergeron et al., 2004]. Establishing the number and size of conserved gene clusters between two genomes also provides an estimate of the similarity between two genomes. One approach for the Ortholog Assignment problem is to

select the gene pairs to maximize the similarity based on conserved gene clusters Bourque et al. [2005], Blin et al. [2006].

Lastly, the study of conserved gene clusters is also interesting from an algorithmic point of view. In most models of conserved gene clusters, the order of the genes does not matter. This gives rise to a new class of string problems that focuses on the character sets of substrings [Uno and Yagiura, 2000, Béal et al., 2004, Heber et al., 2011].

1.2 Thesis Organization and Contributions

The rest of this thesis is organized as follows. In **Chapter 2**, we summarize the related work for Conserved Gene Cluster Discovery and the Ortholog Assignment problem and discusses how it leads to our work. Chapters 3, 4, and 5 then presents the main contributions of thesis.

In **Chapter 3**, we introduce our *Gene Team Tree* (GTT) model, which is a parameter-free hierarchical representation of gene teams for all gap lengths. Gene team is a model for conserved gene clusters that allows for a gap of length at most δ within a cluster. In practice, determining an ideal value of δ is a matter of trial and error. Even worse, there is often no one single "best" value of δ. We propose to eliminate the parameter and simply compute all possible gene teams. It turns out to be possible to do this with the same worst case time complexity as computing the gene team for a specific δ and the computed teams can be represented hierarchically. We compute the GTT for *E. coli* K-12 and *B. subtilis* and confirmed that known *E. coli* K-12 operons corresponds to gene teams with different values of δ.

In **Chapter 4**, we investigated the use of the bidirectional best hit heuristic from sequence analysis for the purpose of identifying conserved gene clusters based on the r-window model. We call this new model *bidirectional best hit r-window model* (BBHRW) and designed a sub-quadratic time algorithm to find all clusters.

We studied how well does the gene clusters modelled by BBHRW and gene team corresponds to known *E. coli* K-12 operons. We found that the two model are complementary; the gene team model has more clusters that corresponds to operons, while the BBHRW model has fewer clusters that do not correspond to any operon. When we rank both sets of clusters and plot their precision and recall, we found that BBHRW model has a higher precision at all levels of recall as compared to the gene team model.

In **Chapter 5**, we studied the identification of conserved genes based on the Ortholog Assignment problem. We present a simple yet effective method (BBH-LS) for the identification of positional homologs from the comparative analysis of two genomes. BBH-LS applies the bidirectional best hit heuristic to a combination of sequence similarity and gene context similarity scores. We applied our method to the human, mouse, and rat genomes and found that BBH-LS produced the best results when using both sequence and gene context information equally. In our comparisons, BBH-LS reported the largest number of true positives and a medium level of false positives as compared to state-of-the-art methods.

We conclude and present a number of open issues in **Chapter 6**.

Chapter 2

Literature Review

In this chapter, we review the related literature and define some common notations and definitions to make the subsequent discussion more concise. We first review the existing models and algorithms for the Conserved Gene Cluster Discovery problem and discuss some of the issues which we addressed in our work. This is followed by a review and discussion of methods for the Ortholog Assignment problem.

This is an extension of the abstract "Survey of Algorithms for Conserved Gene Clusters Discovery" presented at the Asian Association for Algorithms and Computation (AAAC), 2011.

2.1 Basic Definitions and Notations

Our model of a genome is as a sequence of genomic markers for which homology information across the genomes of interest are available. The most common and well annotated type of genomic markers are protein coding genes. Henceforth, we will refer to these genomic elements as *genes*. The methods developed in this thesis work equally well with any kind of genomic feature that is conserved.

A notion that is central for identifying gene clusters is the relationship between genes. In particular, we need to identify genes from different species that have evolved from a common ancestral gene. Such a collection of genes is known as a

gene family [Fitch, 2000].

Definition 1 (Genes and gene families)**.** Let Σ denote the set of gene families. A *gene*, g, is part of a gene family denoted as $\text{fam}(g)$. Furthermore, a gene g in a genome G has a unique location on the genome that starts at $\text{start}(g)$ and ends at $\text{end}(g)$. We represent a gene g textually as $\text{fam}(g)^{\text{start}(g)}$. For simplicity, we omit the position if it is simply the index in the gene order.

The start position and end position can be defined based on either the index of the gene in the whole genome or using the position in base pairs. For small prokaryotic genomes, typically the base pair position is used and for large eukaryotic genomes, typically the index is used.

Definition 2 (Gene order)**.** A *gene order*, G, is a sequence of genes $\langle g_1, g_2, \ldots, g_n \rangle$, in non-decreasing order of their start position. A gene order is a permutation if each gene family occurs at most once, otherwise it is a sequence.

A uni-chromosomal genome can be directly represented as a gene order. Genomes with multiple chromosomes can be represented as a gene order by concatenating the chromosomes together in an arbitrary order and inserting an appropriate gap to separate genes from different chromosomes.

Hence, the input for the Conserved Gene Cluster Discovery problem is a m-tuple of gene orders $\mathcal{G} = (G_1, G_2, \ldots, G_m)$ and the output is a set of gene clusters.

2.2 Models and Algorithms for Conserved Gene Clusters Discovery

The approaches used in the literature can be broadly classified into two categories: algorithms base on a formal model of conserved gene clusters or heuristic methods without a explicit model. In this thesis, we focus on methods with an explicit model of conserved gene clusters.

Intuitively, a conserved gene cluster represents a compact region which contains a large proportion of homologous genes separated by regions that do not contain any shared genes. Due to the effect of rearrangement events, the order of the genes in a conserved gene cluster is usually not conserved and there may be gaps between these genes. Developing a formal definition of such clusters is a non-trivial task due to conflicting cluster properties [Hoberman and Durand, 2005].

The following sections describe a number of formal models that have been proposed in the literature.

2.2.1 Common Intervals and Conserved Intervals

The earliest formal definition of a conserved gene cluster is the *common interval* Uno and Yagiura [2000].

Definition 3 (Interval)**.** Given a gene order $G = \langle g_1, g_2, \ldots, g_n \rangle$ and an *interval* $[i, j]$, $G[i, j]$ denote the subsequence $\langle g_i, g_{i+1}, \ldots, g_j \rangle$.

Definition 4 (Character set)**.** The character set, CS, of a gene order G, is the set of gene families in G. Formally,

$$\mathrm{CS}(G) = \{\mathrm{fam}(g) \mid g \in G\}$$

Definition 5 (Common interval)**.** Given a set of m gene orders, a *common interval* is a m-tuple of intervals within each gene order with the same character set.

Example. Suppose $G = \langle a, b, c, d, e \rangle$ and $H = \langle e, d, b, a, c \rangle$, then $(G[1, 3], H[3, 5])$ is a common interval of G and H which the common character set $\{a, b, c\}$. However, $(G[2, 4], H[3, 5])$ is not a common interval as $\mathrm{CS}(G[2, 4])$ is $\{b, c, d\}$, while $\mathrm{CS}(H[3, 5])$ is $\{d, b, a\}$

Common intervals defines similar regions based on the content and ignores information about the order of the genes, however a class of common intervals called *conserved intervals* makes use of both the order and content of the genes

Problem	Reference	Complexity
Common intervals of 2 perm	Uno and Yagiura [2000]	$O(n+z)$
Common intervals of m perm	Heber et al. [2011]	$O(mn+z)$
Common intervals of 2 seq	Didier [2003]	$O(n^2 \log n)$
Common intervals of m seq	Schmidt and Stoye [2004]	$O(mn^2)$
Conserved intervals of m perm	Bergeron and Stoye [2003]	$O(mn)$

Table 2.1: Summary of algorithms for finding all common/conserved intervals, m is the number of input gene orders, n is the length of each gene order, and z is the output size

in the interval. The concept of conserved intervals was introduced in Bergeron and Stoye [2003] as a type of combinatorial structure that captures both local and global properties of gene orders. Conserved intervals are common intervals with the additional requirement that the order of the two genes at the ends of each conserved interval is the same in all genomes.

Table 2.1 summarizes the algorithms for computing common and conserved intervals and their complexity.

Algorithms for Common Interval of Permutations

The algorithms presented in Uno and Yagiura [2000] are based on the following theorem:

Theorem 1. *Let S be the character set of $G[i,j]$ and $p_{\min}$ be the minimum position in H for the genes in S and $p_{\max}$ be the maximum position in H for the genes in S. Then, $([i,j],[p_{\min},p_{\max}])$ is a common interval of G and H if and only if $p_{\max}-p_{\min}=j-i$.*

Direct application of the theorem gives us an $O(n^2)$ algorithm for finding all common intervals between two permutations of length n by checking all $O(n^2)$ intervals in G to determine if it forms a common interval. The running time of this algorithm can be reduced to $O(n+z)$ time where z is the number of common intervals by eliminating redundant checks using a filtering mechanism [Uno and Yagiura, 2000]. However, a fairly complicated data structure is needed to maintain the information needed for filtering.

Heber et al. [2011] gave a non-trivial extension of the preceding result to find the common intervals of m permutations by defining a novel generating subset of common intervals called *irreducible common intervals*.

Definition 6 (Irreducible common interval)**.** A common interval is a *irreducible common interval* if it is not the union of two overlapping common intervals.

Heber et al. proved that the set of irreducible intervals forms a basis of size $O(n)$ which can be used to generate all z common intervals in $O(z)$ time. Furthermore, the set of irreducible common intervals can be found in $O(mn)$ for m permutations of length n. Landau et al. [2005] proposed a different basis for the set of common intervals called *strong common intervals*.

Definition 7 (Strong common interval)**.** A common interval is a *strong common interval* if it does not overlap any other common intervals.

It is immediate from the definition that the number of strong common intervals is $O(n)$ and we can represent the strong common intervals of a set of permutations using a PQ-tree.

Both the set of irreducible intervals and the set of strong intervals can be used to generate all common intervals by taking the union of intervals. The disadvantage of these two approaches is that they are difficult to implement and require the use of complex data structures. Bergeron et al. [2005] proposed a different kind of generator based on taking the intersection which can be computed with the same complexity and implemented using basic data structures such as stacks and arrays.

Algorithms for Common Interval of Sequences

Most of the techniques used for finding the common intervals of a set of permutations cannot be extended to sequences. In general, problems on sequences have a higher computational complexity as compared to problems on permutations, as

there is a one-to-one correspondence between elements of permutations but not for sequences.

Didier [2003] gave the first $O(n^2 \log n)$ time algorithm for finding the common intervals of two sequences. Schmidt and Stoye [2004] proposed a simpler algorithm, with a time complexity of $O(kn^2)$, for finding the common intervals of k sequences, but it requires more space than Didier's algorithm ($O(n^2)$ instead of $O(n)$). Subsequently, the authors of these two algorithms managed to combine the best of each algorithm and devised an algorithm with $O(n^2)$ time and $O(n)$ space complexity [Didier et al., 2007].

For two gene orders G and H, the algorithm of Schmidt and Stoye [2004] first preprocess H to compute POS(f) and NUM(i, j), where POS(f) is a list of occurrence of gene family f in H and NUM(i, j) is the size of the character set of $H[i, j]$. These two structures can be computed in $O(n^2)$ time. The second step is to enumerate all $O(n^2)$ intervals in G incrementally, while maintaining an array to keep track of the corresponding intervals in H using POS. Each time an interval $[i, j]$ in H is found where NUM(i, j) is the size of the character set for the current interval in G, a common interval is found.

2.2.2 Gene Teams

Both common intervals and conserved intervals assumes that genes in the same cluster are contiguous. In other words, these two models do not consider the existence of gaps between genes in a conserved gene cluster. Bergeron et al. [2002] formalized the concept of *gene teams*, which is a generalization of common intervals that accounts for gaps. Gene teams are also referred to as *max-gap clusters* and they are commonly used in practice [Overbeek et al., 1999, Hoberman and Durand, 2005].

Gaps are simply section of the genome that lies between two genes in the same cluster. To formalize this notion, we define the distance between two genes. Recall that a gene's location is modelled as an interval [start(g), end(g)] along the

genome. Hence, the distance between two genes is simply the distance between two intervals.

Definition 8 (Distance between two genes)**.** The *distance* between two genes g and h that are on the same genome is defined as

$$\Delta(g, h) = \max\{0, \max(\text{start}(g), \text{start}(h)) - \min(\text{end}(g), \text{end}(h))\}$$

Note that the notion of distance only applies to genes on the same genome.

Definition 9 (Gene team)**.** Given a set of m gene orders and a max-gap parameter δ, a *gene team* is a maximal m-tuple of subsequences one in each gene order such that all m subsequences share the same character set and the distance between any pair of neighboring genes in a subsequence is at most δ.

Example. Consider the following two gene orders,

$$G = \left\langle a^1, b^2, a^6, c^8, b^9 \right\rangle$$
$$H = \left\langle c^1, c^4, b^5, a^6, b^8, b^9 \right\rangle$$

The gene teams of G and H for $\delta = 2$ are $(\langle a^1, b^2\rangle, \langle b^5, a^6, b^8, b^9\rangle)$, $(\langle c^8\rangle, \langle c^1\rangle)$, and $(\langle a^6, c^8, b^9\rangle, \langle c^4, b^5, a^6, b^8, b^9\rangle)$

The first algorithm for finding the gene teams of m permutations is an $O(mn \log^2 n)$ time algorithm [Bergeron et al., 2002]. Their algorithm is based on a divide and conquer strategy, which first locates a gap of size greater than δ and use it to split the permutations into two partitions. Then the algorithm recursively find the gene teams in each partition. If there are no gaps of size greater than δ in a particular partition, then all the genes in the partition form a gene team. Refining the algorithm using Hopcroft's partitioning framework improves the time complexity to $O(mn \log n \log \delta')$, where δ' is the maximum number of genes in an interval of length δ over all gene orders [Béal et al., 2004]. Subsequently, Wang and Lin [2011] proposed an output dependent algorithm that makes use of job

queues instead of recursive calls. A careful analysis showed that their algorithm has a time complexity of $O(mn \lg z)$ where z is the number of gene teams.

He and Goldwasser [2005] extended the first algorithm to work for sequences. However, as there is no one-to-one correspondence between the genes, in the divide step the resulting subsequences do not form a partition of the original sequences. By using a clever marking strategy, they were able to compute the gene teams for two sequences with n_1 and n_2 genes respectively in $O(n_1 + n_2)$ space and $O(n_1 n_2)$ time. Recently, Wang et al. [2012] did a more careful extension of the basic algorithm to sequences and showed that it has a worst case running time of $O(\min n_1 n_2, z \lg(n_1 + n_2))$.

So far all the algorithms use a top down decomposition, Ling et al. [2008] presented an algorithm following the "candidate generation and verification" approach from data mining. Their algorithm iteratively merge candidate clusters to form larger teams. They observed that their bottom up approach is more efficient when the actual gene teams are small as the top down methods spend too much time breaking down the problem. Unfortunately, they do not have a bound on the running time of their algorithm.

While the basic gene team model has received considerable attention from the research community, there has been several attempts to relax some of the constraints of the model. For instance, when considering multiple gene orders, it is quite difficult to maintain the condition that a team must be present in every gene order. The following two variants of gene teams, attempt to overcome this problem.

Domain team

The *domain teams* model was proposed in Pasek et al. [2005] as a generalisation of the gene teams model. It only requires that a team exist in at least one of the gene orders. Unfortunately, the definition may result in an exponential number

of domain teams. However, it is shown in Pasek et al. [2005] that real-life examples involving thousands of genes can be computed efficiently in reasonable time, although no details of the algorithm was given in the paper.

A more general model is to impose a quorum parameter q so that a team exists in least q out of the m input gene orders [Parida, 2007, Ling et al., 2009]. Parida [2007] proposed an algorithm with a worst case time complexity that is output sensitive. The same problem was considered in Ling et al. [2009] and they developed an algorithm based on the Apriori heuristic from data mining.

Hybrid Gene Pattern

A different approach to the issue of generalising the gene teams model to multiple gene orders was taken by Kim et al. [2005]. They proposed a hybrid gene pattern mining strategy similar to the classic Apriori algorithm for mining association rules [Agrawal and Srikant, 1994]. Their algorithm is based on a level-wise enumeration of gene sets, utilizing the Apriori property for pruning.

One difficulty with applying their approach is that it requires the specification of four parameters, namely, the parameter δ, the number of gene orders which contains the gene set and satisfy the max-gap constraint, the number of gene orders which contains the gene set, and the minimum number of genes in a gene set.

2.2.3 r-window Clusters

Another type of cluster definition which allows for gaps between genes in a cluster is the *r-window* clusters. Similar to gene teams, it is also a generalisation of common intervals.

Definition 10 (r-window cluster)**.** Given a set of m gene orders and two parameters r and k, a *r-window cluster* is a m-tuple of intervals one in each gene order. The length of each interval is at most r and the m intervals contains at least k homologous genes.

Similar definitions have been used in Friedman and Hughes [2001], Cavalcanti et al. [2003] for the study of segmental duplications, but the first formal definition appeared in Durand and Sankoff [2003] together with a probabilistic analysis of such clusters.

A naïve algorithm method is to generate all intervals of length r in one genome and compare it with all intervals of length r in the second genome. This suggest an $O(n^2r)$ time algorithm to determine all r-window gene clusters. Algorithms based on heuristics, such as the CloseUp algorithm [Hampson et al., 2003], have also been proposed for finding r-window gene clusters, however, such algorithms are not guaranteed to find all r-window gene clusters.

Yang et al. [2010] proposed a simple formulation of r-window cluster based on finding all maximal clusters. Under their formulation a cluster does not have to exist in all the input gene orders. Unfortunately, this formulation is NP-hard even for the simplest case where each gene order is a permutation. They showed that that restricted version where clusters are ordered and must appear in each gene order can be reduced to the problem of find the longest path in a directed acyclic graph. They also describe an exponential time algorithm for the general case.

2.2.4 Discussion

The relatively simple models such as common intervals and conserved intervals have received considerable attention from the community. However, the inability to model clusters with gaps is a serious drawback.

In Hoberman and Durand [2005], the authors presented a comparison between gene teams and r-window gene clusters with regards to several desirable properties of gene clusters. Some of the cluster properties they considered include, size (number of homologous genes in a cluster), length (total number of genes in a cluster), global density (size of cluster/length of cluster) and local density (variance in gap sizes between consecutive genes in a cluster). The size and length of gene teams are not constrained whereas r-window gene clusters have a size of at least

m genes and a length of at most r genes. Base on the definition of r-window clusters, it is clear that each cluster has a global density of at least m/r, however the gap size may be as large as $r - m$. In the case of gene teams, the gap size is at most δ but it is difficult to constrain the global density of the clusters. This shows that different cluster definitions allows us to model different characteristics of conserved gene clusters.

It would be simple to come up with a model that include all of the desirable properties but it would need to have so many parameters as to be unusable. Most extensions of existing models introduce additional parameters to increase the flexibility of the model. However, this pushes the burden of modelling to the user of the model instead of the designer. Models with many parameters also do not generalize well. In this thesis, we adopt the opposite approach of trying to reduce the number of parameters in a model.

2.3 Algorithms for the Ortholog Assignment Problem

The problem of finding the set of positional homologs between two genomes is known as the Ortholog Assignment problem [Fu et al., 2007]. Algorithms for the Ortholog Assignment problem fall into three categories: distance minimization, similarity maximization, and rule-based.

2.3.1 Distance minimization

Distance minimization methods relies on the parsimony principle. They assume that the removal of all the genes except for the positional homologs minimizes the genomic distance (usually some form of edit distance with genomic operations) between two genomes. Genomic distance measures such as the reversal distance [Hannenhalli and Pevzner, 1999] and breakpoint distance [Watterson et al., 1982] have been considered using a branch-and-bound approach [Sankoff, 1999] as the

corresponding computational problems are NP-hard [Bryant, 2000]. MSOAR2 [Shi et al., 2010] uses a number of heuristic algorithms to assign positional homolog pairs in several phases to minimize the number of reversals, translocations, fusions, fissions, and gene duplications between two genomes.

2.3.2 Similarity maximization

Closely related to distance minimization are the *similarity maximization* approaches. By identifying conserved structures between genomes, we can determine the similarity between them. We can model the Ortholog Assignment problem as finding the set of positional homologs that maximize the degree of similarity between two genomes. Bourque et al. [2005] uses heuristics for the MAX-SAT problem to maximize the number of common or conserved intervals. The problem of maximizing the number conserved intervals is NP-hard [Blin and Rizzi, 2005]. Blin et al. [2006] proposed a greedy method based on algorithms for global alignment that first finds a set of anchors and then recursively match genes found in large common intervals.

2.3.3 Heuristics/rule-based

A widely used method for finding pairwise orthologs based on sequence similarity is the *bidirectional best hit (BBH)* heuristic. Two genes g and h in different species form bidirectional best hits if the similarity between g and h is greater than that between g and any other gene (h is the best hit for g) and vice versa. In Burgetz et al. [2006], a pair of BBHs are positional homologs if they are next to another pair of BBHs. Subsequently, Jun et al. [2009] relaxed this condition and defined a local synteny test to determine whether a given pair of genes is a positional homolog pair. A gene pair passes the local synteny check if there are at least two pairs of genes (excluding the gene pair being tested) nearby with a sequence similarity above a certain threshold. Note that the local synteny test does not consider the sequence similarity between the gene pair being tested.

Since positional homologs are a subset of all orthologs, other rule based methods designed for finding orthologs [Schneider et al., 2007, Ostlund et al., 2010] can also be used to identify positional homologs by restricting ourselves to one-to-one orthologous groups.

2.3.4 Discussion

Methods based on distance minimization and similarity maximization assumes that gene families have been computed. Computing gene families is typically accomplished using sequence similarity search followed by clustering of similar genes [Li et al., 2003]. After that, sequence similarity is essentially reduced to a simple binary relation; two genes are equivalent if they are in the same gene family and different otherwise. The main step uses heuristics to find a subset of genes that optimizes an NP-hard problem on gene orders. In short, these methods use sequence similarity to build gene families and gene order information to further refine the gene families to get one-to-one gene matchings.

In contrast, rule-based methods typically do not need to build gene families. However, they only make use of gene order/gene context information in a limited way. Instead of treating gene context information as a simple binary condition, a more uniform method is to compute a numeric score to denote the level of gene context similarity. This allows us to treat both sequence similarity and gene context similarity in a unified way.

Chapter 3

A Parameter-Free Max-Gap Gene Cluster Model

Most extensions of the gene team model focused on improving the running time or generalizing the model. We focused on a different issue, that of making the gene team model easier to use in practice. A crucial issue is the problem of choosing the right value of the max-gap parameter δ.

Selecting the value to use for the max-gap parameter is often a matter of trial and error. There is an inherent structure between gene teams for different values of δ that is not captured by considering different values of δ independently. In this chapter, we propose the Gene Team Tree model which is a parameter-free, hierarchical representation of gene teams over all possible values of δ and we present efficient algorithms for computing the Gene Team Tree that has the same complexity as algorithms for computing gene teams for a single value of δ.

This chapter is based on Zhang and Leong [2008, 2009]. An implementation of the algorithms described in this chapter and the datasets used in the experiments can be downloaded from `http://gtt.assembla.me`.

3.1 Motivation

The *gene team* model [Bergeron et al., 2002] generalizes the common intervals model [Uno and Yagiura, 2000] to allow gaps between genes in the same cluster. It is widely used in practice [Hoberman and Durand, 2005] and there are efficient algorithms for its discovery. Current algorithms require the specification of the parameter δ, which is the maximum gap between adjacent genes in a team.

Determining suitable values for this parameter is nontrivial as it depends on the distribution and arrangement of genes on the genome. As discussed in He and Goldwasser [2005], a large value of δ may result in many false positives, while an overly conservative value may miss many potential conserved clusters. In addition, due to varying rates of rearrangement, different regions of the genome may require different values of δ to discover useful gene clusters. The value of δ also depends on the application one is interested in, for example, finding operons or detecting segmental duplications.

In the experimental study presented in He and Goldwasser [2005], the approach used to determine an appropriate value of δ was to select a small number of known operons and pick the minimum value of δ at which the selected operons were reconstructed. There are two drawbacks with this method. Firstly, there may not be any known operons in the genome we are interested in. Furthermore, it is unclear how to select a representative set of known operons.

In a study of the same dataset by Ling et al. [2008], the gene teams for a range of δ values were analyzed to identify some new patterns, such as clusters spanning multiple operons. This illustrates the utility of considering different values of δ in order to discover interesting gene clusters.

Instead of trying to determine a single "best" value of δ, we propose computing the gene teams for all possible values of δ. The results can be represented compactly in a tree structure, which we call the *Gene Team Tree*. Subsequently, statistical tests [Hoberman et al., 2005] or integration with other information on gene interactions can be used to validate or rank the discovered teams.

Our algorithm for computing the GTT extends existing gene team mining algorithms without increasing their time complexity. We compute the GTT for *E. coli* K-12 and *B. subtilis* and show that *E. coli* K-12 operons are modelled by gene teams with different values of δ. We demonstrate the scalability of our method and the trade-off involved when comparing more than two genomes, through a comparative study using five gamma-proteobacteria genomes. Lastly, we describe how to compute the GTT for multi-chromosomal genomes and illustrate by computing the GTT for the human and mouse genomes.

We first present a formal definition of the problem of computing all gene teams (Section 3.2). This is followed by a description of our proposed Gene Team Tree model, corresponding algorithms (Section 3.3) and experimental results (Section 3.4). Finally, we summarize our contributions and discuss extensions and future works (Section 3.5).

3.2 Problem Definition

3.2.1 Notations and definitions

In the following definitions, we define a δ-team as a tuple of sequences instead of as a set of gene families, as originally defined in Bergeron et al. [2002]. This is because once we allow multiple genes from the same family in a single gene order, the same set of gene families may correspond to different subsequences of the input gene orders. Furthermore, in practice, we are interested in the genes that are part a gene cluster, rather than just the set of gene families involved.

Definition 11 (δ-sequence)**.** A gene order $G = \langle g_1, g_2, \ldots, g_n \rangle$ is a *δ-sequence* if and only if every pair of adjacent genes in G are separated by a distance of at most δ, i.e. $\forall i \in [1, n-1], \Delta(g_i, g_{i+1}) \leq \delta$.

Definition 12 (δ-cluster)**.** Given a collection of gene orders $\mathcal{G} = (G_1, G_2, \ldots, G_m)$, a *$\delta$-cluster* is a m-tuple of δ-sequences $(G'_1, G'_2, \ldots, G'_m)$, such that $\forall i \in [1, m], G'_i$

is a subsequence of G_i and $\mathrm{CS}(G'_1) = \mathrm{CS}(G'_i)$, where $\mathrm{CS}(G)$ is the set of gene families in the gene order G.

Definition 13 (δ-team)**.** A *δ-team* is a *maximal* δ-cluster. A δ-cluster t, is maximal if there is no other δ-cluster t' such that every one of the δ-sequence of t' is a supersequence of the corresponding δ-sequence of t.

Example. Consider the following two gene orders,

$$G = \langle a^1, b^2, a^6, c^8, b^9 \rangle$$
$$H = \langle c^1, c^4, b^5, a^6, b^8, b^9 \rangle$$

where the letters represent gene families and the superscripts denote the start position (for simplicity, the end positions are the same as the start positions). The 2-teams of G and H are $(\langle a^1, b^2 \rangle, \langle b^5, a^6, b^8, b^9 \rangle)$, $(\langle c^8 \rangle, \langle c^1 \rangle)$, and $(\langle a^6, c^8, b^9 \rangle, \langle c^4, b^5, a^6, b^8, b^9 \rangle)$

3.2.2 The AllGeneTeams problem

Given a m-tuple of input gene orders, $\mathcal{G}$, compute the set of δ-teams over all possible values of δ.

Formally, given $\mathcal{G}$, compute

$$\mathcal{T} = \bigcup_{\delta=0}^{\infty} \textsc{GeneTeams}(\mathcal{G}, \delta)$$

where $\textsc{GeneTeams}(\mathcal{G}, \delta)$ denote the δ-teams of $\mathcal{G}$.

3.3 Gene Team Tree Model and Algorithms

A naïve approach is to iterate over the range of δ and for each value of δ apply one of the existing algorithms [Bergeron et al., 2002, He and Goldwasser, 2005] to compute the gene teams. This is very inefficient as the parameter space is large.

3.3.1 A motivating example

In order to develop some insights about the structure of gene teams for different values of δ, consider the following example.

Example. Given $G = \langle a^1, b^2, a^6, c^8, b^9\rangle$ and $H = \langle c^1, c^4, b^5, a^6, b^8, b^9\rangle$, the 3-teams are

$$(\langle a^1, b^2\rangle, \langle b^5, a^6, b^8, b^9\rangle), (\langle a^6, c^8, b^9\rangle, \langle c^1, c^4, b^5, a^6, b^8, b^9\rangle)$$

and the 2-teams are

$$(\langle a^1, b^2\rangle, \langle b^5, a^6, b^8, b^9\rangle), (\langle c^8\rangle, \langle c^1\rangle), (\langle a^6, c^8, b^9\rangle, \langle c^4, b^5, a^6, b^8, b^9\rangle)$$

We observed that $(\langle a^1, b^2\rangle, \langle b^5, a^6, b^8, b^9\rangle)$ is both a 2-team and a 3-team. Furthermore, the large 3-team, $(\langle a^6, c^8, b^9\rangle, \langle c^1, c^4, b^5, a^6, b^8, b^9\rangle)$, is split into two smaller 2-teams.

From the above example, we observe two properties of gene teams that allows us to improve upon the naïve approach.

Firstly, gene teams computed for one value of the parameter can be used to compute the gene teams for other values. As maximum allowed gap length (δ) decreases, existing gene teams are split into smaller teams. This allows us to represent the set of gene teams for all values of δ compactly in a tree structure.

Secondly, some teams remain unchanged when δ is decreased. This suggests that the GTT can be computed more efficiently, than the brute force approach of trying every value of δ. In particular, existing teams are split into smaller teams when the value of δ decreases below the maximum gap within the team.

3.3.2 Gene Team Tree (GTT)

The above observations motivates us to model the set of all gene teams using a tree structure, which we call the *Gene Team Tree*.

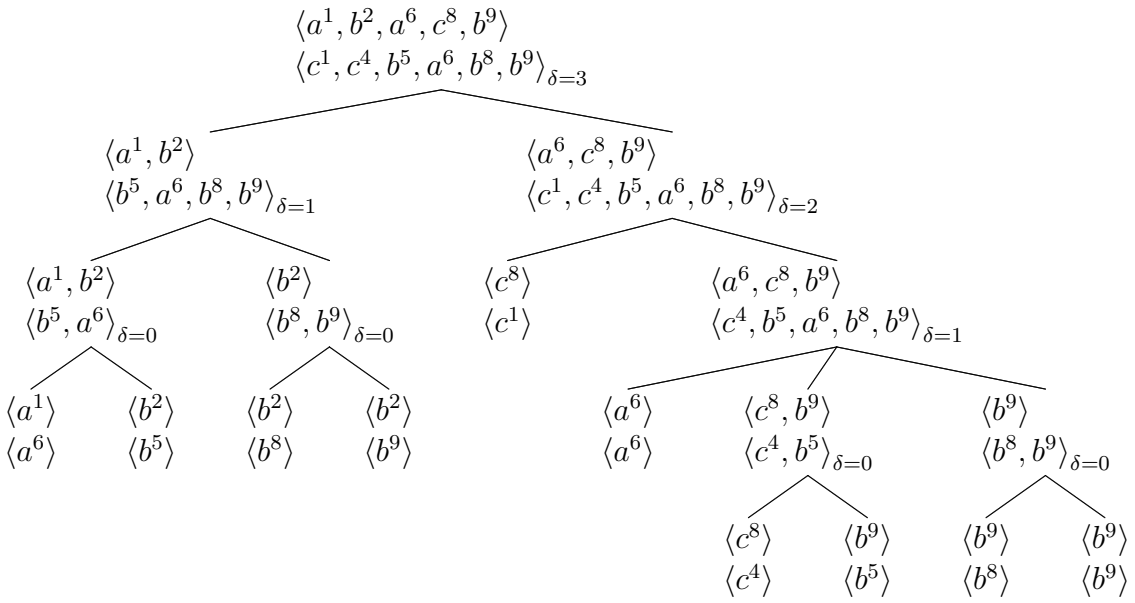

Fig. 3.1: GTT for $\langle a^1, b^2, a^6, c^8, b^9\rangle$ and $\langle c^1, c^4, b^5, a^6, b^8, b^9\rangle$. The value of δ used to split each gene team is shown in subscripts.

Definition 14 (Gene Team Tree)**.** The *Gene Team Tree* for a tuple of gene orders $\mathcal{G}$ is a tree consisting of δ-teams for all values of δ. The root of the GTT is $\mathcal{G}$ and a gene team t' is a descendant of a gene team t if all the sequences of t' are subsequences of t.

Example. Figure 3.1 shows the GTT for the pair of gene orders presented in the previous example.

3.3.3 Properties of the GTT

In the following sections, we describe some of the formal properties of the GTT structure and suggest some biological interpretation.

Space complexity

Leaf nodes of the GTT consists of a single gene from each gene order, all from the same family. Thus, the number of leaf nodes for a particular family f is the product of the size of the family in each gene order, *i.e.*, $\prod_{G\in\mathcal{G}} \mathrm{occ}(G, f)$, where $\mathrm{occ}(G, f)$ is the number of genes in G from gene family f. It follows that the total number of leaf nodes, denoted by L, is the sum of the leaf nodes over all families,

i.e.,

$$L = \sum_{f \in \Sigma} \prod_{G \in \mathcal{G}} \mathrm{occ}(G, f)$$

. This shows that the total number of nodes is $\Theta(L)$. In particular, when the input gene orders are permutations of length n, the number of leaf nodes is n and the total number of nodes is $\Theta(n)$. As the number of genes from the same family increases, so does the size of the GTT.

A δ-sequence of gene order G can be represented as a triple (G, g_s, g_e), where g_s/g_e is the leftmost/rightmost gene in the δ-sequence [He and Goldwasser, 2005]. This allows us to represent each gene team in $\Theta(m)$ space and the entire GTT in $\Theta(mL)$ space.

Range of a gene team

Observe that a gene team t, is not only a δ-team, but it is really a $[\delta_{\min}, \delta_{\max})$-team, where $[\delta_{\min}, \delta_{\max})$ is the range of δ values for which t is a δ-team. For simplicity, we call $[\delta_{\min}, \delta_{\max})$ the range of the gene team t.

Let $\textsc{MaxGap}(\mathcal{G})$ be the largest distance between adjacent genes for every gene order in $\mathcal{G}$. From the definition of GTT, $\delta_{\min} = \textsc{MaxGap}(t)$ and $\delta_{\max} = \textsc{MaxGap}(t')$, where t' is the parent of t in the GTT.

Intuitively, a gene team with a large range is more "robust" than a team that only exists for a few values of δ.

Shared neighborhood

Most methods which make use of the similarity in genomic context between two genes, restrict their attention to a fix number of genes upstream and downstream of the genes of interest. Recall that a leaf node represents a m-tuple of genes of the same family, one from each gene order. We observe that the gene teams on the path from the leaf to the root represent the maximal shared neighborhoods of these genes. Thus, the GTT can be used for analyzing the similarity in genomic context between homologous genes without fixing the size of the neighbourhood

in advance.

3.3.4 Algorithm SimpleGTT

Clearly, the largest possible gene team is $\mathcal{G}$. $\mathcal{G}$ is a gene team when δ is greater than or equal to the largest distance between adjacent genes in each of the m gene orders.

Our algorithm SimpleGTT takes in a tuple of gene orders $\mathcal{G}$ and returns the GTT of $\mathcal{G}$. Initially, we start with the gene team $\mathcal{G}$. In each call to SimpleGTT, we construct a tree node v which stores the gene team. Then, we compute $\text{MaxGap}(\mathcal{G}) - \epsilon$, which is the largest value of δ that will cause the current gene team to be partitioned into smaller sub gene teams. We then make use of the FindTeams algorithm [Bergeron et al., 2002, He and Goldwasser, 2005] to partition t into a set of smaller gene teams T. For each gene team in T, recursively apply SimpleGTT to get a tree of gene teams and make it a subtree of v. Finally, the algorithm returns v, which is the root of the GTT. The pseudocode for the algorithm is shown in Algorithm 1.

In practice, if the positions are integers, we set ϵ to be 1. This is the case in our examples and experiments. Otherwise, ϵ can be set to some number that is smaller than the distance between the closest pair of adjacent genes.

Algorithm 1 SimpleGTT($\mathcal{G}$)

Ensure: Returns the GTT for $\mathcal{G}$
```
team(v) := G
children(v) := ∅
if |team(v)| > 1 then
  δ := MaxGap(G) − ε
  T := FindTeams(G, δ)
  for each gene team t ∈ T do
    children(v) := children(v) ∪ SimpleGTT(t)
  end for
end if
return v
```

3.3.5 Correctness of SimpleGTT

Let SimpleGTT($\mathcal{G}$) denote the result of running algorithm SimpleGTT on the tuple of gene orders $\mathcal{G}$. In order to show that the algorithm SimpleGTT is correct, we need to prove that

$$\textsc{SimpleGTT}(\mathcal{G}) = \mathcal{T}$$

First, we show that only certain values δ will lead to new gene teams.

Lemma 1.

$$\textsc{GeneTeams}(\mathcal{G}, \delta) \begin{cases} = \{\mathcal{G}\} & \textit{if } \delta \geq \textsc{MaxGap}(\mathcal{G}), \\ \neq \{\mathcal{G}\} & \textit{otherwise} \end{cases}$$

Proof. Since MaxGap($\mathcal{G}$) is the largest distance between adjacent genes, when $\delta \geq$ MaxGap($\mathcal{G}$), $\mathcal{G}$ clearly satisfies the definition of a gene team.

Otherwise, there exists a pair of adjacent genes g, g' such that $\Delta(g, g') =$ MaxGap($\mathcal{G}$). The genes g and g' cannot be in the same gene team, thus $\mathcal{G}$ cannot be a gene team. □

The above result can be generalized to an arbitrary gene team, therefore not all values of δ lead to new gene teams. There are certain critical values of δ which lead to the formation of new gene teams, these are the values of δ which are just less than the maximum gap between adjacent genes in a gene team.

The following lemma shows that once a pair of genes is in two different gene teams for a particular value of δ say δ_1, then they will always be in different gene teams when the value of δ decreases to say δ_2. This allows us to apply a divide-and-conquer strategy because gene teams form independent subproblems.

Lemma 2. *If* $\delta_1 > \delta_2$ *and* GeneTeams($\mathcal{G}, \delta_1$) $= T_1$ *and* GeneTeams($\mathcal{G}, \delta_2$) $= T_2$ *then for any two genes* g *and* g' *if they are in the different gene teams in* T_1 *they are also in different gene teams in* T_2

Proof. Suppose on the contrary that g and g' are in different gene teams in T_1 but they are in the same gene team in T_2. Let t be the gene team in T_1 that contains g and t' be the gene team in T_1 that contains g'. Since g and g' are in the same gene team in T_2 it follows that $t \cup t'$ must also be a gene team in T_1. This contradicts the maximality of t and t'. □

Theorem 2 (SimpleGTT is correct)**.**

$$\textsc{SimpleGTT}(\mathcal{G}) = \mathcal{T}$$

Proof. SimpleGTT finds gene teams for values of δ in decreasing order. Lemma 1 ensures that by using the value of $\delta = \textsc{MaxGap}(\mathcal{G}) - \epsilon$ to partition the current gene team, we do not miss out on any gene teams. After partitioning the gene team t, we can consider the sub gene teams in a divide-and-conquer fashion since each sub gene team is independent of the others as a consequence of Lemma 2. □

3.3.6 Time Complexity of SimpleGTT

Given set of m gene orders, the time complexity of FindTeams is $O(mn \lg^2 n)$ [Bergeron et al., 2002] when the gene orders are permutations of length n and $O(n_{\max}^m)$ for general sequence [He and Goldwasser, 2005], where $n_{\max}$ is the length of the longest gene order.

A simple implementation of MaxGap performs a linear scan over each gene order to determine the largest distance between adjacent genes. It has a time complexity of $O(mn_{\max})$.

Let s_i denote the size of the ith gene team generated by FindTeams and let $T(n)$ denote the time complexity of SimpleGTT, then

$$T(n) = \begin{cases} c & \text{if } n = 1, \\ \sum_{i=1}^{k} T(s_i) + \text{time for FindTeams} + \text{time for MaxGap} & \text{otherwise} \end{cases}$$

The running time for FindTeams dominates that of MaxGap. The worst

case occurs when FindTeams splits the gene team into two uneven partitions.

$$T(n) = T(1) + T(n-1) + \text{time for FindTeams} + \text{time for MaxGap}$$
$$= \begin{cases} O(mn^2 \lg^2 n) & \text{if the input gene orders are permutations,} \\ O(n_{\max}^{m+1}) & \text{otherwise} \end{cases}$$

3.3.7 Algorithm FastGTT: Speeding up SimpleGTT

The basic algorithm presented in the previous section treats the FindTeams procedure as a black box. It turn out that we can directly compute the GTT within FindTeams to reduce the time complexity.

The key idea is to adjust the value of δ dynamically during the recursion in the FindTeams procedure. When FindTeams reports a gene team t, instead of terminating the recursion, we will dynamically reduce the value of δ to $\text{MaxGap}(\mathcal{G}) - \epsilon$ and continue recursing. Initially, the value of δ is set to ∞.

The improved algorithm, FastGTT, is presented in Algorithm 2 and 3.

Algorithm 2 FastGTT($\mathcal{G}$)

Ensure: Returns the GTT of $\mathcal{G}$
  team(v) := $\emptyset$ {Setup a dummy root node}
  children(v) := $\emptyset$
  return ModifiedFindTeams($\mathcal{G}, \infty, v$)

Algorithm 3 ModifiedFindTeams($\mathcal{G}, \delta, u$)

  δ' := MaxGap($\mathcal{G}$) $- \epsilon$
  if $\delta' \leq \delta$ **then**
    {$\mathcal{G}$ is a gene team}
    team(v) := $\mathcal{G}$
    children(v) := $\emptyset$
    children(u) := children(u) $\cup \{v\}$
    ModifiedFindTeams($\mathcal{G}, \delta', v$)
    return v
  else
    $\mathcal{G}'$:= ExtractRun($\mathcal{G}, \delta$)
    ModifiedFindTeams($\mathcal{G}', \delta, u$)
    ModifiedFindTeams($\mathcal{G} - \mathcal{G}', \delta, u$)
    return u
  end if

Recall that ExtractRun [Bergeron et al., 2002] takes in as input the tuple of gene orders $\mathcal{G}$ and the parameter δ. It splits one of the gene orders of $\mathcal{G}$ across a gap of length greater than δ into two subsequences, keeping the shorter subsequence, G'. It then returns the tuple of gene orders obtained by extracting from each gene order in $\mathcal{G}$ the subsequence containing the same gene families as G'.

In order for our FastGTT algorithm to be efficient, the complexity of MaxGap should be at most that of ExtractRun. The complexity of ExtractRun in He and Goldwasser [2005] is $O(mn_{\max})$, therefore the simple linear scan implementation of MaxGap suggested in the previous section suffices. However, the complexity of ExtractRun in Bergeron et al. [2002] is $O(mp \lg p)$ where p is the size of the smaller sub problem.

We can reduce the time complexity of MaxGap by storing the length of the gaps between adjacent genes in a priority queue. Then, the complexity of MaxGap becomes $O(1)$, however we incur overhead maintaining the priority queue.

Creating the priority queue for the subproblem of size p requires mp insertions. The priority queue for the subproblem of size $n-p$ can be obtained by modifying the original priority queue. Extracting a single gene may involve merging two gaps, and is accomplished by deleting the two gaps and inserting the new combined gap into the priority queue. The total number of insertions and deletions needed to update the priority queue is $O(mp)$. Using a binary heap for our priority queue requires $O(\lg n)$ operations for deletions/insertions. Therefore, the total overhead is $O(mp \lg n)$, which is more than $O(mp \lg p)$.

Hence, the running time of our algorithm for m permutation is given by the following recurrence relation,

$$T(n) = T(n-p) + T(p) + cmp \lg n, 1 \leq p \leq n/2$$

Similar to the analysis presented in Bergeron et al. [2002], in the worst case, $p = n/2$ and $T(n) = O(mn \lg^2 n)$.

We analyzed the expected running time based on the assumption that at each

stage , the size of the smaller subproblem, p, is uniformly distributed between 1 and $n/2$. Let $E(n)$ denote the expected running time of the algorithm. Then,

$$\begin{aligned}
E(n) &= \frac{1}{n/2}\sum_{p=1}^{n/2} E(n-p) + E(p) + cmp\lg n \\
nE(n) - (n-2)E(n-2) &= 2E(n-1) + 2E(n-2) + cmn\lg n \\
E(n) &\geq \frac{n+2}{n}E(n-2) + cm\lg n \\
&\geq cm(n+2)\sum_{x=1}^{(n-3)/2}\frac{\lg(2x+1)}{2x+1} \\
&\geq cm(n+2)\int_{1}^{(n-1)/2}\frac{\lg(2x+1)}{2x+1}\,dx \\
&= \Omega(mn\lg^2 n)
\end{aligned}$$

Since $T(n) = O(mn\lg^2 n)$ in the worst case, thus $E(n) = O(mn\lg^2 n)$. Therefore, the expected running time is $\Theta(mn\lg^2 n)$.

Our improved FastGTT algorithm has a time complexity of $O(mn\lg^2 n)$ for m permutations of length n and $O(n_{\max}^m)$ for m sequences of length at most $n_{\max}$. Surprisingly, we were able to compute the GTT with the same time complexity as computing the gene teams for a single value of δ.

3.3.8 Handling multiple chromosomes

A single chromosome can be directly represented using a gene order, however many genomes are multi-chromosomal. In practice, we would like to compare entire multi-chromosomal genomes and genes in a gene team should not be spread across several chromosomes.

A simple strategy to map multi-chromosomal genomes into a single linear gene order is to order the chromosomes linearly and insert appropriate gaps between the chromosomes. The order does not matter since the chromosomes will be separated immediately.

Let $l_{\max}$ be the length of the longest chromosome, then it suffices to insert a

gap of length $l_{\max} + \epsilon$ between chromosomes. Base on the definition of the GTT, the root consists of all the genes in the genome and the children of the root are GeneTeams($\mathcal{G}, l_{\max}$). This will separate the genes in each chromosome but allows teams which consist of an entire chromosome. The subsequent subproblems will only be within a single chromosome.

3.4 Experimental Results

In the following, we apply our approach to the comparison of several biological datasets. Our purpose is to illustrate the limitation imposed by considering only a single value of δ and to demonstrate the practicality of our algorithm.

We implemented the FastGTT algorithm presented in Section 3.3 in Java and performed all computations on a Intel Core 2 Duo E6550 (2.33GHz) processor with 2GB of RAM running Linux.

We validated the teams from the computed GTTs against biologically significant sets of genes from manually curated databases. As it is difficult to have an exact match, we use the *Jaccard coefficient* to determine the level of similarity between two sets of genes. To determine how well a particular set of genes is represent in a GTT, we compute its *Jaccard score* with respect to the GTT.

Definition 15. The *Jaccard coefficient* [Jaccard, 1908] of two sets O and T is defined as $\frac{|O \cap T|}{|O \cup T|}$. It gives a value between zero and one. A value of one indicates an exact match, *i.e.*, $O = T$.

Definition 16. The *Jaccard score* of a set of genes O with respect to a set of teams $\mathcal{T}$ is the highest Jaccard coefficient between O and some team in $\mathcal{T}$.

3.4.1 *E. coli* K-12 and *B. subtilis* Dataset

In this study, we compared how well a specific type of conserved gene clusters in prokaryotes are modelled by gene teams. Our objective is to determine the improvement in representational power when we consider gene teams over all possible

Table 3.1: Number of genes and gene families in the *E. coli* K-12 and *B. subtilis* dataset. A common gene family is a gene family that is present in both genomes.

Genome	*E. coli* K-12	*B. subtilis*
#genes	4132	4105
#genes from a common gene family	2250	2364
#gene families	2140	1801
#common gene families	1168	1168

values of δ instead of only a single value.

The dataset consists of two prokaryotic genomes, namely *E. coli* K-12 (GenBank accession number NC000913) and *B. subtilis* (NC000964). We downloaded the protein table file for each genome from the NCBI Microbial Genomes Resource.[1] The start and end position of each gene were determined from the location of the gene in base pairs, and gene families were approximated using the COG [Tatusov et al., 2001] labels in the protein table files. Table 3.1 shows the number of genes and gene families in this dataset.

We only consider genes from gene families that are common to both genomes. Hence, the size of the input gene orders is indicated in the second row of Table 3.1. Computing the GTT took 4 seconds and the resulting tree has 17793 nodes. A portion of the GTT showing gene teams containing at least 10 families is illustrated in Figure 3.2.

[1] ftp://ftp.ncbi.nih.gov/genomes/Bacteria/

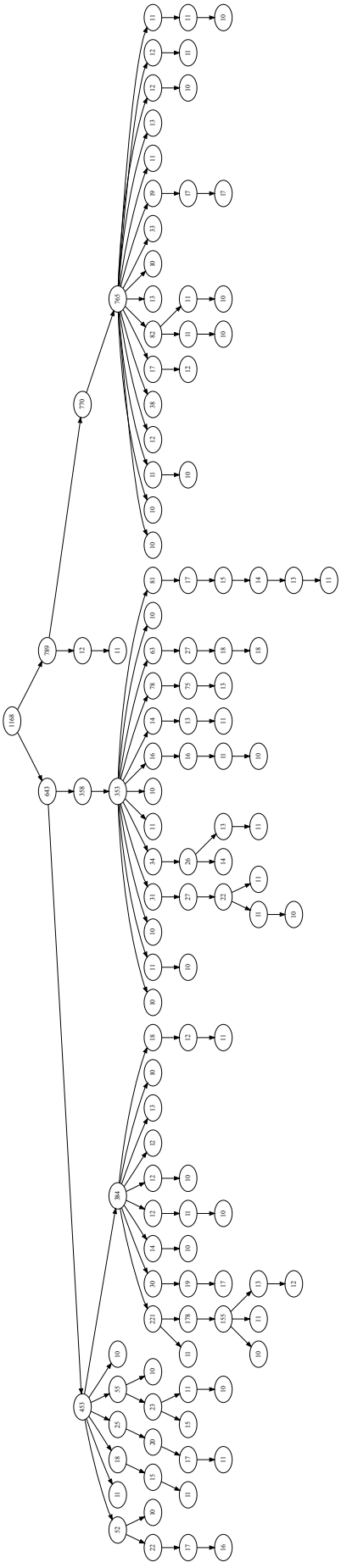

Fig. 3.2: GTT for *E. coli* K-12 and *B. subtilis* showing gene teams with at least 10 families. The number in each node of the tree represents the number of families in the corresponding gene team.

An *operon* consists of several genes that are transcribed into a single polycistronic mRNA [Jacob et al., 1960]. The requirement that these genes be transcribed into a single strand of mRNA forces them to be in close proximity to one another. Operons form one possible explanation for the existence of conserved gene clusters in prokaryotes. Hence, we make use of the manually curated database of *E. coli* K-12 operons from RegulonDB [Gama-Castro et al., 2008] to validate the computed teams.

We consider an operon to be "identified" if its Jaccard score is above a certain threshold. Figure 3.3 shows the number of identified operons for different values of the threshold. We observe that many operons are identified at Jaccard score of 1/3, 1/2, and 2/3. This can be seen in Figure 3.3 as a sudden decrease in the number of identified operons at these Jaccard score thresholds. These values are possible candidates for the threshold. We felt that 1/3 and 1/2 were too low and would allow all operons with two genes to be identified since at the lowest δ we obtain gene teams of a single gene. Therefore, we decided to use a threshold of 2/3. Using a threshold of 2/3 also solves the problem of double counting as one operon can only be identified based on one gene team.

There are a total of 836 *E. coli* K-12 operons in RegulonDB with at least two genes. Since only a subset of the *E. coli* K-12 genes is considered in this study, we consider how many operons can be identified based on these set of genes. We consider an operon to be "identifiable" if at least 2/3 of its genes is in our subset of *E. coli* K-12 genes; there are 322 such operons. By considering gene teams for all values of δ, we identified 168 (52.2%) out of 322 operons.

In subsequent analyses, we focus our attention on these 168 operons. Recall that each team is associated with a range of δ values, *i.e.*, $[\delta_{\min}, \delta_{\max}]$. For each of the 168 identified operons, we considered the range of δ values it is identified. This is obtained by combining the ranges of all teams with a Jaccard coefficient of at least 2/3 with the operon. Given this list of ranges, the operons identified by a particular value of δ are those whose ranges contain δ.

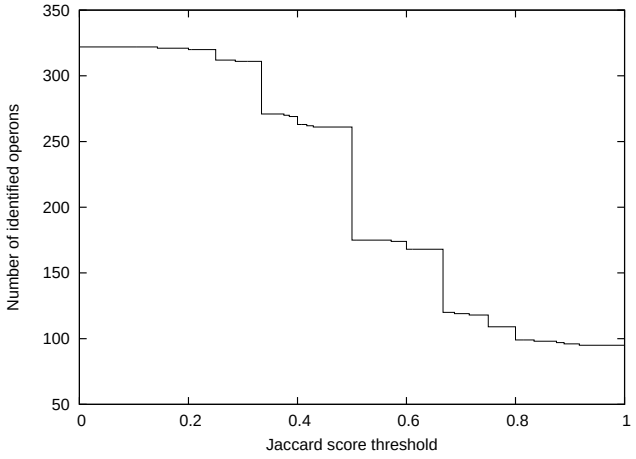

Fig. 3.3: Number of identified operons for different values of the Jaccard score threshold.

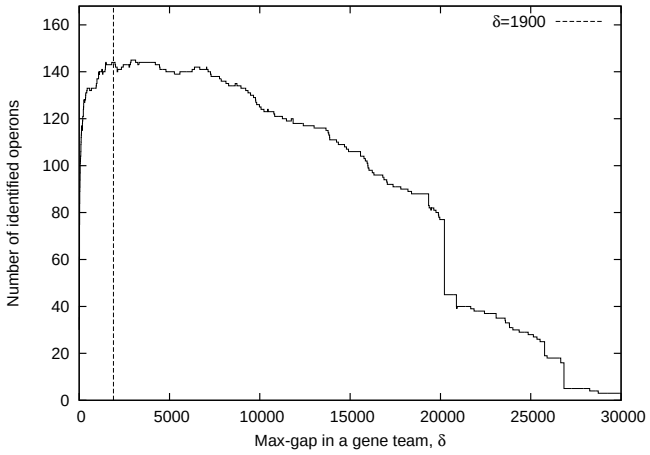

Fig. 3.4: Number of identified operons for different values of the max-gap parameter, δ. The dashed line indicates the value of δ suggested in He and Goldwasser [2005] for this dataset.

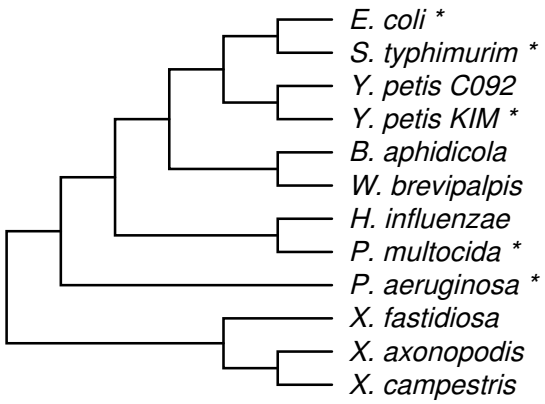

Fig. 3.5: Phylogeny of the gamma-proteobacteria from Lerat et al. [2003]. Marked species are included in our study.

Figure 3.4 shows the number of identified operons for various values of δ. Out of the 168 operons, a maximum of 145 (86.3%) can be identified using a single value of δ. Note that setting δ to be 1900, as suggested in He and Goldwasser [2005], identifies 144 operons, which is one less than the maximum.

We conclude that gene teams for single value of δ cannot model all *E. coli* K-12 operons present in the comparison of *E. coli* K-12 and *B. subtilis*. By considering gene teams for all values of δ, we were able to identify an additional 23 operons.

3.4.2 Gamma-Proteobacteria Dataset

In this experiment, we investigate the scalability of our GTT algorithm and how well gene team models *E. coli* K-12 operons conserved across multiple species.

The dataset we are using is a subset of the one used in Lerat et al. [2003]. It consists of the following five genomes downloaded from the NCBI Microbial Genomes Resource:

- *E. coli* K-12 (EC, GenBank accession number NC000913)
- *S. typhimurium* (ST, NC003197)
- *Y. pestis* KIM (YPK, NC004088)
- *P. multocida* (PM, NC002663)

Table 3.2: Sizes of the input and output for the five datasets and their running time (denoted by t).

	#genes from a common gene family						
$\mathcal{G}$	EC	ST	YPK	PM	PA	$\lvert\mathcal{T}\rvert$	t (s)
(EC)	3558	-	-	-	-	3376	3
(EC,ST)	3255	3304	-	-	-	6783	4
(EC,ST,YPK)	2876	2911	2733	-	-	30484	8
(EC,ST,YPK,PM)	2175	2214	2021	1535	-	25110	10
(EC,ST,YPK,PM,PA)	1951	1974	1802	1343	2542	821048	209

- *P. aeruginosa* (PA, NC002516)

As in the previous section, we included *E. coli* K-12 so we can validate the computed teams against the *E. coli* K-12 operons from RegulonDB. The other four species were selected to provide a range of phylogenetic distance from *E. coli* K-12. We excluded the branch containing *B. aphidicola* and *W. brevipalpis* due to extensive gene loss along this branch.

Based on the phylogeny of the gamma-proteobacteria suggested in Lerat et al. [2003] (see Figure 3.5), we computed the GTT for the following combinations of input gene orders: (EC), (EC, ST), (EC, ST, YPK), (EC, ST, YPK, PM), and (EC, ST, YPK, PM, PA). Each tuple of gene orders corresponds to genomes in a particular subtree of the phylogeny. Table 3.2 summarizes the sizes of the different combinations of gene orders and the number of nodes in the GTT.

Note that as an optimization, we discarded teams which consists solely of genes from a single family. Without this optimization, the last dataset which consists of five gene orders could not run to completion due to lack of memory.

As shown in Table 3.2, the number of genes in each gene order decreases as more genomes are considered due to a reduction in the number of common gene families. Despite this phenomenon, adding more genomes result in an increase in the size of the GTT and a corresponding increase in the running time. This combinatorial explosion in the size of the GTT is a major limiting factor of our approach. In this case, we were only able to scale our method up to a maximum of five genomes.

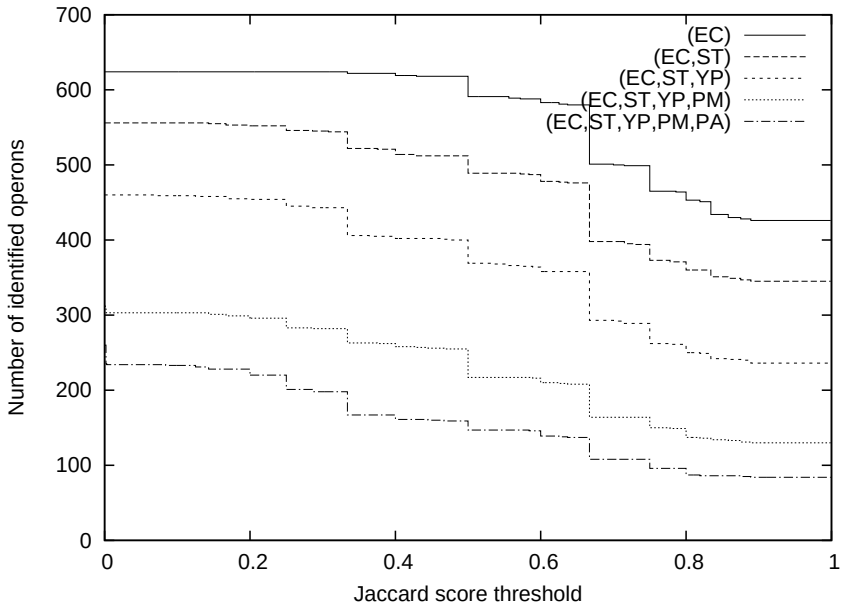

Fig. 3.6: Number of identified operons for different values of the Jaccard score threshold for each of the five input tuples.

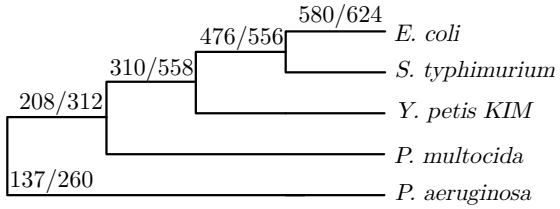

Fig. 3.7: Number of identified operons over number of identifiable operons for each of the five input tuples based on a Jaccard score threshold of 2/3.

Using the same methodology as in the previous section, we compared the GTTs for each of the five input tuples against the set of *E. coli* K-12 operons from RegulonDB using a minimum Jaccard score threshold. Figure 3.6 shows the number of identified operons for different values of the threshold, it shows a general trend that is similar to Figure 3.3. In particular, in this figure, we observe the effect of increasing the number of gene orders considered. Adding more gene orders increases the significance of the gene teams but because we require a gene team to exist in all gene orders, it also reduces the number of operons that can be identified.

Using a Jaccard score threshold of 2/3, we computed the number of identified operons (those with Jaccard score of at least 2/3) and the number of identifiable operons (those with at least 2/3 of its genes in the subset of *E. coli* K-12 genes) for

each of the five input tuples. Figure 3.7 presents this information together with the phylogeny of these five species. Gene teams that are present in many species are more significant as they show greater conservation. However, as more species are added, the number of common gene families decreases. There is a trade-off between improving the significance of clusters using multiple species and the loss of signal that results from only considering gene families that are common to all species.

3.4.3 Human and Mouse Dataset

In Section 3.3.8, we proposed a simple method for applying our method to multi-chromosomal genomes. In this section, we demonstrate the practicality of the method using the human and mouse dataset.

We downloaded the human and mouse genomes from the MSOAR [Fu et al., 2007] website[2] and formed gene families using the MSOAR hit graph. There are 12662 common gene families, with 13965 genes in the human genome and 14195 genes in the mouse genome.

We computed the GTT for these two genomes using the method for handling multi-chromosomal genomes described in Section 3.3.8. The computation took approximately 9 seconds to generate a tree with 34490 nodes. A portion of the GTT with showing gene teams from chromosome X is illustrated in Figure 3.8.

[2]http://msoar.cs.ucr.edu/

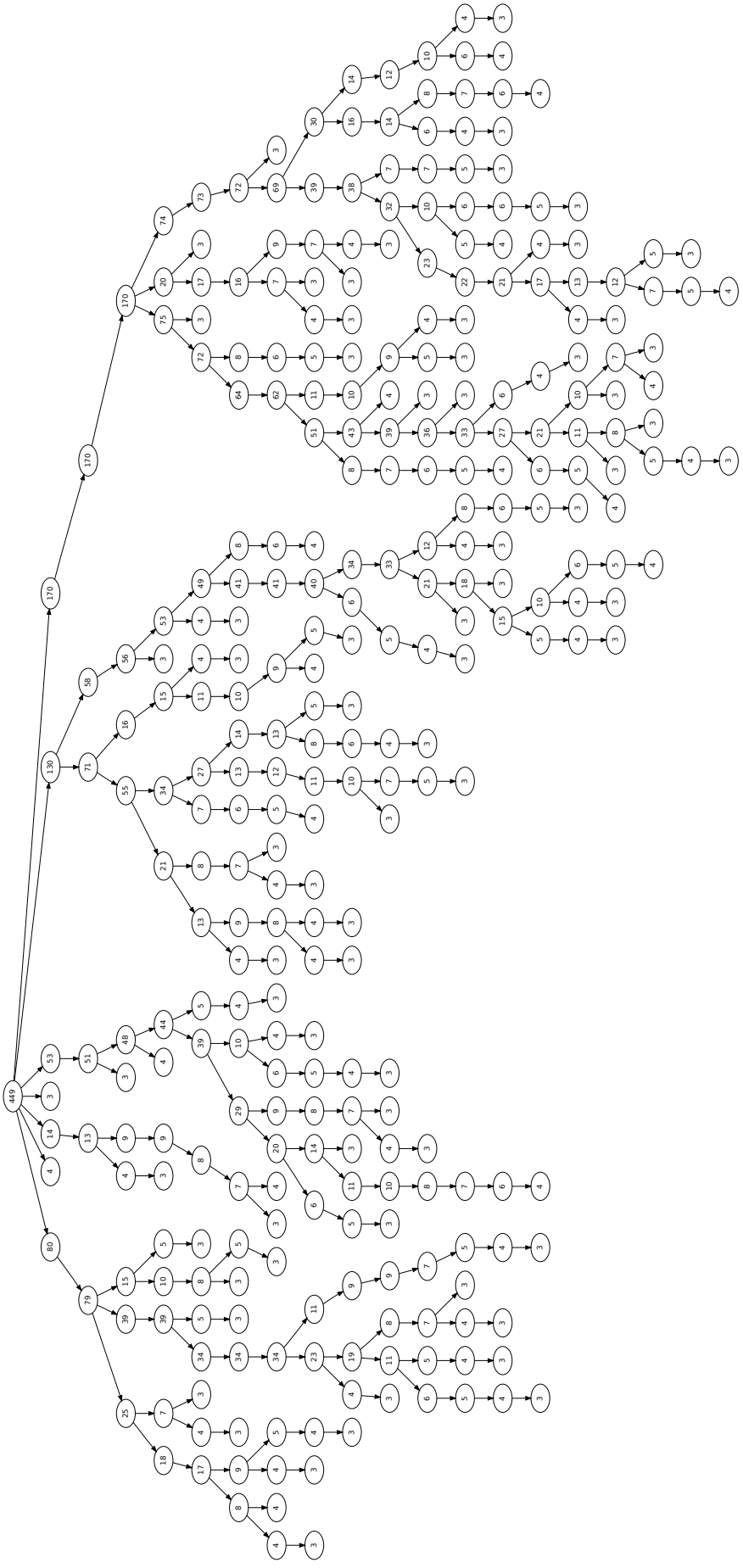

Fig. 3.8: Subtree of the GTT for human and mouse genomes containing genes from chromosome X. Due to space limitations, only gene teams with at least 3 families are shown.

Ling et al. [2008] have shown that gene teams can be used for finding synteny blocks (homologous regions sharing the same set of genes) between the human and mouse genomes. Their analysis is based on two specific values of δ, namely 300K (bp) and 1M (bp). Computing the GTT for these two genomes, allows us to generate the teams for any value of δ efficiently, by performing an intersection query on the range of δ values for each team. In effect, we can think of computing the GTT as a preprocessing step, which allows us to perform efficient analysis over a range of δ values.

3.5 Conclusion and Extensions

The gene team tree represents the set of gene teams for all values of the parameter δ. Organizing the gene teams in the form of a tree allows us to visualize the relationship between the various gene teams and makes explicit the hierarchical structure of gene teams. We have developed an efficient algorithm for computing the gene team tree, whose worst case time complexity is the same as existing algorithms based on a fixed value of δ. We also showed that our algorithm is practical by running our algorithm on real datasets. Our analysis of the gene teams which correspond to known *E. coli* K-12 operons showed that considering gene teams for all values of δ improves the representational power of the model. As shown in the experiment on the gamma-proteobacteria dataset, our method is applicable to comparisons of three or more gene orders, provided the size of the GTT is reasonable.

After the work presented in this chapter was published, a faster algorithm for computing the GTT of permutations was presented in Wang and Lin [2011] based on a novel data structure, called the maximum-gap data structure. Using this data structure and enhancements to the original gene team algorithm, the new algorithm can compute the GTT for m permutations in $O(mn \lg n \lg \lg n)$ time in the worst case.

In the concluding remarks of the journal version of this work, we noted that size of the GTT depends on the number of duplicates in each gene order. Hence, we expect that the computational complexity of computing the GTT for sequences with very few duplications to be close to that of permutations. However, there appears to be a large gap between the worst case complexity of the algorithms for permutations and sequences. Based on this observation, we posed an open problem, as to whether there is an algorithm for computing the GTT whose running time is proportional to the size of the GTT.

We are happy to report that the answer is YES and algorithm is based on the improved algorithm for computing the gene team of sequences proposed in Wang et al. [2012]. They noted in their concluding remarks that one can combine their algorithm with the maximum-gap data structure to come up with an algorithm for computing the GTT for sequences that runs in time $O(S \lg n \lg \lg n)$, where S is the size of the GTT.

Chapter 4

A Constrained Max-Length Gene Cluster Model

In this chapter, we investigate a different generalization of common intervals, the max-length model. Bidirectional best hit (BBH) is a heuristic widely used in sequence comparison to identify homologous genes and conserved gene clusters are essentially homologous genomic regions. Hence, it is natural to consider using the BBH heuristic to identify conserved gene clusters. We proposed the BBH r-window (BBHRW) model that uses the BBH heuristic to eliminate the need to specify the minimum level of similarity in the r-window gene cluster model [Durand and Sankoff, 2003]. Ranking BBHRW clusters and gene teams using statistical tests, we found that BBHRW clusters have a higher level of precision at all levels of recall.

This chapter is based on Zhang and Leong [2010] and Le et al. [2011].

4.1 Motivation

While max-gap models, such as gene team [Bergeron et al., 2002], imposes a constraint on the distance between adjacent genes in a cluster, the size (number of genes) and length (distance between the two furthest genes) of the resulting clusters are unbounded. In contrast, under max-length models, such as r-window

[Durand and Sankoff, 2003], clusters have length at most r and contains at least k genes.

Furthermore, it is unclear which combinatorial model is a better representation of conserved gene clusters as there are few results that compares different models empirically on real genomes. Hoberman and Durand [2005] laid the groundwork by providing a characterization of the desirable properties of gene clusters and a detailed analysis of the difference between max-gap clusters based on the gene team model and those produced by heuristics.

The r-window model was first applied to the study of block duplications within a genome by comparing all pairs of windows of length r [Friedman and Hughes, 2001, Cavalcanti et al., 2003]. The statistical properties of the r-window model are well understood and the significance of discovered clusters can be evaluated using statistical tests [Durand and Sankoff, 2003]. In contrast, exact computation of the significance of gene teams is still an open problem, though upper and lower bounds are known [Hoberman et al., 2005].

Motivated by the popular bidirectional best hit heuristic [Moreno-Hagelsieb and Latimer, 2008] for finding corresponding genes in several species, we developed a novel gene cluster model called *bidirectional best hit r-window* (BBHRW) that combines the notion of bidirectional best hits with the r-window model in Durand and Sankoff [2003]. The key idea is to (a) capture the "frequency of common genes" in an r-window (interval of length at most r) of each genome and (b) to further strengthen it by the *bidirectional best hit* heuristic. The bidirectional best hit (BBH) constraint removes the need to specify the minimum number of shared genes in the r-window model and improves the relevance of the results. We define two variants of BBHRW using two different similarity measures to define the "frequency of common genes" in two r-windows. Then the algorithmic problem is as follows: Give two genomes of length n and m, and an integer r, compute all the BBHRW clusters.

A straight-forward algorithm for solving this problem is an $O(nm)$ algorithm

that compares all pairs of r-windows. In this chapter, we present faster algorithms (SWBST and SWOT) for solving these two BBHRW variants. Algorithm SWBST is a simpler algorithm that solves the first variant of the BBHRW, while algorithm SWOT solves both variants of the BBHRW. Both algorithms have running time $O((n+m)r \lg r)$. The algorithmic speed-up is achieved via a sliding window approach combined with efficient data structures. We also present results for the first empirical study between max-gap and max-length clusters.

4.2 The BBH r-window Gene Cluster Mining Problem

In this section, we introduce a number of relevant definitions and formulate the computational problem of finding all BBHRW clusters.

The main feature of max-length models, such as BBHRW, is the maximum length of a gene cluster. This is imposed by only considering windows of length at most r on a genome.

Definition 17 (r-window)**.** An *r-window* of a gene order G is a substring of G where the distance between the first and last gene is at most r.

In Durand and Sankoff [2003], a gene cluster is defined as a set of k genes that are found in a r-window. The parameter k is a constraint on the number of common genes between two r-windows. However, it is unclear how to determine the minimum number of genes in a gene cluster as it depends on the length of the clusters and the evolutionary distance between the genomes. Too low a value will introduce too many false positives, while a more conservative value may exclude weakly similar clusters.

In this chapter, we adopt a different constraint on the r-window gene cluster, namely, bidirectional best hit. This overcomes the problem having to decide the number of common genes in a cluster by making use of the relative similarities between the r-windows. The bidirectional best hit (BBH) heuristic is routinely used

when identifying homologous DNA sequences between two species using BLAST. It is natural to extend this heuristic to the identification of conserved gene clusters, which are essentially homologous chromosomal segments.

For a given r-window w_G of a gene order G and a specific similarity measure, the r-windows of another gene order H that is the most similar to w_G are called the *best hits* of w_G.

Definition 18 (Best hits of a r-window)**.** Given an r-window w_G on a gene order G, a gene order H, and a similarity measure sim, the *best hits* of w_G are the r-windows w_H of H that satisfies

$$\text{sim}(w_G, w_H) > \text{sim}(w_G, w'_H) \text{ for all } w'_H \text{ which does not contain } w_H$$

Depending on the definition of the similarity measure, it is possible for a r-window w_G to have multiple hits with the same similarity. In the case where one of these r-window contains the other, we prefer the minimal one. That is why in the above definition, we require that w'_H does not contain w_H.

Bidirectional best hits are simply pairs of r-windows that are each other's best hit. Hence, we define bidirectional best hits as follows:

Definition 19 (BBH r-window gene cluster)**.** Given two gene orders G and H, a maximum window length r, and a similarity measure sim, a pair of r-windows (w_G, w_H) is a bidirectional best hit r-window cluster if and only if w_G is the only best hit of w_H with respect to sim and w_H is the only best hit of w_G with respect to sim.

The corresponding computational problem is to find all BBHRW gene clusters.

Definition 20 (BBH r-window gene cluster mining problem)**.** Given two gene orders, G and H, a similarity measure, sim, and a maximum window length, r, compute the set of all BBHRW clusters.

A naïve algorithm for the above problem is compare all r-windows of gene order G against all r-windows of gene order H. This is a general algorithm since

it does not depend on the form of the similarity measure, but it requires at least quadratic time. In the following, we consider two specific similarity measures and exploit the specific properties of the similarity measures to design more efficient algorithms for computing all BBHRW clusters.

4.3 A Generic Algorithmic Framework for BBH r-window Gene Cluster Mining

Our algorithms for computing BBHRW clusters for two gene orders G and H consists of three main steps:

1. computing best hits of r-windows in G
2. computing best hits of r-windows in H
3. keeping only the bidirectional best hits

We first find the best hits from G to H and vice versa, then filter the results to only retain the bidirectional best hits. We store the best hits from H to G in a hash table. For each best hit from G to H, we access the table to check if it is also a best hit from H to G. The pseudocode for the algorithm is shown in Algorithm 4. The most time-consuming step is computing the best hits. In the following, we will focus on how we compute the best hits of r-windows in G.

4.3.1 Finding best hits with a sliding window algorithm

We observe that for a given r-window w_G in G, most of the r-windows in H do not have any genes in common with w_G since only some of the genes in H are also in w_G. Our approach is to construct a data structure to represent the gene in gene order H that are also in w_G that allows us to query for the best hit efficiently.

There are $O(nr)$ r-windows in G so building this data structure for every r-window is time-consuming. The key is to enumerate the r-windows in G in a

Algorithm 4 BBHWindows(G, H, r)

Ensure: Compute the set of BBHRW clusters between G and H
  $BBH := \emptyset$ {set of bidirectional best hits}
  $BH_{G,H} :=$ BestHitWindows(G, H, r)
  $BH_{H,G} :=$ BestHitWindows(H, G, r)
  {Store the best hits from H to G in a hash table M}
  for each (w_H, w_G) in $BH_{H,G}$ **do**
    $M[w_H] := w_G$
  end for
  {Compute the bidirectional best hits}
  for each (w_G, w_H) in $BH_{G,H}$ **do**
    if $M[w_H] = w_G$ **then**
      $BBH := BBH \cup \{(w_G, w_H)\}$
    end if
  end for
  return BBH

specific order. For every position in G, we consider the r-windows that starts at that position in increasing length. This is because the r-window of length l that starts at position i is just the r-window of length $l-1$ starting at position i with an additional gene. Suppose we have computed the data structure and found the best hit, the next r-window is the previous one with an additional gene. This means that we can update our data structure by inserting the additional genes instead of computing it from scratch.

We enumerate the r-windows in G by starting from each gene and incrementally add genes in increasing order of their position as long as the window length is less than or equal to r. We use a data structure T to represent the set of genes in gene order H that are also in w_G. Each time we consider a different window w_G, we need to update our data structure by adding the corresponding genes in H to our data structure. To determine the list of genes to be added, we preprocess H to compute the list of genes for each homology family. Finally, for each window w_G, we make use of our data structure to determine the best hit in H.

The pseudo code for this algorithm is shown in Algorithm 5.

Algorithm 5 BestHitWindows(G, H, r)

Ensure: Determine for each r-window in G the best hit in H
  $BH := \emptyset$ {set of best hits from G to H}
  {Determine list of genes for each family in H and store in gs}
  for i from 1 to n_H **do**
    {h_i is the ith gene in H}
    $f_i := \text{fam}(h_i)$
    $gs[f_i] := gs[f_i] \cup \{h_i\}$
  end for
  {Enumerate r-windows in G and compute best hits}
  for i from 1 to n_G **do**
    {g_i is the ith gene in G}
    $e := i - 1$
    $w_G := \emptyset$
    initialize data structure T
    while $\Delta(g_i, g_{e+1}) \leq r$ **do**
      $e := e + 1$
      $w_G := w_G \cup \{g_e\}$
      $f_e := \text{fam}(g_e)$
      for each gene $h \in gs[f_e]$ **do**
        insert(T, h)
      end for
      $w_H := \text{besthit}(T)$
      $BH := BH \cup \{(w_G, w_H)\}$
    end while
  end for
  return BH

4.4 BBHRW using similarity measure COUNT

4.4.1 Similarity measure COUNT

The first similarity measure we consider is COUNT. $\text{COUNT}(w_G, w_H)$ is the number of genes in w_H that are also in w_G.

Definition 21 (Similarity measure COUNT)**.**

$$\text{COUNT}(w_G, w_H) = |\{i \mid \text{fam}(w_H[i]) \in \text{CS}(w_G)\}|$$

where $\text{CS}(w_G)$ is the character set of w_G.

This is an asymmetric function since we consider all the genes in w_H but treat genes in w_G as a set. For example, $\text{COUNT}(\langle a, b, c\rangle, \langle a, a, b\rangle)$ is 3 because there are two as and one b in $\langle a, a, b\rangle$. In contrast, $\text{COUNT}(\langle a, a, b\rangle, \langle a, b, c\rangle)$ is 2 since c is not in $\langle a, a, b\rangle$.

Example. Consider the following two gene orders,

$$G = \left\langle a^1, b^3, c^5, d^6, e^9, c^{10}, b^{11}, a^{12}, b^{13}\right\rangle$$
$$H = \left\langle c^1, a^3, d^4, b^6, e^7, b^8, c^9, a^{11}, d^{12}\right\rangle$$

where the letters represent homology families and the superscripts denote the position.

The non-singleton BBH 3-window gene clusters of G and H are $(\langle c^5, d^6\rangle, \langle c^1, a^3, d^4\rangle)$ and $(\langle e^9, c^{10}, b^{11}\rangle, \langle b^6, e^7, b^8, c^9\rangle)$

4.4.2 Algorithm SWBST

Algorithm SWBST (Sliding Window with Binary Search Tree) is based on the generic sliding window framework and an augmented binary search tree data structure for finding best hits based on COUNT.

Each node in the tree is a gene in H that is also in the current r-window. We order the nodes according to their position in H. We augment each node u, with $s = \text{COUNT}(w_G, H[p, p + r - 1])$, where p is the position of u, and smax, the maximum value of s in the subtree rooted at u. This gives us a simple algorithm to find the best hit by starting from the root and following the node with the largest smax. If we find more than one node with the largest smax than we can stop, since the best hit is not unique.

The main difficulty is in updating our binary search tree when we want to add an additional gene. Since our data structure only considers genes in H that are also in w_G, $\text{COUNT}(w_G, H[p, p + r - 1])$ is also the number of nodes in the tree whose position is in $[p, p + r - 1]]$. In order to add a new node at position p, we first compute the value of s for the new node by counting the number of nodes in the tree whose position is in $[p, p + r - 1]$. Secondly, we increase the value of s by 1 for those nodes in the tree whose position is in $[p - r + 1, p]$. In both cases, we need to deal with nodes in a certain range. This, we augment each node u with the range of positions of the genes in its subtree (pmin and pmax) and a modifier (d) to indicates the increase to the value of s in all nodes in this subtree. These allows us to execute range queries and updates efficiently. The details are given in the following sections.

In summary, the algorithm for insertion consist of the following three steps:

1. compute the similarity measure s of the new node using a range query to count the number of nodes in a given interval

2. update the nodes affected by the new node using a range update

3. insert the new node into the binary search tree

Range query and update algorithm

We make use of the range query algorithm (Algorithm 6) to compute the result of a query or to update the similarity measure in an interval. Figure 4.1

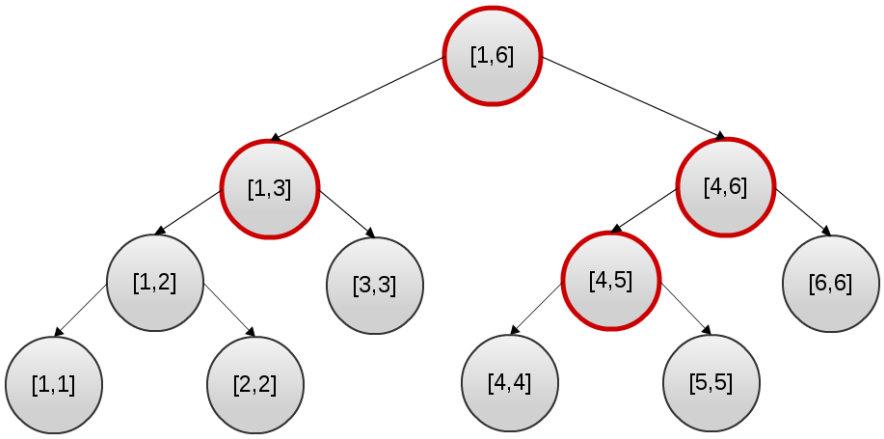

Fig. 4.1: The nodes with bold outline are visited by Algorithm 6 during a range query on the interval $[1, 5]$.

shows a possible tree and the nodes visited during a range query operation. Let ProcessSubtree be the procedure to query/update the subtree rooted at u, ProcessNode be the procedure to query/update the node u, and CombineSubtree be the procedure to combine the results from the two subtrees. We use this algorithm to compute the similarity measure s for the new node and to update the data structure.

Algorithm 6 Algorithm to query/update nodes in a given interval

Require: a node u in the tree, an interval $[s, e]$ to query/update
Ensure: update the subtree rooted at u and return the answer to the query
  if position of u is in $[s, e]$ **then**
    ProcessNode: Query/update the attributes in this node
  else if interval of u lies inside $[s, e]$ **then**
    ProcessSubtree: Query/update the attributes in this node
  else
    if interval of the left child of u intersects with $[s, e]$ **then**
      Recursive call to the left child
    end if
    if interval of the right child of u intersects with $[s, e]$ **then**
      Recursive call to the right child
    end if
    CombineSubtree: Combine the result of the recursive calls and/or update the node to maintain consistency
  end if

Our structure is based on the interval decomposition idea [Bentley, 1977]. Therefore, the worst case time complexity of the range query/update is $O(\lg |T|)$

where $|T|$ is the size of the tree.

Computing the similarity measure of the new node

Computing the similarity measure of the new node makes use of Algorithm 6 and the corresponding procedures are as follows:

ProcessSubtree: return the number of nodes in the subtree.

ProcessNode: add 1 to the result for this node.

CombineSubtree: return the result plus the sum of the recursive calls.

Updating the similarity measure of affected nodes

Insertion of the new node increased the similarity measure for nodes in the interval $[p - r + 1, p]$ by 1, where p is the new node's position. Updating the similarity measure for affected nodes requires a range update.

ProcessSubtree: increase d by 1 to indicate that the value of s in the subtree has increased by 1 without actually changing s in each node.

ProcessNode: increase s by 1.

CombineSubtree: update the smax according to the value of smax and d in the left and right subtrees.

Inserting the new node into the tree

Insertion a node follows the same procedure as a standard binary search tree. In case where rotations might be needed to keep the tree balanced, the augmented attributes of a node are updated using the attributes of its subtrees.

4.4.3 Time complexity analysis of algorithm SWBST

For simplicity, we assume that the number of genes from a gene family is bounded by a constant. In the worst case, the size of our binary search tree is $O(r)$. Hence,

all operations on the tree have a worst case time complexity of $O(\lg r)$ where r is the maximum window length.

Using the above data structure, the worst case time complexity to compute best hits for all r-windows with the same starting position in G is $O(r \lg r)$. We have to compute best hits twice: once from G to H which takes $O(nr \lg r)$ time and once from H to G which takes $O(mr \lg r)$. Hence, the worst case time complexity of the entire algorithm is $O((n+m)r \lg r)$ where n and m are the length of G and H.

4.4.4 Results and discussion

In order to determine the practicality of the BBHRW (COUNT) model, we evaluated how well the gene clusters between *E. coli* K-12 and *B. subtilis* corresponds to known *E. coli* K-12 operons.

We make use of the *E. coli* K-12 and *B. subtilis* dataset described earlier in Section 3.4.1. After some preprocessing, we obtain two gene orders of 2250 genes (*E. coli* K-12) and 2364 genes (*B. subtilis*) from 1168 gene families. Gene positions are assigned based on the index of the gene in the complete genome, thus the distance between two genes represents the number of intervening genes, including those genes that are not found in both genomes.

We implemented algorithm SWBST, which finds all BBHRW (COUNT) clusters between two gene orders, in Java. All computations were performed on a Intel Core 2 Duo E6550 (2.33GHz) processor with 2GB of RAM running Linux.

We computed the BBHRW (COUNT) clusters between these two genomes and compared our results against known *E. coli* K-12 operons from RegulonDB [Gama-Castro et al., 2008]. As it is difficult to obtain an exact match, we compute the Jaccard score (Definition 16) for each operon and we consider an operon to be identified if its Jaccard score is above a certain minimum Jaccard score threshold. Figure 4.2 shows the number of identified operons for different values of the threshold when the maximum window length is 6.

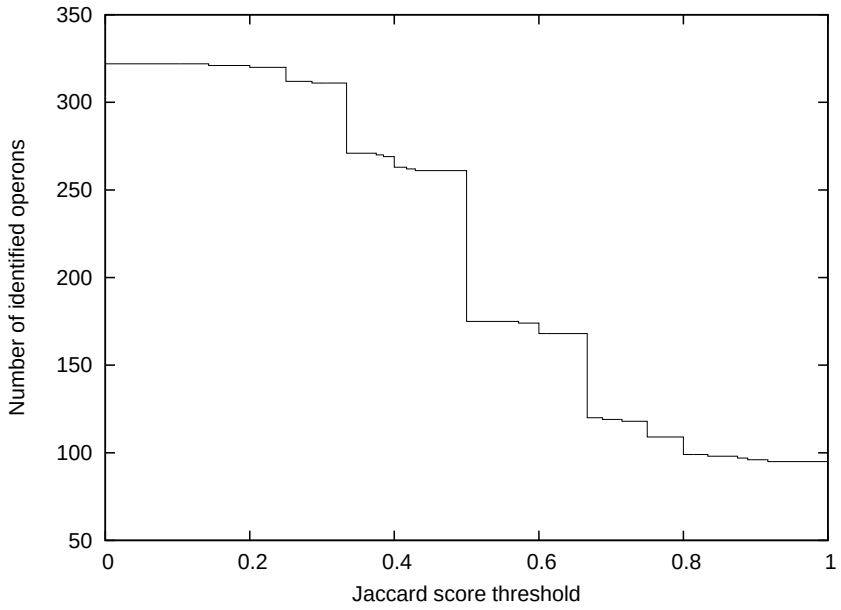

Fig. 4.2: Number of identified operons versus Jaccard score threshold for BBHRW (COUNT, $r = 6$) clusters.

Based on Figure 4.2, we chose a value of 2/3 for the threshold. There are 253 operons with at least two genes and at least 2/3 of its genes are common to both *E. coli* K-12 and *B. subtilis*. This is an upper bound on the number of operons that can be identified based on the input.

Effect of window length

Our gene cluster model has a single parameter r, which is the maximum length of a window. A natural question that arises is the effect of this parameter on the resulting clusters. We ran our algorithm for a range of window lengths from 1 to 30, each run took approximately 8 seconds to complete.

As shown in Figure 4.3, the percentage of identified operons increases as the window length increases from 1 to 6 and decreases for larger values of r. At the peak, when the maximum window length is 6, our method identified 34% of the operons (85 out of 253). It is interesting to note that there is a core of about 60 operons that are identified across the entire range of the parameter r. The dashed line shows the number of BBHRW (COUNT) clusters that are not matched to any operon; it ranges between 70% to 80%. This illustrates the difficulty of using only

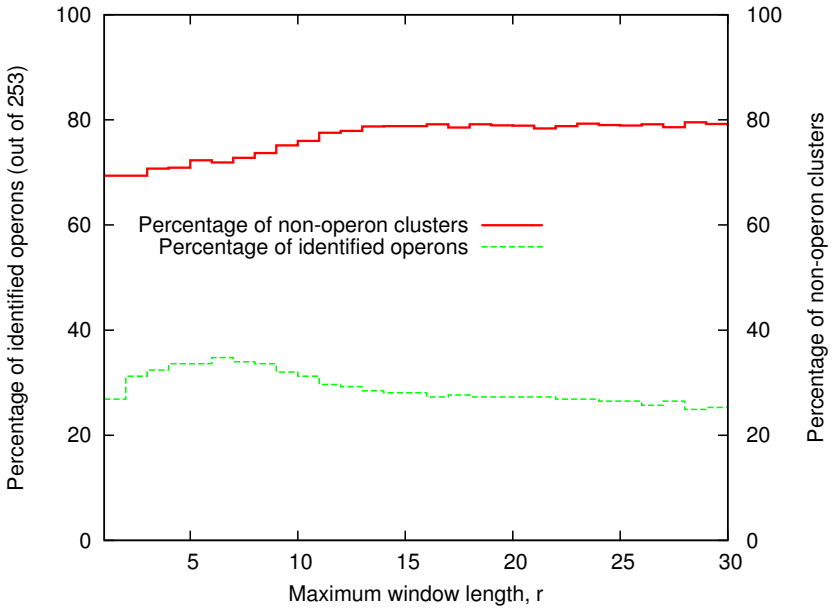

Fig. 4.3: Percentage of identified operons and percentage of non-operon clusters versus maximum window length for the BBHRW (COUNT) model.

Table 4.1: Significant BBHRW (COUNT) clusters and corresponding operons. Nine out of the top twelve based on $\log E$ value and corresponding operons. Numbers in brackets indicate number of genes in the cluster over number of genes in the operon.

$\log \mathbf{E}$	**BBHRW (count) cluster**	**Operon**
−13	atpC,atpD,atpG,atpA,atpH,atpF,atpE,atpB	atpIBEFHAGDC (8/8)
−12	secE,nusG,rplK,rplA,rplJ,rplL,rpoB,rpoC	secE-nusG (2/2), rplKAJL-rpoBC (6/6)
−11	hisG,hisD,hisB,hisH,hisA,hisF,hisI	hisLGDCBHAFI (7/8)
−10	fliE,fliF,fliG,fliH,fliI,fliJ,fliK	fliFGHIJK (7/6)
−9	menE,menC,menB,yfbB,menD,menF	menFD-yfbB-menBCE (6/6)
−9	rbsD,rbsA,rbsC,rbsB,rbsK,rbsR	rbsDACBKR (6/6)
−8	pnp,rpsO,truB,rbfA,infB,nusA,yhbC	metY-yhbC-nusA-infB-rbfA-truB-rpsO-pnp (7/7)
−8	dppF,dppD,dppC,dppB,dppA,yhjX	dppABCDF (6/5)
−7	oppA,oppB,oppC,oppD,oppF	oppABCDF (5/5)

spatial information to distinguish between the two kinds of genes clusters: those that are due to the evolutionary proximity of the two species and those that are under selective pressure.

Analysis of significant clusters

Table 4.1 shows nine of the top twelve BBHRW (COUNT) clusters in increasing order of $\log E$, the logarithm of the expected number of clusters between two random genomes.

Most of the clusters are an exact match to a specific operon, except for the second cluster which consists of a combination of two operons. Two of our clusters

contains an additional gene that is not part of the operon.

The fliE-K cluster includes the additional fliE gene that is not part of the fliF operon. The fliE gene is known to be a monocistronic transcriptional unit that is adjacent to the fliF operon, it forms part of the flagellar of *E. coli* K-12 together with the fliF operon [Muller et al., 1992]. This evidence supports our grouping of fliE together with the fliF operon.

The cluster matched to the dpp operon contains an addition yhjX gene. yhjX is a hypothetical protein with an unknown function predicted to be a transporter [Rudd, 2000]. This prediction gives yhjX a similar function as the dpp operon, which function as a dipeptide transporter, and gives support to our cluster.

4.5 BBHRW using similarity measure MSINT

4.5.1 Similarity measure MSINT

The previous similarity measure COUNT is an asymmetric function. In this section, we consider a symmetric similarity measure that takes into account the multiplicity of the gene families.

We make use of the multiplicity by considering the *character multiset* (CMS) instead of character set and let the similarity measure be the size of the character multiset intersection.

Definition 22 (Similarity measure MSINT)**.**

$$\text{MSINT}(w_G, w_H) = |\text{CMS}(w_G) \cap \text{CMS}(w_H)|$$

The following illustrates the difference between COUNT and MSINT

- $\text{COUNT}(\langle a, b, a, c, f\rangle, \langle b, f, c, b, a\rangle) = 5$ and
  $\text{MSINT}(\langle a, b, a, c, f\rangle, \langle b, f, c, b, a\rangle) = 4$
- $\text{COUNT}(\langle a, b, a, c, f\rangle, \langle f, c, b, a, a\rangle) = 5$ and
  $\text{MSINT}(\langle a, b, a, c, f\rangle, \langle f, c, b, a, a\rangle) = 5$

For the same pair of gene orders, the value of COUNT is at least as large as MSINT. In the above example, $\langle b, f, c, b, a\rangle$ and $\langle f, c, b, a, a\rangle$ give the same value under COUNT, and $\langle f, c, b, a, a\rangle$ has a higher value compared to $\langle b, f, c, b, a\rangle$ under MSINT. Intuitively, $\langle f, c, b, a, a\rangle$ is a more similar to $\langle a, b, a, c, f\rangle$ as both of them have two copies of the gene a, whereas $\langle b, f, c, b, a\rangle$ has two copies of gene b but only a single copy of gene a. This illustrates that we can better distinguish between two gene orders when we make use of the multiplicity.

4.5.2 Algorithm SWOT

The general approach for this algorithm is the same as in algorithm SWBST. The difference is in how we compute best hits. The method for computing best hits used in algorithm SWBST cannot be extended for MSINT because the range query used for counting the number of genes in an r-window of H does not take into account the multiplicity. Thus, we need a new approach to handle the similarity measure MSINT.

The key insight is that instead of storing the similarity measure in the data structure, we store the intervals to be updated. These intervals depend on the order of insertion, so we have to insert genes from left to right. We call these intervals the *update interval* of a gene.

Definition 23. An *update interval* of a gene g is the interval where the similarity measure of every r-window starting in the interval is increased by the addition of g.

The key is to recast the problem of finding best hits to the problem of finding the position where the largest number of update intervals overlap. This position is the start of the best hit r-window. We keep track of the update intervals using the segment tree data structure.

The algorithm for inserting a new gene and computing best hits consist of three steps:

1. computing the update interval of the next gene

2. inserting the update interval and updating the segment tree

3. determining the best hit

Computing the update interval of the next gene

As discussed earlier, inserting a new gene increases the similarity measure of the r-windows containing the gene. In the case of MSINT, the similarity measure increases by 1. The following gives the formula to compute the update interval for similarity measure MSINT.

Definition 24 (Update interval for similarity measure MSINT)**.** Let H_f denote the list of genes from family f in H in increasing position, and l denote the number of genes from family f in w_G Consider a gene from family f at position p which has index k in H_f, its *update interval* is

$$[\max(p - r + 1, p' + 1), p]$$

where $\text{fam}(H[p']) = f$ has index $k - l$ in H_f. If $k - l$ is less than 1, let p' be 0.

Observe that every r-window whose start position is in the interval $[p-r+1, p]$ is affected by $H[p]$ except those r-windows that already have enough occurrence of family f. Position p' is the last position that has as many occurrences of family f as r-window w_G, therefore any position at or before p' is not affected by the insertion of a gene at position p.

Figure 4.4 illustrates the update intervals for each gene of the gene order H for a given r-window w_G. Note that this formulation of the problem is also able to handle similarity measure COUNT. Using similarity measure COUNT, the update interval for a new gene at position p is simply $[\max(1, p - r + 1), p]$.

From the above definition, we can compute the update interval of a gene in constant time.

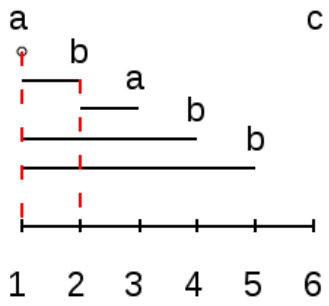

Fig. 4.4: The update intervals corresponding to each gene in $H = \langle a, b, a, b, b, c\rangle$ with the red line representing largest overlap with respect to $w_G = \langle a, b, b\rangle$ and $r = 5$. There is no update interval for gene c since it does not occur in w_G.

Inserting the update interval and updating the segment tree

Insertion of the update interval follows from standard segment tree insertion. In order to compute the best hit efficiently, we augment each node to keep track of the largest number of overlapping intervals in its subtree. This augmented value is updated during insertion.

Determining the best hit

Computing the region covered by the largest number of update intervals incrementally is the dynamic version of the standard stabbing query problem Bentley [1977]. In our case, since we know the largest number of overlapping interval in the subtree of each node, we can simply start from the root of the segment tree and follow the child that has the largest number of overlapping interval. The leaf node that we reach at the end of this process is the interval with the largest overlap and equivalently the highest similarity measure according to MSINT.

4.5.3 Time complexity analysis of algorithm SWOT

For simplicity, we assume that the number of genes from a gene family is bounded by a constant.

Consider two gene orders of length n and m respectively, pre-computing gene order H takes $O(m)$ time. Computing the update interval for each gene takes $O(1)$. The worst case time complexity of updating the segment tree and traversing the

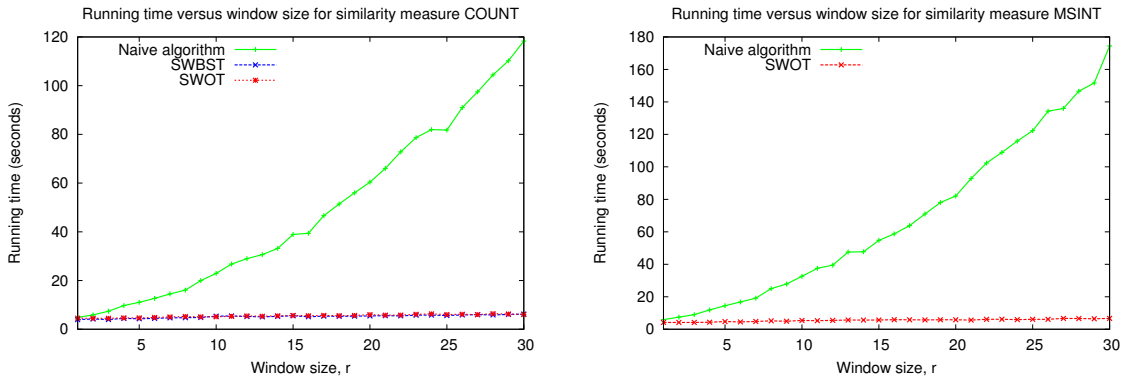

Fig. 4.5: Comparison of the running time between the naïve algorithm, algorithm SWBST and algorithm SWOT for the two similarity measures, COUNT (left) and MSINT (right). Note that algorithm SWBST cannot be used for similarity measure MSINT.

tree to find the start position of the best hit r-window is $O(\lg r)$ [Bentley, 1977] as there are at most $O(r)$ intervals in the tree.

Therefore, the worst case time complexity for computing best hits in H for every r-window of G is $O(m + nr \lg r)$, which is the same as the method for computing best hits used in algorithm SWBST. Hence, the worst case time complexity of the entire algorithm is $O((n + m)r \lg r)$ (same as algorithm SWBST).

4.5.4 Results and Discussion

In this section, we are interested in the improvement in the running time of our algorithms compared to the naïve algorithm and the difference between the two similarity measures.

We computed the BBHRW gene clusters between *E. coli* K-12 and *B. subtilis* genome (details in Section 3.4.1) using both similarity measures for r ranging from 1 to 30 (on an Intel Core 2 Duo T7300 2Ghz with 2GB of RAM running Ubuntu 10.10).

Comparison of the running time

As shown in Figure 4.5, the running time of SWBST and SWOT are almost the same when we fix the two gene orders and increase r from 1 to 30. This is expected

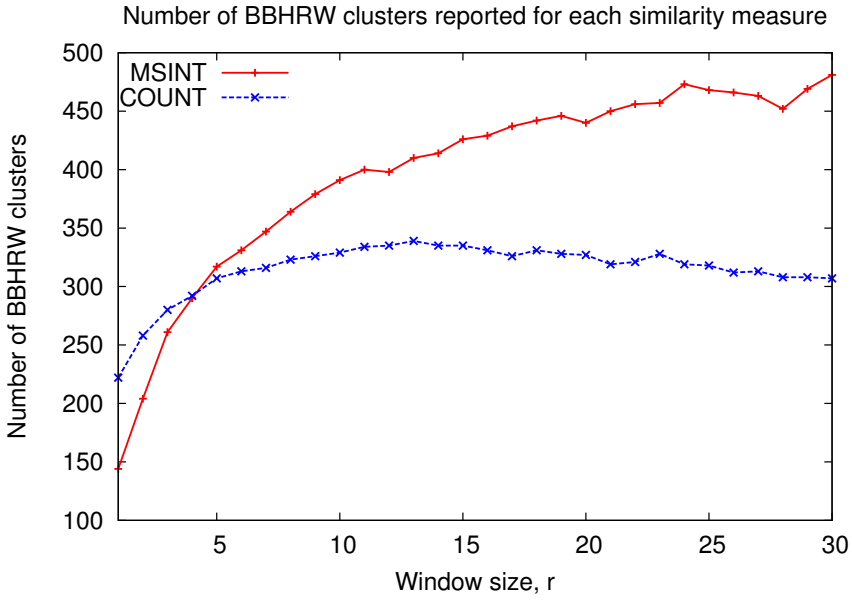

Fig. 4.6: Number of reported BBHRW clusters for both similarity measures and r varying from 1 to 30.

from their complexities. In contrast, running time of the naïve algorithm is roughly quadratic with respect to r. From this result, we conclude that our algorithms are much more scalable than the naïve algorithm.

Number of clusters reported

Figure 4.6 shows the number of clusters produced by our algorithm for the two similarity measures for r from 1 to 30. The two curves shows completely different trends. The number of results for similarity measure COUNT peak at r equal to 13 and decreases slightly as r increases to 30. In contrast, for similarity measure MSINT the number of results increases with the parameter r.

This phenomenon is due to the fact that MSINT is symmetric while COUNT is not. When the similarity measure is symmetric, it is always possible to find a bidirectional best hit when starting from any pair of r-windows. Suppose (w_G, w_H) is not a bidirectional best hit and without loss of generality assume that w_G is not the best hit of w_H. Then, there exist another r-window w'_G which is the best hit of w_H and $\text{MSINT}(w'_G, w_H)$ is greater than $\text{MSINT}(w_G, w_H)$. If (w'_G, w_H) is not a bidirectional best hit, we simply repeat the previous argument. Each time we do

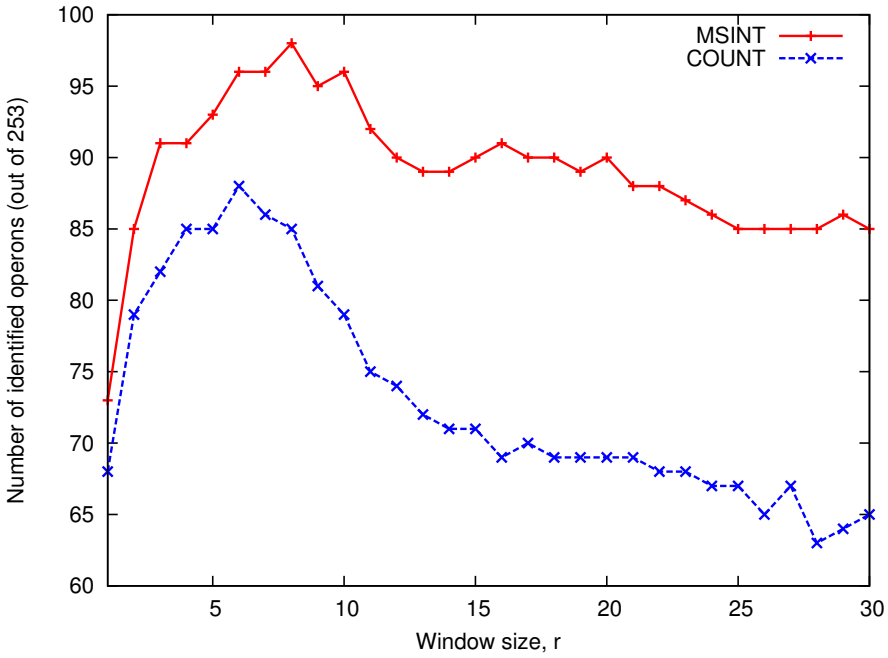

Fig. 4.7: Percentage of identified operons and percentage of non-operon clusters versus maximum window length for both variants of the BBHRW model.

this the similarity measure strictly increases. Since the similarity cannot increase indefinitely, we eventually reach a maxima and find a bidirectional best hit. The same argument cannot be applied to an asymmetric similarity measure because while $\textsc{count}(w'_G, w_H)$ is greater than $\textsc{count}(w_G, w_H)$, $\textsc{count}(w_H, w'_G)$ may be *smaller* then $\textsc{count}(w_G, w_H)$. Therefore, it is easier to find bidirectional best hits when the similarity measure is symmetric and this is reflected by the greater number of BBHRW clusters found using the similarity measure MSINT.

Number of clusters that match known operons

We used the Jaccard score cut-off of 2/3 to determine whether an operon can be identified based on the clusters reported. There are 253 operons with at least 2 genes and at least 2/3 of its genes are common to both *E. coli* K-12 and *B. subtilis*. This is an upper limit on the number of operons that can be identified.

We ran our algorithm on the above dataset, varying the window size from 1 to 30. Figure 4.7 show the difference in the number of identified operons for the two similarity measures. MSINT identified a maximum of 38.7% (98 out of 253) of the

operons for window size of 8, while COUNT identified a maximum of 34% (85 out of 253) for window size of 6. Using similarity measure MSINT, we identified more operons. In addition, MSINT is less sensitive to the choice of the window size. As the window size increase to 30, the number of operons identified by MSINT decreases but not as dramatically as for COUNT. The difference in the number of identified operons between MSINT and COUNT increases as window size increases.

The similarity measure MSINT generates a greater number of clusters and allows us to identify more operons. In order to get a more comprehensive analysis that takes into account both the number of clusters produces and the number of operons identified, we rank the clusters produced. To do this we compute the expected number of r-window gene clusters with k genes between two random genomes (equation 55 in Durand and Sankoff [2003]) and use it to rank the two sets of BBH r-window gene clusters to produce a list of ranked clusters. From these two lists, we compute the precision and recall at all possible cut-offs, where the precision is the number of identified operons divided by the number of clusters and the recall is the number of identified operons divided by the number of identifiable operons (253).

Figure 4.8 shows the precision versus recall curve for the gene clusters found using the two similarity measures. We observe that the two curves are comparable, but similarity measure MSINT has a slightly lower precision at various levels of recall but higher overall recall. This is due to the fact that the additional clusters reported using similarity MSINT do not correspond to known operons.

Based on Figure 4.8, we conclude when using the BBH heuristic, the theoretically simpler symmetric similarity measure MSINT does not perform as well as the asymmetric similarity measure COUNT. In the following section, we perform the same comparison between BBHRW (COUNT) clusters and gene teams.

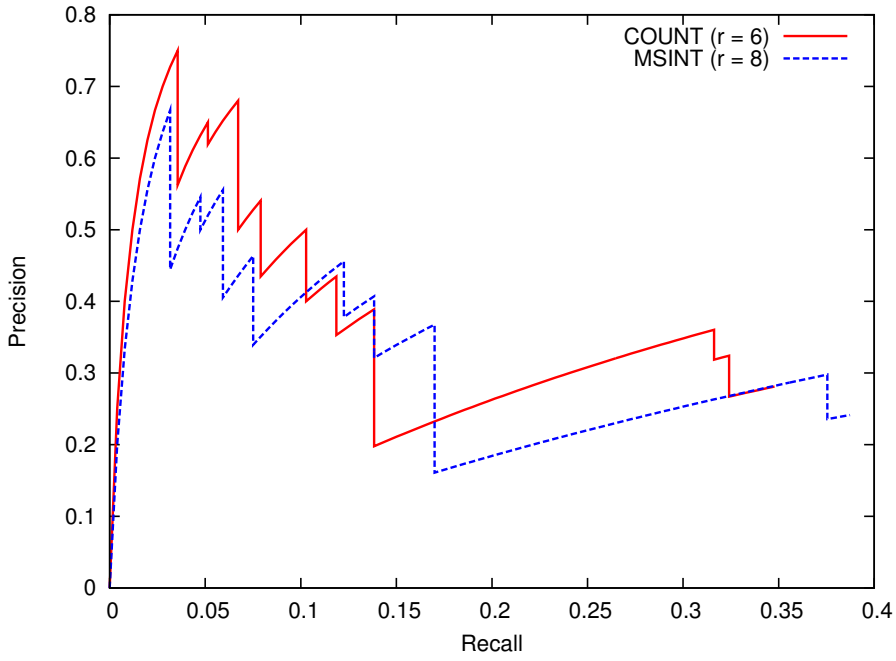

Fig. 4.8: Precision versus recall curve for BBHRW (COUNT, $r = 6$) and BBHRW (MSINT, $r = 8$) clusters for the identification of *E. coli* K-12 operons.

4.6 Comparison between BBHRW (COUNT) and Gene Team

In this section, we present the first empirical comparison between two different conserved gene cluster models that allows non-contiguous clusters.

We computed the gene team tree (Chapter 3) for the *E. coli* K-12 and *B. subtilis* dataset (ignoring singleton teams) and found that a maximum of 47% of the operons (119 out of 253) was identified when δ is 3 (see Figure 4.9). Recall that an operon is identified if its Jaccard score is more than 2/3.

This is slightly higher than the 34% achieved by BBHRW (COUNT). However, at all parameter values the percentage of non-operon teams is much higher for the gene team model. This suggests that only a small percentage of the gene teams are identified as operons, due to the property that every gene is part of some gene team. In addition, we observe that over the same range of parameter values, variation in the number of identified operons for BBHRW (COUNT) clusters is lower than that for gene teams. This means that the BBHRW (COUNT) model

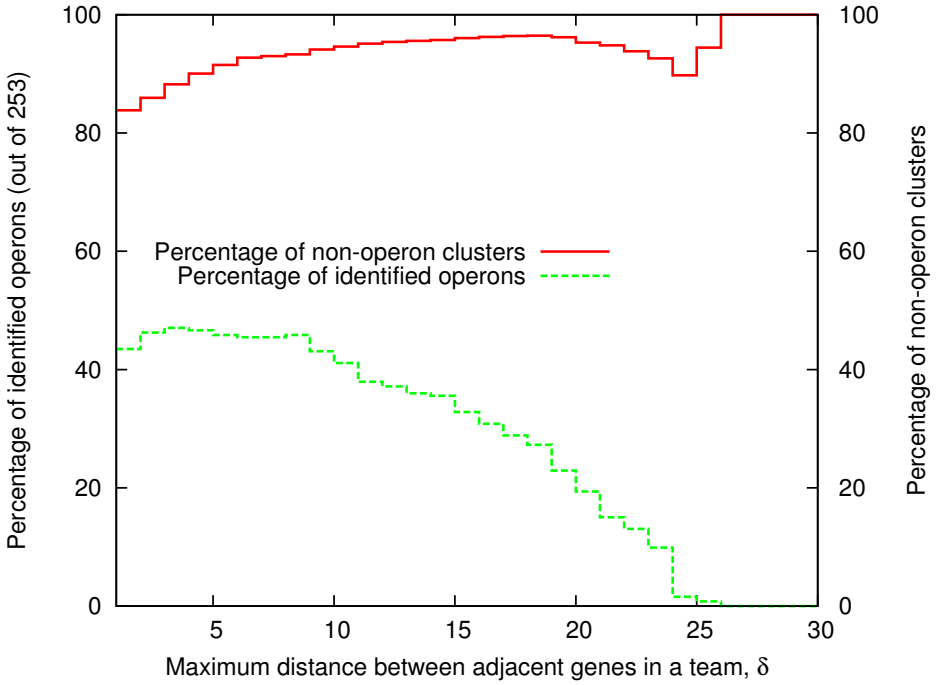

Fig. 4.9: Percentage of identified operons and percentage of non-operon clusters versus maximum distance between adjacent genes in a team for the gene team model.

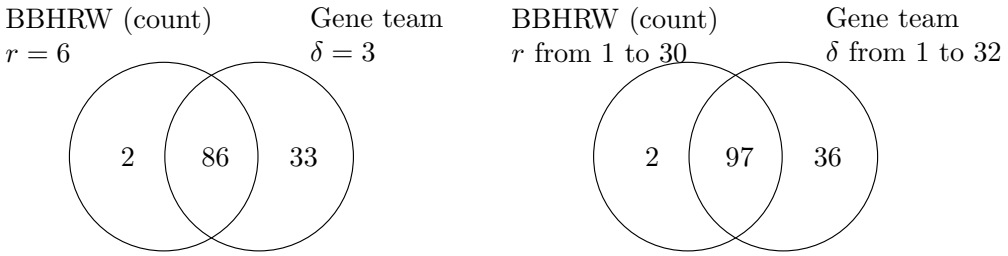

Fig. 4.10: Venn diagram showing the overlap between the operons identified based our BBHRW (COUNT) model and the gene team model for a single parameter value ($r = 6, \delta = 3$) and over a range of parameter values ($r \in [1, 30], \delta \in [1, 32]$).

is more robust to changes to the value of its parameter as compared to the gene team model.

Figure 4.10 consists of two Venn diagrams which illustrates the overlap between the operons identified by the BBHRW (COUNT) model and the gene teams model for a single value of the parameter and over a range of parameter values. Considering a range of different parameter values for both models did not significantly increase the number of identified operons as most operons can be modelled by a small range of parameters.

The operons identified using BBHRW (COUNT) are mostly a subset of the

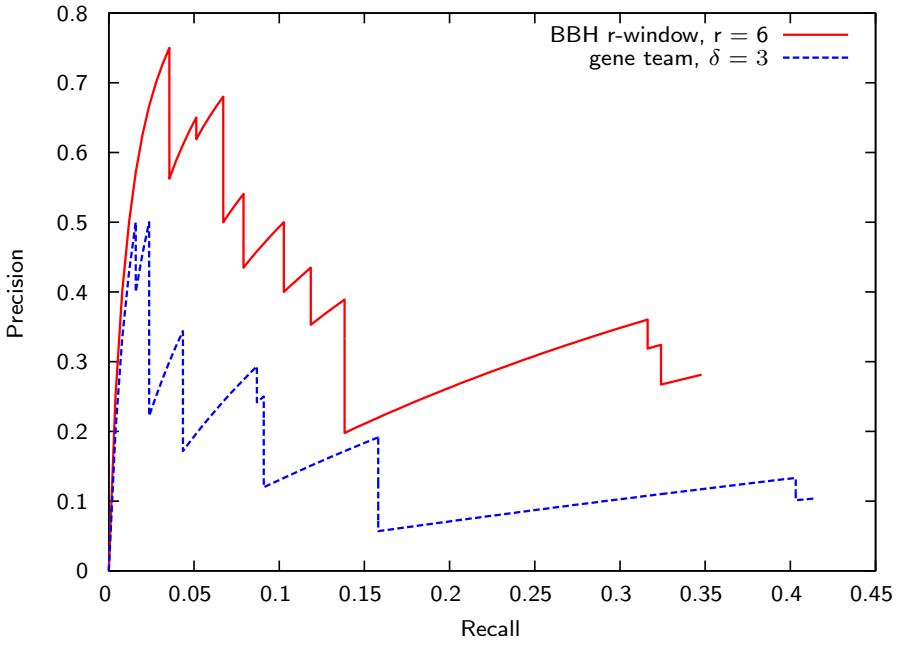

Fig. 4.11: Precision versus recall curve for BBHRW (COUNT, $r = 6$) and gene teams ($\delta = 3$) for identification of *E. coli* K-12 operons.

ones identified using the gene team model. Both models agree on a common set of 86 operons. This result is expected since the BBHRW (COUNT) model is more restrictive. Hence, the recall, which is percentage of identified operons, is lower as compared to gene teams.

However, an advantage of r-window gene clusters is the availability of exact statistical tests to evaluate the significance of putative clusters. We computed the expected number of r-window gene clusters with k genes between two random genomes (equation 55 in Durand and Sankoff [2003]) and use it to rank the BBHRW (COUNT) clusters when r is 6. We also ranked each of the gene teams when δ is 3 using the probability of forming a gene team of size k (equation 3 in He and Goldwasser [2005]). Given these two list of ranked gene clusters, we computed the precision and recall at all possible cut-offs, where the precision is the number of identified operons divided by the number of clusters and the recall is the number of identified operons divided by the number of identifiable operons (253).

Although our BBHRW (COUNT) clusters had a slightly lower recall as compared to gene teams, our model has a much higher precision. Figure 4.11 plots the

precision versus recall curve for both gene cluster models; it clearly shows that at any given recall, the precision of our model is *always* higher than the gene team model. For example, at a recall of 0.05, 65% of the BBHRW (COUNT) clusters match 5% of the identifiable operons, whereas only 20% of the gene teams match the same number of operons. Similarly, at a recall of 0.1, 50% of the BBHRW (COUNT) clusters match 10% of the identifiable operons, while only 13% of the gene teams match the same number operons.

4.7 Conclusion

In this chapter, we proposed a novel variant of the r-window gene cluster model based on the bidirectional best hit constraint. The bidirectional best hit heuristic is most commonly used for identifying families of homologous DNA sequences from BLAST hits. We extend this notion to identify homologous chromosomal segments/conserved gene clusters.

We proposed the BBHRW gene cluster model and two variants based on different similarity measures. Similarity measure MSINT is symmetric and takes into consideration the number of copies of each gene, whereas COUNT is asymmetric and based on simple counting. We designed an efficient algorithm for each variant by making use of fast data structures that support incremental updates.

In our experiments using *E. coli* K-12 and *B. subtilis* genomes, we found that BBH based on a symmetric similarity measure results in many more cluster being reported as compared to an asymmetric similarity measure. Ranking the results revealed that overall the clusters reported using the theoretically simpler, symmetric, similarity measure MSINT does not model operons as well as the asymmetric similarity measure COUNT when using the BBH heuristic.

Comparing BBHRW (COUNT) against the gene team model revealed that the operons identified by our more restrictive BBHRW (COUNT) model is a subset of the operons identified by the gene team model. However, as a result of the BBH

constraint, we were able to achieve a higher level of precision at all levels of recall as compared to the gene team model. In addition, a detailed analysis of the most significant BBHRW (COUNT) clusters show that the top ranking results match known *E. coli* K-12 operons.

Chapter 5

Ortholog Assignment based on Sequence and Spatial Similarity

In this chapter, we investigate the identification of conserved genes based on the Ortholog Assignment problem. We present a simple yet effective method (BBH-LS) for the identification of positional homologs from the comparative analysis of two genomes. BBH-LS applies the bidirectional best hit heuristic to a combination of sequence similarity and gene context similarity scores. We applied our method to the human, mouse, and rat genomes and found that BBH-LS produced the best results when using both sequence and gene context information equally. Compared to the state-of-the-art algorithms, such as MSOAR2, BBH-LS is able to identify more positional homologs with fewer false positives.

This chapter is based on Zhang and Leong [2011, 2012].

5.1 Motivation

Genome-wide comparative analysis of the different species is only feasible and meaningful if we can identify elements that are conserved across species boundaries [Koonin, 2005]. For many studies, the elements under consideration are the set of protein coding genes. Therefore, the identification of corresponding genes between different species is an important step in any genome-wide comparative analysis.

In particular, one-to-one correspondences between genes in different species are preferred in certain applications such as transfer of function annotation [Friedberg, 2006] and genome rearrangement studies [Sankoff, 1999] as they greatly simplify subsequent analysis.

Consider a set of extant genomes and their most recent common ancestor (MRCA). For each gene in the MRCA, there is at most one direct descendant of the gene in each of the extant genomes. The direct descendants of a gene in the MRCA form a set of *positional homologs* [Burgetz et al., 2006]. A single ancestral gene may have multiple descendants due to gene duplication, or no descendants because of gene loss. In the case of gene duplication, we distinguish between the gene that remains in the original location and the copy inserted into a new location. The gene that retains its ancestral location is the direct descendant. Positional homologs represent a set of genes in one-to-one correspondence with each other where each member best reflect the original location of the ancestral gene in the MRCA. Figure 1.2 shows the gene tree for three genes found in two genomes and it illustrates the concept of positional homologs, orthologs, and paralogs.

As discussed in Section 2.3, existing methods for the Ortholog Assignment problem used sequence similarity information and gene order separately. We propose to combine sequence similarity and gene context similarity into a single similarity score and identify positional homologs using the bidirectional best hit heuristic. This has the advantage that the method is easy to implement and computationally efficient. Furthermore, we can easily vary the weight of each type of similarity.

5.2 Inferring Positional Homologs as Bidirectional Best Hits of Sequence and Gene Context Similarity

Our approach is to approximate positional homologs as bidirectional best hits using a scoring scheme that integrates both sequence and gene context similarity scores.

Bidirectional (or reciprocal) best hits (BBH) is a widely used heuristic for finding orthologs between two species. Altenhoff and Dessimoz [2009] compared a number of ortholog inference algorithms and found that BBH's overall performance is surprisingly good despite the simplicity of the method. In particular, they found that orthologs predicted by BBH show close functional relatedness. Another advantage of BBH is that it is easy to compute and commonly used in the literature.

However, using sequence similarity alone is *not enough* to identify the positional homolog among several orthologs [Burgetz et al., 2006]. In such cases, gene context can be used to disambiguate between the paralogs because positional homologs tend to have more similar gene context as evidenced by the presence of large synteny blocks (see Figure 5.1).

Furthermore, [Notebaart et al., 2005] showed that in 29–38 percent of the orthologs they investigated in bacteria, the gene pair with the lower sequence similarity have a higher gene context similarity. Hence, they advised combining gene context information with protein sequence information to predict functional orthologs. In this chapter, we integrate sequence similarity score with a gene context similarity score that reflects the shared gene neighborhood between two genes.

In the following subsections, we give the details for computing sequence and gene context similarity scores and explain how to combine them to compute bidirectional best hits.

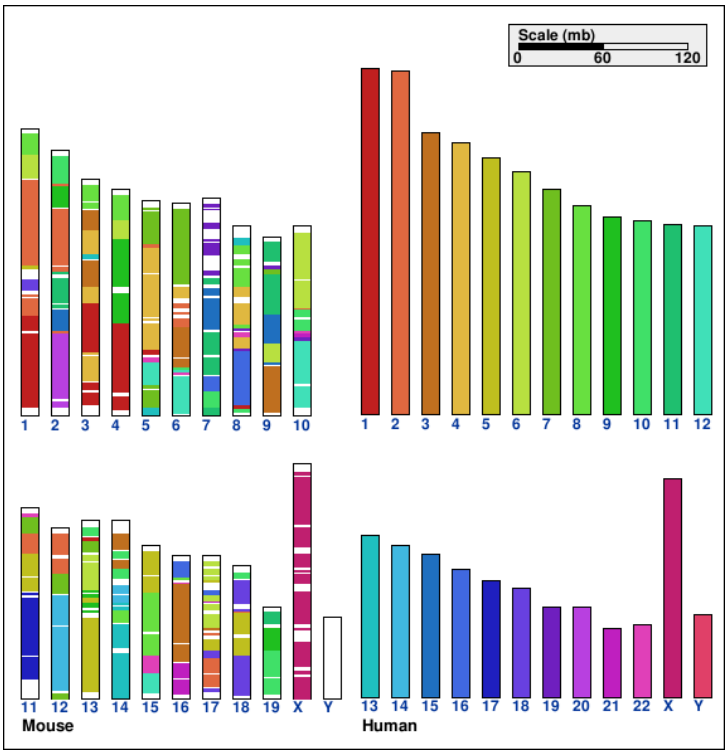

Fig. 5.1: Conserved synteny blocks between human and mouse genome generated by the Cinteny web server [Sinha and Meller, 2007]

5.2.1 Computing sequence similarity scores

We define the sequence similarity score between two genes as the Smith-Waterman alignment score between the respective peptide sequences. As a gene may have multiple transcripts, we use the transcript with the longest peptide sequence to represent the gene. We use the SSEARCH program from the FASTA v36 package [Pearson, 1991] to compute the Smith-Waterman alignment score between all pairs of peptide sequences using default parameters optimized for high sensitivity (BLOSUM50 substitution matrix and E-value cutoff of 10).

We use peptide sequences as the basis of sequence comparison as they have a number of advantages over using nucleotide sequences [Roth et al., 2008]. Peptide sequences are not affected by synonymous substitution and hence able to detect more distant homology. Furthermore, the alignments are faster to compute since the peptide sequence is only one third the length of the nucleotide sequence. Heuristic search algorithms, such as BLAST, are often used to find homologous

sequences since they avoid computing the expensive dynamic programming alignment. However, a serious drawback is that the derived scores (bit score or E-value) are not symmetric and are difficult to easily integrate with other scores. On the other hand, the Smith-Waterman alignment score is symmetric and modern implementations are sufficiently fast for our purpose.

Since we want to integrate both sequence similarity and gene context similarity scores, we normalize the Smith-Waterman scores so that it ranges from 0 to 1 with a score of 1 indicating maximum sequence similarity. The Smith-Waterman alignment score is roughly linearly proportional to the length of the peptide sequences compared; longer peptide sequences tend to have higher alignment scores. Therefore, we remove this dependence on the length of the peptide sequence and normalize the score to range between 0 and 1 by dividing by the maximum Smith-Waterman score of the two self alignments. We formally defined the normalized Smith-Waterman score, $\mathrm{sw}_{\mathrm{norm}}$, as follows:

$$\mathrm{sw}_{\mathrm{norm}}(g,h) = \frac{\mathrm{sw}(g,h)}{\max\{\mathrm{sw}(g,g), \mathrm{sw}(h,h)\}}$$

where $\mathrm{sw}(g,h)$ is the Smith-Waterman alignment score between the peptide sequences of genes g and h.

5.2.2 Computing gene context similarity scores

Gene context similarity refers to the similarity in the genomic context of two genes. In contrast to sequence similarity, there is no widely accepted method to determine the level of gene context similarity between two genes. Jun et al. [2009] proposed a *local synteny test* that considers three genes upstream and downstream of two genes of interest to decide if they are orthologs. They modelled the sequence similarity between the two sets of six genes as a bipartite graph; there is an edge between two genes if their BLASTP E-value is less than $1e^{-5}$. They then compute a maximum matching of the graph. Two genes are *putative orthologs* if

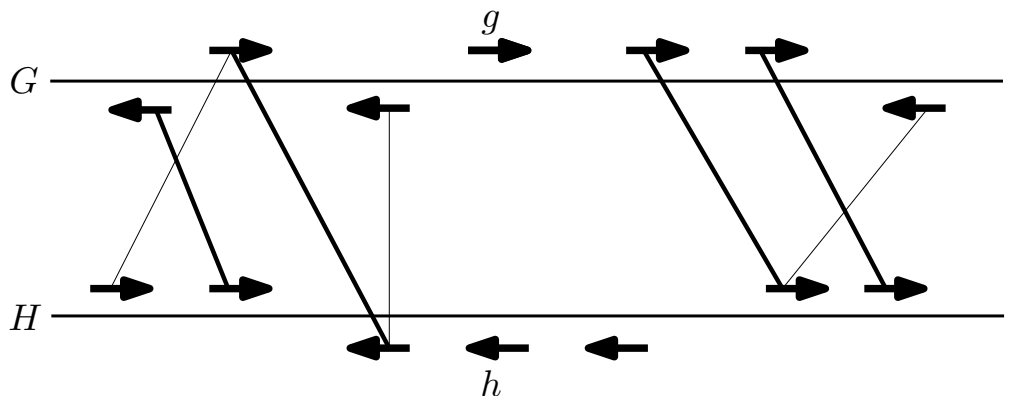

Fig. 5.2: Computing the local synteny score for g and h. We consider three genes upstream and downstream of the two genes of interest and add an edge between two genes if their BLASTP E-value is less than $1e^{-5}$. The thick edges show one of the possible maximum matching. The local synteny score of g and h is 4 since there are 4 edges in the maximum matching.

the size of the maximum matching is greater than one. In other words, they test if there is at least two other matching gene pairs in the vicinity of the gene pair of interest. They determine the BLASTP threshold and size of gene neighborhood by finding the values that maximizes the agreement with InParanoid [Ostlund et al., 2010] and Ensembl Compara [Hubbard et al., 2009] orthologs. They found that 93 percent of sampled inter-species pairs in five mammalian genomes (human, chimpanzee, mouse, rat, and dog) identified by their local synteny test are also found by InParanoid. By analyzing the remaining seven percent of the pairs, they conclude that the use of a local synteny test can resolve ambiguous many-to-many orthologous groups into one-to-one pairs.

While the binary test proposed in Jun et al. [2009] detects the presence of other matching gene pairs in the vicinity of g and h, it does not capture the *strength* of the gene context similarity nor does it make use of the sequence similarity of the gene pair being tested. Thus, it may cause errors in special cases: (a) *false positives* when the local context similarity is high, but the sequence similarity is low, or (b) *false negatives* when the local context similarity is low (only one other matching pair), while the sequence similarity is high.

We define the *local synteny score*, $\mathrm{lss}(g, h)$, as an extension of the binary test proposed in Jun et al. [2009], to capture the degree of gene context similarity

between g and h. The *local synteny score* of two genes g and h is the size of the maximum matching between the six genes surrounding g and h (see Figure 5.2). This gives us a number between 0 and 6, which we normalize by dividing by 6. This is similar to the gene-neighborhood conservation score [Notebaart et al., 2005].

Formally, we define the normalized local synteny score, lss_{norm}, as follows:

$$\text{lss}_{\text{norm}}(g,h) = \frac{|\text{max matching of graph } G = (U \cup V, E)|}{6}$$

where U is the set of six genes around g, V is the set of six genes around h and there is an edge (u,v) in E if the BLASTP E-value of u and v is less than $1e^{-5}$.

5.2.3 Combining bidirectional best hits

Given the normalized sequence similarity scores (sw_{norm}) and normalized gene context similarity scores (lss_{norm}), we combine them into a single similarity score (sim) with a parameter α to represent the weight of gene context similarity. Formally, we define the combined similarity score, sim , as follows:

$$\text{sim}(g,h) = (1-\alpha) \times \text{sw}_{\text{norm}}(g,h) + \alpha \times \text{lss}_{\text{norm}}(g,h)$$

Using the combined score, we compute the set of bidirectional best hits by sorting all gene pairs in decreasing score and scanning this list once. A gene pair (g,h) is a bidirectional best hit if $\text{sim}(g,h)$ is strictly greater than $\text{sim}(g,h')$ for all other genes h' and $\text{sim}(g,h)$ is strictly greater than $\text{sim}(g',h)$ for all other genes g'. This guarantees that the set of bidirectional best hits is always one-to-one.

5.2.4 Reducing the number of false positives

A drawback of the bidirectional best hit criteria is that it does not take into account the actual similarity between two genes. This may lead to false positives when two

genes with very low similarity form bidirectional best hits simply because there are no other similar genes. We found that we can quantify the strength of a particular bidirectional best hit by comparing the similarity of the best hit and the second best hit. Based on this observation, we define the strength of a bidirectional best bit pair (g, h) as:

$$\text{strength}(g,h) = \sqrt{(\text{sim}(g,h) - \text{sim}(g,h')) \times (\text{sim}(g,h) - \text{sim}(g',h))}$$

where h' is the second best hit of g and g' is the second best hit of h. When there is only one hit, the similarity between a gene and its second best hit is defined to be 0.

We can reduce the number of false positives by only keeping those bidirectional best hits whose strength is greater than a minimum strength threshold β.

5.3 Results and Discussion

We evaluate our BBH-LS method by applying it to the human, mouse, and rat genomes. For each pair of genome, we compared the performance of BBH-LS, BBH using only normalized Smith-Waterman score (BBH), MSOAR2 [Shi et al., 2010], InParanoid 4.0 [Ostlund et al., 2010], OMA [Roth et al., 2008], Ensembl Compara [Hubbard et al., 2009], and OrthoMCL[Li et al., 2003].

5.3.1 Experimental setup

Ideally, we should verify the inferred positional homologs by checking whether they perform the same function. Due to the lack of experimentally verified orthologs, we adopt the evaluation strategy used by MSOAR2. The idea is to make use of the official gene symbols for each gene as a proxy for its function. Gene symbols are manually curated based on gene function [Bruford et al., 2008]. However, in the absence of experimentally verified function, genes may be manually assigned a gene symbol based on their sequence/structural similarity to other genes. This

implies that there is some correlation between sequence similarity and gene symbols. Nevertheless, the additional manual curation put into the assignment of gene symbols makes them a reasonable measure.

Using this approach, we can classify the predicted positional homolog pairs into the following three categories:

- true positive: both genes share a common gene symbol
- false positive: gene symbols are completely different
- unknown: either one of the two genes have not been assigned a meaningful[1] gene symbol

The peptide sequences and locations of genes in each of three genomes were download from the Ensembl Release 60[2]. There are 20801, 22842, and 22925 genes in the human, mouse and rat genome. We downloaded official gene symbols from the following species specific databases: HUGO Gene Nomenclature Committee[3], Mouse Genome Informatics[4], and Rat Genome Database[5].

5.3.2 Parameter tuning for BBH-LS

Our scoring scheme uses the parameter α to controls the weight of gene context similarity score. If α is 1, then we only use gene context similarity. If α is 0, then we only use sequence similarity.

We want to determine the optimal value of the parameter α on the human-mouse and mouse-rat dataset. To do this, we ran BBH-LS on the human-mouse and mouse-rat dataset over a range of values of α and tabulated the number of true positives, false positives and unknown pairs for each value. Figure 5.3 and Figure 5.4 shows how the number of true positives, false positives and unknown pairs varies as a function of α for each dataset.

[1] We filter away symbols matching the regular expression "orf" in human genes, "Rik$" or "^GM[0-9]+$" in mouse genes, and "^LOC[0-9]+$" or "^RGD[0-9]+$" in rat genes.
[2] retrieved from ftp://ftp.ensembl.org/pub/release-60/fasta/ in Nov 2010
[3] retrieved from http://www.genenames.org in Dec 2010
[4] retrieved from http://www.informatics.jax.org in Dec 2010
[5] retrieved from http://rgd.mcw.edu in Dec 2010

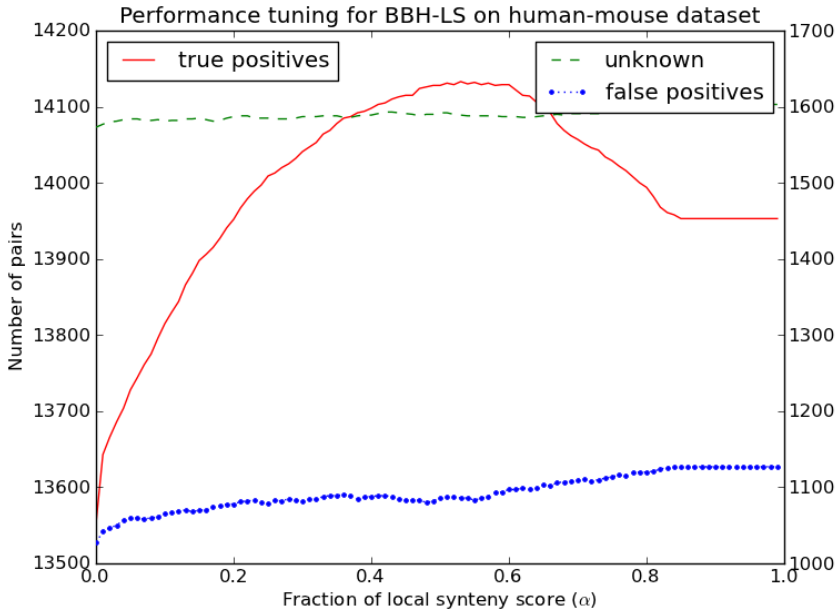

Fig. 5.3: Performance of BBH-LS for different weight of gene context similarity to sequence similarity on the *human-mouse* dataset. Left axis indicates the number of pairs of true positives and the right axis indicate the number of unknown pairs and false positives.

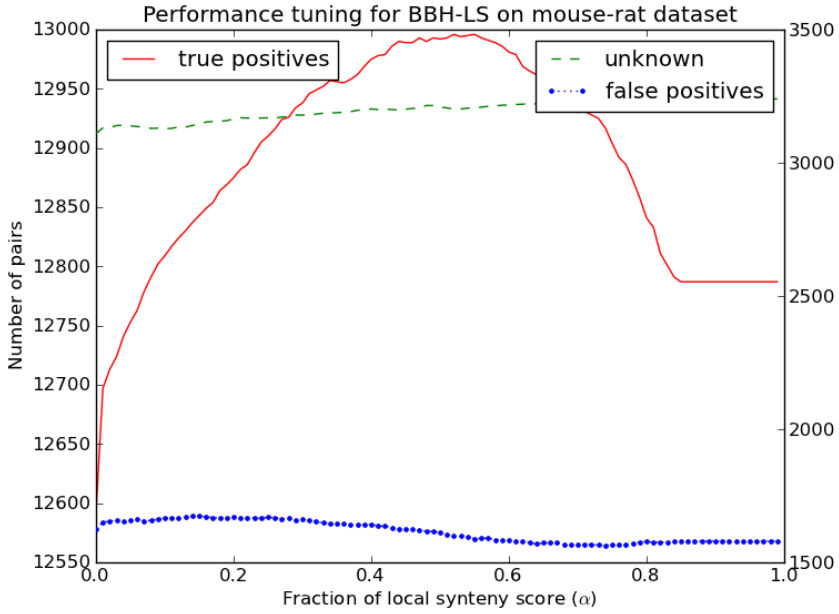

Fig. 5.4: Performance of BBH-LS for different weight of gene context similarity to sequence similarity on the *mouse-rat* dataset. Left axis indicates the number of pairs of true positives and the right axis indicate the number of unknown pairs and false positives.

For the human-mouse dataset (Figure 5.3) we observe that the number of true positives increases rapidly as α increases and then decreases at the same rate after reaching a maximum of 14133 when α is 0.53. However, the number of false positives and unknown pairs also increased slightly as α increases. The same general trend for the true positives is observed for the mouse-rat dataset shown in Figure 5.4 (maximum of 12996 when α is 0.52).

We initially thought that the weight of gene context similarity score should be much lower that of the sequence similarity as many existing methods make use of sequence similarity but not gene context similarity. To our surprise, we found that setting α close to 0.50 maximizes the number of true positives for both datasets. In the following experiments, we set α as 0.50.

It is estimated that the last common ancestor of human and mouse existed 87 million years ago while the mouse-rat ancestor existed 16 million years ago [Springer et al., 2003], furthermore there are 297 large scale rearrangement events between human and mouse but only 106 rearrangement events between mouse and rat [Bourque et al., 2004]. Despite the difference in the genomic distance in these two datasets, the best value of α is consistently around 0.50. Additional experiments using genomes of varying evolutionary distances will be necessary to determine whether this observation holds more generally.

Similarly, we considered the effect of the strength threshold β using the human-mouse dataset. Figure 5.5 shows how the number of true positives, false positives and unknown pairs varies as a function of β. All three quantities decrease to zero as the threshold increases. Importantly, the number of true positives decreases slowly for small values of β while the number of false positives and unknown pairs drops significantly. This shows that our definition for the strength of a BBH pair is effective at reducing the number of false positives without too much effect on the number of true positives.

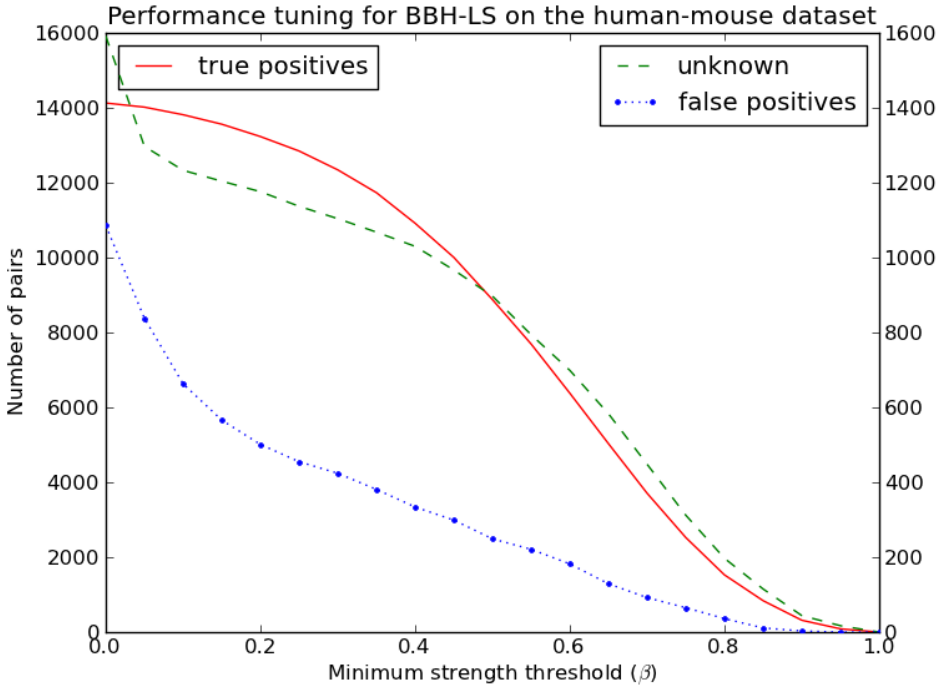

Fig. 5.5: Performance of BBH-LS for different strength threshold β on the *human-mouse* dataset. Left axis indicates the number of pairs of true positives and the right axis indicate the number of unknown pairs and false positives.

5.3.3 Comparison of BBH-LS against existing methods

We obtained the output of the methods in our comparison by running the respective programs on the input data, except for OMA and Ensembl Compara as we did not have access to the programs. We downloaded the orthologs predicted by OMA[6] and Ensembl Compara[7] from their respective websites. InParanoid, OMA, and Ensembl Compara produces pairs of orthologous groups instead of positional homolog pairs. We get ortholog pairs by post-processing the output. InParanoid builds its groups from pairs of seed orthologs, we extract the seed orthologs from each group. For OMA, Ensembl Compara, and OrthoMCL, we use only the one-to-one groups.

Figure 5.6 shows the number of true positives (TP) and false positives (FP) for each method on three datasets. The results for OrthoMCL were not included in Figure 5.6 as is an outlier; for human-mouse dataset OrthoMCL has 8936 TP and

[6]retrieved from http://omabrowser.org in Dec 2010
[7]retrieved from http://www.ensembl.org in Dec 2010

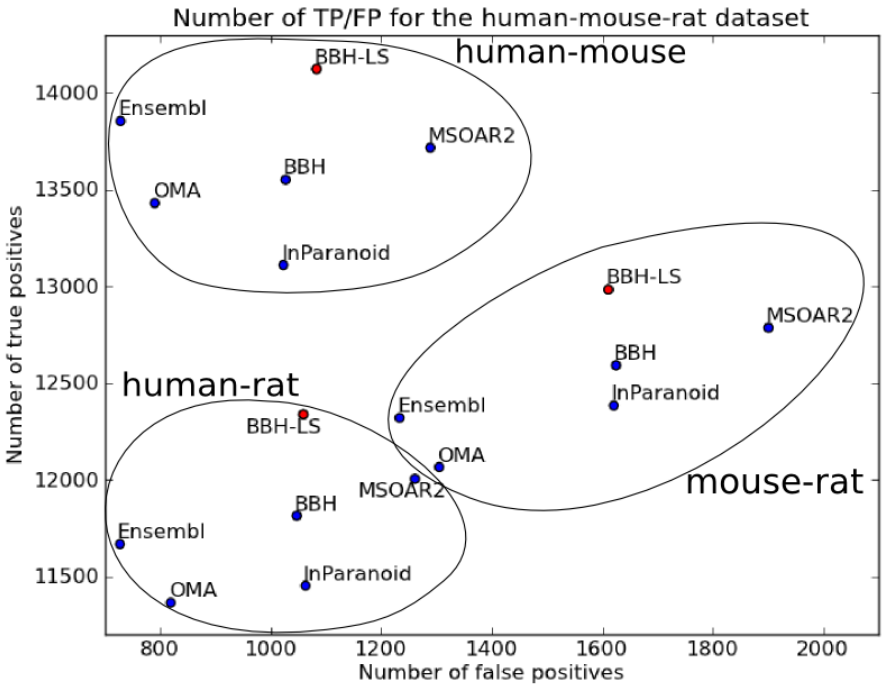

Fig. 5.6: Plot of number of true positives vs number of false positives in the output of BBH-LS ($\alpha = 0.50, \beta = 0.00$), BBH, MSOAR2, InParanoid, OMA, and Ensembl Compara for the human-mouse, human-rat, and mouse-rat dataset

498 FP, for human-rat dataset there are 7409 TP and 530 FP, and for mouse-rat dataset there are 7812 TP and 819 FP.

For the human-mouse dataset, BBH-LS ($\alpha = 0.50, \beta = 0.00$) identified the largest number of true positives (14128), followed by Ensembl Compara (13856), and MSOAR2 (13718). InParanoid which uses BLAST to compute sequence similarity does significantly worst that BBH using normalized Smith-Waterman alignment scores. In terms of the number of false positives, the methods we evaluated fall into three categories: low false positives (OrthoMCL, OMA, Ensembl Compara), medium false positives (InParanoid, BBH, BBH-LS), and high false positives (MSOAR2). We can reduce the number of false positives to 838 (low false positives), by increasing β to 0.05. The corresponding number of true positives is 14018, which is still the highest among all the methods compared. OMA and Ensembl Compara performed surprisingly well given that we only consider the one-to-one groups that were generated.

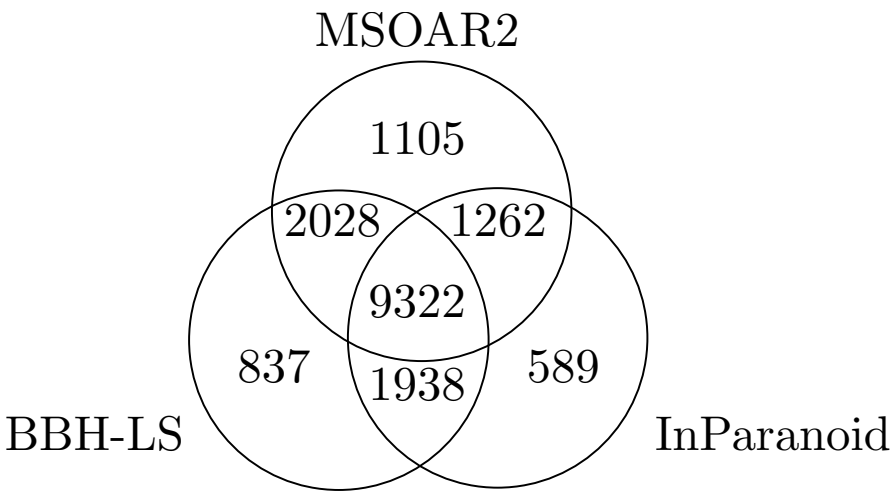

Fig. 5.7: Venn diagram showing the overlap between the true positives reported by BBH-LS, MSOAR2, and InParanoid for the human-mouse dataset.

The results for the human-rat dataset shown in Figure 5.6 is similar to that of the human-mouse data except that the number of true positives produced by Ensembl Compara and OMA has decrease relative to the other methods, but Ensembl Compara still has more true positives than InParanoid. For the mouse-rat dataset (Figure 5.6), OMA and Ensembl Compara is now worse than InParanoid. Another interesting characteristic of the mouse-rat dataset is the higher number of false positives, roughly doubled that of the human-mouse or human-rat dataset for all the methods.

Overall, in all three experiments, our BBH-LS method consistently produced the highest number of true positives as validated using gene symbols with a medium level of false positives. The number of false positives can be further reduced by removing BBH pairs with low strength.

Figure 5.7 shows a more detailed comparison of the true positives reported by BBH-LS, MSOAR2, and InParanoid. A total of 17081 true positives pairs are identified by at least one of the three methods and 57.5 percent (9322/17081) are identified by all three methods. Therefore, only slightly over half of the true positive pairs exhibit strong signals and are easy to detect. The rest require the combination of a number of different sources of information.

In the following, we illustrate a number of specific instances where gene context

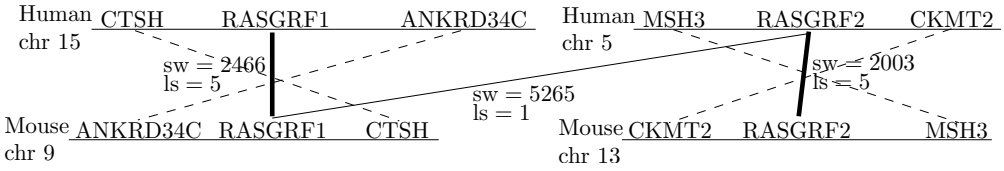

Fig. 5.8: BBH erroneously paired RASGRF2 (human) to RASGRF1 (mouse) because of high Smith-Waterman score, this was corrected by BBH-LS with the help of local synteny score. Bold edges are the pairing from BBH-LS, thin edges are the pairing from BBH, sw = Smith-Waterman score, lc = local synteny score

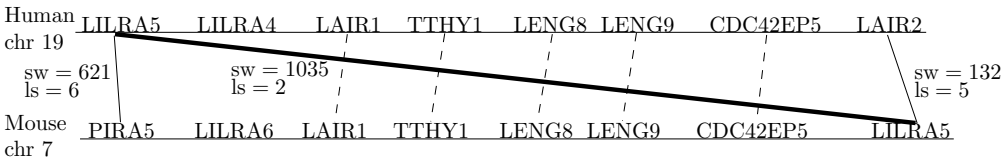

Fig. 5.9: BBH-LS paired LILRA5 (human) with PIRA5 (mouse) and LAIR2 (human) with LIRA5 (mouse) due to the high local synteny produced by the five pairs of genes in between. The correct pairing should be LILRA5 (human) with LILRA5 (mouse) and this was picked up by BBH using just the normalized Smith-Waterman score.

similarity made a significant differences. Figure 5.8 shows an instance where the high gene context similarity between related genes overcame the low sequence similarity between the true positives and converted what would be a false positive for BBH to a pair of true positives. There are a total of 12 of such cases. However, in four cases, the local synteny score caused a true positive identified by BBH to become a pair of false positives. Figure 5.9 illustrates one of these cases.

5.4 Conclusion

The Ortholog Assignment problem is challenging in practice due to gene duplications and gene loss. Several sophisticated methods, which make use of complex heuristics (InParanoid) or require solving computationally hard problems (MSOAR2), have been proposed to tackle this problem. However, we show in this paper that the simple bidirectional best hit heuristic, coupled with a scoring scheme that combines both sequence and gene context similarity, is surprisingly good at identifying positional homologs. In all three pairwise comparison between human, mouse, and rat genomes, our BBH-LS method identified the largest

number of positional homolog (as validated using gene symbols) with a medium number of false positives.

We are current investigating the application of our method for ortholog assignment in plant genomes (see Section A.4).

Chapter 6

Conclusion

In this thesis, we studied the problem of identifying conserved gene clusters and one-to-one conserved genes (positional homologs). We conclude by presenting a summary of the contributions and some possible future work.

6.1 Summary of Contributions

We considered two different models of conserved genes clusters that allow for gaps within the cluster. Our Gene Team Tree (GTT) model is a hierarchical representation of gene teams for all gap lengths. It allows us to visualize the relationship between the various gene teams and makes explicit the structure of gene teams. The identification of all gene teams makes it possible to apply other more intuitive measures to select interesting teams with different maximum gap length. Our algorithms extends existing algorithms for computing the gene teams based on a specific max-gap without increasing the time/space complexity. These algorithms have been superseded by faster, more sophisticated algorithms developed by other researchers.

The bidirectional best hit (BBH) heuristic is commonly used for identifying families of homologous DNA sequences from BLAST hits. As conserved gene clusters are also homologous genomic regions, it is natural to use BBH to constrain the r-window model. Using the BBH heuristic removes the need to specify the

minimum level of conservation in a r-window cluster. The constrained model also enabled us to develop efficient algorithms for finding the clusters as we only have to find the best hit for each r-window. We found that compared to the gene teams, a greater percentage of BBHRW clusters corresponds to known conserved gene clusters. This demonstrates that incorporating domain specific constraints such as BBH can help to improve the relevance of the clusters.

The last part of this thesis considered the problem of identifying conserved genes, specifically positional homologs. Inspired by the success of the BBHRW model, we also make use of the BBH heuristic for this problem. The twist was we added gene context similarity into the mix to improve upon the standard approach that was based on only sequence similarity. A surprising result is that the best performance on the human-mouse-rat data was obtained using sequence similarity and gene context similarity with equal weight. We experimented with more sophisticated definitions of gene context similarity based on conserved gene clusters but they did not improve the results.

6.2 Future Work

GTT for many genomes: Computing the GTT for a large number of genomes in a reasonable time is still very challenging as the size of the GTT increases dramatically when more genomes are considered. It doesn't make sense to compute the whole GTT when only a small number of nodes are potentially useful. A possible approach is to make use of a score function to guide the growth of the GTT and only compute the parts necessary to find interesting gene teams. A related problem is to combine the quorum parameter with the GTT.

Conserved gene clusters without gene families: A general problem with existing models of conserved gene clusters is their reliance on accurate gene families. We observed that both gene team and BBHRW can be reformulated in terms of a more general gene similarity relation. Under this framework, we can model

gene families as an equivalence relation. One possible gene similarity relation is to use well known sequence similarity measures such as the Smith-Waterman distance. This allows us to circumvent the computation of gene families and still compute conserved gene clusters.

Ranking of gene clusters using domain knowledge: Most algorithms generally produce a large number of gene clusters. Ranking the generated clusters is an important way to make sense of a large number of results and focus on the most interesting clusters. The ranking function is also the right place to include expert/domain knowledge about gene clusters without complicating the model.

Bibliography

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.

Adrian Altenhoff and Christophe Dessimoz. Phylogenetic and functional assessment of orthologs inference projects and methods. *PLoS Computational Biology*, 5(1):e1000262, 2009.

Marie-Pierre Béal, Anne Bergeron, Sylvie Corteel, and Mathieu Raffinot. An algorithmic view of gene teams. *Theoretical Computer Science*, 320(2-3):395–418, 2004.

Jon L. Bentley. Solutions to Klee's rectangle problems. *Unpublished manuscript, Dept of Comp Sci, Carnegie-Mellon University, Pittsburgh PA*, 1977.

Anne Bergeron and Jens Stoye. On the similarity of sets of permutations and its applications to genome comparison. In Tandy Warnow and Binhai Zhu, editors, *Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 68–79. Springer, 2003.

Anne Bergeron, Sylvie Corteel, and Mathieu Raffinot. The algorithmic of gene teams. In Roderic Guigó and Dan Gusfield, editors, *Algorithms in Bioinformatics*, volume 2452 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 2002.

Anne Bergeron, Mathieu Blanchette, Annie Chateau, and Cédric Chauve. Reconstructing ancestral gene orders using conserved intervals. In Inge Jonassen and Junhyong Kim, editors, *Algorithms in Bioinformatics*, volume 3240 of *Lecture Notes in Computer Science*, pages 14–25. Springer, 2004.

Anne Bergeron, Cédric Chauve, Fabien de Montgolfier, and Mathieu Raffinot. Computing common intervals of k permutations, with applications to modular decomposition of graphs. In Gerth Stølting Brodal and Stefano Leonardi, editors, *European Symposium on Algorithms*, volume 3669 of *Lecture Notes in Computer Science*, pages 779–790. Springer, 2005.

Guillaume Blin and Romeo Rizzi. Conserved interval distance computation between non-trivial genomes. In Lusheng Wang, editor, *Computing and Combinatorics*, volume 3595 of *Lecture Notes in Computer Science*, pages 22–31. Springer, 2005.

Guillaume Blin, Annie Chateau, Cédric Chauve, and Yannick Gingras. Inferring positional homologs with common intervals of sequences. In Guillaume Bourque and Nadia El-Mabrouk, editors, *Comparative Genomics*, volume 4205 of *Lecture Notes in Computer Science*, pages 24–38. Springer, 2006.

Guillaume Bourque, Pavel A. Pevzner, and Glenn Tesler. Reconstructing the genomic architecture of ancestral mammals: Lessons from human, mouse, and rat genomes. *Genome Research*, 14(4):507–516, 2004.

Guillaume Bourque, Yasmine Yacef, and Nadia El-Mabrouk. Maximizing synteny blocks to identify ancestral homologs. In Aoife McLysaght and Daniel H. Huson, editors, *Comparative Genomics*, volume 3678 of *Lecture Notes in Computer Science*, pages 21–34. Springer, 2005.

Elspeth A. Bruford, Michael J. Lush, Mathew W. Wright, Tam P. Sneddon, Sue Povey, and Ewan Birney. The HGNC Database in 2008: a resource for the human genome. *Nucleic Acids Research*, 36(suppl 1):D445, 2008.

David Bryant. The complexity of calculating exemplar distances. In David Sankoff and Joseph H. Nadeau, editors, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 207–212. Kluwer Academic Publishers, 2000.

Ingrid J. Burgetz, Salimah Shariff, Andy Pang, and Elizabeth R.M. Tillier. Positional homology in bacterial genomes. *Evolutionary Bioinformatics Online*, 2: 42–55, 2006.

Andre R.O. Cavalcanti, Ricardo Ferreira, Zhenglong Gu, and Wen-Hsiung Li. Patterns of gene duplication in Saccharomyces cerevisiae and Caenorhabditis elegans. *Journal of Molecular Evolution*, 56(1):28–37, 2003.

Francis S. Collins, Ari Patrinos, Elke Jordan, Aravinda Chakravarti, Raymond Gesteland, and LeRoy Walters. New goals for the US human genome project: 1998-2003. *Science*, 282(5389):682, 1998.

Gilles Didier. Common intervals of two sequences. In Gary Benson and Roderic D.M. Page, editors, *Algorithms in Bioinformatics*, volume 2812 of *Lecture Notes in Computer Science*, pages 17–24. Springer, 2003.

Gilles Didier, Thomas Schmidt, Jens Stoye, and Dekel Tsur. Character sets of strings. *Journal of Discrete Algorithms*, 5(2):330–340, 2007.

Dannie Durand and David Sankoff. Tests for gene clustering. *Journal of Computational Biology*, 10(3-4):453–482, 2003.

Maria D. Ermolaeva, Owen White, and Steven L. Salzberg. Prediction of operons in microbial genomes. *Nucleic Acids Research*, 29(5):1216–1221, 2001.

Walter M. Fitch. Homology a personal view on some of the problems. *Trends in Genetics*, 16(5):227–231, 2000.

Iddo Friedberg. Automated protein function prediction–the genomic challenge. *Briefings in Bioinformatics*, 7(3):225, 2006.

Robert Friedman and Austin L. Hughes. Gene duplication and the structure of eukaryotic genomes. *Genome Research*, 11(3):373–381, 2001.

Zheng Fu, Xin Chen, Vladimir Vacic, Peng Nan, Yang Zhong, and Tao Jiang. Msoar: A high-throughput ortholog assignment system based on genome rearrangement. *Journal of Computational Biology*, 14(9):1160–1175, 2007.

Socorro Gama-Castro, Verónica Jiménez-Jacinto, Martín Peralta-Gil, Alberto Santos-Zavaleta, Mónica I Peñaloza-Spinola, Bruno Contreras-Moreira, Juan Segura-Salazar, Luis Muñiz-Rascado, Irma Martínez-Flores, Heladia Salgado, César Bonavides-Martínez, Cei Abreu-Goodger, Carlos Rodríguez-Penagos, Juan Miranda-Ríos, Enrique Morett, Enrique Merino, Araceli M Huerta, Luis Treviño-Quintanilla, and Julio Collado-Vides. Regulondb (version 6.0): gene regulation model of escherichia coli k-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. *Nucleic Acids Research*, 36(Database issue):D120–D124, 2008.

Steve Hampson, Aoife McLysaght, Brandon Gaut, and Pierre Baldi. LineUp: statistical detection of chromosomal homology with application to plant comparative genomics. *Genome Research*, 13(5):999–1010, 2003.

Sridhar Hannenhalli and Pavel A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutation by reversal). *Journal of the ACM*, 46:1–27, 1999.

Xin He and Michael H. Goldwasser. Identifying conserved gene clusters in the presence of homology families. *Journal of Computational Biology*, 12(6):638–656, 2005.

Steffen Heber, Richard Mayr, and Jens Stoye. Common intervals of multiple permutations. *Algorithmica*, 60(2):175–206, 2011.

Rose Hoberman and Dannie Durand. The incompatible desiderata of gene cluster properties. In Aoife McLysaght and Daniel H. Huson, editors, *Comparative*

Genomics, volume 3678 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2005.

Rose Hoberman, David Sankoff, and Dannie Durand. The statistical analysis of spatially clustered genes under the maximum gap criterion. *Journal of Computational Biology*, 12(8):1083–1102, 2005.

Tim J.P. Hubbard, Bronwen L. Aken, Sarah C. Ayling, Benoit Ballester, Kathryn Beal, Eugene Bragin, Simon Brent, Yuan Chen, Peter Clapham, Laura Clarke, Guy Coates, Susan Fairley, Stephen Fitzgerald, Julio Fernandez-Banet, Leo Gordon, Stefan Gräf, Syed Haider, Martin Hammond, Richard C.G. Holland, Kevin L. Howe, Andrew M. Jenkinson, Nathan Johnson, Andreas Kähäri, Damian Keefe, Stephen Keenan, Rhoda Kinsella, Felix Kokocinski, Eugene Kulesha, Daniel Lawson, Ian Longden, Karine Megy, Patrick Meidl, Bert Overduin, Anne Parker, Bethan Pritchard, Daniel Rios, Michael Schuster, Guy Slater, Damian Smedley, William Spooner, Giulietta Spudich, S. Trevanion, Albert J. Vilella, Jan Vogel, Simon White, Steven P. Wilder, Arek Zadissa, Ewan Birney, Fiona Cunningham, Val Curwen, Richard Durbin, Xosé M. Fernández-Suarez, Javier Herrero, Arek Kasprzyk, Glenn Proctor, James Smith, Stephen M.J. Searle, and Paul Flicek. Ensembl 2009. *Nucleic Acids Research*, 37(suppl 1):D690, 2009.

Paul Jaccard. Nouvelles recherches sur la distribution florale. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 44:223–270, 1908.

François Jacob, David Perrin, Carmen Sanchez, and Jacques Monod. Operon: a group of genes with the expression coordinated by an operator. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 250:1727, 1960.

Jin Jun, Ion I. Mandoiu, and Craig E. Nelson. Identification of mammalian orthologs using local synteny. *BMC Genomics*, 10(1):630, 2009.

Sun Kim, Jeong-Hyeon Choi, and Jiong Yang. Gene teams with relaxed proximity

constraint. In *Proceedings of the IEEE Computational Systems Bioinformatics Conference*, pages 44–55. IEEE Computer Society, 2005.

Eugene V. Koonin. Orthologs, Paralogs, and Evolutionary Genomics1. *Genetics*, 39(1):309, 2005.

Gad M. Landau, Laxmi Parida, and Oren Weimann. Gene proximity analysis across whole genomes via pq trees. *Journal of Computational Biology*, 12(10): 1289–306, 2005.

Jeffrey G. Lawrence. Selfish operons: the evolutionary impact of gene clustering in prokaryotes and eukaryotes. *Current Opinion in Genetics & Development*, 9 (6):642–648, 1999.

Trong Dao Le, Melvin Zhang, and Hon Wai Leong. Algorithms for computing bidirectional best hit r-window gene clusters. In Mikhail J. Atallah, Xiang-Yang Li, and Binhai Zhu, editors, *Frontiers in Algorithmic and Algorithmic Aspects in Information and Management*, volume 6681 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 2011.

Emmanuelle Lerat, Vincent Daubin, and Nancy A. Moran. From gene trees to organismal phylogeny in prokaryotes: the case of the gamma-proteobacteria. *PLoS Biology*, 1(1), 2003.

Li Li, Christian J. Stoeckert, and David S. Roos. OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome research*, 13(9):2178, 2003.

Xu Ling, Xin He, Dong Xin, and Jiawei Han. Efficiently identifying max-gap clusters in pairwise genome comparison. *Journal of Computational Biology*, 15 (6):593–609, 2008.

Xu Ling, Xin He, and Dong Xin. Detecting gene clusters under evolutionary constraint in a large number of genomes. *Bioinformatics*, 25(5):571–577, 2009.

Gabriel Moreno-Hagelsieb and Kristen Latimer. Choosing BLAST options for better detection of orthologs as reciprocal best hits. *Bioinformatics*, 24(3):319, 2008.

Volker Muller, Christopher J. Jones, Ikuro Kawagishi, Shin-ichi Aizawa, and Robert M. Macnab. Characterization of the fliE genes of Escherichia coli and Salmonella typhimurium and identification of the FliE protein as a component of the flagellar hook-basal body complex. *Journal of Bacteriology*, 174(7):2298–2304, 1992.

Richard A. Notebaart, Martijn A. Huynen, Bas Teusink, Roland J. Siezen, and Berend Snel. Correlation between sequence conservation and the genomic context after gene duplication. *Nucleic Acids Research*, 33(19):6164, 2005.

Gabriel Ostlund, Thomas Schmitt, Kristoffer Forslund, Tina Kostler, David N. Messina, Sanjit Roopra, Oliver Frings, and Erik L.L. Sonnhammer. InParanoid 7: new algorithms and tools for eukaryotic orthology analysis. *Nucleic Acids Research*, 38(Database issue):D196, 2010.

Ross Overbeek, Michael Fonstein, Mark D'Souza, Gordon D. Pusch, and Natalia Maltsev. The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences of the United States of America*, 96(6):2896–2901, 1999.

Laxmi Parida. Gapped Permutation Pattern Discovery for Gene Order Comparisons. *Journal of Computational Biology*, 14(1):45–55, 2007.

Sophie Pasek, Anne Bergeron, Jean-Loup Risler, Alexandra Louis, Emmanuelle Ollivier, and Mathieu Raffinot. Identification of genomic features using microsyntenies of domains: Domain teams. *Genome Research*, 15(6):867–874, 2005.

William R. Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11(3):635–650, 1991.

Alexander C.J. Roth, Gaston H. Gonnet, and Christophe Dessimoz. Algorithm of OMA for large-scale orthology inference. *BMC Bioinformatics*, 9(1):518, 2008.

Kenneth E. Rudd. EcoGene: a genome sequence database for Escherichia coli K-12. *Nucleic Acids Research*, 28(1):60, 2000.

David Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11): 909–17, 1999.

David Sankoff. Rearrangements and chromosomal evolution. *Current Opinion in Genetics & Development*, 13(6):583–7, 2003.

Thomas Schmidt and Jens Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. In Süleyman Cenk Sahinalp, S. Muthukrishnan, and Ugur Dogrusöz, editors, *Combinatorial Pattern Matching*, volume 3109 of *Lecture Notes in Computer Science*, pages 347–358. Springer, 2004.

Adrian Schneider, Christophe Dessimoz, and Gaston H. Gonnet. OMA Browser—exploring orthologous relations across 352 complete genomes. *Bioinformatics*, 23(16):2180, 2007.

Guanqun Shi, Liqing Zhang, and Tao Jiang. MSOAR 2. 0: Incorporating tandem duplications into ortholog assignment based on genome rearrangement. *BMC Bioinformatics*, 11(1):10, 2010.

Amit U. Sinha and Jaroslaw Meller. Cinteny: flexible analysis and visualization of synteny and genome rearrangements in multiple organisms. *BMC Bioinformatics*, 8(1):82, 2007.

Mark S. Springer, William J. Murphy, Eduardo Eizirik, and Stephen J. O'Brien.

Placental mammal diversification and the cretaceous–tertiary boundary. *Proceedings of the National Academy of Sciences of the United States of America*, 100(3):1056, 2003.

Roman L. Tatusov, Darren A. Natale, Igor V. Garkavtsev, Tatiana A. Tatusova, Uma T. Shankavaram, Bachoti S. Rao, Boris Kiryutin, Michael Y. Galperin, Natalie D. Fedorova, and Eugene V. Koonin. The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Research*, 29(1):22–28, 2001.

Takeaki Uno and Mutsunori Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.

J. Craig Venter, Mark D. Adams, Granger G. Sutton, Anthony R. Kerlavage, Hamilton O. Smith, and Michael Hunkapiller. Shotgun sequencing of the human genome. *Science*, 280(5369):1540, 1998.

Biing-Feng Wang and Chien-Hsin Lin. Improved algorithms for finding gene teams and constructing gene team trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(5):1258–1272, 2011.

Biing-Feng Wang, Chung-Chin Kuo, Shang-Ju Liu, and Chien-Hsin Lin. A new efficient algorithm for the gene-team problem on general sequences. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(2):330–344, 2012.

Geoffrey A. Watterson, Warren J. Ewens, Thomas E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99:1–7, 1982.

Qingwu Yang, Gangman Yi, Fenghui Zhang, Michael R. Thon, and Sing-Hoi Sze. Identifying Gene Clusters within Localized Regions in Multiple Genomes. *Journal of Computational Biology*, 17(5):657–668, 2010.

Melvin Zhang and Hon Wai Leong. Gene team tree: A compact representation of all gene teams. In Craig E. Nelson and Stéphane Vialette, editors, *Comparative*

Genomics, volume 5267 of *Lecture Notes in Computer Science*, pages 100–112. Springer, 2008.

Melvin Zhang and Hon Wai Leong. Gene team tree: A hierarchical representation of gene teams for all gap lengths. *Journal of Computational Biology*, 16(10): 1383–1398, 2009.

Melvin Zhang and Hon Wai Leong. Bidirectional best hit r-window gene clusters. *BMC Bioinformatics*, 11(Suppl 1):S63, 2010.

Melvin Zhang and Hon Wai Leong. Identifying positional homologs as bidirectional best hits of sequence and gene context similarity. In *IEEE International Conference on Systems Biology*, pages 117–122, 2011.

Melvin Zhang and Hon Wai Leong. BBH-LS: An algorithm for computing positional homologs using sequence and gene context similarities. *BMC Systems Biology*, 2012. to appear.

Appendix A

Other research work undertaken during the candidature

In addition to the results presented in the main chapters of the thesis, I also contributed to the following joint research.

A.1 Phylogeny from Gene Order Web Application

We developed a web application, `http://pgo.comp.nus.edu.sg`, that allows users to run different algorithms for phylogenetic reconstruction from gene orders on their own dataset. A user submits a list of gene orders via the website and the system will compute phylogenetic trees using different combinations of phylogenetic reconstruction algorithms. The system generates a html report that shows the tree produced by each algorithm and a comparison of the all the computed trees. The report is then sent to the user via email.

This is joint work with Fan Chang Hao and Hon Wai Leong (my PhD advisor). We put up a poster describing this system at the 21st International Conference on Genome Informatics (GIW), 2010.

A.2 On Two Variations of the Reversal Median Problem

The Reversal Median Problem (RMP) is the problem of finding an ancestral genome (called the median) given the gene orders of three genomes. RMP is commonly encountered when doing phylogenetic reconstructions. We developed an exact algorithm that solves certain instances of the RMP when provided with the optimal sorting sequences between every pair of genomes. Two variations of the RMP were considered: In the first variation, we are given one sorting sequence for each pair of genomes. However, in general, there can be many different optimal sorting sequences. Hence, in the second variation, we make use of a compact representation of all possible optimal sorting sequences for each pair of genomes. Our algorithm is able to provide an exact solution (the median genome) or determine that it is not able to do so for every instance of the problem.

This is joint work with Zhou Zhong and Hon Wai Leong, the results are published in the proceedings of International Conference on Mathematical and Computational Biology (ICMCB), 2011.

A.3 Dynamic Programming Algorithms for Efficiently Computing Cosegmentation between Biological Images

We propose two dynamic programming algorithms for the so-called tree assignment problem, which generalizes bipartite matchings to trees. We formulate restricted versions that are tractable by a dynamic programming algorithm. Furthermore, we describe a second dynamic programming algorithm that deals with the efficient computation of certain weights between so-called component trees that can be

applied to obtain certain cosegmentations in bioimaging applications. Our investigations indicate that our algorithms are both efficient and effective, supported by evaluating the influence of the restrictions imposed by the dynamic programming formulation on a collection of image data.

This is joint work with Xiao Hang, Axel Mosig, and Hon Wai Leong, the results will appear in the proceedings of Workshop on Bioinformatics Algorithms (WABI), 2011.

A.4 Ortholog Assignment for Plant Genomes

We are interested in evaluating different algorithms to solve the Ortholog Assignment problem in plant genomes. The goal is to find the positional homologs for C3/C4 photosynthesis genes and use them to study the evolution of the C3/C4 photosynthesis pathway. Ortholog Assignment is particular difficulty in plant genomes because of extensive gene duplications and whole genome duplication. So far, the results of our BBH-LS algorithm looks promising as compared to the other algorithms in the study.

This is joint work with the Plant System Biology group at the CAS-MPG Partner Institute for Computational Biology.

A.5 Genome Sorting with Bridges

We developed a new heuristic framework for genome sorting based on the concept of bridges. The idea is have an algorithm that is able to handle arbitrary gene order without any restrictions by performing a series of reductions to simplify the problem and reduce the gene orders to simple permutations. We have identified several structures in the breakpoint graph that can be exploited to reduce the number of rearrangement operations needed to sort two genomes.

This is joint work with Fan Chang Hao and Hon Wai Leong.