OrthoCluster: A New Tool for Mining Synteny Blocks and Applications in Comparative Genomics

Xinghuo Zeng[†] Jian Pei[†] Ismael A. Vergara[‡]
Matthew J. Nesbitt[‡] Ke Wang[†] Nansheng Chen[‡]
[†] School of Computing Science [‡] Department of Molecular Biology and Biochemistry
Simon Fraser University, Burnaby, BC Canada V5A 1S6
[†]{xzeng, jpei, wangk}@cs.sfu.ca [‡]{iav, mnesbitt, chenn}@sfu.ca

ABSTRACT

By comparing genomes among both closely and distally related species, *comparative genomics analysis* characterizes structures and functions of different genomes in both conserved and divergent regions. Synteny blocks, which are conserved blocks of genes on chromosomes of related species, play important roles in comparative genomics analysis. Although a few tools have been designed to identify synteny blocks, most of them cannot handle some challenging application requirements, particularly the strandedness of genes, gene inversions, gene duplications, and comparison of more than two genomes. We developed a data mining tool, OrthoCluster, which can handle all those challenges. It is publicly available at http://genome.sfu.ca/projects/orthocluster. OrthoCluster takes the annotated gene sets of candidate genomes and pairwise orthologous relationships as input and efficiently identifies the complete set of synteny blocks. In addition, OrthoCluster identifies four types of genome rearrangement events namely inversion, transposition, insertion/deletion, and reciprocal translocation. To be flexible in various application scenarios, OrthoCluster comes with a systematic set of parameters such as the synteny block size, number of mismatches allowed, whether the strandedness is enforced, whether gene ordering is preserved. Furthermore, OrthoCluster can be used to identify segmental duplication in a genome. In this paper, we introduce the major technical ideas, and present some interesting findings using OrthoCluster.

1. INTRODUCTION

Genome sequences contain ultimate genetic information and instructions defining essentially all aspects of different forms of living organisms ranging from a simple virus to highly sophisticated humans. Genome analysis enables us to understand mechanisms underlying biological processes like development, aging, learning and memory, as well as disease conditions such as cancer, diabetes, and many types of devastating neurodegenerative diseases.

EDBT'08, March 25–30, 2008, Nantes, France.

Since the initiation of the unprecedented Human Genome Project in 1980s, an increasing number of genomes have been fully sequenced, assembled, and annotated due largely to major technological advances in DNA sequencing. Milestones include the complete genome sequences of the bacterium *Haemophilus influenza* [13], the budding yeast *Saccharomyces cerevisiae* [14], the multicellular organism nematode worm *Caenorhabditis elegans* [11], the popular genetics model organism fruit fly *Drosophila melanogaster* [1, 23], and humans (*Homo sapiens*) [38, 19] were published between 1995 and 2001. More recently, the sequencing of James Watson's genome[1] and Craig Venter's diploid genome [20] were published in 2007.

The availability of a large number of genome sequences offers a huge potential to comparative genomics analysis projects, which aim to understand structures and functions of genomic features and their roles in gene expression in closely or distally related species [22, 16]. Through such comparative analysis, we can relate knowledge gained in well-studied genomes or simple genomes of model organisms (the "reference" genomes) to newly sequenced genomes or more complex genomes (the "target" genomes). We can also discover novel knowledge about yet unidentified functional elements in the genomes under analysis. Furthermore, comparative analysis facilitates understanding of driving forces for molecular evolution and speciation.

Generally, comparative genomics analysis highlights two categories of functional elements. The first category is widely known as "ultraconserved elements", which are a few hundred bases (bp) long and show 100% identity between elements in genomes of closely related species and are conserved in other species [3]. Another category is synteny blocks, which are tens of kilobases (Kb) to multiple megabases (Mb) long and contain usually multiple genes. A synteny block is a conserved block of genes on chromosomes of related species.

Using the newly developed methods, hundreds of ultraconserved elements have been identified in the human genome [3]. More recently, ultraconserved elements in repetitive regions and transposable elements of the human genome were identified [39]. Those discoveries have led to some critical insights into the regulatory function of non-coding DNAs and have prompted reevaluation of 95% of the intergenic genomic sequences in the human genome, which were once widely called junk DNAs.

In contrast to the comprehensive identification of short ultraconserved elements, the progress on identification of

[1] http://www.454.com/news-events/press-releases.asp?display=detail&id=68

synteny blocks among genomes, especially the genomes of higher organisms such as animals and plants, has been rather limited. On the one hand, there are well documented cases of synteny blocks where genes are often co-regulated, and share similar functions. For example, synteny blocks in simple bacteria genomes are usually organized as blocks of genes called operons. Because of the small size of the prokaryote genomes and the high density of genes, it is rather straightforward to identify synteny blocks in those genomes. Identification and characterization of those prokaryote operons have greatly facilitated the understanding of gene functions and regulations in prokaryote species such as bacteria. It has been known that genes within a bacterial operon are under the control of a single regulatory signal or sequence [18].

On the other hand, synteny blocks in higher eukaryote genomes are much more complex. Operons, like those identified in prokaryote genomes, have not been found in higher animal and plant genomes except for a small number of genomes such as the genome of the *C. elegans* [5]. About $1,000$ operons, which contain only 15% of all genes in the *C. elegans* [5] genome, have been identified, each containing two or more genes.

Nevertheless, synteny blocks are critical for comparative genomics research. The accumulating evidence strongly suggests that genes in the eukaryote genomes are not randomly distributed. Instead, they form various types of functional clusters [17] and topological arrangements [9]. Because of the functional significance of those gene blocks, they may be under evolutionary pressure that prevents genes within a block from escaping the block. Therefore, identification of such synteny blocks may provide important insights into the functions of genes and gene blocks. Successful development and application of data mining tools for identifying such large scale functional elements can also assist in identifying copy number genomic variations (CNVs) [28] where segments of DNA are present at variable copy numbers in comparison with a reference genome. Some recently identified genomic variations have been associated to many disease conditions including the autism spectrum disorders (ASDs) [29]. The tools also play an essential role in interpreting results from large scale projects including the ENCODE projects [4], the cancer genomics atlas projects (http://cancergenome.nih.gov/), and the HapMap projects [12].

Earlier efforts in identification of synteny blocks often used ad hoc methods. Those ad hoc methods tend to be slow, not fully reproducible, unaware of strandedness that indicates the DNA double-helix strand in which a gene resides, and inappropriate for general applications. In the last five years, a few computational methods have been developed for identifying synteny blocks between genomes. Compared to the early ad hoc methods, those methods are generally more effective. However, since most of them were developed to tackle specific problems, they usually lack many important functionalities and, therefore, may not be general for other applications.

We developed a data mining tool OrthoCluster, which is capable of handling multiple challenging scenarios. OrthoCluster is publicly available at http://genome.sfu.ca/projects/orthocluster. Briefly, OrthoCluster takes the annotated gene sets of candidate genomes and pairwise orthologous relationships as the input and efficiently computes the complete set of synteny blocks. In addition, OrthoCluster also identifies four types of genome rearrangement events namely inversion, transposition, insertion/deletion, and reciprocal translocation. Users can obtain desirable output by setting the related parameters including the synteny block sizes, mismatches allowed, whether the strandedness is enforced, whether gene ordering is preserved, etc. Furthermore, OrthoCluster can be used to identify segmental duplication in a genome. In contrast to the existing methods, OrthoCluster can resolve one-to-many relationships and work on multiple genomes. Moreover, it is aware of strandedness and ordering of genes. It can allow different levels of order differences. Another important feature is that OrthoCluster distinguishes between two types of mismatches: "in-map" mismatches and "out-map" mismatches. This distinction is very critical for separation of two different types of genome rearrangement events: insertion/deletion and transposition.

In this paper, we introduce the major technical ideas in OrthoCluster and present some interesting findings using OrthoCluster. We make the following contributions. First, we identify some important challenges in data mining tools for comparative genomics. Second, we introduce OrthoCluster, a novel tool for mining blocks in comparative genomics. The tool is publicly available and has been used by working biologists in comparative genomics research. Third, we present some case studies using OrthoCluster and an empirical evaluation of the data mining techniques in OrthoCluster. Last, we systematically compare OrthoCluster with the existing major comparative genomics data mining tools.

The rest of the paper is organized as follows. In Section 2, we introduce OrthoCluster. In Section 3, we present the case studies using OrthoCluster and an empirical evaluation of the major techniques. OrthoCluster is compared with the existing major comparative genomics data mining tools in Section 4. Section 5 concludes the paper.

2. ORTHOCLUSTER

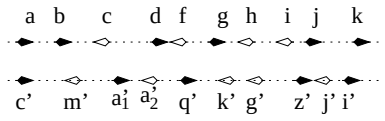
In this section, we first describe the problem of mining synteny blocks. Then, we present the core algorithm in OrthoCluster to tackle the problem. We also discuss how the core algorithm is extended to provide a wide range of synteny block mining functionalities for various applications in comparative genomics research.

OrthoCluster can compare multiple chromosomes or compare a chromosome to itself. To keep our presentation concise, in this paper, we often elaborate ideas using examples comparing two chromosomes. Our discussion can be straightforwardly extended to the general case where more than two chromosomes are compared, as OrthoCluster is implemented.

2.1 The Synteny Block Mining Problem

Conceptually, a chromosome can be modeled as a sequence of genes. Roughly speaking, the synteny block mining problem in genome comparison is to compare genes in multiple genomes and identify fragments that contain similarly ordered genes as synteny blocks.

In order to conduct the comparison, a mapping between genes in the genomes under comparison is assumed. Biologically, mapping between genes in two genomes is made by "connecting" orthologous gene pairs. Orthologous gene pairs can be simply taken as the "same" genes in different genomes due to speciation, i.e., the orthologous genes share a common ancestor. The majority of the mappings between



Mapping

Gene in Chromosome 1	–	Gene in Chromosome 2
a	–	a'_1
a	–	a'_2
c	–	c'
g	–	g'
i	–	i'
j	–	j'
k	–	k'

Figure 1: Two chromosomes and the mapping of genes.

two closely related species such as the worms *C. elegans* and *C. briggsae*, or human and mouse, are called one-to-one relationships. Sometimes gene duplication can occur after speciation so that a gene in one species is duplicated while the ortholog of the gene in the other species remains a single copy. Such gene duplication events create numerous one-to-many or many-to-one relationships.

Generally, for two genomes C_1 and C_2, let $G(C_1)$ and $G(C_2)$ be the sets of genes in C_1 and C_2, respectively. When it is clear from context, we also write $G(C_1)$ and $G(C_2)$ as G_1 and G_2, respectively. A *mapping* between genes in C_1 and C_2 is a set of tuples $M_{C_1,C_2} = \{(g, g')\}$ where $g \in G(C_1)$ and $g' \in G(C_2)$. Gene g is called a *correspondence* of g' and g' a correspondence of g, too.

In general, a gene in a genome may have zero, one, or more than one correspondence gene in another genome. A gene is called an *in-map gene* if it has at least one correspondence gene in the other genome under analysis. Otherwise, it is called an *out-map gene*.

Each chromosome of a genome has two strands: a positive one and a negative one. A gene stays in one strand. Thus, a gene carries a "direction": either positive or negative.

For example, Figure 1 shows the segments of two chromosomes. The direction of a gene is represented using an arrow in the figure. A gene may be duplicated in a chromosome, such as a'_1 and a'_2 in the figure.

Structurally, a *synteny block* is a set of fragments from multiple genomes under analysis, one fragment from each genome, such that the genes in the fragments are well preserved (i.e., very similar to each other). Technically, we can model a synteny block as follows.

Given two genomes C_1 and C_2, a *synteny block* is a pair (S_1, S_2) such that S_1 and S_2 are subsequences in genomes C_1 and C_2, respectively, and S_1 and S_2 are similar to each other. To measure the similarity between S_1 and S_2, we consider the following factors.

Set similarity.

Let $\#(S_1/S_2)$ be the number of genes in S_1 that have at least a correspondence gene in S_2. Then, $\frac{\#(S_1/S_2)}{|S_1|}$ measures how well S_1 is presented by S_2 under the mapping. Similarly, $\frac{\#(S_2/S_1)}{|S_2|}$ reflects how well S_1 presents S_2 under the mapping. A user may specify a *set-similarity threshold* α. (S_1, S_2) is a cluster if $\frac{\#(S_1/S_2)}{|S_1|} \geq \alpha$ and $\frac{\#(S_2/S_1)}{|S_2|} \geq \alpha$.

Strandedness-aware similarity.

In this case, we consider the strandedness of the corresponding genes. Two cases may arise.

Case 1: Consistent strandedness. That is, a gene g in S_1 and a gene g' in S_2 are considered matched if g and g' are correspondences of each other, and g and g' are in the same type of strands (i.e., in the same direction, both positive or both negative) in the two genomes.

Case 2: Reversed strandedness. That is, a gene g in S_1 and a gene g' in S_2 are considered matched if g and g' are correspondences of each other, and g and g' are in the different type of strands (i.e., in different directions, one in positive and the other one in negative) in the two genomes.

In sequel, let $\#_{strand}(S_1/S_2)$ be the number of genes in S_1 that have at least a matched gene in S_2 under the consistent (or reversed) strandedness constraint. (S_1, S_2) is a synteny block if $\frac{\#_{strand}(S_1/S_2)}{|S_1|} \geq \beta$ and $\frac{\#_{strand}(S_2/S_1)}{|S_2|} \geq \beta$, where β is a user-specified *strandedness similarity threshold*.

Order-preserving similarity.

In this case, we consider the ordering of the corresponding genes. Again, due to the existence of the two strands, two cases may arise.

Case 1: Consistent order-preserving. That is, the genes in the fragments of the two genomes follow the same order under the mapping. Technically, a sequence of genes $g_1 \cdots g_n$ in S_1 and $g'_1 \cdots g'_n$ in S_2 are considered matched if g_i and g'_i $(1 \leq i \leq n)$ are correspondence genes of each other.

Case 2: Inverted order-preserving. That is, the genes in the fragments of the two genomes follow the inverted orders under the mapping. Technically, a sequence of genes $g_1 \cdots g_n$ in S_1 and $g'_1 \cdots g'_n$ in S_2 are considered matched if g_i and g'_{n+1-i} $(1 \leq i \leq n)$ are correspondence genes of each other.

In the above two cases, $g_1 \cdots g_n$ and $g'_1 \cdots g'_n$ are called the (order-preserving) *correspondence* sequences to each other. Let $\triangle(S_1, S_2)$ be the longest sequences of genes in S_1 which has a consistent (or inverted) order-preserving correspondence sequence in S_2. It is easy to see that $|\triangle(S_1, S_2)| = |\triangle(S_2, S_1)|$.

To measure the similarity, let $\#_{order}(S_1, S_2) = |\triangle(S_2, S_1)|$. (S_1, S_2) is a synteny block if $\frac{\#_{order}(S_1,S_2)}{|S_1|} \geq \gamma$ and $\frac{\#_{order}(S_1,S_2)}{|S_2|} \geq \gamma$, where γ is a user-specified *order-preserving similarity threshold*.

Often, synteny blocks of short fragments are insignificant in comparative genomics. For example, (g, g') is a trivial synteny block if g and g' are correspondence genes to each other. To avoid the triviality, a user can specify a minimum length constraint l_{min}. A synteny block (S_1, S_2) is *significant* if $|S_1| \geq l_{min}$ and $|S_2| \geq l_{min}$.

Simultaneously, a user can also specify a maximum length constraint l_{max} such that only synteny blocks (S_1, S_2) are returned if $|S_1| \leq l_{max}$ and $|S_2| \leq l_{max}$. The constraint is useful for closely-related genomes where the genes are relatively well preserved. Without the maximum length constraint, the whole chromosomes may be reported as a trivial synteny block when the settings of the other parameters are loose.

If both (S_1, S_2) and (S'_1, S'_2) are synteny blocks and $S'_1 \subset S_1$ and $S'_2 \subset S_2$., (S_1, S_2) is more interesting in comparative genomics since longer fragments of chromosomes often indi-

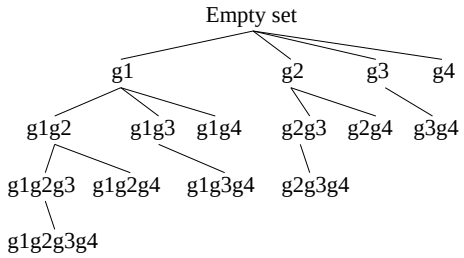


Figure 2: A set enumeration tree enumerating all synteny blocks of $G(C) = \{g_1, g_2, g_3, g_4\}$.

cate greater significance in biology. Larger synteny blocks often suggest stronger evolutionary constraint for maintaining the synteny blocks, while smaller nested blocks may be due to stochastic events.

To avoid the redundancy and focus on synteny blocks most interesting to users, we call a synteny block (S_1, S_2) *maximal* if there does not exist another synteny block (S_1'', S_2'') such that $S_1 \subseteq S_1''$, $S_2 \subseteq S_2''$ and at least one inequality holds. OrthoCluster outputs only maximal synteny blocks.

Although the above discussion involves only two chromosomes, it can be straightforwardly generalized to the synteny blocks among more than two genomes.

Problem definition. Given a set of genomes, the mapping between genes in the genomes, the similarity requirements, the user-specified similarity thresholds, and the minimum and/or maximum length threshold. The problem of *mining synteny blocks* is to find the complete set of maximal significant synteny blocks. ∎

The above problem definition can be easily extended to incorporate more user requirements. As will be discussed later, OrthoCluster provides a systematic set of parameters for users to specify various constraints. In the next subsection, we will first introduce a core algorithm using the set similarity. In Section 2.3, we will discuss how the core algorithm can be extended to handle other constraints.

2.2 The Core Mining Algorithm

In this section, we present the core data mining algorithm in OrthoCluster. The core algorithm tackles the essential version of the synteny block mining problem as follows.

Given two genomes C_1 and C_2, the mapping M between genes in the genomes, a set similarity threshold α, a maximum length threshold l_{max}, find the complete set of synteny blocks.

Consider the genes in C_1, $G(C_1) = \{g_1, \ldots, g_n\}$. We only need to consider the in-map genes in $G(C_1)$. For out-map genes in $G(C_1)$, we can replace them with a dummy symbol "−". To keep our presentation simple, let us overload $G(C_1)$ as the set of in-map genes in C_1 as well.

The set of synteny blocks can be divided into the following n subsets according to their fragments on C_1: the blocks containing g_1, the blocks containing g_2 but not g_1, the blocks containing g_3 but not g_1 or g_2, ..., and the blocks containing g_n but not $g_1, \ldots, g_{n-1}$.

Using a set enumeration tree [26], we can enumerate all possible blocks systematically. For example, suppose we have a genome C containing only 4 in-map genes, i.e., $G(C) = \{g_1, g_2, g_3, g_4\}$. Figure 2 shows a set enumeration tree to enumerate all synteny blocks according to their fragments on genome C. Since we are interested in only the synteny blocks whose fragments have up to l_{max} genes, we only need to grow the set enumeration tree up to l_{max} levels.

A straightforward method to find the complete set of synteny blocks checks every node in the set enumeration tree of $G(C_1)$. For each node which represents a subset X of $G(C_1)$, we check whether there exist some fragments S_1 in C_1 such that $X \subseteq S$ and $\frac{|X|}{|S_1|} \geq \alpha$. If so, for each such a fragment S_1, we check whether there exist some fragments S_2 in C_2 such that the following two conditions hold: (1) S_2 contains a subset of genes Y such that every gene in X has exactly one correspondence in Y under the mapping M; and (2) $\frac{|Y|}{|S_2|} \geq \alpha$. If so, for each such a S_2, (S_1, S_2) is a synteny block. After all synteny blocks are found, only the maximal ones are output.

The above straightforward method can be very costly when a chromosome has many genes and l_{max} is not small: the set enumeration tree is huge. Generally, if $G(C_1)$ has m in-map genes, there are in total $\sum_{i=1}^{l_{max}} \binom{m}{i}$ nodes in the tree. Searching all nodes in a huge set enumeration tree is computationally prohibitive.

In OrthoCluster, we developed an efficient algorithm to tackle the challenges in searching the set enumeration tree. It searches genome C_1 gene by gene from left to right.

As the preprocessing, we scan C_1 and C_2 once. For each tuple (g, g') in the mapping M, we find the occurrences of g and g' in C_1 and C_2, respectively. Therefore, after the preprocessing, we can find the locations of g (g') in C_1 (C_2) in constant time.

When OrthoCluster meets an in-map gene g_1 in genome C_1, we check a window $W(g_1)$ of size $(2l_{max} - 1)$ on C_1 with g_1 as the center, as shown in Figure 3. Window $W(g_1)$ contains both in-map and out-map genes. Let $Tail(g_1)$ be the set of in-map genes in $W(g_1)$. The genes in $Tail(g_1)$ may find correspondences in genome C_2.

After we generate the initial tail of $\{g_1\}$, we refine $Tail(g_1)$ by removing from it the genes that cannot form a synteny block with g_1. This refining process works as follows.

For a tuple (g_1, g_1') in the mapping M, we find the occurrences of g_1' in genome C_2. For each occurrence g_1', we also check a window $W(g_1')$ of size $(2l_{max} - 1)$ on C_2 with g_1' as the center, as shown in Figure 3. Let $Tail(g_1')$ be the set of in-map genes in $W(g_1')$.

Then, for each gene $g_i \in Tail(g_1)$, we count the number of genes that is located between g_1 and g_i and do not have a correspondence in $Tail(g_1')$. If this number exceeds $(1 - \alpha)l_{max}$, then g_i is removed from $Tail(g_1)$. This is because if g_1 and g_i form a synteny block, its set similarity must be less than α.

Symmetrically, we can remove genes in $Tail(g_1')$. We apply the above process iteratively until no more genes can be removed from either $Tail(g_1)$ or $Tail(g_1')$. We call this process the *iterative refining* process.

After refining $Tail(g_1)$ and $Tail(g_1')$, if at least one of the tail sets is empty, then (g_1, g_1') is a synteny block. If both tail sets are not empty, we search the set enumeration tree of $Tail(g_1)$ in a depth-first manner. For each gene $g_i \in Tail(g_1)$, we generate a child $X = \{g_1\} \cup \{g_i\}$ of $\{g_1\}$ in the set enumeration tree. Suppose an order of genes is adopted to enumerate gene combinations as in a set enumeration

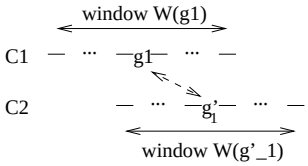


Figure 3: Matching two genes.

tree. We form $Tail(X)$ as the set of genes in $Tail(g_1)$ that is ordered after g_i. Only those genes can be used to extend X to a large block. We refine $Tail(X)$ in the same way as we refine $Tail(g_1)$. After the refinement, the children of X are generated and searched in a recursive way.

If the subtree of node X is searched and no other synteny block is found in the subtree, or if X does not have children at all, we check the minimal fragments $W(M(X))$ in $W(g_1')$ where $M(X)$ is the set of correspondences of genes in X. If the set similarity requirement is satisfied, i.e., $\frac{|X|}{|W(X)|} \geq \alpha$ and $\frac{|M(X)|}{|W(M(X))|} \geq \alpha$, then $(W(X), W(M(X)))$ is a synteny block and is output. This guarantees the maximality of the synteny block.

We have also developed two pruning techniques which can effectively prune the search space.

First, a synteny block $(W(X), W(M(X))$ already found can be used to prune the search of the set enumeration tree. A block containing a subset of gene set Y cannot contain any gene not in $Tail(Y)$. When searching a node Y in the set enumeration tree such that $Y \subset X$, if $Tail(Y) \subseteq X$, then a block containing Y but not some genes in X cannot be maximal, and thus can be pruned.

Second, consider a gene $g_2 \in Tail(g_1)$ such that both g_2 and the correspondence gene g_2' occur at the same side (i.e., either to the left or to the right) of g_1 and g_1', respectively. If the subtree of g_2 is searched and no synteny blocks are found, then, for any gene g_3 and the correspondence gene g_3' such that $g_1 \cdots g_2 \cdots g_3$ $(g_3 \cdots g_2 \cdots g_1)$ and $g_1' \cdots g_2' \cdots g_3'$ $(g_3' \cdots g_2' \cdots g_1')$ appear in $W(g_1 g_3)$ and $W(g_1' g_3')$, respectively, it is impossible to have a synteny block containing $g_1 g_3$, since such a block, if exists at all, can be extended by g_2 to form a larger block. This contradicts with the precondition that there is no synteny block having $\{g_1, g2\}$.

The above mining procedure is run recursively until the depth-first search is completed. Limited by space, we omit the implementation details. The OrthoCluster is publicly available. Further details can be found in the technical document comes with the software.

Comparing to the straightforward method, the core algorithm in the OrthoCluster can prune the search space substantially. It uses the l_{max} constraint to focus on only the genes which can form synteny blocks. It further prunes search branches using the synteny blocks found before. Our empirical study and the case studies show that OrthoCluster is efficient for large genome (e.g., containing thousands of genes). The straightforward method cannot run on those large genome since the set enumeration tree on all genes is too huge and there are too many nodes to search.

2.3 Extensions for Various Constraints

In this section, we discuss how to extend the core algorithm to handle various constraints.

2.3.1 Minimum Length Constraint l_{min}

If the minimum synteny block length constraint is specified, it can be incorporated into the core algorithm.

First, we do not output any synteny blocks that fail the minimum length constraint. Second, when a node X in the set enumeration tree is searched, if $|X| + |Tail(X)| < l_{min}$, then the node and its subtree can be pruned since it cannot satisfy the minimum length constraint.

2.3.2 Strandedness-aware Constraint

When the strandedness-aware constraint presents, the core algorithm is revised as follows.

When an in-map gene g_1 is met, we consider all correspondences of g_1. For a correspondence g_1' of g_1 such that g_1 and g_1' have the same direction, we use the consistent strandedness to map fragments in $W(g_1)$ and $W(g_1')$. On the other hand, if g_1' has a direction different from g_1, then we simply use the reversed strandedness to map fragments in $W(g_1)$ and $W(g_1')$.

2.3.3 Order-preserving Constraint

Meeting an order-preserving constraint is a little more complicated.

When a gene g_1 is met, we cannot determine the order yet. However, when we search a child node of the root in the set enumeration tree of $Tail(g_1)$, the order (i.e., either consistent order or inverted order) can be determine.

Suppose $g_2 \in Tail(g_1)$ is searched whose correspondence is g_2'. If $g_1 \cdots g_2$ $(g_2 \cdots g_1)$ and $g_1' \cdots g_2'$ $(g_2' \cdots g_1')$ appear in $W(g_1)$ and $W(g_1')$, respectively, then the consistent order should be used in the matching in the subtree of $g_1 g_2$. Otherwise, i.e., $g_1 \cdots g_2$ $(g_2 \cdots g_1)$ and $g_2' \cdots g_1'$ $(g_1' \cdots g_2')$ appear in $W(g_1)$ and $W(g_1')$, respectively, then the inverted order should be used.

Moreover, when searching a child $X \cup \{g\}$ of node X, i.e., a new gene $g \in Tail(X)$ is added into X, consider a gene $g_i \in Tail(X \cup \{g\})$. Let g' and g_i' be the correspondence of g and g_i, respectively. If g_i' appears in a side of g' different from g_i appears to g, then g_i and g_i' can be removed from $Tail(X \cup \{g\})$ if consistent order is used, since they violate the order. Symmetrically, we have a pruning rule for the inverted order.

2.4 Genome Rearrangements Identification

OrthoCluster also integrates the functionalities for identifying genome rearrangements including *reciprocal translocation*, *transposition*, *inversion* and *insertion/deletion*. Here we briefly describe their definitions and the operations OrthoCluster performs to identify them. These definitions are consistent to those described previously [10, 34].

Reciprocal translocation. Two nonhomologous chromosomes that exchange chunks of DNA by recombinations are called reciprocal translocations. For detecting reciprocal translocations, people usually merge synteny blocks such that they cannot be further merged to form longer blocks that have only been fractured by duplications, inversions, or transpositions [10, 27]. OrthoCluster provides users with the functions to achieve that goal.

Transposition. A transposition is a chunk of DNA that is excised from one chromosome and inserted into a nonhomologous chromosome. For each adjacent pair of blocks CL_a and CL_b in the reference genome, OrthoCluster searches the region between their corresponding fragments in the other

genome, CL'_a and CL'_b. If a fragment of less than 50 genes is found between CL'_a and CL'_b, then a transposition is identified.

Inversion. An inversion is a DNA segment inverted in the genome. For each synteny block in the reference genome, if the order of genes in the corresponding fragment of the other genome is reversed, then an inversion is identified.

Insertion/deletion. An insertion/deletion is a DNA segment that is inserted into or deleted from a genomic region. For every two adjacent synteny blocks in the reference genome, if the out-map genes are found between the two blocks, then an insertion/deletion is reported.

3. CASE STUDIES AND EMPIRICAL EVALUATION

In this section, we present three cases studies of biologically meaningful findings using OrthoCluster. We also evaluate the data mining techniques in OrthoCluster using real data sets.

3.1 Data Preparation

The datasets used for case studies and other analysis are obtained from WormBase (http://www.wormbase.org/), an integrated biological and genomics database for the model organism nematode *C. elegans* [8], which is the first multiple cellular organism whose genome was subject to whole genome sequencing [11]. We apply OrthoCluster to identify synteny blocks in the genomes of *C. elegans* and its sister species *C. briggsae* [34], two species split in evolution about 100 million years ago (MYA), for the following reasons. 1) The *C. elegans* genome is the only multiple cellular organism whose genome sequence is complete, with no remaining gaps. 2) The *C. elegans* genome is the most extensively curated genome and the *C. briggsae* genome has been well curated as well. 3) Some synteny blocks have been identified in previous studies [34, 10]. The results of these studies will be compared against the results produced by OrthoCluster.

Four datasets have been prepared for testing the performance of OrthoCluster and case studies: (1) *C. elegans* genes ordered according to their genomic coordinates, (2) ordered *C. briggsae* genes, (3) a *C. elegans*-*C.briggsae* gene mapping file, and (4) a file listing duplicated genes in the *C. elegans* genome. To obtain these datasets, we installed a local WormBase mirror site, which hosts a WormBase stable release WS170, according to the mirror installation procedure available at WormBase. A series Perl scripts, which use Perl modules in BioPerl [33] and gbrowse [35], were written to extract ordered genes in both *C. elegans* and *C. briggsae* genomes.

There are approximately the same numbers of genes ($\sim 20,000$) in these two genomes. The *C. elegans*-*C. briggsae* mapping file was based on the orthologous relationships between genes in these two genomes. Genes from different species are orthologous if these genes share common ancestors. Three types of orthologous relationships are considered here. The one-to-one relationship represents a relation that a gene in one species (e.g., *C. elegans*) has only one orthologous gene in another species (e.g., *C. briggsae*). The one-to-many and the many-to-one relationships represent relations that a gene in one species has multiple orthologous genes in another species and vice versa. This happens when a gene is duplicated in one genome, while the same gene in the other species remains as a single copy. The orthologous relationships were produced using a widely used program InParanoid [24] and gene sets from WormBase WS170. The file listing duplicated genes in *C. elegans* was generated by carrying out all-against-all BLAST [2] searches for all *C. elegans* genes. Perl scripts were used to parse the BLAST results. Two genes are regarded to be duplicated genes if they show certain level of identity.

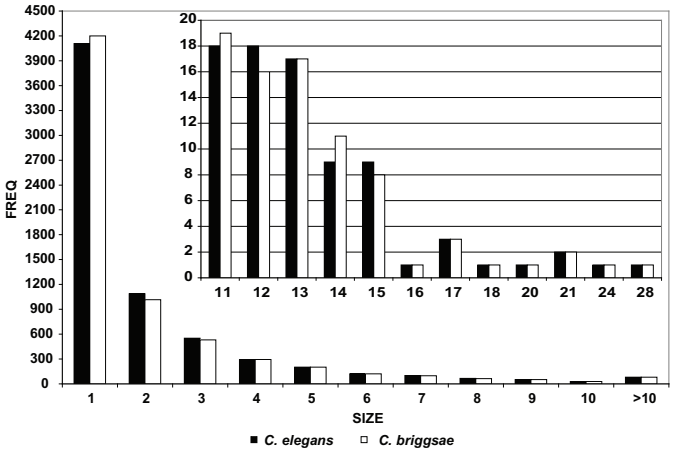


Figure 4: Size distribution of perfectly conserved non-nested blocks, obtained by OrthoCluster preserving order and strandedness (rs), $l_{max} = 1000$.

3.2 Case Studies

We report three interesting case studies conducted by working biologists using OrthoCluster.

3.2.1 Perfectly conserved synteny blocks

Genomes of different species may show little similarity in an obvious way – they may have different genome sizes, different number of chromosomes, and different number of protein-coding genes-because of extensive genome rearrangement events in evolution. Nevertheless, genome segments perfectly conserved among these species can be found and these segments may play essential role. Additionally, genes within these conserved blocks may be co-regulated due to some locus control regions (LCR), which are regulatory genomic sequences that control the expression of a group of genes, in contrast to many other regulatory genomics sequences that control the expression of single genes. Identification of these blocks, especially an extreme class of so called perfectly conserved synteny blocks (defined as blocks of genes that share exactly same order, strandedness and contain no mismatches) may therefore provide insights into the function and regulation of genes.

Before the completion of the *C. briggsae* genome and using ad hoc programs, [10] estimates $5,816$ perfectly conserved blocks between the genomes of *C. elegans* and *C. briggsae*. These segments contain a variable number of genes, ranging from 1 to 19 genes.

Using OrthoCluster and a complete genome of *C. briggsae*, we have identified $6,690$ perfectly conserved synteny blocks (shown in Figure 4), comparable to the estimated number by Coghlan and Wolfe [10]. However, the largest perfectly conserved synteny blocks identified by OrthoClus-

ter are much larger than those identified in the previous project, partly due to the availability of only 13% of the *C. briggsae* genome sequence to the previous project. The largest perfectly conserved block identified using OrthoCluster contains 28 genes, in comparison to 19 genes identified previously. The chromosome that contains the largest number and the highest density of perfectly conserved synteny blocks in *C. elegans* is chromosome V (0.077 blocks/kb) and the chromosome that contains the least number and the lowest density of perfectly conserved synteny blocks is X (0.036 blocks/kb). Interestingly, the largest synteny blocks reside in these two chromosomes as well, with the largest synteny block (containing 28 genes) in chromosome V and a synteny block of size 24 in chromosome X. The significance of these blocks and their existence on these two chromosomes will be subject of further biological analysis.

Figure 4 shows the size distribution of perfectly conserved blocks. The number of blocks decreases as the size of blocks become larger, which is explained by the fact that smaller blocks accomplishing the constraints imposed by the parameters of preserving order and strandedness (rs) are easier to find. In the same figure, a zoom for the size distribution of the largest blocks is shown. A few considerably large blocks are found, which should be subject of biological scrutiny in order to detect functionality associated to the blocks.

3.2.2 Inversions

Inversion is a type of common genome rearrangement event, in which the corresponding synteny blocks of two genomes reside on opposite genome strands. Since most of the existing synteny block identifying programs (Table 2 in Section 4) do not take into account gene strandedness, those programs are not appropriate for identifying inversions.

Using OrthoCluster, we have identified $1,837$ perfectly conserved inverted synteny blocks between the genomes of *C. elegans* and *C. briggsae*. These inverted synteny blocks contain no insertions or deletions and do not contain any mismatches. The number of genes in these synteny blocks varies from 2 to 24. About 75% of these blocks contain 3 or less genes. The results are consistent with [10], which indicates that the majority of the inversions are shorter than 25 kb. The largest perfectly conserved inverted segments correspond to three blocks containing 18, 21 and 24 genes in chromosome X and one block of 21 genes in chromosome V.

When we allow some mismatches (5% in-map mismatches and 20% out-map mismatches), we identified 1265 inverted synteny blocks composed of two or more genes. The number is similar compared to the number of inverted synteny blocks reported before (938 by [10] and $1,384$ by [34]) using *ad hoc* methods, both allowing mismatches in inverted synteny blocks. The blocks allowing mismatches, however, are much larger compared to those perfectly conserved inverted blocks. About 44% of these blocks contain 3 or more genes. The largest corresponding inverted synteny block contains 134 genes in *C. elegans* and 116 genes in *C. briggsae*. Again, these numbers are much larger than the largest inverted synteny blocks in *C. elegans* and *C. briggsae* genomes [10]. The effective identification of inverted synteny blocks will facilitate analysis of inversion breakpoints.

3.2.3 Segmental duplications

Segmental duplication, which is defined as low-copy repeats of DNA segments, has been proposed to be important in genome organization and evolution [25]. Genome analysis has revealed large range segmental duplications in species ranging from yeast to human. However, segmental duplications have not been extensively investigated in the *C. elegans* genome. Previous studies produced only a limited number of small (three) segmental duplications (called "regional duplications") [30]. Each of these predicted segmental duplications contains only three genes.

Using OrthoCluster, we have found many large segmental duplications, in which each gene in one segment shows at least 90% identity at the protein sequence level to its corresponding gene in the duplicated segment, in the *C. elegans* genome. There are 44 duplicated segments that each contains three or more genes, with the largest duplicated segment containing 9 genes. About 880 (approximately 4.4% of all genes in the *C. elegans* genome) genes are contained in these duplicated segments. If we relax the protein level identity requirement from 90% to 80%, 75 duplicated segments that contain three or more genes can be identified, with the largest block containing 16 genes. The number of duplicated segments with three or more genes increases to 123 with the largest one containing 26 genes when the identity requirement is further relaxed to 70%. The number of duplicated segments that contain three or more genes and the maximum size reaches a plateau when the protein identity requirement is relaxed to below 70%.

These duplicated segments are as "perfectly conserved" ones since no mismatches were allowed in the predictions. The largest duplicated segments, each contains a surprising large number of 26 genes, are located on chromosome V (Figure 5). Two genome segments are adjacent on chromosome V. Each segment is 109 Kb long and contains 26 genes. In the figure, the numbers indicate genome coordinates. Each glyph in the "Gene Models"" track represents a single gene. Within each gene, the boxes represent exons and the lines represent introns. The genes on the positive strand are shown in purple, while genes on the negative strand are shown in blue. These images are taken from WormBase. These two segments, which are in tandem, may be recently duplicated because of the perfect conservation of orders. The structure and the function of these duplicated blocks warrant further detailed experimental analysis.

3.3 Performance Evaluation

OrthoCluster is implemented using C++ with STL support. The source code is publicly available. We conducted extensive empirical evaluations of OrthoCluster on various genome data sets. Here we report some selected results.

All experiments were conducted on a PC computer running the Fedora Core 6 operating system, with a 1.6 GHz Pentium 4 CPU, 512 MB main memory, and a 60 GB hard disk. The programs were compiled using gcc/g++.

3.3.1 Parameters

Table 1 lists the important parameters of OrthoCluster. By setting the r (or s) option, a user can find order-preserving (or strandedness-preserving) synteny blocks. Additionally, the rs option can be used if a user is interested in finding only those blocks with consistent order and consistent strandedness, and with inverted order and reversed strandedness.

The similarity thresholds α, β and γ can be set by com-

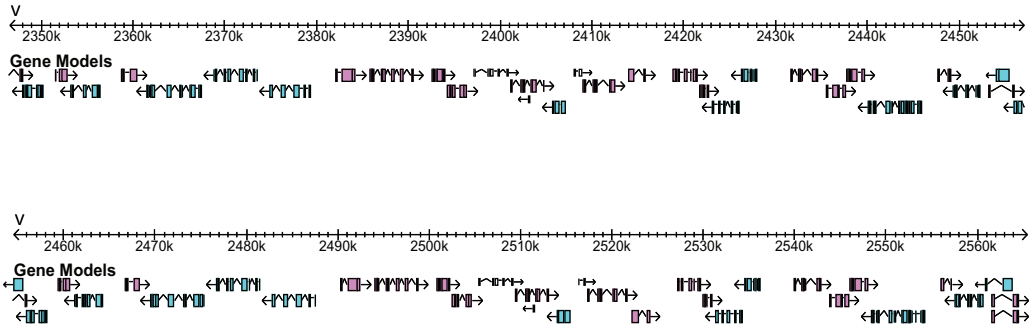


Figure 5: The largest duplicated segments in the *C. elegans* genome

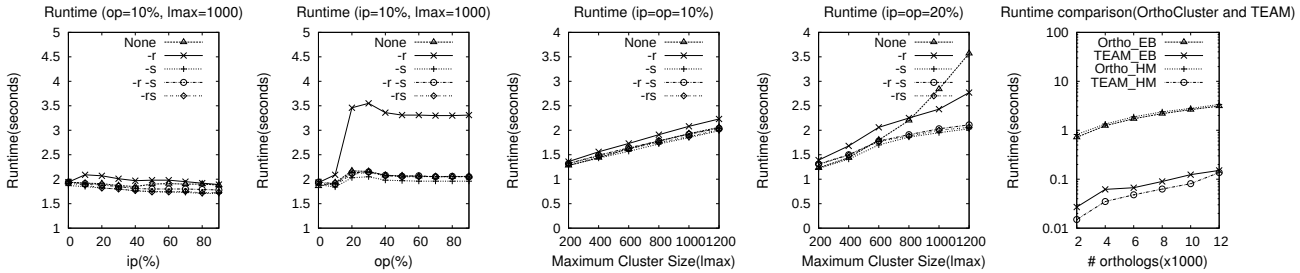


Figure 6: The runtime of OrthoCluster on *C. elegans* and *C. briggsae* data set with respect to ip, op and $lmax$, and runtime comparison with TEAM.

bining parameters ip, op, s and r. In particular, ip is the maximal percentage of mismatched in-map genes allowed in a synteny block, and op is the maximal percentage of out-map genes allowed in a synteny block. By setting ip (or op), a user can control the number of genes involved in transpositions (or insertions/deletions) in a synteny block. Thus, if a user wants to find synteny blocks with *set similarity* $\geq \alpha$ but does not care about the order and the strandedness, s/he can set ip and op such that $ip + op = 1 - \alpha$. Similarly, if a user is interested in synteny blocks with *Strandedness-ware similarity*$\geq \beta$, s/he can set ip and op such that $ip+op = 1-\beta$ and choose the s option at the same time.

For large synteny blocks consisting of dozens of genes, the parameters ip and op work well. To find small synteny blocks with many mismatched (either in-map or out-map) genes, however, setting ip and op to high values is not a good idea since many large synteny blocks with low similarity may also be identified. To overcome this problem, OrthoCluster provides two parameters, i and o, which specify the absolute number of in-map and out-map mismatches allowed. A set of genes is still a valid synteny block as long as it satisfies the i(or o) constraint even if it violates the ip (or op) constraint.

3.3.2 Runtime

Figure 6 shows the runtime of OrthoCluster on two genomes: *C. elegans* and *C. briggsae*. Please note that the runtime reported here includes the time for loading and cleaning the data sets, mining all clusters, detecting genome rearrangements and writing the final output.

Parameter	Functionality
l_{max}	the upper bound on the number of genes in each cluster
l_{min}	the lower bound on the number of genes in each cluste
ip	maximal percentage of mismatched in-map genes allowed
i	maximal number of mismatched in-map genes allowed
op	maximal percentage of out-map genes allowed
o	maximal number of out-map genes allowed
r	find order-preserving clusters
s	find strandedness-preserving clusters
rs	find order and strandedness preserving blocks

Table 1: Parameters of OrthoCluster.

We run OrthoCluster with four conservation levels: non-order and non-strandedness preserving (none), order preserving (-r), strandedness preserving (-s), both order and strandedness preserving (-r -s), and consistent order and consistent strandedness or inverted order and reversed strandedness (-rs). On each pair of genomes, we first set the pa-

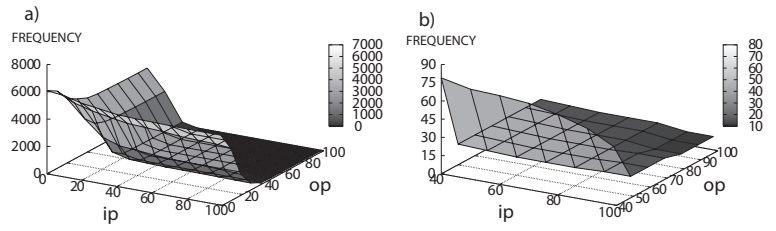


Figure 7: Number of synteny blocks with respect to *ip* and *op*.

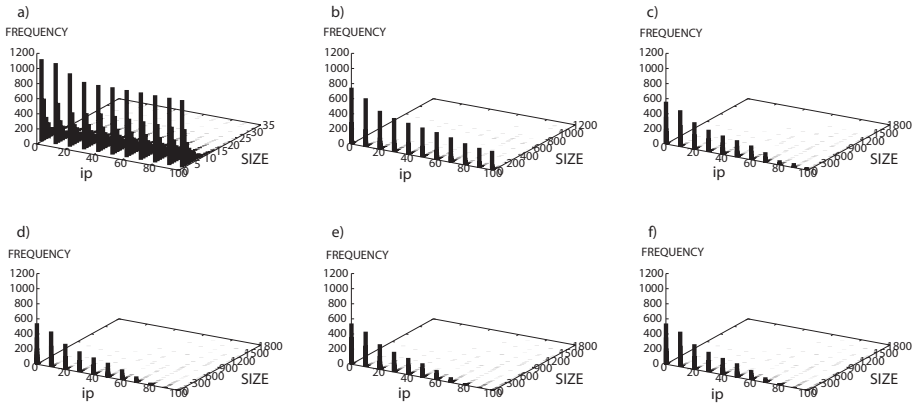


Figure 8: The distribution of synteny block size with respect to *ip*. $l_{max} = 1000$. (a) $op = 0$, (b) $op = 20$, (c) $op = 40$, (d) $op = 60$, (e) $op = 80$, (f) $op = 99$. Synteny blocks of size one are not plotted for clarity.

rameter $ip = 10\%$ and vary the parameter *op*, then we set $op = 10\%$ and vary the parameter *ip*. Figures 6(a) and (b) show that, in a wide range of parameter settings, OrthoCluster finishes within less than 4 seconds.

It is worth mentioning that the runtime for different conservation levels does not show significant differences, though non-order and non-strandedness option typically takes less time, while the order-preserving option takes relatively longer time. The figure also shows that relaxing out-map mismatches *op* and in-map mismatches *ip* does not necessarily increase the search time.

The above observations may seem to be counter-intuitive, since the larger *op* or *ip*, or the looser the conservation is, the more genes would be included in the tail, thus the size of the search tree increases. However, it should also be noted that relaxing *ip* and *op* can increase the sizes of the synteny blocks, and thus more search sub-trees can be pruned by our techniques using those synteny blocks already found.

Figures 6(c) and (d) show the runtime of OrthoCluster on the *C. elegans* and *C. briggsae* data sets when the maximal synteny block size l_{max} varies. In Figure 6(c), both *ip* and *op* are set to 10% while, in Figure 6(d), both are set to 20%. The trend in the two figures shows that the runtime increases as l_{max} increases, but the increase is mild: the runtime is still less than 4 seconds when l_{max} is set to more than $1,000$. This figure shows the effectiveness of our pruning techniques. Again, for different conservation levels, the runtime does not differ significantly, as observed in Figure 6.

Figure 6(e) shows the runtime comparison between OrthoCluster and another tool TEAM [21]. The functionalities of OthoCluster and TEAM are quit different and the differences are discussed in Section 4. They produce the same results only when OrthoCluster runs with $i = o = ip = op = 0$ and TEAM runs with the maximal number of genes in the gap allowed equals to 0. In the experiment we range the number of orthologs, and run both tools using the above settings on two pairs of genomes: *C. elegans* and *C. briggsae* (EB), and *Human* and *Monkey*(HM). Figure 6(e) shows that both OrthoCluster and TEAM scale well when the number of ortholog increases, and OrthoCluster takes more time than TEAM. The reason for the runtime difference is that, to handle one to many mapping, the input data of OrthoCluster is split into three files, while the input of TEAM can be presented in one file. Thus OrthoCluster needs more IO cost than TEAM. In addition, with three input files, OrthoCluster needs more preprocessing time. Third, OrthoCluster is implemented in a recursive fashion. The cost of recursive calls also contributes to the runtime difference. Nevertheless, as both tools can finish in a reasonable amount of time (less than 5 seconds), to the users they do not make big difference.

We also conducted experiments on other data sets such as *Human* and *Chimp*, *Human* and *Monkey*. Similar trends were observed. Limited by space, we omit the details here.

3.3.3 Sensitivity analysis

To evaluate the robustness and the consistency of the output from OrthoCluster, we conducted sensitivity analysis by running OrthoCluster using various parameters. The number of synteny blocks decreases when more mismatches are allowed within a block (Figure 7), but the blocks are larger. In Figure 7, $l_{max} = 1000$.

In addition, the distribution of the size of synteny blocks tends to be composed of fewer and larger synteny blocks when *ip* increases and *op* is fixed (Figure 8), and when *op* increases and *ip* is fixed (Figure 9).

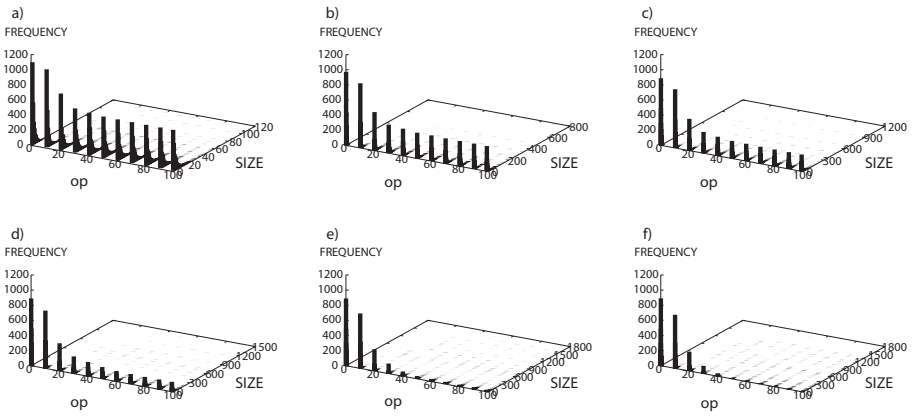


Figure 9: The distribution of synteny block size with respect to *op*. $l_{max} = 1000$. (a) $ip = 0$, (b) $ip = 20$, (c) $ip = 40$, (d) $ip = 60$, (e) $ip = 80$, (f) $ip = 99$. Synteny blocks of size one are not plotted for clarity.

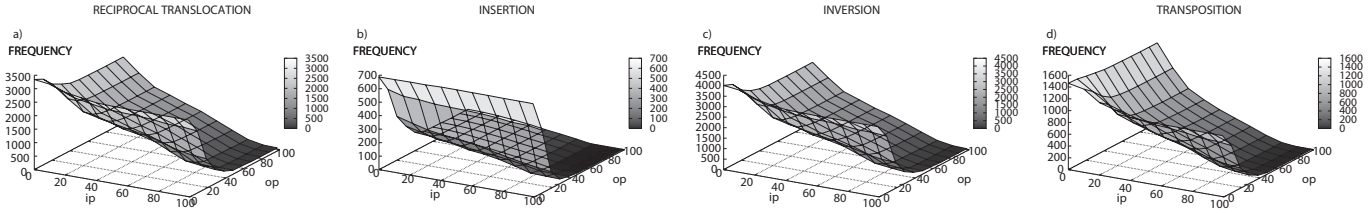


Figure 10: Number of rearrangements with respect to *ip* and *op*. $l_{max} = 1000$.

Software	Order	Strandedness	Flexibler r/s	Mismatches	Duplications	Multi-genomes
Cinteny	YES	YES	NO	YES	NO	YES
FISH	NO	NO	NO	YES	YES	NO
TEAM	NO	NO	NO	YES	NO	YES
DAGchainer	YES	NO	NO	YES	NO	NO
ADHoRe	YES	YES	NO	YES	YES	NO
OrthoCluster	YES	YES	YES	YES	YES	YES

Table 2: Criteria applied on different tools for comparison.

The number of rearrangements between genomes decreases when more mismatches are allowed within a synteny block (Figure 10). Allowing more mismatches produces fewer and larger blocks, hence reduces the possibility of all types of rearrangements being detected. In the case of insertions, the value of *ip* is stable when $op = 0$. This is expected since the variation of the number of in-map mismatches should have no effect on such type of rearrangement when no out-map mismatch occur.

Figure 10(a) shows the frequency of reciprocal translocations with respect to the number of mismatches. The number of reciprocal translocations decreases when more mismatches are allowed. An increasing number of mismatches implies fewer and larger blocks. Figure 10(b) shows the frequency of insertions with respect to the number of mismatches. The number of insertions decreases when more mismatches are allowed. An increasing number of mismatches implies fewer synteny blocks, which are larger and hence are composed by more out-map mismatches. Figure 10(c) shows the frequency of inversions with respect to the number of mismatches. The number of inversions decreases when more mismatches are allowed. An increasing number of mismatches implies larger synteny blocks. Figure 10(d) shows the frequency of transpositions as a function of mismatches. The number of transpositions decreases when more mismatches are allowed. An increasing number of mismatches implies fewer blocks, which are larger and hence are composed by more in-map mismatches.

4. COMPARISON WITH OTHER TOOLS

In this section we compare several programs for identifying synteny blocks with OrthoCluster. The tools presented here cover a wide range of different approaches for finding synteny blocks, and are currently widely used. Table 2 summarizes the comparison.

Many existing synteny block identification tools like FISH [6], SyMAP [32], ADHoRe [37] and DiagHunter [7] represent homology between two genomes in a matrix or dot plot, where a non-zero value is assigned to a pair of markers that are homolog with each other, and a zero is assigned otherwise. By doing so, synteny blocks are visualized as diagonals in the matrix, and the purpose of the algorithms is to automatically detect those diagonals in a way that allows for some deviations from collinearity, such as duplications (represented by horizontal or vertical line in the matrix), and rearrangements such as insertions and

deletions. A distance measure (typically a Manhattan or Manhattan-like distance) may be defined and used to detect diagonal elements, and a dynamic programming algorithm is applied in order to detect synteny blocks. Those tools share a few important weaknesses. First, those tools work well with pairwise genome analysis, but they are incapable of identifying synteny blocks among more than two genomes. Second, none of those methods handles one-to-many relationships between genes properly. Third, they are unaware of strandedness and therefore they cannot resolve inversions properly. In particular, the program SyMAP is specifically designed for aligning physical map to genomics sequences. In sequel, some parameters have been predefined for this purpose. Hence, it is not ready for general synteny block detection.

Other programs, like Cinteny [31] or TEAM [21], define different frameworks for detecting synteny blocks.

Also, some other programs have been developed specifically to understand the set and the number of rearrangements that occurred between two genomes since their divergences from a common ancestor species. For example, in GRIMM [36], genomes are represented as signed permutations, i.e., one or more vectors of signed numbers, with each number representing a gene. The algorithm computes the minimum number of rearrangement steps to transform one genome to the other. No insertions or deletions are allowed.

Although these comparative genomics programs can be used to identify synteny blocks and some genome rearrangement events in many cases, they suffer from some important limitations. For example, most of these programs cannot handle the strandedness of genes. Therefore, they cannot be used to identify genes inversions. Neither can they resolve one-to-many relationships on genes where gene duplications happen in some genomes but not others. In addition, most of those programs cannot work on more than two genomes. Finally, most of the programs are hard to use, and their output are hard to interpret.

The criteria used to compared the software were the following when searching for synteny blocks: whether order preserving, whether strandedness-aware, whether flexible with respect to the handling of order (r) and strandedness (s), whether allowing mismatches, whether duplications between genomes can be handled, and whether multiple genomes can be handled.

Cinteny [31] represents genes in a tree structure in which each node is a character that conforms the official symbol of each gene. At the leaves of the tree, the homologous groups can be found vertically and the linear order in which each gene is located on each chromosome can be found horizontally, for each species. Hence, blocks can be obtained by "walking" horizontally through the leaves. Even though this program can handle multiple genomes and considers order and strandedness when generating synteny blocks, it does not allow the user to vary these two last properties to its convenience. Unlike OrthoCluster, this program allows for mismatches by means of a parameter that corresponds to a maximum gap measured in kb, rather than number or percentage of genes within blocks. In addition, it does not identify transposition and skips small scale rearrangement events. Finally, this program allows the user to handle duplications (paralogs) between genomes with different strategies, but all of them are such that consider only one instance of the duplicate and discard the rest, or remove all of them.

FISH [6] (Fast Identification of Segmental Homology) is a program conceived to detect highly diverged synteny blocks. For each pair of contigs, a matrix is built. Each cell of each matrix is a "1" when features are homologs, and a "0" otherwise. In this way, blocks can be seen on each matrix as a clump of closely spaced points ("1s") in a roughly diagonal line. The objective is to discern when a clump of closely spaced points is unlikely to have occurred by chance. To achieve this, they define a null model for homologies among individual features in the absence of segmental homology and compute a p-value of observing a clump of size k. Given the focus of this program on highly diverged sequences, it does not consider order or strandedness when finding synteny blocks. Also, it fails in handling multiple genomes and, even though it allows for mismatches, the parameter for achieving this feature corresponds to a probability, which makes it quite indirect as method of control of mismatches within blocks.

TEAM [21] focuses on finding all gene sets that are placed closely in all the input genomes regardless the order and strandedness of each gene. In each genome, this set of genes can be separated by gaps, but the size of the maximal gap should not exceed a user-defined parameter δ. Like OrthoCluster, TEAM can work on multiple genomes. The functionality of TEAM is also similar to OrthoCluster when order and strandedness are not enforced. However, there are several critical differences between TEAM and OrthoCluster. First, unlike OrthoCluster, users of TEAM have little control over the compactness of the set of genes found: even δ is set to 1, the number of genes in the gap can be almost as large as the number of genes in the block itself. In addition, TEAM is unable to find order-preserving or strandedness-preserving sets of genes. Third, limited by its two previous disadvantages, TEAM cannot detect inversion, transposition and deletion. Finally, TEAM cannot handle duplicate genes.

DAGchainer [15] identifies chains of genes sharing conserved order in all the input genomes. It works by constructing a directed acyclic graph (DAG) and then extracting the path having highest scores using a dynamic programming approach. Compared with OrthoCluster, it can only work on two genomes and it does not allow part of the block to be inverted, thus it is unable to find highly diverged blocks. It does not take care of the strandedness of genes; neither does it provide the transposition and insertion/deletion detection functionalities.

ADHoRe [37] (Automatic Detection of Homologous Regions) is a tool that detects genomic regions with statistically significant conserved gene content and order. It represents homologies in a matrix and consists of three main steps: 1) Preprocessing of the data, in which irrelevant points are removed and tandem duplications are remapped, 2) Clustering of genes and blocks of genes, in which blocks are found separately for pairs of genes with the same orientation and pairs of genes with opposite orientation, and 3) postprocessing, in which the blocks obtained are tested for statistical significance and the results for the 2 classes of orientation are combined. This program is not flexible about the inclusion/exclusion of strandedness and order by the user, even though it considers these properties when finding synteny blocks. It handles duplicates to some extent by collapsing all tandem duplications of a gene with the same orientation and within a certain distance when preprocessing data, thus

making easier the search for diagonals in the matrix. Finally, it fails to handle multiple genomes.

5. CONCLUSIONS

In this paper, we present the major technical ideas and some interesting findings of a novel data mining tool OrthoCluster for comparative genomics analysis. OrthoCluster addresses several important challenges that the previous synteny block finding tools cannot handle. OrthoCluster has been adopted by working biologists and bioinformatics researchers.

As the next step, the web-based version of OrthoCluster will be released in the near future, which will also serve as a public database of synteny blocks. Moreover, the next version of OrthoCluster is in plan which can incorporate domain knowledge and known patterns about synteny blocks in finding new blocks.

Acknowledgement

We are grateful to the anonymous reviewers for their constructive comments.

X. Zeng and J. Pei were supported in part by an NSERC Discovery grant, an SFU CTEF grant, and an IBM Faculty Award. N. Chen was supported in part by an NSERC Discovery grant and an SFU CTEF grant. N. Chen is an MSFHR Scholar. K. Wang was supported in part by an NSERC Discovery grant. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

6. REFERENCES

[1] M. D. Adams, et al. The genome sequence of drosophila melanogaster. *Science*, 287(5461):2185–95, 2000.

[2] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25:3389–3402, 1997.

[3] G. Bejerano, et al. Ultraconserved elements in the human genome. *Science*, 304(5675):1321–5, 2004.

[4] E. Birney, et al. Identification and analysis of functional elements in 1genome by the encode pilot project. *Nature*, 447(7146):799–816, 2007.

[5] T. Blumenthal. Operons in eukaryotes. *Brief Funct Genomic Proteomic*, 3(3):199–211, 2004.

[6] P. P. Calabrese, S. Chakravarty, and T. J. Vision. Fast identification and statistical evaluation of segmental homologies in comparative maps. *Bioinformatics*, 19(Supplement 1):i74–80, 2003.

[7] S. B. Cannon, et al. Diaghunter and genopix2d: programs for genomic comparisons, large-scale homology discovery and visualization. *Genome Biology*, 4(10):R68, 2003.

[8] N. Chen, T. W. Harris, I. Antoshechkin, C. Bastiani, T. Bieri, D. Blasiar, K. Bradnam, P. Canaran, and J. Chan C. K. Chen, et al. Wormbase: a comprehensive data resource for caenorhabditis biology and genomics. *Nucleic acids research*, 33:D383–389, 2005.

[9] N. Chen and L.D. Stein. Conservation and functional significance of gene topology in the genome of caenorhabditis elegans. *Genome Res*, 16(5):606–17, 2006.

[10] A. Coghlan and K. H. Wolfe. Fourfold faster rate of genome rearrangement in nematodes than in drosophila. *Genome research*, 12:857–867, 2002.

[11] Consortium. Genome sequence of the nematode c. elegans: a platform for investigating biology. *Science*, 282(5396):2012–8, 1998.

[12] J. Couzin. Human genome. hapmap launched with pledges of $100 million. *Science*, 298(5595):941–2, 2002.

[13] R.D. Fleischmann, et al. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, 269(5223):496–512, 1995.

[14] A. Goffeau, et al. Life with 6000 genes. *Science*, 274(5287):546, 563–7, 1996.

[15] B. J. Haas, A. L. Delcher, J. R. Wortman, and S. L. Salzberg. DAGchainer: a tool for mining segmental genome duplications and synteny. *Bioinformatics*, 20(18):3643–6, 2004.

[16] R. C. Hardison. Comparative genomics. *PLoS Biol*, 1(2):E58, 2003.

[17] L. D. Hurst, C. Pal, and M.J. Lercher. The evolutionary dynamics of eukaryotic gene order. *Nat Rev Genet*, 5(4):299–310, 2004.

[18] F. Jacob, et al. Operon: a group of genes with the expression coordinated by an operator. *C R Hebd Seances Acad Sci*, 250:1727–9, 1960.

[19] E. S. Lander, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

[20] S. Levy, et al. The diploid genome sequence of an individual human. *PLoS Biol*, 5(10):e254, 2007.

[21] N. Luc, et al. Gene teams: a new formalization of gene clusters for comparative genomics. *Computational Biology and Chemistry*, 27(1):59–67, 2003.

[22] W. Miller, et al. Comparative genomics. *Annu Rev Genomics Hum Genet*, 5:15–56, 2004.

[23] E. W. Myers, et al. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–204, 2000.

[24] K. P. O'Brien, M. Remm, and E. L. Sonnhammer. Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic acids research*, 33:D476–480, 2005.

[25] S. Ohno. *Evolution by Gene Duplication*. Springer-Verlag, New York, 1970.

[26] R. Rymon. Search through systematic set enumeration. In *Proc. 1992 Int. Conf. Principle of Knowledge Representation and Reasoning (KR'92)*, pages 539–550, Cambridge, MA, 1992.

[27] D. Sankoff. Comparative mapping and genome rearrangement. *From Jay Lush to genomics: Visions for animal breeding andgenetics*, pages 124–134, 1999.

[28] J. Sebat. Major changes in our dna lead to major changes in our thinking. *Nat Genet*, 39(7 Suppl):S3–5, 2007.

[29] J. Sebat, et al. Strong association of de novo copy number mutations with autism. *Science*, 316(5823):445–9, 2007.

[30] C. Semple and K.H. Wolfe. Gene duplication and gene conversion in the caenorhabditis elegans genome. *Journal of molecular evolution*, 48:555–564, 1999.

[31] A. U. Sinha and J. Meller. Cinteny: flexible analysis and visualization of synteny and genome rearrangements in multiple organisms. *BMC Bioinformatics*, 8, 2007.

[32] C. Soderlund, et al. Symap: A system for discovering and viewing syntenic regions of fpc maps. *Genome Research*, 16(9):1159–68, 2006.

[33] J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, J. G. Gilbert, I. Korf, and H. Lapp, et al. The bioperl toolkit: Perl modules for the life sciences. *Genome research*, 12:1611–1618, 2002.

[34] L. D. Stein, Z. Bao, D. Blasiar, T. Blumenthal, M. R. Brent, N. Chen, A. Chinwalla, L. Clarke, C. Clee, and A. Coghlan, et al. The genome sequence of caenorhabditis briggsae: a platform for comparative genomics. *PLoS Biol*, 1:E45, 2003.

[35] L. D. Stein, C. Mungall, S. Shu, M. Caudy, M. Mangone, A. Day, E. Nickerson, J. E. Stajich, T. W. Harris, and A. Arva, et al. The generic genome browser: a building block for a model organism system database. *Genome research*, 12:1599–1610, 2002.

[36] G. Tesler. GRIMM: genome rearrangements web server. *Bioinformatics*, 18(3):492–493, 2002.

[37] K. Vandepoele, et al. The automatic detection of homologous regions (adhore) and its application to microcolinearity between arabidopsis and rice. *Genome Research*, 12(11):1792–1801, 2002.

[38] J. C. Venter, et al. The sequence of the human genome. *Science*, 291(5507):1304–51, 2001.

[39] X. Xie, et al. Systematic discovery of regulatory motifs in conserved regions of the human genome, including thousands of ctcf insulator sites. *Proc Natl Acad Sci U S A*, 104(17):7145–50, 2007.