Formal models of gene clusters

Anne Bergeron, Yannick Gingras
CGL, Université du Québec à Montréal

Cedric Chauve
CGL, Université du Québec à Montréal
Department of Mathematics, Simon Fraser University

To appear in *Bioinformatics Algorithms: Techniques and Applications* edited by I. Mandoiu and A. Zelikovsky, *Wiley Book Series on Bioinformatics*

Formal models of gene clusters

Anne Bergeron[1], Cedric Chauve[1,2], and Yannick Gingras[1]

[1] CGL, Université du Québec à Montréal, Canada
[2] Department of Mathematics, Simon Fraser University, Burnaby, Canada

1 Introduction

Genomes evolve through small-scale events, like point mutations in the DNA sequence, and large-scale events, known as *rearrangements*, that reorganize the genetic material along chromosomes [23, 39, 47]. Such rearrangements can involve very few genes, like the mutation/loss of genes due to the accumulation of point mutations, tandem duplications or short reversals, but can also be at a much higher scale, like large reversals or even whole-genome duplications. It results from these evolutionary events that, when comparing two or more genomes in terms of their gene orders, *i.e.* the order of genetic markers along their chromosomes, it is very unlikely that these gene orders are identical, even for very close species; see for example recent works on primates [43], or on different strains of a same bacterial organism [35]. However, gene orders are not random, and the comparison of gene orders of related species shows genome segments that exhibit homogeneous gene content, with sometimes similar gene order. These groups of segments are usually called *conserved gene clusters*, or gene clusters, the notion of conserved being implicit.

Conserved gene clusters can be the result of several biological mechanisms but, basically, they could be defined as "genomic regions that share a common ancestor" [28]. For example, they can result from functional pressure that requires that a group of genes stays close along the genomes. The most widely studied example of such groups of genes are operons in prokaryotic genomes, that are transcribed in a single messenger RNA and need then to have their genes located contiguously in the genome [35, 36, 51]; it has also been suggested that being part of a given biochemical network [45] or co-expression [42] could be correlated with belonging to a same gene cluster. Segments of genomes with homogeneous gene content can also result very mechanically from the evolutionary proximity between the genomes: there was not enough time from the speciation events leading to observed genomes to break such groups of genes. This phenomenon can add noise in the detection of functional gene clusters, but it is worth noting that the resulting clusters carry an important information for computing evolution scenarios [6, 7], reconstructing ancestral genomes [9], or identifying positional and ancestral homologs [12, 17].

The detection of conserved gene clusters is a challenge for both the biological and mathematical communities, with applications in comparative genomics, annotation of genomes, and phylogenomics. There are essentially two families of methods for detecting conserved gene clusters. A first approach attacks this problem on very pragmatic grounds, primarily based on the detection of short conserved genomic segments, such as consecutive gene pairs, that are easy to detect. This collection of conserved segments is then processed, in general using an heuristic, to obtain a set of gene clusters. See [37] for a survey on this topic. However, these methods lack a formal definition of the notion of gene cluster in terms of genomic segments and sets of genes involved in clusters. This can produce incoherent results as clustering short conserved segments can form genomic segments that have very different gene content, but are grouped in a same cluster.

In this chapter, we survey a different approach in the detection of gene clusters that has been developed in the last years by various groups of researchers in gene order analysis. This approach relies on 1) formal definitions of what is a set of genomic segments that defines a conserved gene cluster, and 2) algorithms that search and compare genomes of a dataset to detect all the sets of genomic segments that satisfy a given definition. Starting with conserved adjacencies and common intervals, we will work through several variants that allows for duplications and missing genes. We will also show that most variants are necessary, in the sense that they try to capture a biological reality that does not care about formal models. We will also see that the algorithmic challenges in detecting gene clusters are non-trivial. Depending on the model, efficient solutions can be easy to obtain, can require a lot of ingenuity, or sometimes do not even exist.

2 Genome plasticity

2.1 Genome representations

The genetic information of species is stored in molecules called *chromosomes* that consist of two complementary strands of DNA. Each DNA strand is a string of basic units, the nucleotides, that have two extremities, called the 5' and 3' extremities. Consecutive nucleotides are connected by joining the 3' extremity of one to the 5' extremity of the other. Figure 1 illustrates a small segment of chromosome: note that the 5' and 3' extremities of the two strands of DNA are in opposite directions. Nucleotides on a strand come in four different types, cytosine, guanine, adenine and thymine, abbreviated respectively by the letters **C**, **G**, **A** and **T**. The nucleotides on the complementary strand are uniquely determined by the Watson-Crick complement relation in which **C** is always paired with **G**, and **A** is always paired with **T**.

From an information processing perspective, the knowledge of only one strand is sufficient to recover the whole molecule. However, the information contained on a strand is decoded in a sequential way going from the 5' extremity to the 3' extremity, allowing both strands to carry biologically meaningful information. It is thus customary to represent the information present on a chromosome by identifying a substring of one strand of a chromosome, and specifying in which *orientation* it must be read, positive or negative. These substrings can correspond to any type of features found on a chromosome: genes, domains, operons, synteny blocks, banding patterns, etc. We will refer collectively to such features

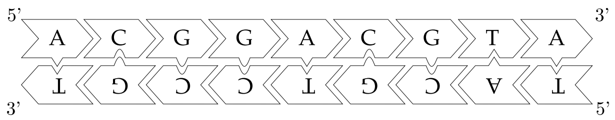


Fig. 1. The double strand structure of chromosomes.

as "genes", using a more detailed terminology when necessary. We will also assume that genes are non-overlapping substrings of chromosomes. This is not always the case, especially for small organisms such as viruses that have to compress their genetic information in very short chromosomes, but this assumption constitutes a reasonable compromise. A chromosome, or part of it, can thus be represented by a string such as:

$$(a \quad b \quad -c \quad d \quad -e \quad f),$$

in which letters stand for genes, and a negative sign signals a negative orientation.

The comparison of gene content and order between two species also relies in a fundamental way on our ability to tell when two genes are the "same" or not. This is not a trivial task, and numerous approaches have been tried since the discovery of the first rearranged chromosomes. In [22], Dobzhansky and Sturtevant divided the genome of fruit flies in 100 arbitrary sections that were called "genes" and that were recognizable under the microscope by their banding patterns. Two genes from different genomes were "equal" if they shared the same pattern.

When the term gene is taken in its modern sense of a sequence of nucleotides that is transcribed into RNA, most of the techniques of identification of similar genes rely on sequence alignments [44], either of the nucleotide sequences themselves, or the corresponding amino acid sequences if the RNA molecules are translated into proteins. This produces reliable results, especially for slow evolving genes such as those found in animal mitochondria [14], for example.

In some recent studies, such as [16], based on very large synteny blocks, equivalent segments of chromosomes are detected using small anchors whose order and orientation is conserved between species. This approach allows to ignore rearrangements whose size fall below a fixed threshold.

Whatever the approach used to define equal genes, comparing gene orders ultimately boils down to compare strings of signed letters. For a fixed set $\mathcal{S}$ of genes, when strings representing two genomes contain exactly one occurrence of each gene in $\mathcal{S}$, in either orientation, we refer to these strings as *permutations*. For example, the following strings

$$G_1 = (a \quad b \quad -c \quad d \quad -e \quad f) \text{ and } G_2 = (-a \quad d \quad -e \quad b \quad -c \quad f)$$

are permutations of the set of genes $\{a, b, c, d, e, f\}$. The strings

$$G_3 = (a \quad b \quad -c \quad a \quad d \quad -e \quad f) \text{ and } G_4 = (-a \quad d \quad -e \quad -c \quad f)$$

are not permutations since gene a is duplicated in genome G_3, and gene b is missing in genome G_4.

The choice of representing genomes by permutations or strings with duplicated genes has implications on both the biological and computational sides of genome analysis. Indeed, as we will see later, the computational complexity of handling strings with duplicated genes is higher than for permutations. At the same time, from a biological point of view, it is a strong hypothesis to assume that a gene is present in a single copy in a set of genomes. Thus, aside from a few exceptions, such as animal mitochondrial genomes that have few duplicated genes, representing a set of genomes by a set of permutations often implies a non-trivial preliminary analysis that clears ambiguities due to duplicated genes; examples of such "preprocessing" can be found in the alignment process used to define synteny blocks [16], or in the process of ortholog assignments [5, 24].

2.2 Genome rearrangements

The goal of this section is to give the reader some intuitions about the nature and extent of genome rearrangements that happened during evolution.

Rearrangements that preserve gene content These type of rearrangements can be described as the result of a series of cut and join operations performed on the chromosomes of a genome. Breaking and repairing chromosomes into new configurations can affect gene order and gene orientation. While the first modification is easy to understand, the change in gene orientation deserves a little more attention since it is a consequence of the double strand structure of the chromosomes.

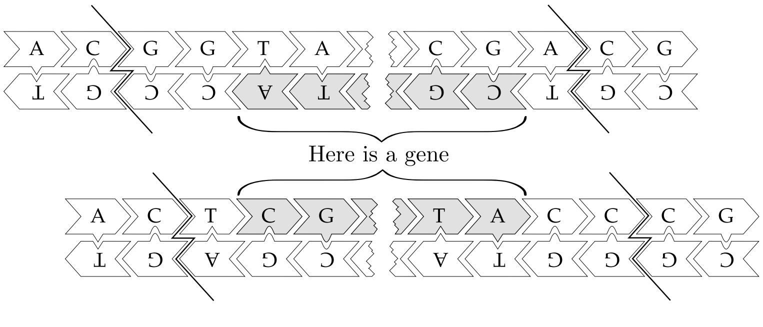
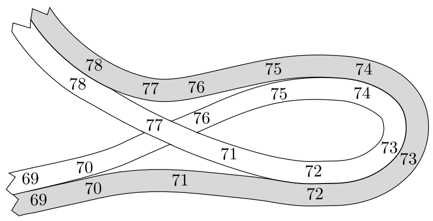


Fig. 2. An inversion changes the orientation of the genes of the inverted segment.

The upper part of Figure 2 shows a chromosome which is broken at two places. The segment in the middle can either be correctly repaired, or repaired in the wrong direction, as shown in the lower part of Figure 2. However, since a 3' extremity can only be reconnected to a 5' extremity, this results in the exchange of strands. Genomic features that are on one of the strands of the inverted segment thus change their orientation, but are otherwise intact.

In one of the first papers on genome rearrangements [22], Dobzhansky and Sturtevant observed *inversions* of large segments of chromosomes in closely related species of *Drosophila* (see Fig. 3). When comparing more distant species, hundreds of small and large inversions can be detected [16].



69 70 71 72 73 74 75 76 77 78
69 70 76 75 74 73 72 71 77 78

Fig. 3. Fragment of the dataset constructed in 1938 by Dobzhansky and Sturtevant to compare whole genomes.

Closely related to inversions are *translocations* between chromosomes. This happens when two different chromosomes exchange parts of their genetic material. Even in relatively close species, translocations can be extensive. For example, a comparison between human and concolor gibbon chromosomes reveals that at least 31 translocations are necessary to explained the observed rearrangements [31]. *Fusions* and *fissions* of chromosomes are special cases of translocations that modify the total number of chromosomes of an organism. A striking example of this type of rearrangement is given by the human chromosome 2 which is the result of a fusion of two smaller chromosomes found in most great apes [30].

A last rearrangement operation that does not change gene content is the *transposition*, in which a genomic segment is displaced within a chromosome, but keeps its original orientation. Animal mitochondrial genomes, for example, provide numerous examples of transposed elements [48].

Rearrangements that modify gene content Gene content is modified by three main processes: duplications, losses, and exchanges of genetic material with close or symbiotic species.

Duplications can affect whole genomes: the bread wheat (*Triticum aestivum*) has 21 pairs of chromosomes that can be grouped in 7 groups of triplicated chromosomes. This organization is quite recent, and the triplicated chromosomes retain a high similarity [41]. More ancient duplications are more difficult to establish, since many genes are lost or change function. For example, evidence of an ancient whole genome duplication was discovered in yeasts [50]. Having access to extra genetic material meant that the evolving yeasts could "experiment" with part of their genetic material while retaining vital functions: as a result, modern yeasts with duplicated genomes are adapted to many different environments, but different species can have very different gene content.

On a smaller genomic scale, duplications are extensive. Most sequenced genomes contain segments that are similar at the nucleotide level, but at different places along the same chromosome or even on different chromosomes. Domains, genes, or even groups of genes, can be present in several hundred copies in a genome. Table 1 gives examples of the number of occurrences of four different domains in two species of α-proteobacteria, as identified by the Pfam database [3].

PFAM Family	*Agrobacterium tumefaciens*	*Rhizobium meliloti*
ABC_tran PF00005	497	262
BPD_transp1 PF00528	420	189
HTH_1 PF000126	149	95
LysR_substrate PF03466	148	89

Table 1. Number of occurrences of four domains in two species of α-proteobacteria.

3 Basic concepts

Before introducing formally the notion of gene cluster, we give an intuitive definition. Basically, a gene cluster is a *set of genes* that, for biological reasons, have been kept "more or less together", or in other words, *in a same segment* in different genomes. There are several biological reasons that can prevent a set of genes to be dispersed by the rearrangements that occur during the evolution: co-transcription, as operons for example [36, 51], co-regulation as part of a biochemical network [45], or evolutionary proximity.

From the combinatorial point of view, it is important to note the very different nature of the two notions used to define a cluster, that is sets – with no order between the genes in a given set – and genome segments, that are strings of genes, and then totally ordered. The key point in a formal definition of a gene cluster is then the relationship between the set of genes and the genomes segments that defines the cluster. We develop this approach in the next section.

3.1 Highlighting sets of genes

In order to formalize the notion of sets of genes that are "more or less together" in different genomes, we must first look a the problem in a more general context: How do sets of letters behave in strings or permutations? For example, *highlighting* the set of letters {e, g, n} in the sentence "rearrangements of genomes involve genes and chromosomes",

reveals groups that, at least in the case of the substrings "gene" and "gen", share a common – semantic – function. On the other hand, highlighting the set of letters {a, c, v} in the same sentence

"rearrangements of genomes involve genes and chromosomes",

does not seem to produce any interesting result.

These examples illustrate a major pitfall that one has to face in defining clusters as sets of genes: there is an exponential number of sets of genes. As we will see it with max-gap clusters in strings for example, this can result in models that define an exponential number of gene clusters, ruling out any hope of a polynomial time detection of all gene clusters in such models. However, depending on both the combinatorial nature of the considered data (strings or permutations) and the definition of genome segments that define clusters, it is possible to define gene cluster models that are tractable.

A second question is what is an *occurrence* of a gene cluster. In our example, it seems reasonable to consider the substrings "nge", "gene" and "gen" as occurrences of the set {e, g, n} but, given the proximity of the substrings "nge" and "en" in the word "rearrangement" – they are separated by a single letter –, it might be interesting to join them together in a single occurrence by bridging the *gap* of one letter. This would reduce the number of occurrences of the cluster defined by the set {e, g, n} to two: "gene" and "ngemen", this last occurrence being a non-exact occurrence due to the letter "m". Such a decision to accept an outsider in an occurrence of a cluster may also be wise from a biological point of view since genes can be gained or lost, or can change function, which translates, in terms of strings, as gaps or as missing genes.

This example illustrates the importance of the definition of the genome segments that define a cluster. In this chapter, we take the point of view to consider the intruder genes in occurrences of clusters under the point of view of the gaps they create, which was followed in most recent approaches[3]. We thus have the following definition:

Definition 1. *Given a set of genes S, and a chromosome C represented by the string $g_1 g_2 \ldots g_k$, an* occurrence *of the set S is a substring $g_i \ldots g_j$ such that:*

1) Both g_i and g_j belong to the set of genes S.

2) The set of genes S is contained in the multiset $\{g_i, \ldots, g_j\}$.

3) If a substring of $g_i \ldots g_j$ contains no gene of S, then its length must be smaller or equal to δ, a fixed integer that represents the maximal allowed gap *size.*

4) The flanking substrings $g_{i-1-\delta} \ldots g_{i-1}$ and $g_{j+1} \ldots g_{j+1+\delta}$ contain no gene in S. (Extremities of chromosomes are padded with characters not in S as necessary.)

When $\delta = 0$, we usually refer to occurrences as "without gaps".

[3] Note however that a different point of view was taken in [19].

With a gap size $\delta = 0$, the three occurrences of the set {e, g, n} in the sentence "rearrangements of genomes involve genes and chromosomes" are the substrings "nge", "gene" and "gen"; with a gap size $\delta = 1$, there are again three occurrences, but the first occurrence is now the substring "ngemen".

The next basic concept is illustrated by the following example. When the set of letters {m, o, s} is highlighted in the sentence

"rearrangements of genomes involve genes and chromosomes",

the two occurrences of the set with gap size $\delta = 1$ are the substrings "omes" and "omosome". The presence of the letter 'e' in both occurrences suggests that a more meaningful set of genes to consider could be the set {e, m, o, s}. The relationship between these two sets is captured by the following definition.

Definition 2. *Given a set of genes S, and genomes represented by a set of strings $\mathcal{G}$, a set of genes T is an* extension *of the set S if:*
1) $S \subset T$,
2) Each occurrence of S in $\mathcal{G}$ is a substring of an occurrence of T.

In the example sentence, with $\delta = 1$, the two occurrences of the set {e, m, o, s} are "enomes" and "omosomes", both of which contain an occurrence of {m, o, s} as a substring. Since {m, o, s} has only two occurrences, {e, m, o, s} is an extension of {m, o, s}. On the other hand, the set {e, g, n, m} is not an extension of the set {e, g, n} since there is one occurrence of {e, g, n}, namely the occurrence within the word "gene", that is not a substring of an occurrence of {e, g, n, m}.

The point of view of this chapter will be purely combinatorial. Once gene clusters are identified, it is necessary to distinguish clusters whose existence could be merely due to random factors, from those whose existence rests on other causes. These aspects are studied, for example, in [29].

3.2 An elementary model: conserved segments

Various formal models for gene clusters are obtained by imposing requirements on the subsets of genes that are allowable, on the type of strings that are considered, and on the number and nature of the occurrences.

Perhaps the simplest model is when we assume that genomes are permutations of each other, and require occurrences of clusters to be equal. Because strings represent DNA molecules, the notion of equality must be adapted to capture the fact that an inverted string is the "same" as the original. Two strings $g_1 g_2 \ldots g_k$ and $h_1 h_2 \ldots h_k$ are *equal* if either 1) for all i, $g_i = h_i$, or 2) for all i, $g_i = -h_{k-i+1}$. We have:

Definition 3. *Let $\mathcal{G}$ be a set of signed permutations on the set of genes $\mathcal{S}$. A subset of $\mathcal{S}$ is a* conserved segment *if it has an occurrence in each permutation of $\mathcal{G}$, without gaps, and all occurrences are equal.*

Conserved segments capture the notion of sets of genes that occur in the same order and same orientation in different genomes. For example, consider the following permutations:

$$G_1 = (a \quad b \quad -c \quad d \quad -e \quad f) \text{ and } G_2 = (-a \quad d \quad -e \quad f \quad c \quad -b).$$

Apart from the trivial conserved segments formed by singletons, the sets $\{b, c\}$, $\{d, e\}$, $\{e, f\}$, $\{d, e, f\}$ are all conserved segments. We are usually interested in *maximal* conserved segments, that is, conserved segments without extension. In this example, these are $\{a\}$, $\{b, c\}$ and $\{d, e, f\}$. In each permutation, the boundary between two occurrences of maximal conserved segments are called *breakpoints*. For example, the breakpoints of G_1 with respect to G_2 are

$$G_1 = (a \;\|\; b \quad -c \;\|\; d \quad -e \quad f).$$

Maximal conserved segments form a partition of the set $\mathcal{S}$, and appear as consecutive elements in each permutation of $\mathcal{G}$, which are very desirable properties from a computational point of view. Identifying them can be done in $\mathcal{O}(Kn)$ time complexity, where K is the number of permutations in $\mathcal{G}$, and n the number of genes in $\mathcal{S}$. The computation involves the following steps:

1) Choose an arbitrary ordering I of the set of genes $\mathcal{S}$.

2) For each permutation in $\mathcal{G}$, construct a table indexed by I which gives the position and sign of the corresponding gene.

3) Choose one permutation G in $\mathcal{G}$ and test, for each consecutive pair of genes $g_i g_{i+1}$ of G, whether it is a conserved segment. If it is not, mark a breakpoint between g_i and g_{i+1}.

When using very large datasets, it is a current practice to consider conserved segments as a single "gene", since there are no rearrangements within these segments. This is especially true when comparing close species, which often share very long conserved segments. For example, in the following three permutations, each integer represents a large conserved block in the chromosomes X of the human, mouse and rat; reverse orientation is indicated by overlined integers. This dataset is adapted from [16].

$$\begin{array}{lll} Human & = & (1 \; 2 \; 3 \; 4 \; 5 \; 6 \; 7 \; 8 \; 9 \; 10 \; 11 \; 12 \; 13 \; 14 \; 15 \; 16) \\ Mouse & = & (\overline{6} \; \overline{5} \; 4 \; 13 \; 14 \; \overline{15} \; 16 \; 1 \; \overline{3} \; 9 \; \overline{10} \; 11 \; 12 \; \overline{7} \; 8 \; \overline{2}) \\ Rat & = & (\overline{13} \; \overline{4} \; 5 \; \overline{6} \; \overline{12} \; \overline{8} \; \overline{7} \; 2 \; 1 \; \overline{3} \; 9 \; 10 \; 11 \; 14 \; \overline{15} \; 16) \end{array}$$

By construction, this set of permutations do not have any conserved segment. However, the comparison of these permutations two by two reveals conserved segments for each pair of species. For example, rat and mouse share the segment $\{4, 5, 13\}$, rat and human share $\{5, 7\}$, and mouse and human share $\{5, 6\}$.

This type of analysis was used, for example, in phylogenetic studies and for the reconstruction of ancestral genomes [11]. For larger sets of permutations, it is possible to relax the definition of conserved segments, and ask for occurrences in at least J permutations in the set $\mathcal{G}$, but not necessarily in all. This is done at the cost of added algorithmic complexity.

Most properties of conserved segments do not hold anymore when the chromosomes contain duplicated genes. Indeed, the definition of conserved segment can be extended to strings, but they do not form a partition of the set $\mathcal{S}$ of genes, neither do they define clear breakpoint regions (see [13] for an example of using conserved segments to define breakpoints in a set of strings). We observe a similar phenomenon with the various gene clusters models in the next section.

4 Models of gene clusters

4.1 Common intervals in permutations

The notion of common intervals is a first generalization of conserved segments in which we relax the conditions that genes appear in the same order or the same orientation. It was first introduced by Uno and Yagiura [49] in the case of 2 permutations, and various efficient algorithms have since been developed for K permutations. Since orientation is not necessarily conserved, in this section we ignore the signs of genes.

Definition 4. *Let $\mathcal{G}$ be a set of permutations on the set of genes $\mathcal{S}$. A subset of $\mathcal{S}$ is a* common interval *if it has an occurrence in each permutation of $\mathcal{G}$, without gaps.*

Intuitively, a common interval is a set whose elements are consecutive in both permutations, thus intervals in both permutations. Note that singletons are always common intervals, and are sometimes referred to as *trivial* common intervals. Consider the following two permutations:

$$G_1 = (\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ 10\ 11\ )$$
$$G_2 = (\ 4\ \ 2\ \ 1\ \ 3\ \ 7\ \ 8\ \ 6\ \ 5\ 11\ \ 9\ 10\ ).$$

The common intervals of G_1 and G_2, except for the singletons, are underlined in the second permutation. When, as it is the case in this example, one of the permutation is the identity permutation, all common intervals are sets of consecutive integers since their occurrence in the identity permutation is an interval.

Underlining the same common intervals in the identity permutation highlights some of the properties of common intervals of permutations:

$$G_2 = (\ 4\ \ 2\ \ 1\ \ 3\ \ 7\ \ 8\ \ 6\ \ 5\ 11\ \ 9\ 10\ )$$
$$G_1 = (\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ 10\ 11\ ).$$

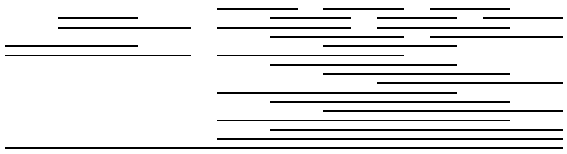
For example, if two common intervals have a non trivial intersection, such as $\{1, 2, 3, 4, 5, 6, 7, 8\}$ and $\{5, 6, 7, 8, 9, 10, 11\}$, then the intersection of these common intervals is also a common interval, since it is an interval in both permutations. More formally we have:

Definition 5. *Two sets S and T* overlap *if:*
1) their intersection is non empty, and
2) neither S is contained in T, nor T is contained in S.

Proposition 1. *Let S and T be two overlapping common intervals of a set of permutations $\mathcal{G}$, then $S \cap T$, $S \cup T$, $S \setminus T$ and $T \setminus S$ are all common intervals of the set $\mathcal{G}$.*

The number of common intervals of a set of permutations on n elements is $\mathcal{O}(n^2)$. This bound is achieved, for example, when one compares two equal permutations: each interval is then a common interval and there are $(n+1)(n/2)$ of them. On the other hand, from a biological perspective, a huge number of common intervals merely reflects that parts of genomes under study are very similar. This fact has hindered the use of common intervals in comparative genomics, because most of common intervals in "real" genomes are very repetitive. For example, the following permutations

$$\begin{array}{l} G_3 = (\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ 10\ 11\ ) \\ G_4 = (\ 4\ \ 2\ \ 3\ \ 1\ 11\ 10\ \ 9\ \ 8\ \ 7\ \ 6\ \ 5\ ) \end{array}$$

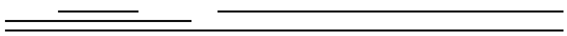


have 26 non trivial common intervals, but most of them seems to be related, such as $\{5,6\}$, $\{6,7\}$, and $\{5,6,7\}$. A significant advance was made in recent years to formally distinguish interesting common intervals from less interesting ones [33]. It is the identification of *strong* common intervals:

Definition 6. *Let $\mathcal{G}$ be a set of permutations, a* strong *common interval is a common interval of $\mathcal{G}$ that do not overlap any other common intervals of $\mathcal{G}$.*

For example, permutations G_3 and G_4 have only 4 strong common intervals, that are underlined in G_4, and these illustrate very nicely the respective structures of the two permutations:

$$\begin{array}{l} G_3 = (\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ 10\ 11\ ) \\ G_4 = (\ 4\ \ 2\ \ 3\ \ 1\ 11\ 10\ \ 9\ \ 8\ \ 7\ \ 6\ \ 5\ ). \end{array}$$



Strong intervals have very rich and deep combinatorial properties [10], as well as they capture relevant biological relations [7,33]. One of their most attractive features, from both point of views, is that there are few of them, and these few can be used to generate all common intervals. The proof of the following proposition is worth reading, since it introduce a basic construction.

Proposition 2. *Let $\mathcal{G}$ be a set of permutations on n elements, the number of strong common intervals of $\mathcal{G}$ is in $\mathcal{O}(n)$.*

Proof. Two strong common intervals are either disjoint, or one is contained in the other. All singletons and the set $\{1, 2, \ldots, n\}$ are strong common intervals. Consider the tree in which each strong common interval is the child of the smallest strong common interval that properly contains it. Each node of this tree has thus at least two children, its root is the set $\{1, 2, \ldots, n\}$, and its leaves are the singletons. Therefore, the total number of its internal nodes is less than n.

It is thus possible to report and display the strong common intervals as a tree using the inclusion relation. For example, the tree in Figure 4 corresponds to the strong common intervals of G_3 and G_4.

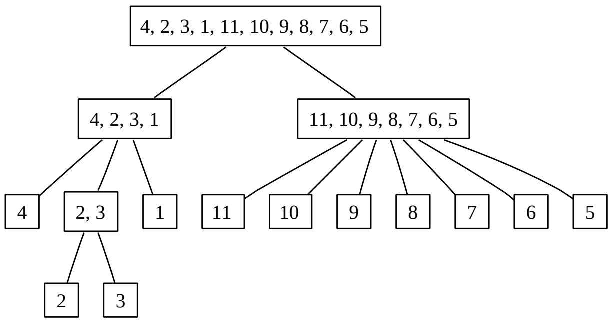


Fig. 4. The tree of strong common intervals of the permutations G_3 and G_4. Leaves are ordered according to G_4.

Such trees are examples of a general structure, known as PQ-trees [15], that was developed to represent sets of permutations. These are ordered trees whose nodes are classified as either P-nodes or Q-nodes. In the tree of strong common intervals of a set of permutations $\mathcal{G}$, if the leaves are ordered according to one of the permutation in $\mathcal{G}$, then for each node N, exactly one of the following is true:

1) [Q-nodes] Any union of consecutive children of N is a common interval of $\mathcal{G}$.

2) [P-nodes] No union of consecutive children of N is a common interval of $\mathcal{G}$, except the union of all its children – in this case the union equals N itself.

In PQ-trees, the P-nodes are traditionally depicted as roundish boxes, and the Q-nodes as rectangular boxes. The tree of Figure 4 has only Q nodes. A more general example is given by the tree of Figure 5 which represents the strong common intervals of the permutations:

$$
\begin{array}{lcccccccccccl}
G_5 = (& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 &) \\
G_6 = (& 1 & 4 & 2 & 5 & 3 & 11 & 10 & 8 & 9 & 7 & 6 &).
\end{array}
$$

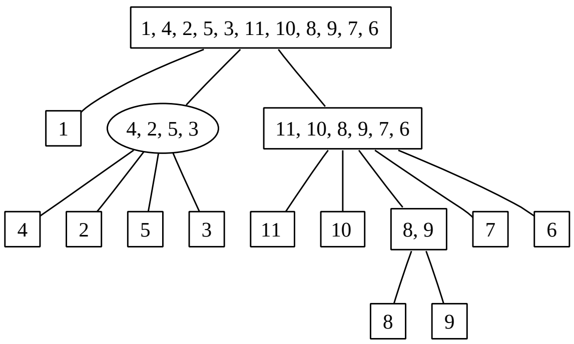


Fig. 5. The tree of strong common intervals of the permutations G_5 and G_6. Leaves are ordered according to G_6.

In Figure 5, the node corresponding to the strong common interval $\{4, 2, 3, 5\}$ is a P-node since no union of consecutive children is a common interval. This representation of strong common intervals allows them to serve as a basis for generating all common intervals of a set of permutations. We have:

Proposition 3. *[10] Let T be the PQ-tree of the strong common intervals of a set $\mathcal{G}$ of permutations, ordered according to one of the permutations in $\mathcal{G}$. A set S is a common interval of $\mathcal{G}$ if and only if it is the union of consecutive nodes of children of a Q-node, or the union of all children of a P-node.*

Computing common intervals and strong intervals The algorithmic history of efficient computation of common and strong intervals has an interesting twist. From the start, Uno and Yagiura [49] proposed an algorithm to compute the common intervals of two permutations whose theoretical running time was $\mathcal{O}(n + N)$ where n is the number of elements of the permutation, and N is the number of common intervals of the two permutations. Such an algorithm can be considered as optimal since it runs in time proportional to the sum of the size of the input and the size of the output. However, the author acknowledged that their algorithm was “quite complicated” and that, in practice, simpler $\mathcal{O}(n^2)$ algorithms run faster on randomly generated permutations.

Building on Uno and Yagiura’s work, Heber and Stoye [27] proposed an algorithm to generate all common intervals of a set of K permutations in time proportional to $Kn + N$, based on Uno and Yagiura analysis. They achieved the extension to K permutations by considering the set of *irreducible* common intervals, that are common intervals that are *not* the union of two overlapping common intervals. As for the strong intervals, the irreducible common intervals also form a basis of size $\mathcal{O}(n)$ that generates the common intervals by unions of overlapping irreducible intervals.

The drawback of these algorithms is that they use complex data structures that are difficult to implement. A simpler way to generate the common intervals is to compute a basis that generate intervals using intersections instead of unions:

Definition 7. *Let $\mathcal{G}$ be a set of K permutations on n elements, that contains the identity permutation. A* generator *for the common intervals of $\mathcal{G}$ is a pair (R, L) of vectors of size n such that:*
1) $R[i] \geq i$ and $L[j] \leq j$ for all $i, j \in \{1, 2, \ldots, n\}$,
2) $(i..j)$ is a common interval of $\mathcal{G}$ if and only if $(i..j) = (i..R[i]) \cap (L[j]..j)$.

It is not immediate that such generators even exist, but it turns out that they are far from unique, and some of them can be computed using elementary data structures such a stacks and arrays [10]. The algorithms are easy to implement, and the theoretical complexity is $\mathcal{O}(Kn+N)$. The strong common intervals can also be computed in $\mathcal{O}(Kn)$.

The use of common intervals in comparative genomics Datasets based on permutations that use real "genes" are not frequent in comparative genomics since real genes are often found in several copies within the genome of an organism. In order to obtain permutations, it is possible to eliminate all duplicates, or, even better, to retain only one copy [38]. However, in some small chromosomes, such as animal mithocondrion genomes, genes are not often duplicated, and it is possible to extract permutations from the gene order without sacrificing too much information. In [1], for example, common intervals are used to define a distance between plant chloroplast genomes.

On the other hand, datasets that are constructed using big synteny blocks, such as in [16], naturally yield permutations. In this context, common intervals, and strong common intervals, have been used, for example, for the evaluation of proposed evolution scenarios between species [6], or for the construction of evolution scenarios [7]. In these applications, the impact of rearrangement operations on the structure of common intervals is taken into account to propose a rearrangement scenario between two genomes.

For example, Figure 6 represents the strong common intervals of two signed permutations representing conserved block in the chromosomes X of the rat and mouse. Each strong interval is marked either by a '+' or a '-' sign, using the sign of the element of the permutation for the leaves, and the fact that the numbers are increasing or decreasing for the other nodes. Inverting all strong intervals that have a sign different from their parent yields a scenario that conserves, for each intermediate scenario, all common intervals of the two permutations.

$$
\begin{array}{ll}
Mouse & = (1\ 2\ 3\ 4\ \ 5\ \ \ 6\ \ \ 7\ \ \ 8\ \ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16) \\
Rat & = (\overline{4}\ \overline{3}\ \overline{2}\ 1\ \overline{13}\ \overline{15}\ 14\ \overline{16}\ 8\ \ 9\ \ 10\ \overline{11}\ 12\ \ 5\ \ \ 6\ \ \ 7)
\end{array}
$$

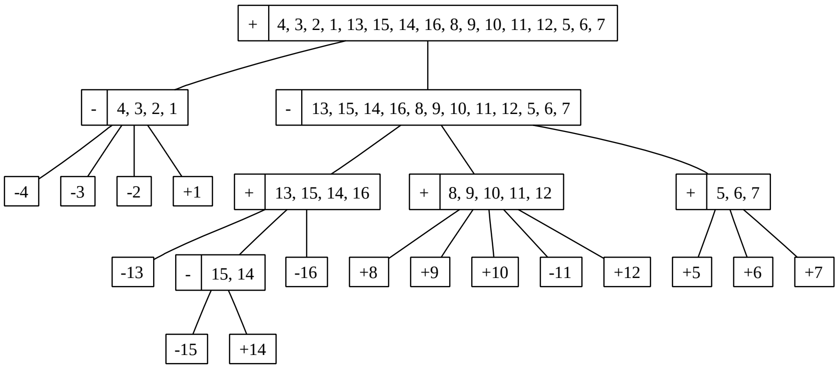


Fig. 6. Comparing the rat and mouse X chromosomes.

4.2 Max-gap clusters in permutations

A first formal model of gene clusters in permutations that allows gaps was introduced in [8] under the name of *gene teams*. Even if the range of its applications is limited, it remains the only model whose formal properties can be fully analyzed, as in [28] where they are called *max-gap clusters*, whose output size is reasonable, and for which there exists efficient algorithms.

Definition 8. *Let $\mathcal{G}$ be a set of strings on the set of genes $\mathcal{S} \cup \{*\}$, such that each string is a permutation of $\mathcal{S}$ when the symbols $\{*\}$ are removed. Let $\delta \geq 0$ a fixed integer. A subset of $\mathcal{S}$ is a* gene team *if it has an occurrence with maximum gap size δ in each permutation of $\mathcal{G}$, and has no extension.*

Consider, for example, the two following strings:

$$
\begin{aligned}
G_1 &= (\ 1 * 2\ 3\ 4 * * 5\ 6\ 7 * * 8\ ) \\
G_2 &= (\ 8 * 4 * 2\ 1\ 5\ 3\ 6 * 7 * * \ ).
\end{aligned}
$$

The gene teams of G_1 and G_2 with gap size $\delta = 1$ are $\{1, 2, 3, 4\}$, $\{5, 6, 7\}$ and $\{8\}$. It is important to note that occurrences of teams can overlap, Figure 7 illustrates this fact by joining the various member of a team with arcs.

$$
\begin{array}{lcccccccccccccc}
G_1 = (& 1 & * & 2 & 3 & 4 & * & * & 5 & 6 & 7 & * & * & 8\,) & \\
G_2 = (& 8 & * & 4 & * & 2 & 1 & 5 & 3 & 6 & * & 7 & * & * &).
\end{array}
$$

Fig. 7. Gene teams of G_1 and G_2 with gap size $\delta = 1$

Even though occurrences of gene teams can overlap, they always form a partition of the set of genes $\mathcal{S}$. This is a consequence of the following proposition.

Proposition 4. *Let S and T be two gene teams of the set of strings $\mathcal{G}$. If $S \cap T \neq \emptyset$, then $S = T$.*

Proof. Consider a substring $g_i \ldots g_j$ in one of the string G of $\mathcal{G}$ that contains all genes of $S \cup T$, and such that both g_i and g_j are in $S \cup T$. We first show that each substring of $g_i \ldots g_j$ of length greater than δ contains at least a gene from $S \cup T$, implying that $g_i \ldots g_j$ is an occurrence of $S \cup T$. In order to prove this, suppose that a substring $g_k \ldots g_{k+\delta}$ contains no gene in $S \cup T$, and both flanking genes are in $S \cup T$. Then these genes cannot be both in S, or both in T, since S or T would not have an occurrence in the string G. Therefore, one is an element of S and the other of T, and the occurrences of S and T in this string are on different sides of the substring $g_k \ldots g_{k+\delta}$, implying that $S \cap T = \emptyset$.

Now, since the string G was arbitrary, we have shown that $S \cup T$ is an extension of both S and T, which is ruled out by definition, thus $S = T$.

A first consequence of Proposition 4 is that the number of gene teams is $\mathcal{O}(n)$, where n is the number of elements of $\mathcal{S}$. This will be our last model with such a desirable property. Proposition 4 also allows the computation of gene teams through partition refinement techniques. The best complexity achieved up to now is $\mathcal{O}(Kn \log^2(n))$ [4], where K is the number of strings in $\mathcal{G}$.

4.3 Common intervals in strings

In the previous section, we generalized the notion of common intervals in permutations by allowing gaps in occurrences of a set of genes defining a cluster. We now describe another natural extension of common intervals in permutations, which is the notion of common intervals in strings.

Definition 9. *Let $\mathcal{G}$ be a set of strings on the set of genes $\mathcal{S}$. A subset of $\mathcal{S}$ is a* common interval *if it has an occurrence in each string of $\mathcal{G}$, without gaps.*

The above definition is identical to the definition of common intervals in permutations (Def. 4) except for one word: “strings” instead of “permutations”, and this model is used to analyze genomes with many duplicates. This representation is particularly adapted to the study of genomes at the level of “genes” and “domains”, under the classical biological definition of these two words.

For example, in

$$G_1 = (f \;\; e \;\; c \;\; e \;\; b \;\; e \;\; d) \text{ and } G_2 = (a \;\; b \;\; b \;\; e \;\; c \;\; b \;\; c \;\; d \;\; b \;\; e \;\; c)$$

the set $\{b, e\}$ is a common interval that has one occurrence in G_1, $(e \;\; b \;\; e)$, and two occurrences in G_2, $(b \;\; b \;\; e)$ and $(b \;\; e)$. The other common intervals are $\{b\}$, $\{c\}$, $\{e\}$, $\{c, e\}$, $\{b, c, e\}$ and $\{b, c, d, e\}$.

Properties of common intervals in strings. A major difference with common intervals of permutations, due to the possible repetition of genes in strings, is that a common interval S of the strings $\mathcal{G}$, can have two or more occurrences in a genome, that do not overlap due to the point 3 of Definition 1. Hence using common intervals of sequences allows to detect gene clusters that have been rearranged by duplication events, either internal duplications resulting in several copies of a gene of the cluster, or large-scale duplications resulting in several copies of the whole clusters; this is an important strength of this model.

The first question raised by the definition of gene clusters as common intervals in strings is the computational cost of such a higher biological accuracy. It happens that this model is very tractable, and in fact, only slightly more costly to handle than common intervals of permutations. Indeed, since every occurrence of a common interval is without gap, every substring of a genome of $\mathcal{G}$ can be occurrence of at most one gene cluster, which immediately leads to the following proposition.

Proposition 5. *Let N be the sum of the lengths of the strings in $\mathcal{G}$. The number of common intervals in $\mathcal{G}$ and the number of occurrences of all common intervals in $\mathcal{G}$ are in $\mathcal{O}(N^2)$.*

Hence the maximum number of common intervals is quadratic in the size of the genomes in both cases, permutations and strings, which is a direct consequence of considering occurrences without gaps. However, an important difference with permutations relies in the internal structure of the set of common intervals in strings. In permutations, the existence of linear space basis of the set of common intervals, irreducible intervals and strong intervals, is central in the linear time and space complexity of algorithms. Currently, no such basis has been found for common intervals in strings (see [33] for an attempt to use PQ-trees with strings, that is applicable when very few duplicated genes exist).

Detecting common intervals in strings. Due to the lack of internal structure, the algorithms used to detect the set of common intervals in a set of strings $\mathcal{G}$, and the occurrences of each common interval, are very different in nature than the algorithms for detecting common intervals in permutations, and much less elegant. They rely on the enumeration of the *fingerprints* of $\mathcal{G}$.

Given a substring $g_i \dots g_j$ of a string of $\mathcal{G}$, its *fingerprint* is the subset of $\mathcal{S}$, the gene alphabet, containing the genes appearing in $g_i \dots g_j$. For example, the fingerprint of $(b\ \ b\ \ e\ \ c\ \ b\ \ c)$ is $\{b, c, e\}$. The *fingerprints* of $\mathcal{G}$ is the set of all the fingerprints of all substrings of $\mathcal{G}$. It follows immediately that a common interval of $\mathcal{G}$ is a fingerprint of $\mathcal{G}$ that appears in each string of $\mathcal{G}$, which implies that computing the common intervals of $\mathcal{G}$ reduces to the appropriate filtering of the set of fingerprints of $\mathcal{G}$.

Detecting the set of fingerprints of a set of strings was first considered by Amir *et al.* in [2]. The key of their algorithm is an efficient encoding scheme for fingerprints that associates, to each fingerprint, a unique name computed from the elements of $\mathcal{S}$ it contains. Their encoding is efficient in the sense that, if two fingerprints differ only by one element, computing the name of the second one from the first one requires a limited complexity ($\mathcal{O}(\log(n))$), where n is the number of elements of $\mathcal{S}$. This property allows to compute the number of different fingerprints by using a sliding window on the considered string of total length N, in time $\mathcal{O}(nN\log(n)\log(N))$. Two recent papers used the same principle, but improved the fingerprint naming technique introduced by Amir *et al.* to compute the set of fingerprints of $\mathcal{G}$ respectively in time $\mathcal{O}(nN\log(n))$ (see [21]) and $\mathcal{O}((occ+N)\log(n))$ where occ is the total number of occurrences of the fingerprints of $\mathcal{G}$ (see [32]). Another family of algorithms, initiated by Didier [20], and improved in [40] and [21], allows to compute the fingerprints of $\mathcal{G}$ in time $\mathcal{O}(N^2)$; note that the property to be independent from the number of genes is important as in most comparative genomics applications, n is in $\mathcal{O}(N)$.

These two families of algorithms offer, as far as we know, the only non-trivial ways to compute the set of common intervals of a set of strings.

Using common intervals in strings. Common intervals of strings have been used in two problems: detection of conserved gene clusters, which was the initial goal they were designed for, and assignment of positional homologs. We conclude briefly this section by describing these applications, which will allow us to outline the advantages and shortcomings of this formal model.

Detection of conserved gene clusters. The algorithm of Schmidt and Stoye, described in [40], and adapted to compute intervals that are common to a subset of the strings in $\mathcal{G}$, is the basis of the algorithm used in the software GECKO [25]. But GECKO brings additional important features that address weaknesses of the common intervals model.

A first improvement of the basic common interval model relies on the ability of GECKO to avoid displaying clusters that have an extension (see Def. 2). This is an important practical feature, as it reduces significantly the size of the output that can be huge otherwise. A second improvement addresses the main problem of common intervals as models of gene clusters: it is very unlikely that a cluster keeps the same gene content in all the genomes of a dataset, especially if they span a large evolutionary spectrum. In such a case, these clusters cannot be detected by common intervals. To solve this problem, GECKO uses an heuristic post-processing that agglomerates common intervals in groups such that all common intervals in a given group share at least a minimum fraction – a user defined parameter – of their genes with a particular common intervals of this group. This way of grouping common intervals, in order to to define less strict clusters, is inspired from the heuristic methods used to compute gene clusters, and adds flexibility. The price to pay is more noisy output, since a given common interval can belong to several groups.

Inferring positional homologs. When genomes contain duplicated genes, a major question is to elucidate the evolutionary relationships between the several copies of a given gene, that form a family of homologous genes. Several kinds of relationships have been defined, the most commonly considered being the notions of *orthologs* and *paralogs*, that are defined in terms of the evolutionary history of gene families. This problem has attracted a lot of attention since such knowledge has been shown to be helpful in understanding the function of the genes of a gene family. Recently, several other notions of homology have been defined in terms of gene order, such as *positional homologs* [18] and *ancestral homologs* [38], that are used to propose interesting putative pairs of orthologs. In the gene order approaches, these homologs are deduced from a pairing between the genes of two compared genomes that defines a signed permutation, and that optimize a given combinatorial criterion, for example a genetic distance [24], or the number of common intervals in the resulting permutation [17].

In [12], it was shown that using directly common intervals in strings as anchors of a global matching, in a way that is similar to methods for whole-genomes alignments, offers an interesting alternative to the parsimonious approach to infer positional homologs from gene orders, and performs at least as well as methods based on more complicated combinatorial objects, such as the breakpoint graphs in [24].

4.4 Max-gap clusters in strings

Max-gap clusters in strings are the most general formal model of gene clusters: gaps are allowed, duplications and missing genes are taken care of, and there is no requirement that a cluster be present in all species under study. Unfortunately, there is little hope that the automatic detection of these type of clusters will be computationally feasible for large datasets.

Definition 10. *Let $\mathcal{G}$ be a set of strings on the set of genes $\mathcal{S}$, and $\delta \geq 0$ a fixed integer. A subset of $\mathcal{S}$ is a* max-gap cluster *if it has an occurrence with maximum gap size δ in at least one string of $\mathcal{G}$, and has no extension.*

Each type of clusters that we examined in the preceding sections provides examples of max-gap clusters, but the general notion seems indeed truly adapted for certain type of biological data. For example, in the genome of *Nostoc sp. PCC 7120*, we have the following five occurrences with gap size $\delta = 1$ of a set of three domains:

$$CABC, \quad CA * ABCC, \quad ABCC, \quad A * ABC * CB, \quad A * BCCABC,$$

where stars '*' stand in for domains not in the set $\{A, B, C\}$. These five patterns are among dozens of variations on the associations of these three domains that are actually found in bacteria. Thus there is clearly a need to be able to consider models that allow for gaps and duplicate genes. However, as we will see in the next paragraphs, known bounds for the size of the output – that is, the number of max-gap clusters in a given set of strings $\mathcal{G}$ – are currently exponential in the number of genes in $\mathcal{S}$. This implies that the running time of algorithms that detect max-gap clusters can be unreasonably long.

Algorithmic complexity Clearly, testing for all possible subsets as candidate max-gap clusters is out of the question for any realistic set of genes. Some reasonable hypothesis, such as limiting the size of the gap, and considering sets that have at least one occurrences in the genomes under study, are helpful and allow the computation of max-gap clusters for interesting sets of genomes with thousands of genes [34]. For the comparison of exactly two genomes, and with the additional hypothesis that a cluster must have an occurrence in both genomes, He and Goldwasser [26] describe a polynomial algorithm to compute them.

However, even with a gap size equal to 1, there can be an exponential number of subsets that have at least one occurrence, and possibly no extension, as the following example shows. Consider a genome segment of the form:

$$ag_1 \dots g_n z$$

in which all the genes are different. We will compute T_n, the number of subsets of $\{a, g_1, \dots, g_n, z\}$ that contain genes a and z, and that have an occurrence with maximum gap size of 1 in the segment.

For $n = 0$, the number of such subsets is 1, and for, $n = 1$, there are two of them: $\{a, z\}$ and $\{a, g_1, z\}$. Suppose now that $n \geq 2$. A subset that has an occurrence in the segment, and contain both a and z, either:

1) contains both g_{n-1} and g_n, or
2) contains g_{n-1} but not g_n, or
3) contains g_n but not g_{n-1}.

Clearly, the three cases are disjoint. In the first case, removing genes g_{n-1} and g_n from the subset yields a subset that has an occurrence in the segment $ag_1 \dots g_{n-2}z$. Conversely, adding genes g_{n-1} and g_n to such a subset yields a subset that have an occurrence in the segment $ag_1 \dots g_{n-1}, g_n z$. Thus, the number of subsets covered by case 1) is T_{n-2}.

The two remaining cases are treated together. Consider the T_{n-1} subsets that have an occurrence in $ag_1 \dots g_{n-1}z$. All subsets that contains gene g_{n-1} also have occurrences with gap size 1 in $ag_1 \dots g_{n-1}, g_n z$, by skipping gene g_n, and all those that do not contain gene g_{n-1} can be extended to occurrences in $ag_1 \dots g_{n-1}, g_n z$ by adding gene g_n, since they must contain gene g_{n-2}. The inverse bijection is obtained by removing gene g_n in all subsets covered by case 3). Thus the total number of subsets covered by case 2) and 3) is T_{n-1}, yielding the recurrence equation:

$$T_n = T_{n-1} + T_{n-2},$$

with initial conditions $T_0 = 1$ and $T_1 = 2$.

This equation has, unfortunately, a well known solution that grows exponentially with n. Its closed form is:

$$T_n = \frac{(1+\sqrt{5})^{n+2} - (1-\sqrt{5})^{n+2}}{2^{n+2}\sqrt{5}},$$

and the sequence $\{T_n\}$ is known as the *Fibonacci* sequence. As a measure of its growth, we have, for example, $T_{10} = 144$ and $T_{20} = 17711$, and $T_{30} = 2178309$.

The combinatorial beauty of nature In the preceding section, we proved that the theoretical number of max-gap clusters could be exponential with respect to the number of genes. How do real genomes behave? It seems that, at least at certain levels of complexity, nature does experiment in a combinatorial way.

The STRING database [46] predicts functional associations between genes or domains in prokaryotic genomes, based on the identification of genomic segments where they appear close together. Using STRING, we looked at occurrences of subsets of the following 5 domains: A) COG2202 - PAS/PAC domain, B) COG0642 - Signal transduction histidine kinase, C) COG0784 - CheY-like receiver, D) COG2203 - GAF domain, and E) COG2200 - EAL domain.

The first of these five domains, COG2202, was chosen among the few dozens that have many duplicates in bacterial genomes. The next four are the best scoring for functional association with COG2202, as predicted by STRING. This choice is somewhat arbitrary, and other sets of domains were also shown to exhibit the same combinatorial behavior.

If we exclude trivial subsets consisting of 1 elements, there are 26 possible subsets of the set $\{A, B, C, D, E\}$. Of these, we found that 19 had at least one occurrence with $\delta = 2$ among 11 different genomes. Table 1 gives examples of occurrences for each of the 19 subsets. It illustrates in a striking way why the general model of max-gap clusters must eventually be replaced by heuristics in some applications. With more and more organisms to compare, it is tempting to predict that all the "not found" lines in Table 2 will eventually be filled up.

5 Conclusion

We described in this chapter a hierarchy of formal models that have been defined to detect conserved gene clusters. One of the important point that we can outline is the strong link between the biological complexity that these models try to capture and their computational complexity. It appears very clearly that the limiting factors are the presence of duplicated genes (*i.e.* strings vs. permutations) and the existence of clusters whose occurrences do not have the same gene content: indeed, the detection of common intervals in strings and max-gap clusters in permutations is very tractable. It then remains open to define general models of gene clusters that consider both non-exact occurrences and strings, and that are tractable.

A first way to attack this problem could be to try to extend the notion of common intervals, that is the only known tractable model handling strings, to define a model with non-exact occurrences but without using the notion of gap, as it was shown that even the shortest possible gap of 1 can lead to an exponential number of clusters. A first attempt was done in [19], with a model of *common intervals with errors* that adds flexibility by bounding the number of genes of an occurrence of a cluster that do not belong to the cluster. But even in this model the number of generated clusters can be exponential.

Subset	Occurrence	Species
$\{A, B\}$	$ABAB$	*Nostoc sp. PCC 7120*
$\{A, C\}$	AC	*Leptospira interrogans*
$\{A, D\}$	$A * ADA$	*Methanosarcina acetivorans*
$\{A, E\}$	AE	*Escherischia coli*
$\{B, C\}$	$BC * *C$	*Xanthomonas campestris*
$\{B, D\}$	DB	*Leptospira interrogans*
$\{B, E\}$	(not found)	
$\{C, D\}$	$CC * D * *C$	*Nostoc sp. PCC 7120*
$\{C, E\}$	$E * *C$	*Bacillus halodurans*
$\{D, E\}$	$E * D$	*Xanthomonas axohopodis*
$\{A, B, C\}$	CAB	*Xanthomonas campestris*
$\{A, B, D\}$	$ADADB$	*Methanosarcina acetivorans*
$\{A, B, E\}$	$B * A * E$	*Pseudomonas aeruginosa*
$\{A, C, D\}$	$ADA * C$	*Sinorhizobium meliloti*
$\{A, C, E\}$	$AC * E$	*Agrobacterium tumafaciens*
$\{A, D, E\}$	(not found)	
$\{B, C, D\}$	$DBCB$	*Nostoc sp. PCC 7120*
$\{B, D, E\}$	(not found)	
$\{B, C, E\}$	$C * EBB * C$	*Xylella fastidiosa*
$\{C, D, E\}$	(not found)	
$\{A, B, C, D\}$	$DAD * CAB$	*Nostoc sp. PCC 7120*
$\{A, B, C, E\}$	$CE * ABC$	*Pseudomonas aeruginosa*
$\{A, B, D, E\}$	(not found)	
$\{A, C, D, E\}$	(not found)	
$\{B, C, D, E\}$	(not found)	
$\{A, B, C, D, E\}$	$ADABCECB$	*Nostoc sp. PCC 7120*

Table 2. Occurrences of various subsets of domains in prokaryotic genomes. Stars '*' stand in for domains not in the set $\{A, B, C, D, E\}$.

Another possibility would be to rely on combinatorial properties of gene clusters in order to refine gene clusters models and then reduce the number of produced clusters by eliminating non significant clusters. For example, one of the reasons for the definition of common intervals with errors is that it ensures a minimum *density* [28] of each occurrence, roughly defined as the ratio between genes belonging to the cluster and intruders. Other properties of gene clusters that are worth to be investigated are also described in [28].

Finally, it would be very natural to include the phylogenetic information, when available, in the definition of gene cluster models. This approach has been shown to be very interesting in a heuristic approach and in a probabilistic framework [52], but it is not obvious how it would fit in the purely combinatorial framework we described in this chapter.

References

1. Adam, Z., Turmel M., Lemieux C., Sankoff D.: Common Intervals and Symmetric Difference in a Model-Free Phylogenomics, with an Application to Streptophyte Evolution. *Lecture Notes in Bioinformatics*, 4205:63–74, Springer, 2006.
2. Amir, A., Apostolico, A., Landau, G.M., Satta, G.: Efficient text fingerprinting via Parikh mapping. *Journal of Discrete Algorithms*, 1(5–6):409–421, 2003.
3. Bateman, A., Coin, L., Durbin, R., Finn, R.D., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall. M., Moxon, S., Sonnhammer, E.L., Studholme, D.J., Yeats, C., Eddy, S.R.: The Pfam protein families database. *Nucleic Acids Research*, 32:D138–D141, 2004.
4. Béal, M.-P., Bergeron, A., Corteel, S., Raffinot, M.: An algorithmic view of gene teams. *Theoretical Computer Science*, 320(2-3): 395-418, 2004.
5. Belda, E., Moya, A., Silva, F.J.: Genome rearrangement distances and gene order phylogeny in γ-proteobacteria. *Molecular Biology and Evolution*, 22(6):1456–1467, 2005.
6. Bérard, S., Bergeron, A., Chauve, C.: Conserved structures in evolution scenarios. *Lecture Notes in Bioinformatics*, 3388:1–15, Springer, 2005. To appear in IEEE/ACM Transactions on Computational Biology and Bioinformatcs.
7. Bérard, S., Bergeron, A., Chauve, C., Paul, C.: Perfect sorting by reversals is not always difficult. *Lecture Notes in Bioinformatics*, 3692:228–238, Springer, 2005.
8. Bergeron, A., Corteel, S., Raffinot, M.: The Algorithmic of Gene Teams. *Lecture Notes in Computer Science*, 2452: 464–476, Springer, 2002.
9. Bergeron, A., Blanchette, M., Chateau, A., Chauve, C.: Reconstructing ancestral gene orders using conserved intervals. *Lecture Notes in Bioinformatics*, 3240:14–25, Springer, 2004.
10. Bergeron, A.,Chauve, C., de Montgolfier, F., Raffinot, M.: Computing Common Intervals of K Permutations, with Applications to Modular Decomposition of Graphs. *Lecture Notes in Computer Science*, 3669: 779–790, Springer, 2005.
11. Blanchette, M., Kunisawa, T., and D. Sankoff, Gene order breakpoint evidence in animal mitochondrial phylogeny, *Journal of Molecular Evolution*, 49:193–203, 1999.
12. Blin, G., Chateau, A., Chauve, C., Gingras, Y.: Inferring positional homologs with common intervals of sequences. *Lecture Notes in Bioinformatics*, 4205:24–38, Springer, 2006.

13. Blin, G., Chauve, C., Fertin, G.: Gene order and phylogenetic reconstruction: application to γ-Proteobacteria. *Lecture Notes in Bioinformatics*, 3678:11–20, Springer, 2005.
14. Boore, J.L.: Animal mitochondrial genomes. *Nucleic Acids Research*, 27(8): 1767–1780, 1999.
15. Booth, K. S., Lueker, G. S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.
16. Bourque, G., Pevzner, P.A., Tesler, G.: Reconstructing the genomic architecture of ancestral mammals: Lessons from human, mouse, and rat genomes. *Genome Research*, 14(4):507–516, 2004.
17. Bourque, G., Yacef, Y. El-Mabrouk, N.: Maximizing synteny blocks to identify ancestral homologs. *Lecture Notes in Bioinformatics*, 3678:21–34, Springer, 2005.
18. Burgetz, I.J., Shariff, S., Pang, A., Tillier, E.R.M.: Positional homology in bacterial genomes. *Evolutionary Bioinformatics Online*, 2:42–55, 2006.
19. Chauve, C., Diekmann, Y., Heber, S., Mixtacki, J., Rahmann, S., Stoye, J.: On Common Intervals with Errors. Report 2006-02, Technische Fakultt der Universitt Bielefeld, Abteilung Informationstechnik, 2006.
20. Didier, G.: Common intervals of two sequences. *Lecture Notes in Bioinformatics*, 2812:17–24, Springer, 2003.
21. Didier, G., Schmidt, T., Stoye, J., Tsur, D.: Character sets of strings. To appear in *Journal of Discrete Algorithms*.
22. Dobzhansky, T., Sturtevant, A.T.: Inversions in the Chromosomes of *Drosophila pseudoobscura*. *Genetics*, 23:28–64, 1938.
23. Eichler, E.E., Sankoff, D.: Structural dynamics of eukaryotic chromosome evolution. *Science*, 301(5634):793–797, 2003.
24. Fu, Z., Chen, X., Vcici, V., Nan, P., Zhong, Y., Jiang, T.: A parsimony approach to genome-wide ortholog assignment. *Lecture Notes in Bioinformatics*, 3909:578–594, Springer, 2006.
25. Gecko. Gene Clusters Identification in Prokaryotes. Software available on the website `http://bibiserv.techfak.uni-bielefeld.de/gecko`.
26. He, X., Goldwasser, M.H.: Identifying Conserved Gene Clusters in the Presence of Homology Families. *Journal of Computational Biology*, 12(6):638-656, 2005.
27. Heber, S., Stoye, J.: Finding All Common Intervals of k Permutations. *Lecture Notes in Computer Science*, 2089:207–218, Springer, 2001.
28. Hoberman, R., Durand, D.: The incompatible desiderata of gene cluster properties. *Lecture Notes in Bioinformatics*, 3678:73–87, Springer, 2005.
29. Hoberman, R., Sankoff, D., Durand, D.: The Statistical Analysis of Spatially Clustered Genes under the Maximum Gap Criterion. *Journal of Computational Biology*, 12(8):1081-1100, 2005.
30. Ijdo, J.W., Baldini, A., Ward, D.C., Reeders, S.T., Wells R.A.: Origin of Human Chromosome 2: An Ancestral Telomere-Telomere Fusion. *Proceedings of the National Academy of Sciences of the United States of America*, 88:9051-9055, 1991.
31. Koehler, U., Bigoni, F., Wienberg, J., Stanyon, R.: Genomic reorganization in the concolor gibbon (Hylobates concolor) revealed by chromosome painting. *Genomics*, 20;30(2):287–92, 1995.
32. Kolpakov, R., Raffinot, M.: New algorithms for text fingerprinting. *Lecture Notes in Computer Science*, 4009:342–353, Springer, 2006.
33. Landau, G.M., Parida, L., Weimann, O.: Gene proximity analysis across whole genomes via PQ trees. *Journal of Computational Biology*, 12(10):1289–1306, 2005.

34. Pasek, S., Bergeron, A., Risler, J.-L., Louis, A., Ollivier, E., Raffinot, M.: Identification of genomic features using microsyntenies of domains: domain teams. *Genome Research*, 15(6):867–74, 2005.
35. Price, M.N., Arkin, A.P., Alm, E.J.: The life-cycle of operons. *PLoS Genetics*, 2(6):e96, 2006.
36. Rogozin, I.B., *et al.*: Connected gene neighborhoods in prokaryotic genomes. *Nucleic Acids Research*, 30(10):2212–2223, 2002.
37. Rogozin, I.B., Makarova, K.S., Wolf Y.I., Koonin, E.V.: Computational approaches for the analysis of gene neighborhoods in prokaryotic genomes. *Briefings in Bioinformatics*, 5(2):131–149, 2004.
38. Sankoff, D.: Genome rearrangement with gene families. *Bioinformatics*, 15(11): 909–917, 1999.
39. Sankoff, D.: Rearrangements and chromosomal evolution. *Current Opinion in Genetics & Development*, 13(6):583–587, 2003.
40. Schmidt, T., Stoye, J.: Quadratic time algorithms for finding common intervals in two and more sequences. *Lecture Notes in Computer Science*, 3109:347–358, Springer, 2004.
41. Sears, E.R.: Nullisomic-tetrasomic combinations in wheat. In *Chromosome Manipulation and Plant Genetics*, R. Riley, K.R. Lewis, eds. Oliver and Boyd, Edinburgh, 29–45, 1966.
42. Semon, M., Duret, L.: Evolutionary origin and maintenance of coexpressed gene clusters in mammals. *Molecular Biology and Evolution*, 23(9):1715–1723, 2006.
43. She, X., *et al.*: A preliminary comparative analysis of primate segmental duplications shows elevated substitution rates and a great-ape expansion of intrachromosomal duplications. *Genome Research*, 16(5):576–583, 2006.
44. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
45. Snel, B., Bork, P., Hyunen, M.A.: The identification of functional modules from the genomic association of genes. *Proceedings of the National Academy of Sciences of the United States of America*, 99:5890–5895, 2002.
46. Snel, B., Lehmann, G., Bork, P., Hyunen, M.A.: STRING: a web-server to retrieve and display the repeatedly occurring neighbourhood of a gene. *Nucleic Acids Research*, 28(18):3442-3444, 2000.
47. Tamames, J.: Evolution of gene order conservation in prokaryotes. *Genome Biology*, 2(6):RESEARCH0020, 2001.
48. Thao, M.L., Baumann, L., Baumann, P.: Organization of the mitochondrial genomes of whiteflies, aphids, and psyllids (Hemiptera, Sternorrhyncha). *BMC Evolutionary Biology*, 4:25, 2004.
49. Uno, T., Yagiura, M.: Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.
50. Wolfe, K.H., Shields, D.C.: Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387:708–713, 1997.
51. Xie, G., Keyhani, N.O., Bonner, C.A., Jensen, R.A.: Ancient Origin of the Tryptophan Operon and the Dynamics of Evolutionary Change. *Microbiology and Molecular Biology Reviews*, 67(3):303–342, 2003.
52. Zheng, Y., Anton, B.P., Robert, R.J., Kasif, S.: Phylogenetic detection of conserved gene clusters in microbial genomes. *BMC Bioinformatics*, 6:243, 2005.

Acknowledgments

The different models described in this chapter were developed by a strongly interacting community of researchers. The authors wish to ackowledge that the connected component of the transitive closure of the collaboration relation, as defined by the bibliography of this chapter, and that contains the three authors, also contains 29 other persons. This means that formal definitions of gene clusters is a collective endeavor, and we thank them all.