University of Thessaly
School of Science
Informatics and Computational Biomedicine

Computational Medicine and Biology

How to compute the similarity of genomes: a few observations

Christakaki Agapi

Master Thesis

Supervisor
E. Markou

Examiners
E. Markou
P. Bagos
V. Plagianakos

Lamia

November 9, 2018

Contents

0.1 Abstract

A basic problem in Computational Biology is to compute how similar two genomes are and is usually mentioned as the computation of the distance between two genomes. The distance, $d(G_1,G_2)$, is defined as the minimum number of events like inserting, duplicating, replacing or deleting a part of the genome, transforming one genome G_1 into G_2. Many different models with respect to the distance definition have been studied as breakpoint distance, exemplar distance, reversal distance and the general type of distance which is the edit distance.
We first give biological definitions about genomes, genes and other terms we use. We will talk about some models that compute the distance between two genomes and their complexity. Then we propose a way to compute the distance between more than two genomes which are consisted of singletons. We define some criteria about that way and we give some examples. The basic 'step' is to create a new genome. Then we compute the distances among all the initial genomes through the created one. So we can make a fast estimation about the distances. In some cases we achieve good approximation.

First, we see the case that only reversals are allowed. We give some examples and we take a good approximation of the distances. We do the same in case that only deletions and insertions are allowed and then in case that all operations are allowed. In all cases we try to create a new string (genome), which construction is not very far from the initial strings (genomes) by using the criteria we have defined. Then we compute the distance between every initial genome and the constructed one despite of computing each pair of genomes separately. After that, we can compute all the distances via the constructed genome according to the property $d(G_1,G_2) \leq d(G_1,G^*) + d(G^*,G_2)$, where G_1, G_2 are the initial strings (genomes), and G^* is the constructed one. Finally, we can't guarantee the optimal solution to the rearrangement problem but is an easier and faster way to make a good estimation.

Chapter 1

Introduction

1.1 A view at biology

The computation of the similarity of two genomes has many applications in Biology. Before we discuss those applications we need to define some terminology. We give the following definitions according to Fertin's book, mostly [5] (see also [6], [16],). Genome is the set of all chromosomes. Chromosomes are made of DNA, defined as a double stranded molecule in which each strand is a long succession of nucleotides (sequence). Nucleotides can be of four types, Adenine (A), Thymine (T), Cytosine (C), and Guanine (G). The two DNA strands are coupled in such a way that an Adenine on one strand is coupled with a Thymine on the other strand and a Cytosine on one strand is coupled with a Guanine on the other strand. Those strands are said to be complementary which means that the sequence on one strand determines the sequence on the other one. As about chromosomes, they can be circular, which is often the case in bacteria, or linear which is often the case in animals and plants. A segment of DNA is a part of this molecule made of consecutive nucleotides. A gene is a segment of DNA that contains the information needed to construct the other molecules in the cell.[1] See Figure 1.1.

What accounts for the diversity of living organisms is the principle of molecular evolution. This means that the DNA replicates itself with some inacurracy. A DNA molecule may evolve by mutations at the point of nu-

[1] A cell is the smallest unit of life that can replicate independently, and cells are often called the "building blocks of life".

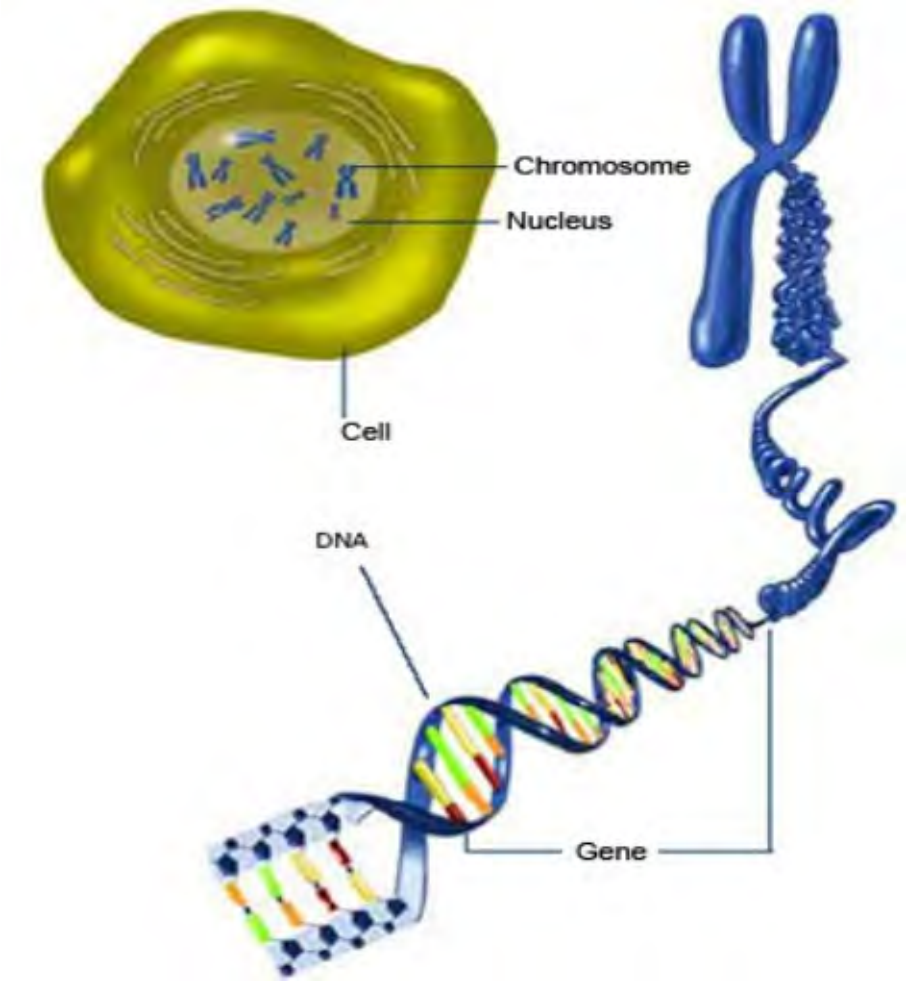

Figure 1.1: Representation of a cell.

cleotides, called point mutations, or by a larger scale mutations (more than one nucleotides), called rearrangements. Detecting the rearrangements is the goal of genome rearrangements problems which we study here.

There exist three different kinds of point mutations:

1. Substitution: one nucleotide is replaced by another.

2. Insertion: a nucleotide is inserted to the sequence.

3. Deletion: a nucleotide is removed from the sequence.

The main rearrangements according to Fertin [5] include the following:

- Deletion: a segment of the genome is lost. (see Figure 1.2)
- Transposition: a segment of the genome moves to another location. (see Figure 1.3)
- Inversion or reversal: a segment of the genome is reversed and strands are exchanged. (see Figure 1.4)
- Duplication: a segment of DNA is copied and inserted in the genome. There are three main standard types of duplications:
  - Tandem duplication: inserts the copy next to the original(see Figure 1.5).
  - Retrotransposition: inserts a copy of a gene at an arbitrary location in the genome.
  - Whole genome duplication: copies either the whole genome or some of its chromosomes.
- Reciprocal translocation: a segment of a chromosome that contains a telomere is exchanged with a segment of another chromosome that also contains a telomere. (see Figure 1.6)
- Fussion: two chromosomes are joined into one. (see Figure 1.7)
- Fission: one chromosome splits into two.
- Horizontal or lateral transfer: a segment of the genome is copied from one genome to another. This is mainly common in unicellular organisms.

Figure 1.2: Deletion of the dotted region in a chromosome.[5]

Figure 1.3: Transposition of the dotted region in a chromosome. [5]

Figure 1.4: Reversal of the underlined segment, resulting in the box segment. [5]

Figure 1.5: Tandem duplication of the dotted region in a chromosome. [5]

Figure 1.6: Reciprocal translocation of the dotted regions in two chromosomes. [5]

All these operations are applied on a genome at the level of DNA segments rather than on nucleotides. This is why a genome is often represented by a sequence of segments in that setting: they are the segments that are found

Figure 1.7: Fussion of two chromosomes.[5]

in an almost identical state in several species, not cut by rearrangements.

Two segments are said to be homologous if they derive from a common ancestor and are distinguished by a replication event (they end up in two different genomes) or by a duplication event (they both belong to the same genome). Homologous genes can also be orthologs or inparalogs.The interested reader can find more information on those subjects in [29], [27].

Genes are often taken as those homologous segments because, due to their functional utility, they are less subjected to small mutations and are rarely cut by rearrangements, which is not the case for other parts of the genome. A multigene family is a set of several similar genes, formed by duplication of a single original gene, and generally with similar biochemical functions. Multigene families typically consist of members with similar sequences and functions, though a high degree of divergence (at the sequence and/or functional level) does not lead to the removal of a gene from a gene family.

For instance, a gene family a, may consist of genes a_1, a_2 *and* a_3. Biologically speaking, we have one famous multigene family whose name is *histones*. The number of genes in this family can be between 100 to 1000.

For a gene family f $\in$ F and a genome G we use F(G,f) to denote the set of genes in G that come from f. We say a gene family f $\in$ F is a singleton in G if |F(G,f)| = 1; otherwise we say f is a multigene family [14], [11].

Generally, from one cell to another, one species to another, the content of DNA molecules is often similar, but their organization often differs dramatically. The mutations that affect this organization are called genome rearrangements.

The genome rearrangement problem is formulated in its general form as follows: given an ordered pair of genomes G_1 and G_2 and a set R of evolutionary events, find a sequence $[a_i,..,a_k]$ of a minimum length k, where $a_i \in R$ that transforms G_1 into G_2.

We are studying how similar two genomes are, in order to see whether

their genes derive from a common ancestor or whether they are homologous genes. If two genomes are similar, they have similar three-dimensional structures (as for example in proteins) and functions.

In the following chapter (Studied models about the rearrangement problem) we talk about different types of distances as breakpoint distance, exemplar distance and reversal distance. Then we discuss about different models that compute the edit distance. First, we mention the case that only reversals are allowed and then we talk about the case that only insertions and deletions are allowed. After that, we discuss about 'the difficult case', where all operations -insertions, deletions and reversals- are allowed. This is an NP-hard problem.

In Chapter: A few observations about the rearrangement problem , we propose a way to compute the distance between more than two genomes without computing all distances pairwise. We define some criteria and then we try to construct a new genome. Then we compute the distance between each genome and the constructed one. With that way we can make an estimation of the distance between all genomes without computing all of them. This is a fast way but we can't guarantee a good approximation of the distance between two genomes. We give some examples and we discuss about the results.

Chapter 2

Studied models

2.1 Representation

We represent genomes and genes as strings and symbols respectively. We use a string of symbols to represent a genome of genes. From now on, we interchangeably use the terms genome and gene or string of symbols, and symbols respectively. Let G_1 and G_2 be two genomes (i.e.,two strings of symbols). Each gene is represented by a signed symbol where the sign indicates the possible transcriptional direction of this gene. We give an example of the representation of genomes G_1 and G_2:

G_1: abcd-a-bf

G_2: abc-af-b

We may transform one genome into another by applying deletion, insertion or reversal operation. We usually refer to the distance between two genomes meaning the minimum number of operations that transform one genome into the other one.

Let's see some how we can transform G_1 into G_2:

- Deletion: A deletion of symbol d which is at the forth position of G_1 is the first operation that take place in order to transform G_1 into G_2. We call the remaining string of G_1 as G'_1 and we represent this operation as

  del(G_1,'4') = abc-a-bf =G'_1.

- Reversal : A reversal of the subsequence which is located between positions 5 and 6 of G'_1 creates the genome G_2. We represent this

operation as rev(G_1','5 6') = abc-afb = G_2.

- Hence, d(G_1,G_2) ≤ 2. It is easy to see that G_1 can not be transformed to G_2 by only one operation (deletion or reversal) and therefore the distance d(G_1,G_2)= 2, when only deletion or reversal operations are allowed.

2.2 Discussion about the model

According to Fertin [5] depending on the operations we want to study, different models can be used. Also, in case that:

a. the order of genes in each genome is known,

b. all genomes share the same set of genes,

c. all genomes contain a single copy of each gene ,

d. all genomes consist of a single chromosome,

genomes are modeled by permutations : each gene can be assigned a unique number and is found exactly once in each genome [5].

In general, genes do not appear exactly once in each genome: due to duplications and deletions, there can be several copies of a gene in a given genome. In that case genomes are modeled by strings (on the alphabet of genes) rather than permutations [5] .

Permutations were the first mathematical objects to serve as formal models for studying arrangements of DNA fragments among several species.

A permutation p is a bijection over the set [1,2,...,n]. The image of i$\in$[1,2,...,n] is denoted by p_i. The elements of p_i of the permutations are called genes when we are talking about genomes and genes.

The classical composition of permutations is applied as the composition of functions from right to left: when writing pos , we first apply p, then s, which results in the permutation (p_{s1},p_{s2}...p_{sn}). For example if p=(3142) and s= (4132) then pos = (2341).(See the following steps). Analytically, we are doing the following steps:

- A p permutation (3142) says : bring the third symbol of a sequence which consists of four symbols at the first position. Then bring the first symbol at the second position , the fourth symbol at the third position and the second symbol at the fourth position.

- The pos = (2341) is a synthesis of p and s.

p=3142

s=4132

In that case the s permutation says: bring the fourth symbol of p (which is 2) at the first position. Then, bring the first symbol of p (which is 3) at the second position, the third symbol of p (4) at the third position and the second symbol of p (1) at the last position. Finally pos =s(p) = (2341).

If we have the string abcd, where abcd are genes, then we applicate the permutation p=(3142) and p(abcd)=cadb. After that, we apply the permutation s=(4132) and s(cadb)=bcda. Or we apply the pos =(2341) in abcd and pos(abcd)=bcda. Finally pos(abcd)=s(p(abcd)).

Signed permutations model the organization of genomes better than unsigned permutations, because they take into account the double helix structure of DNA. Indeed, given one arbitrary starting point for a chromosome, each DNA strand has an orientation, and by complementarity, the orientation of one strand is the reverse of the orientation of the other. This orientation corresponds to the direction in which genes are transcribed on a given strand. Genome rearrangements such as reversals may change the order of the genes in a genome, and also the direction of transcription[5]. Identify the genes with the integers 1, . . . , n, with a plus or minus sign to indicate their orientation. The order and orientation of genomic markers will be represented by a signed permutation of 1, . . . , n, that is, by a bijective function p over [n, n] 0 such that $p_i = p_i$, where p_i = p(i) [12]. A signed permutation on (1,2,..n) is a permutation of the set (-n,...,-2,-1,1,2,...,n) that satisfies p_{-i} = -p_i.

The one-row notation is used for signed permutations. For example, the permutation :

-4 -3 -2 -1 1 2 3 4

3 -1 4 2-2-4 1-3

is simply denoted by (-2-4 1-3) in which we drop the mapping of the negative elements since keeping it would be redundant. For more details look at [5].

Permutations may represent not only genomes, but also mutations (rearrangements) in such a way that a mutation m, will transform one genome G_1 into the genome G_2.

One theory is that if a group of genes is appeared consecutively in several species, then they must have been presented in the same order in the ancestral species, and were not separated during evolution.

This is translated mathematically into the notions of adjacencies and breakpoints which definition varies according to the models they are used. Follows a section with the definitions of adjacencies, breakpoints and others.

2.3 Definitions and examples

2.3.1 Breakpoint and adjacency

The linear extension of a signed or unsigned permutation p of (1,2,....n) is the permutation (0,1,2,...n+1) defined by p'=(0,p_1,p_2,...,p_n, n+1). For the linear extension of signed permutations it is a convention that 0 has a positive sign [5].

According to Kececioglu [21] a breakpoint of a permutation p is a pair of adjacent positions (i, i+1) such that [p_{i+1}-p_i] $\neq$ 1. In other words (i, i+1) forms a breakpoint if values p_i and p_{i+1} are not consecutively increasing or decreasing. To handle the boundaries let p_0 has the value 0, p_{n+1} has the value n+1, as we said above, and allow i to range from 0 to n in the definition. Thus (0,1) is a breakpoint if $p_1 \neq 1$, and (n, n+1) is a breakpoint if $p_n \neq$ n. Notice that the identity permutation has no breakpoints, any other permutation has some breakpoints, and the number of breakpoints is at most n+1. When | p_{i+1}-p_i |= 1 values p_{i+1} and p_i are adjacent and we write $p_{i+1} \sim p_i$.

Let's see a simple example :
p = 5 6 2 1 3 4
Extend p with p_0=0 and p_7=7
Then p' = 0 5 6 2 1 3 4 7
Adjacencies are : (5 6) , (2 1) , (3 4)
Breakpoints are : (0 5), (6 2), (1 3), (4 7)

2.3.2 Matching

A matching between two genomes G_1 and G_2, that may contain multigene families, is defined as a one-to-one correspondence between a subset of genes

in G_1 and a subset of genes in G_2, such that each element of the matching is a pair of homologous genes. [1] If,

G_1 : a_1 a_2 a_3 b_1 c_1

G_2 : a_4 a_5 b_2 c_2

then, a matching between G_1 and G_2 is : [(a_1,a_4),(a_2,a_5),(c_1, c_2)]. The multigene families are: a consisting of a_1 a_2 a_3 (in G_1) and a_4 a_5 (in G_2), b consisting of b_1 (in G_1) and b_2 (in G_2) , c consisting of c_1 and c_2.

Depending on the model the distance can be defined as edit distance, breakpoint distance, inversion or reversal distance, exemplar distance e.t.c.

2.3.3 Edit distance

The edit distance between G_1 and G_2 is defined as the minimum number of events needed to transform G_1 to G_2 like insertions, deletions or reversals [5]. In section *representation* we gave an example where the edit distance between G_1 and G_2 is two because with two operations, one deletion and one reversal, G_1 is transformed to G_2 and this satisfies the definition of edit distance.

2.3.4 Breakpoint distance

The breakpoint distance between G_1 and G_2 is defined as the minimum number of breakpoints among all possible matchings M. [7]. Two genomes G_1 and G_2 are balanced if for any gene family a, there are as many genes that belong to a in G_1 as in G_2. The maximality condition of matching is that in any matching M between balanced G_1 and G_2, every gene of G_1 is matched to a gene of G_2 and conversely. In Figure 2.1 we see two unbalanced genomes G and H. They are unbalanced because genes j (G) and i (H) don't belong to any matching. [7]

Another definition for the breakpoint has been given by Blin et al [7]. So a breakpoint between G_1 and G_2 with respect to M is a pair of consecutive genes in G_1 (resp in G_2) that exactly one of them does not belong to M.

Let's see two balanced genomes G_1 and G_2.

G_1 : a_1 b_1 a_2 c_1

G_2 : a_3 b_2 c_2 a_4

[1] Remind the reader that two segments-genes are said to be homologous if they derive from a common ancestor and are distinguished by a replication event (they end up in two different genomes) or by a duplication event (they both belong to the same genome).

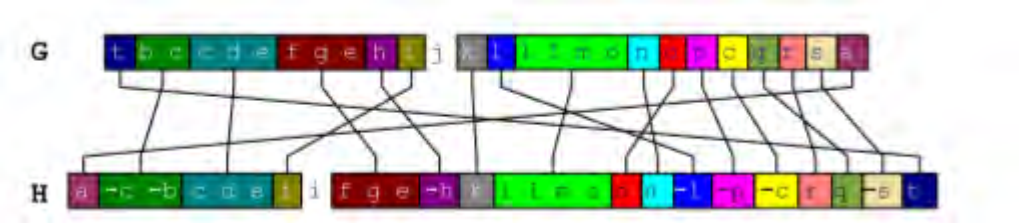

Figure 2.1: Illustration of a gene matching M between two unbalanced genomes G and H. [7]

In this case the breakpoint distance is one. The breakpoint is [(b_1 a_2), (b_2 c_2)]. According to Blin's definition the genes b_1,a_2 and b_2, c_2 are consecutive but a_2 and c_2 don't belong to matching.

2.3.5 Inversion or reversal distance

The inversion or reversal distance between G_1 and G_2 is the minimum number of inversions that must be applied to G_1 in order to produce G_2 [4]. The inversion (reversal) distance is the edit distance when only inversions are allowed.

2.3.6 Exemplar distance

In order to define the exemplar distance between G_1 and G_2 we need to modify G_1 and G_2 in the following way: we delete all genes but one of each gene family in G_1 and G_2. We call exemplar string of G_1 (G_2) the modified genome G_1 (G_2 respectively). Finally, we define as Exemplar Breakpoint Distance, EBD, between G_1 and G_2, the minimum breakpoint distance among all possible exemplar strings (derived from G_1 and G_2) of the same length.

Analogously, we define as Exemplar Reversal Distance, ERD, between G_1 and G_2, the minimum reversal distance among all possible exemplar strings (derived from G_1 and G_2) of the same length.

Sankoff et al [20] study the exemplar distance. In Sankoff's work, a generalized version of the genome rearrangement problem is formulated where each gene may occur in a number of copies. The idea is to delete all but one member of each gene family-its exemplar-in each of the two compared genomes, so as to minimize some rearrangement distance between the two

reduced genomes thus derived. For two exemplar strings of same length n (n=number of genes families), having the smallest possible rearrangement distance , which is called exemplar distance. It can be EBD (exemplar breakpoint distance) or ERD (exemplar reversal distance).

For example, consider the genomes G_1 and G_2 [20]:

G_1 : -b-aba-cdc

G_2 : a-aca-cbd

Based on the exemplar strings (-b-a-cd) and (cabd) the breakpoints are [(-b, c), (-c,b)].So the EBD = 2. Also the reversal is only one, a reversal of -b-a-c transforms the first exemplar string to the second. So the ERD = 1.

Analogously based on the exemplar strings (badc) and (cabd) the EBD=3 and the ERD=3.

The rearrangement problem is difficult to be solved. Trying to compute the edit distance when all operations are allowed, ended that is an NP-hard problem. In order to find a way to solve this difficult problem, new distances have been found like the breakpoint distance, the reversal distance, the exemplar distance, e.t.c., as we said above. We will see some of those in the following section.

2.4 Related work

In this section we will discuss about previously appeared algorithms for the computation of the distance between two genomes when:

1. Insertions and deletions are allowed.
2. Reversals are allowed.
3. Insertions, deletions and reversals are allowed.

2.4.1 Insertions and deletions

Wagner and Fischer [18] in 1974, define a general notion of distance between two strings and present an algorithm for computing the distance within linear time with respect to the product of the lengths of the strings. The operations they consider are:

1. Substitute one character with another single character.
2. Deleting one character.
3. Inserting a single character.

The algorithm may also be used to find the longest common subsequence of characters in two strings.

Eugene Myers [15] develops a simple O(ND) time and space algorithm to compute the distance between two strings where N is the sum of the lengths of the two compared strings (i.e., string A and string B) and D is the size of the minimum edit script for A and B. As script is defined the minimum number of deletions and insertions that transform one sequence into the other. We will talk about this algorithm below.

In case that only insertions and deletions are allowed in order to transform one sequence into another some algorithms have been developed. We are going to present one of them right now. At Myers survey [15] the problem is to find a longest common subsequence of two sequences A and B and a shortest edit script for transforming A into B. Using the perspective of finding the shortest path in an edit graph a simple O(ND) time and space algorithm is developed, as we said above.

Consider two sequences A and B of length N and M respectively. We create an edit graph by taking a grid of (N+1)(M+1) vertices. Horizontal edges connect vertex, (x-1,y) to (x,y) for x ∈ [1,N] and y ∈ [0,M]. Vertical edges connect vertex, (x,y-1) to (x,y) for x ∈ [0,N] and y ∈ [1,M]. If at the same position x of the two sequences the same symbol appears, then we connect the corresponding vertices with a diagonal edge. This way the vertex (x-1, y-1) goes to (x,y). At this edit graph the horizontal edges show the deletions and the vertical edges show the insertions respectively.

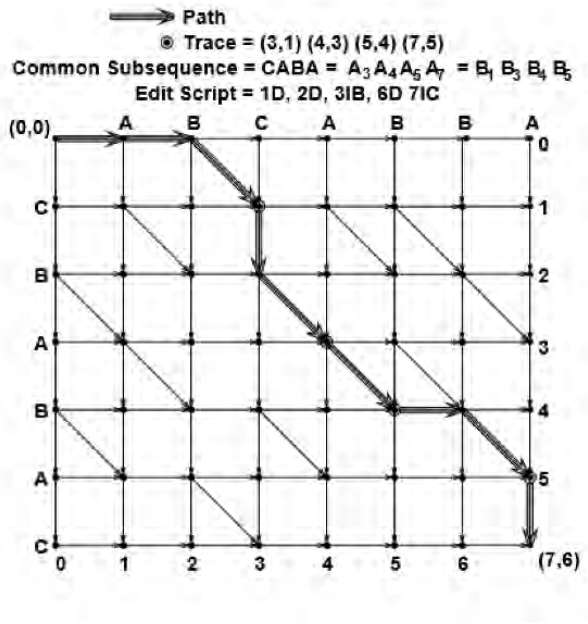

Figure 2.2: An edit graph. [15]

As we will see its suffices to look for a path with the most diagonal edges. It must begin from start (0,0) and ends at (N,M). Given a cost 0 for diagonals and a cost 1 for horizontal and vertical edges we select the path with the smallest cost.

Consider L as the length of the common subsequence of symbols between the two sequences at the path and these common symbols are defined as traces. Also consider D as the minimum number of operations (insertions and deletions) that transform one sequence (A) to another (B). We can prove that

the number of horizontal vertices in the edit graph is N-L and the number of vertical edges is M-L respectively. As we said above horizontal edges represent the deletions and vertical edges represent the insertions. So N-L is the number of deletions and M-L is the number of insertions. Finally, **D=N+M-2L**.

The edges of every path have the following direct interpretations in terms of the corresponding common subsequence and edit script. Each diagonal edge ending at (x,y) gives a symbol, a_x (= b_y), in the common subsequence. Each horizontal edge to point (x,y) corresponds to the delete command 'xD',(where x= the position and D=delete) and a sequence of vertical edges from (x,y) to (x,z) corresponds to the insert command, 'xI b_{y+1},...b_z', (where x=the position and I= insert).

Horizontal edges show the number of deletions and the positions that take place and vertical edges show the number of insertions and the positions that take place.

Let's discuss the provided by Myers example [15]. Consider two sequences A= ABCABBA and B= CBABAC. We construct the edit graph which is shown in Figure 2.2 [15].

As traces are defined the common symbols of A and B at the selected path and edit script is the minimum number of insertions and deletions (D). L is the length of the subsequence of common symbols between the two sequences, so in this path L=4. Horizontal edges show the number of deletions and the positions that take place. So the number of them is N-L=3. Respectively vertical edges show the number of insertions and the positions that take place. The insertions' number is M-L=2. As we can see in Figure 2.2, the edit script is 1D, 2D, 3IB, 6D, 7IC. It means the following: delete the first symbol of A, delete the second symbol of A, insert the symbol B at the third position (without deleting C), e.t.c. We observe that D= N+M-2L (D= 7+6-2*4=5) or D is equivalent to the cost of this path. (Let's remind that cost is 0 for diagonal edges and 1 for horizontal and vertical edges). Finally, D= 1+1+0+1+0+0+1+1=5.

The algorithm that Myers developed was based on Dijkstra and returns within $O((MN)^2)$ time.

2.4.2 Only reversals

Genomes frequently evolve by reversals r(i,j) that transform a gene order $p_1....p_ip_{i+1}....p_{j-1}p_j...p_n$ into $p_1....p_ip_{j-1}....p_{i+1}p_j...p_n$. The reversal distance

between two sequences p and s is the minimum number of reversal operations that transform p into s. Each reversal takes an arbitrary substring of elements, and reverses their order. These sequences or permutations can be signed or unsigned. (We remind the reader that in signed permutations symbols have a sign + or - which represents the direction of the gene).

Back to 1981 Watterson et al. [1] developed an algorithm, called 'rachet algorithm', to compute the reversal distance between two sequences. They proved that |n-2 | reversal operations are sufficient to bring any chromosome of n genes to order.

According to Caprara [3] (sorting by reversals) an unsigned permutation is an NP-hard problem, i.e., is NP-hard to decide whether the transformation can be done within less than r reversals.

Kececioglu and Sankoff gave the first approximation algorithm for sorting by reversals, (for signed permutations), that it never exceeds the minimum number by more than a factor of 2, and has a quadratic-time complexity [21].

Bafna and Pevzner [2] devise a polynomial-time approximation algorithm for sorting by reversals (for signed permutations) with an approximation ratio of 7/4.

Hannenhalli and Pevzner [23] gave the first polynomial-time algorithm for sorting by reversals signed permutations and proposed an $O(n^4)$ implementation of the algorithm. This algorithm is based on a combination of a breakpoint graph and a cycle graph. Let's see how those graphs are defined.

The breakpoint graph of a permutation p is an edge-colored graph G(p) with n+2 vertices $(p_0,p_1,...,p_n,p_{n+1})$ = (0,1,...,n, n+1). We connect vertices p_i and p_{i+1} by a black edge, if $| p_i - p_{i+1} | \neq 1$. We connect by a gray edge the vertices p_i and p_j if $| p_i - p_j | = 1$ and | i-j | $\neq 1$. A cycle in an edge-colored graph G is called alternating if the colors of every two consecutive edges of this cycle are distinct. In what follows, by cycles we mean alternating cycles. The length of a cycle C, l(C), is the number of black (or equivalently gray) edges in it.

Consider a cycle decomposition of G(p) into a maximum number c(p) of edge-disjoint alternating cycles.

Figure 2.3 shows the transformation of permutation p = (3 5 8 6 4 7 9 2 1 10 11) into (1,2,3,4,5,6,7,8,9,10) within a minimum number of 5 reversals.

Figure 2.4 shows the breakpoint graph G(p) of permutation p. C(p) = 4 since G(p) can be decomposed into three short cycles (8,9,7,6,8), (8,5,4,7,8), (3,5,6,4,3) and one long cycle (0,1,10,9,2,3,0). It has been showed by Bafna and Pevzner [2] that d(p) $\geq$ b(p) - c(p) , where d(p) = the reversal distance,

b(p) = the breakpoints or the black edges at G(p) and c(p) = the number of cycles at G(p).

In Figure 2.4 b(p) = 9, c(p) =4 , so d(p) $\geq$ 5.

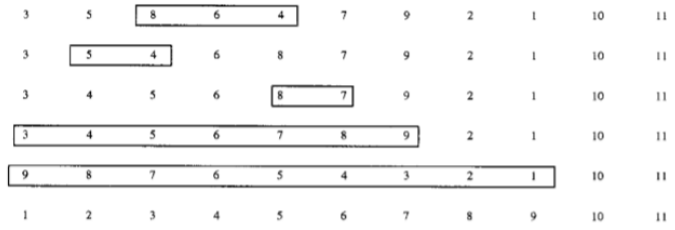

Figure 2.3: Optimal sorting of a permutation (3 5 8 6 4 7 9 2 1 10 11) by 5 reversals. [23]

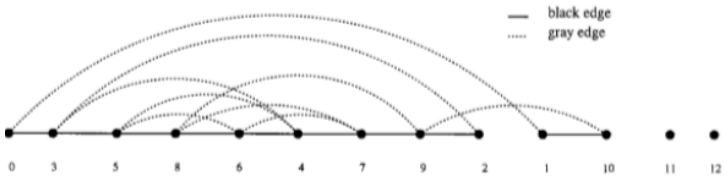

Figure 2.4: Breakpoint graph of the permutation (3 5 8 6 4 7 9 2 1 10 11). [23]

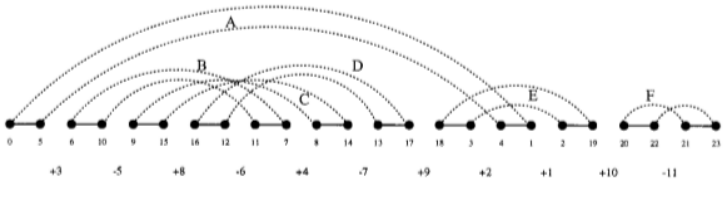

Figure 2.5: Transformation of a signed permutation (+3 -5 +8 -6 +4 -7 +9 +2 +1 +10 -11) into an unsigned permutation p and the breakpoint graph G(p). [17]

Hannenhalli and Pevzner constructed a function h(p) and showed that for signed permutations it holds: b(p) - c(p) + h(p) $\leq$ d(p) $\leq$ b(p) - c(p) +h(p) + 1.

In order to transform a signed permutation of length n into an unsigned permutation of length 2n replace the positive elements +x by 2x-1,2x and the negative elements -x by 2x, 2x-1. Figure 2.5. For more informations look at [23] and [8].

Hannenhalli and Berman [17] exploit a few combinatorial properties of the breakpoint graph G(p) of a permutation and propose an $O(n^2(a(n)))$ implementation of the algorithm where a is the inverse Ackerman function which grows very slowly.

Bader, Moret and Yan [4] present a new linear time algorithm for computing the reversal distance, based on Hannenhalli and Pevzner algorith, which is more efficient than that. It uses only a stack and is very easy to implement.

Finally, in 2010, Swenson, Rajan, Lin and Moret [10] provide two new sorting algorithms, a simple and fast randomized algorithm and a deterministic refinement. They conclude (without proving) that almost all signed permutations can be sorted by reversals in O(nlogn) time.

2.4.3 Insertions, deletions and reversals

Blin et al [7], showed that computing the minimum number of operations that transform a sequence p of length n to a sequence q of length m is NP-hard when insertions, deletions and reversals are allowed. They define the notion of breakpoint distance and show that it is a good approximation of the edit distance.

In order to compute the breakpoint distance, Blin et al [7] designed an algorithm based on the following intuition: long segments from the two genomes that match are likely to belong to a minimum gene matching. The algorithm focuses only on non-trivial segments (see below). If a gene appears in just one genome, it does not belong to any gene matching. If a gene appears exactly once in each genome, then these two occurrences have to be matched together. The main goal of this algorithm is to match parts of non-trivial segments that are common in the genomes. For two genomes G_1 and G_2 and a gene family **a**, the number of occurrences of **a** in G_1 and G_2 is the cardinality of this family. A gene family **a** is said to be trivial if either **a** appears in exactly one genome or **a** has cardinality exactly 2. Otherwise, **a** is said to be non-trivial. Genes are also called trivial or non-trivial analogously.

A segment of G_1 that contains only non-trivial genes is called a non-trivial segment.

The algorithm follows a branch-and-cut strategy. It tries all possible matchings between G_1 and G_2 starting with an empty matching and extending it in all possible ways, until the current matching induces more breakpoints than the best matching previously computed. Blin et al used another algorithm designed by Swenson et al to compute an initial matching that gives an upper bound for the breakpoint distance. Then a so called suffix-tree is used, that allows to find quickly the occurrences of a segment of G_1 that appears in G_2. The problem of computing the breakpoint distance between two genomes, even with just one non-trivial gene family, is NP-hard [7].

Swenson et al [9] propose an algorithm that computes an approximation of the distance of two genomes in the presence of insertions (including duplications), deletions or inversions. The approximation is good enough that it can be used in conjunction with a distance-based phylogenetic reconstruction method to reconstruct trees of reasonable sizes and very large pairwise distances with high accuracy.

Sankoff [20] proposes a branch and bound algorithm in order to calculate the exemplar breakpoint distance (EBD) and the exemplar reversal distance (ERD). This algorithm focuses on one gene family at a time, choosing the pair of exemplars which cause a minimum increase to the distance when inserted into the partial exemplar string already constructed. The monotonicity property [2] allows us to construct the exemplars strings by selecting one gene at a time, efficiently pruning unpromising parts of the search tree.

The strategy of this algorithm is that starts with gene families of size two -i.e., those with only one exemplar in each genome-, then tries those of size three etc. This is the fastest way to build long partial exemplar strings. The exemplar breakpoint distance problem is not only NP-hard but also unlikely to have polynomial time constant ratio approximation. For more information look at [20], [28].

Angibaud et al [19], propose a generic pseudo-boolean approach for computing the exact breakpoint distance between two genomes in presence of duplications and for both the exemplar and the maximum matching methods.

[2]Suppose we delete all occurrences of gene family a from genomes G_1 and G_2. If the exemplar distance decreases or remains the same, we say it is monotonic.

Nguyen et al [28], propose a divide and conquer approach for calculating the exemplar breakpoint distance between two genomes with multiple gene families. They calculate the exemplar breakpoint distance by partitioning the gene families into disjoint subsets such that two different families in different subsets are independent. Then, they find the pair of exemplars of gene families in each subset at a time, and finally merge all the exemplars together to obtain two optimal exemplars of the given genomes. Tests with both simulated and real datasets show that Nguyen's et al divide and conquer approach is much more efficient than the branch and bound approach of Sankoff [20].

More recently, Shao and Moret [13] propose a very fast and exact algorithm to compute the exemplar breakpoint distance, based on new insights in the underlying structure of genome rearrangements and exemplars. This algorithm runs much faster and scales much better than Nguyen's et al algorithm. Shao and Moret define the notion of adjacency. Two genes g and h on the same chromosome form a potential adjacency, written as [g,h], if we can remove all genes between g and h and yet retain at least one gene in every gene family in each genome. Adjacencies are just potential where nothing need be removed. We say a potential adjacency [g,h] is simple if g and h come from different genes families. Given two genomes G_1 and G_2, we say two simple potential adjacencies $[g_1,h_1] \in G_1$ and $[g_2,h_2] \in G_2$ form a Pair of Shared Simple Potential Adjacencies (PSSPA), written as $([g_1,h_1],[g_2,h_2])$ if g_1 and g_2 have the same sign and come from the same gene family and the same holds for h_1 and h_2.

Another method to compute the minimum number of breakpoints and the maximum number of adjacencies between two genomes in presence of duplications is proposed by Angibaud et al in 2008 [25].

Given two genomes G_1 and G_2 and a model (exemplar, intermediate or maximum matching) Angibaud et al wish to find a matching which:

1. Is appropriate to the model.

2. Yields the optimal number of breakpoints/adjacencies, where optimal means minimum for breakpoints and maximum for adjacencies.

Finally Shao and Moret in 2016 [14] proposed very fast exact algorithms for two breakpoint distance problems. The intermediate breakpoint distance

problem (I-BDP) and the maximum matching breakpoint distance problem (M-BDP). These algorithms rely on a compact integer-linear programming.

The M-model says: use as many matched pairs as possible, until all genes in the genome with the smaller number of copies in gene family are matched.

The I-model says: the matching must contain at least one matched pair per gene family -so that the focus is clearly on minimizing the distance, not on meeting constraints on the matchings.

We have seen above the E-model (Exemplar) which says: select exactly one matched pair in gene family. (See Figure 2.6)

For more information look at [14] and [13].

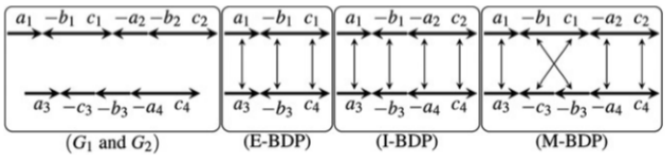

Figure 2.6: Examples for E-BDP, I-MDP and M-BDP. Optimal matchings are shown within each model. [14]

Follows a list of useful bibliography about every model separately. Also, table 2.1 gives us the complexity of the most important models that we have discussed in this chapter.

1. Insertions and Deletions :Fischer [18] ,Eygene Myers [15].
2. Only reversals for unsigned permutations : Caprara [3].
3. Only reversals for signed permutations: Hannenhalli and Pevzner [8], [17], [23], Swenson, Rajan, Lin, Moret, [10].
4. Insertions, Deletions and Reversals: Blin et al [7], Swenson et al [9], Sankoff [20], Angibaud et al [19], [24], [25], Nguyen [28], Shao and Moret [11], [14], [22], [26].

Table 2.1: **Models and complexity.**

Models	**Complexity**
Insertions and Deletions	O(n^2)
Only reversals for unsigned permutations	NP-hard
Only reversals for signed permutations	O(nlogn)
Insertions, Deletions and Reversals	NP-hard

Chapter 3

A few observations

Consider the following scenario. We are given a set of genomes and the goal is to find the distance between any two of them. We would like to investigate a trade-off between "good" approximations of all distances and computation times. A possible approach towards this goal could be the following:

- Construct a *new genome* and compute the distances between the given genomes and the new one.
- Compute an approximation of the distance between any two of the given genomes via the new created genome.

3.1 Only reversals

Let us discuss this approach for a number of models with respect to the allowed operations. Assume the following example. We consider five genomes with only singletons and the same number of genes. Suppose only reversals are allowed. We will construct a new genome in order to compute faster, good approximations of distances. Hence, the first task is how to construct such a new genome. Consider criterion:

1. Fill each position i, of the *new genome* with the gene which appears in the majority of the given genomes in position i, where $1 \leq i \leq$ n and n is the length of the genomes. Ties could be arbitrarily broken or one could look at the subsequences consisting of the selected genes.

2. Compute the distance between the constructed genome and each one of the given five genomes.

Let's give an example as shown in Figure 3.1 where only reversals take place in order to compute all the distances.

In the following Figures by 1,2,3,4,5,6... we mean G_1, G_2, G_3, G_4, G_5, G_6. Also, we call G^* the *new genome*.

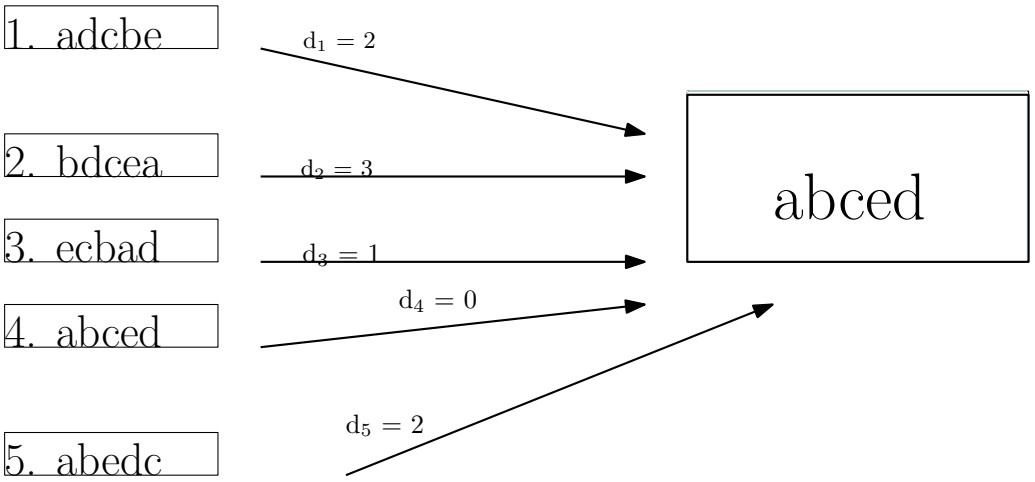

Figure 3.1: Make a *new genome* when only reversals are allowed.

Gene *a* appears at the first position 3 times (i.e. in G_1, G_4, G_5), so put *a* in the first position of the *new genome*. Then, genes *b* and *d* appear at the second position 2 times. Select *b* for this position because the sequence *ab* appears more often than *ad*. Analogously, genes *c*, *e*, *d* take the remaining third, forth and fifth positions respectively. The constructed genome G^*, **abced**, is also one of the initial five genes (G_4).

To compute an estimation of the distance $d(G_1,G_2)$ we compute the distances $d(G_1,G^*)$, $d(G^*,G_2)$ [(or d_1,d_2)]. Then the estimation of the distance $d(G_1,G_2)$ is defined as $d'(G_1,G_2) = d(G_1,G^*) + d(G^*,G_2)$ or $d'(G_1,G_2) = d_1 + d_2$. (For the following text d'= the estimation of distances and d= the optimal distances).

As shown in Figure 3.1, $d_1 = 2$ and $d_2 = 3$. In order to transform G_1 to the *new genome* (G^*) reverse the subsequence dcb , then reverse the subsequence de. Finally we take the new genome abced with two reversals. In order to

transform G_2 to the *new genome* (G^*) we need three reversals. So $d_1= 2$ and $d_2= 3$. The optimal distance $d(G_1, G_2) = 3$ while $d'(G_1, G_2)= d_1+d_2 = 2+3=5$. In that case the approximation ratio is 5/3.

Analogously we compute an estimation of $d(G_1, G_3)$ given by the sum of distances $d_1+ d_3$. In that case $d(G_1, G_3) = d'(G_1, G_3) = d_1 + d_3= 3$ and the approximation ratio is 1.

As it appeared in the above example we compute five distances only (instead of all pair combinations of the five genomes which are ten) and we manage to approximate the optimal distances with remarkable success: we achieve in most of the cases the optimal distances. It follows Table 3.1 with all the distances (d and d') among all the genomes for this example of Figure 3.1 and the approximation ratios.

Table 3.1: Results of all distances-Only reversals

d	d'	Approximation ratio
$d(G_1,G_2)=3$	$d'(G_1,G_2)=d_1+d_2= 5$	5/3
$d(G_1,G_3)=3$	$d'(G_1,G_3)= d_1+d_3=3$	1
$d(G_1,G_4)=2$	$d'(G_1,G_4)= d_1+d_4=2$	1
$d(G_1,G_5)=3$	$d'(G_1,G_5)= d_1+d_5=4$	4/3
$d(G_2,G_3)=3$	$d'(G_2,G_3)= d_2+d_3=4$	4/3
$d(G_2,G_4)=3$	$d'(G_2,G_4)= d_2+d_4=3$	1
$d(G_2,G_5)=3$	$d'(G_2,G_5)= d_2+d_5=3$	1
$d(G_3,G_4)=1$	$d'(G_3,G_4)= d_3+d_4=1$	1
$d(G_3,G_5)=3$	$d'(G_3,G_5)= d_3+d_5=3$	1
$d(G_4,G_5)=2$	$d'(G_4,G_5)= d_4+d_5=2$	1

As we can see in Table 3.1 seven to ten times d'= d. $d(G_1,G_3)= d'(G_1,G_3)$, $d(G_1,G_4)= d'(G_1,G_4)$, $d(G_2,G_4)= d'(G_2,G_4)$, $d(G_2,G_5)= d'(G_2,G_5)$, $d(G_3,G_4)= d'(G_3,G_4)$, $d(G_3,G_5)= d'(G_3,G_5)$, $d(G_4,G_5)= d'(G_4,G_5)$. The average of approximation ratios is 1.14.

In case that only reversals are allowed and all genomes have the same length n, consider n-1 as an easy upper bound for the reversal distance between two genomes. So in that example the worst case would be :$d(G_1,G_2)= d(G_1,G_3)= d(G_1,G_4)= d(G_1,G_5)= d(G_2,G_3)= d(G_2,G_4)= d(G_2,G_5)= d(G_3,G_4)= d(G_3,G_5)= d(G_4,G_5)= 4$. The average of the approximation ratio is 1.73. See Table 3.2. So we have managed a good approximation.

However in general the approximation could be worse.

Table 3.2: Optimal distance -Upper Bound -Only reversals

d	Upper Bound	Approximation ratio
d(G_1,G_2)=3	d(G_1,G_2)=4	4/3
d(G_1,G_3)=3	d(G_1,G_3)= 4	4/3
d(G_1,G_4)=2	d(G_1,G_4)= 4	2
d(G_1,G_5)=3	d(G_1,G_5)= 4	4/3
d(G_2,G_3)=3	d(G_2,G_3)= 4	4/3
d(G_2,G_4)=3	d(G_2,G_4)= 4	4/3
d(G_2,G_5)=3	d(G_2,G_5)= 4	4/3
d(G_3,G_4)=1	d(G_3,G_4)= 4	4
d(G_3,G_5)=3	d(G_3,G_5)= 4	4/3
d(G_4,G_5)=2	d'(G_4,G_5)= 4	2

Let's see another example as in Figure 3.2. In order to create *a new genome* we are looking for which symbol (gene) is located at the same position of each genome more times. So, symbol d appears at the fourth position three times (G_1,G_2,G_5) and we put it at the fourth position of the created genome. Then, we are doing the same for symbol c which appears at the third position more times (G_1,G_5) and we put it at the third position of the created genome. Analogously symbol b goes at the second position. We observe that symbol a appears at the second position two times (G_2,G_5) and at the fifth position two times (G_3,G_4). We choose the fifth position because symbol b obtain the second one. Finally, symbol e goes at the first position because it is the only one which has remained. The *new genome* is **ebcda**.

We compute the distances d_1, d_2, d_3, d_4, d_5 which are the distances between G_1, G_2, G_3, G_4, G_5, and the constructed genome G^*. (Remind that only reversals are allowed.) As we can see in Figure 3.2 d_1= 2 because we need two reversals in order to transform G_1 into G^*. First, reverse the genome G_1 , then reverse the substring dcb and take ebcda. d_2= 3 because we need three reversals in order to transform G_2 into G^*. First reverse 'bad' and take eabdc, then reverse ab and take ebadc After that, reverse adc and take ebcda. d_3= 3 because we need three reversals. Reverse cbde and take edbca, then reverse db and take ebdca and the third reversal is dc and take ebcda. d_4= 1 because it takes place only one reversal , reverse dcbe and take ebcda. d_5= 2 because we have two reversals. First, reverse acdb and take ebdca, then reverse dc and take ebcda.

The optimal distance d(G_1, G_2) = 2. Only two reversals transform G_1 to G_2, while the distance d'(G_1, G_2) via the new genome is d'(G_1, G_2)= d(G_1, G^*) + d(G_2, G^*) = d_1+ d_2 = 2+3 = 5. Also, the optimal distance d(G_2, G_3) = 3, while d'(G_2, G_3)= d(G_2, G^*) + d(G_3, G^*) = d_2+ d_3 = 3+3 = 6. d(G_4, G_5) =3 while d'(G_4, G_5)= d(G_4, G^*) + d(G_5, G^*) = d_4+ d_5 = 1+2 = 3. Doing all comparisons we take as a conslusion that in this example we are far from the optimal distance for most of the pairs. The following Table 3.3 gives the results of all distances (d and d') and the approximation ratios.

Table 3.3: Results of all distances-bad approximation.

d	d'	Approximation ratio
d(G_1,G_2)=2	d'(G_1,G_2)=d_1+d_2= 5	5/2
d(G_1,G_3)=3	d'(G_1,G_3)= d_1+d_3=5	5/3
d(G_1,G_4)=2	d'(G_1,G_4)= d_1+d_4=3	3/2
d(G_1,G_5)=3	d'(G_1,G_5)= d_1+d_5=4	4/3
d(G_2,G_3)=3	d'(G_2,G_3)= d_2+d_3=6	2
d(G_2,G_4)=3	d'(G_2,G_4)= d_2+d_4=4	4/3
d(G_2,G_5)=2	d'(G_2,G_5)= d_2+d_5=5	5/2
d(G_3,G_4)=2	d'(G_3,G_4)= d_3+d_4=4	2
d(G_3,G_5)=3	d'(G_3,G_5)= d_3+d_5=5	5/3
d(G_4,G_5)=3	d'(G_4,G_5)= d_4+d_5=3	1

As we can see in Table 3.3 we have managed a bad approximation of the computation of distances. Only d(G_4,G_5)= d'(G_4,G_5). The average of approximation ratios is 1,75. This is a bad approximation because it is bigger than the upper bound which is 1.73.

We give another example in Figure 3.3. We construct the new genome according to our criteria. We but a and b at the first and second position respectively because they appear in those positions in all genomes. Then we select the positions for c,d,e randomly. We construct the new genome abced which is the same as G_4. (c appears at third position two times (G_1,G_4), as e (G_2, G_5). d appears at fourth position two times (G_1, G_5) as e (G_3,G_4). d and c appear at fifth position two times (G_2,G_4, and G_3,G_5 respectively)).

We compute the distances as previously. So, d_1 = d(G_1, G^*)= 1, because only one reversal ed to de transforms G_1 into G^*. d_2 = d(G_2, G^*)= 1, because only one reversal ec to ce transforms G_2 into G^*. d_3 = d(G_3, G^*)= 1, because only one reversal dec to ced transforms G_3 into G^*. d_4 = d(G_4,

G^*)= 0, because it is the same. d_5 = d(G_5, G^*)= 2, because we need two reversals in order to transform G_5 into G^*, reverse edc to cde and then de to ed. The optimal distance d(G_1, G_2) = 2, while the distance via the new genome is d'(G_1, G_2)= d_1 +d_2= 2. So d(G_1, G_2) = d'(G_1, G_2)= d_1 +d_2. Also, d(G_2, G_3) = d'(G_2, G_3)= d_2 +d_3.

Table3.4 gives us all the results of all distances (d and d') and the approximation ratios.

Table 3.4: Results of all distances-good approximation.

d	d'	Approximation ratio
d(G_1,G_2)=2	d'(G_1,G_2)=d_1+d_2= 2	1
d(G_1,G_3)=2	d'(G_1,G_3)= d_1+d_3=2	1
d(G_1,G_4)=1	d'(G_1,G_4)= d_1+d_4=1	1
d(G_1,G_5)=1	d'(G_1,G_5)= d_1+d_5=3	3
d(G_2,G_3)=2	d'(G_2,G_3)= d_2+d_3=2	1
d(G_2,G_4)=1	d'(G_2,G_4)= d_2+d_4=1	1
d(G_2,G_5)=1	d'(G_2,G_5)= d_2+d_5=3	3
d(G_3,G_4)=1	d'(G_3,G_4)= d_3+d_4=1	1
d(G_3,G_5)=1	d'(G_3,G_5)= d_3+d_5=3	3
d(G_4,G_5)=2	d'(G_4,G_5)= d_4+d_5=2	1

The approximation in that case is 1.6. (The average of approximation ratios is 1.6).

As a conclusion we can say that in case that only reversals are allowed, we have managed good approximations when all the genomes have a common substring. And is even closer to the optimal distance when the created genome is one of the initial ones.

3.2 Insertions-Deletions

Let's discuss the case that only insertions and deletions are allowed. In Figure 3.4 we show an example with five genomes. We create a new genome in order to compute all pairwise distances via the constructed genome. First, we find the common genes in all genomes. Those are a,b,c,d. Then we put them in order according to the criteria we have defined above, criteria 1 and 2 and we create the new genome **abdc**. Finally, we compute the five distances d_1, d_2,

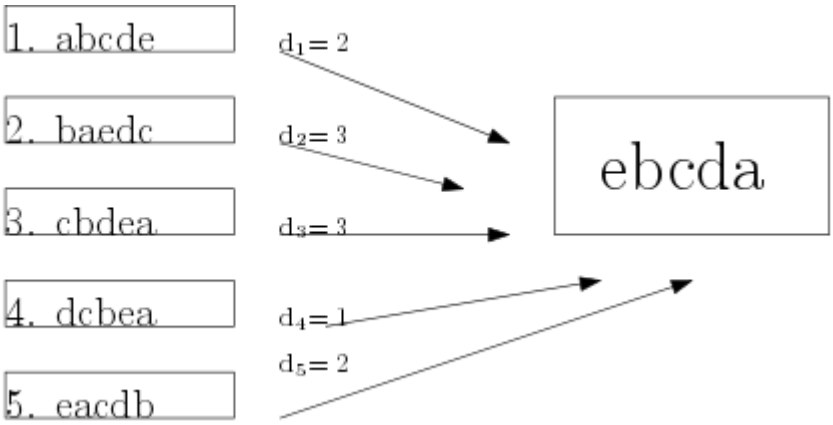

Figure 3.2: Bad approximation when only reversals are allowed.

d_3, d_4, d_5. (Remind that according to Myers we can delete only one symbol every time but we can insert either one symbol or a substring of symbols).

In order to transform G_1 to G^* *(new genome)* we first delete the symbol (gene) f, then we delete the symbol e. So we make two 'moves' and the distance between those two genomes is 2 (d_1=2). Analogously the distance between G_2 and G^* is 3. We first delete the symbol c, then e and insert c at the end next to d. So d_2=3. The distance d'(G_1, G_2) is given by the sum of distances d_1+ d_2 which is 5. The optimal distance between them is 3 because in order to transform G_1 into G_2 directly, we first delete the symbol f from G_1, then symbol d , and we insert it next to symbol b. We have done three 'moves' and the optimal distance is 3. Table 3.5 gives us the results of all distances d and d' and the approximation ratios.

We have achieved a bad approximation because d $\neq$ d' in all cases. The approximation ratio is 1.8. In order to transform one genome into another when only insertions and deletions are allowed, we define n as an easy upper bound. (n= the length of the genome).

By using the upper bound we have defined above, we take the following results, see Table 3.6. The approximation ratio in that case is 1.6. $1.6 < 1.8$ so our method has failed.

Let's create another *new genome* for the same example. This time we take the common symbols that appear in all genomes (strings) and these ones that appear to the most of the strings as we can see in Figure 3.5. So we make the **abdce** string. We are doing the same as before as we see in

Table 3.5: Results of all distances-Only insert-delete.

d	d'	Approximation ratio
$d(G_1,G_2)=3$	$d'(G_1,G_2)=d_1+d_2=5$	5/3
$d(G_1,G_3)=4$	$d'(G_1,G_3)=d_1+d_3=7$	7/4
$d(G_1,G_4)=4$	$d'(G_1,G_4)=d_1+d_4=5$	5/4
$d(G_1,G_5)=3$	$d'(G_1,G_5)=d_1+d_5=5$	5/3
$d(G_2,G_3)=3$	$d'(G_2,G_3)=d_2+d_3=8$	8/3
$d(G_2,G_4)=4$	$d'(G_2,G_4)=d_2+d_4=6$	6/4
$d(G_2,G_5)=2$	$d'(G_2,G_5)=d_2+d_5=6$	3
$d(G_3,G_4)=6$	$d'(G_3,G_4)=d_3+d_4=8$	8/6
$d(G_3,G_5)=4$	$d'(G_3,G_5)=d_3+d_5=8$	2
$d(G_4,G_5)=5$	$d'(G_4,G_5)=d_4+d_5=6$	6/5

Table 3.6: Only insert-delete-Optimal distance-Upper Bound.

d	Upper bound	Approximation ratio
$d(G_1,G_2)=3$	$d(G_1,G_2)=6$	2
$d(G_1,G_3)=4$	$d(G_1,G_3)=6$	6/4
$d(G_1,G_4)=4$	$d(G_1,G_4)=6$	6/4
$d(G_1,G_5)=3$	$d(G_1,G_5)=6$	2
$d(G_2,G_3)=3$	$d(G_2,G_3)=5$	5/3
$d(G_2,G_4)=4$	$d(G_2,G_4)=5$	5/4
$d(G_2,G_5)=2$	$d(G_2,G_5)=5$	5/2
$d(G_3,G_4)=6$	$d(G_3,G_4)=6$	1
$d(G_3,G_5)=4$	$d(G_3,G_5)=6$	6/4
$d(G_4,G_5)=5$	$d(G_4,G_5)=5$	1

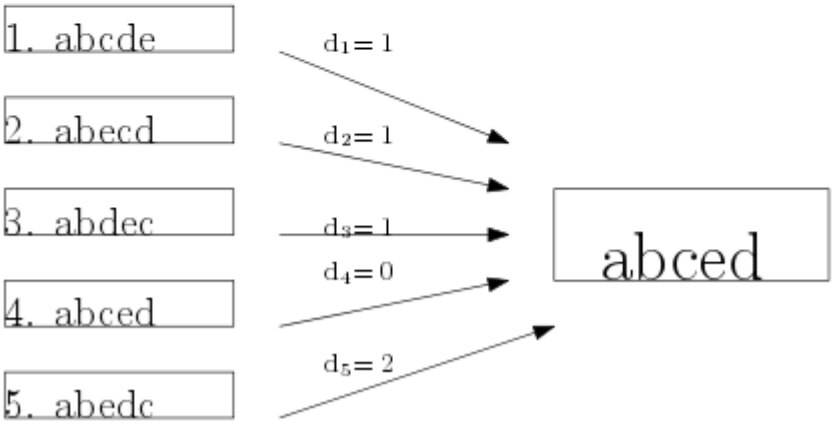

Figure 3.3: Good approximation when only reversals are allowed.

Figure 3.6.

Table 3.7 gives us the results of all distances d an d' and the approximation ratios.

The approximation ratio is 1.4.

Finally we try to construct a new genome by using all symbols that appear to all genomes (a,b,c,d,e,f) according to our criteria and the counters, Fig 3.5.

Figure 3.7 shows the results. Also, Table 3.8 gives us all distances d and d' and the approximation ratios for this example.

The approximation ratio is 1.7. This is a bad approximation again.

As a conclusion we can say that in case that only insertions and deletions are allowed we can't manage good approximations of the computation of the distances. When we construct the new genome by using the common symbols

Table 3.7: Better results of all distances-Only insert-delete.

d	d'	Approximation ratio
$d(G_1,G_2)=3$	$d'(G_1,G_2)=d_1+d_2=3$	1
$d(G_1,G_3)=4$	$d'(G_1,G_3)=d_1+d_3=5$	5/4
$d(G_1,G_4)=4$	$d'(G_1,G_4)=d_1+d_4=5$	5/4
$d(G_1,G_5)=3$	$d'(G_1,G_5)=d_1+d_5=3$	1
$d(G_2,G_3)=3$	$d'(G_2,G_3)=d_2+d_3=6$	2
$d(G_2,G_4)=4$	$d'(G_2,G_4)=d_2+d_4=6$	6/4
$d(G_2,G_5)=2$	$d'(G_2,G_5)=d_2+d_5=4$	2
$d(G_3,G_4)=6$	$d'(G_3,G_4)=d_3+d_4=8$	8/6
$d(G_3,G_5)=4$	$d'(G_3,G_5)=d_3+d_5=6$	6/4
$d(G_4,G_5)=5$	$d'(G_4,G_5)=d_4+d_5=6$	6/5

Table 3.8: Bad results of all distances-Only insert-delete.

d	d'	Approximation ratio
$d(G_1,G_2)=3$	$d'(G_1,G_2)=d_1+d_2=5$	5/3
$d(G_1,G_3)=4$	$d'(G_1,G_3)=d_1+d_3=5$	5/4
$d(G_1,G_4)=4$	$d'(G_1,G_4)=d_1+d_4=6$	6/4
$d(G_1,G_5)=3$	$d'(G_1,G_5)=d_1+d_5=5$	5/3
$d(G_2,G_3)=3$	$d'(G_2,G_3)=d_2+d_3=6$	2
$d(G_2,G_4)=4$	$d'(G_2,G_4)=d_2+d_4=7$	7/4
$d(G_2,G_5)=2$	$d'(G_2,G_5)=d_2+d_5=6$	3
$d(G_3,G_4)=6$	$d'(G_3,G_4)=d_3+d_4=7$	7/6
$d(G_3,G_5)=4$	$d'(G_3,G_5)=d_3+d_5=6$	6/4
$d(G_4,G_5)=5$	$d'(G_4,G_5)=d_4+d_5=7$	7/5

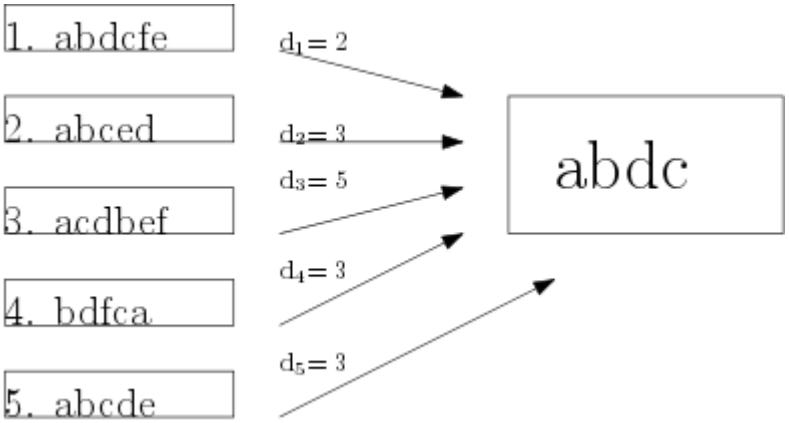

Figure 3.4: Make a *new genome* when only deletions and insertions are allowed.

that appear to all genomes and these ones that appear to the most of them we take better results. Also notice that we have a better approximation when the sum of distances $d_1 + d_2 + d_3 + d_4 + d_5$ is the smallest one.

3.3 Reversals-Deletions-Insertions

Suppose that the given genomes are not of the same length and the set of allowed operations consists of deletions, insertions and reversals.

Let's see an example. Consider five genomes with only singletons. We wish to create a *new genome*. The criteria are the same as above, but the question is how many singletons the *new genome* must have in order to achieve a good approximation. As shown in Figure 3.8 some genomes have five genes while others have six genes. One natural approach is to take all six genes in the *new genome* . We define as L the biggest length of the genomes, which is the genome with the most genes.

For instance, the distance d'(G_1, G_2) via the *new genome* is the sum $d_1 + d_2 = 1 + 3 = 4$, while the optimal distance d(G_1, G_2) is 3. The distance d'(G_1, G_3) via the *new genome* is the sum $d_1 + d_3 = 1 + 1 = 2$. The optimal distance d(G_1, G_3) is also 2 . All the distances d and d' and the approximation ratios for the example of Figure 3.8 are given in Table 3.8.

The approximation ratio is 1.7.

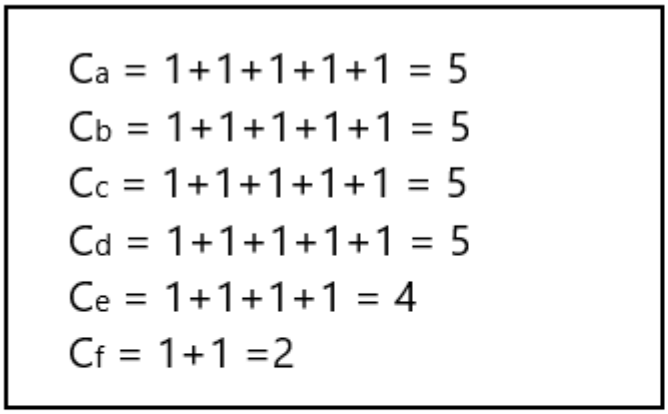

Figure 3.5: Counters of each gene in each genome.

Now we are going to create a new genome with the common genes only. We define as l the smallest length of genes (which are the common ones). See Figure 3.9 and Table 3.10.

The approximation ratio is 1.8.

In both cases we haven't achieved a good approximation. We will try to create an algorithm in order to achieve better approximation. We are doing the following steps:

1. We first find in linear time the total number of occurences for each gene (i.e., a appears five times while f appears two times). See Figure 3.10 . We can find this with appropriate counters (c) in linear time. See Figure 3.5.

2. We select the genes by scanning each sequence that appear more times in the same position in each genome. For example gene a appears in

Table 3.9: Results of all distances when G*=L. Insert-delete-reverse.

d	d'	Approximation ratio
$d(G_1,G_2)=3$	$d'(G_1,G_2)=d_1+d_2=4$	4/3
$d(G_1,G_3)=2$	$d'(G_1,G_3)=d_1+d_3=2$	1
$d(G_1,G_4)=4$	$d'(G_1,G_4)=d_1+d_4=5$	5/4
$d(G_1,G_5)=2$	$d'(G_1,G_5)=d_1+d_5=3$	3/2
$d(G_2,G_3)=4$	$d'(G_2,G_3)=d_2+d_3=4$	1
$d(G_2,G_4)=4$	$d'(G_2,G_4)=d_2+d_4=7$	7/4
$d(G_2,G_5)=1$	$d'(G_2,G_5)=d_2+d_5=5$	5
$d(G_3,G_4)=3$	$d'(G_3,G_4)=d_3+d_4=5$	5/3
$d(G_3,G_5)=3$	$d'(G_3,G_5)=d_3+d_5=3$	1
$d(G_4,G_5)=5$	$d'(G_4,G_5)=d_4+d_5=6$	6/5

Table 3.10: Results of all distances when G*= *l*. Insert-delete-reverse.

d	d'	Approximation ratio
$d(G_1,G_2)=3$	$d'(G_1,G_2)=d_1+d_2=4$	4/3
$d(G_1,G_3)=2$	$d'(G_1,G_3)=d_1+d_3=5$	5/2
$d(G_1,G_4)=4$	$d'(G_1,G_4)=d_1+d_4=5$	5/4
$d(G_1,G_5)=2$	$d'(G_1,G_5)=d_1+d_5=4$	2
$d(G_2,G_3)=4$	$d'(G_2,G_3)=d_2+d_3=5$	5/4
$d(G_2,G_4)=4$	$d'(G_2,G_4)=d_2+d_4=5$	5/4
$d(G_2,G_5)=1$	$d'(G_2,G_5)=d_2+d_5=4$	4
$d(G_3,G_4)=3$	$d'(G_3,G_4)=d_3+d_4=6$	2
$d(G_3,G_5)=3$	$d'(G_3,G_5)=d_3+d_5=5$	5/3
$d(G_4,G_5)=5$	$d'(G_4,G_5)=d_4+d_5=5$	1

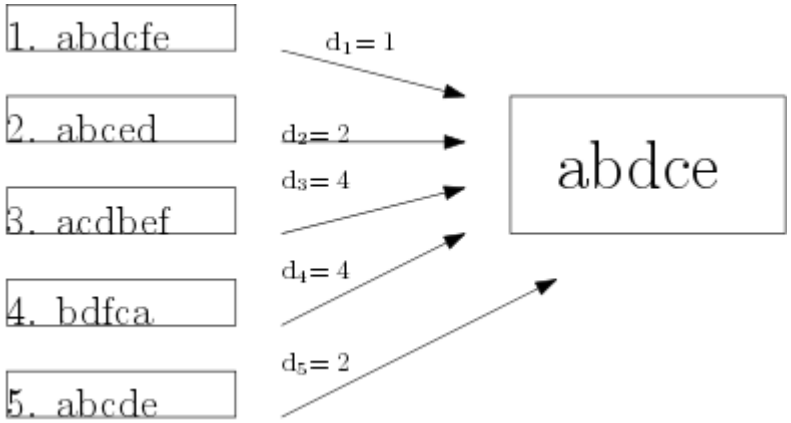

Figure 3.6: Make a better *new genome* when only insertions and deletions are allowed.

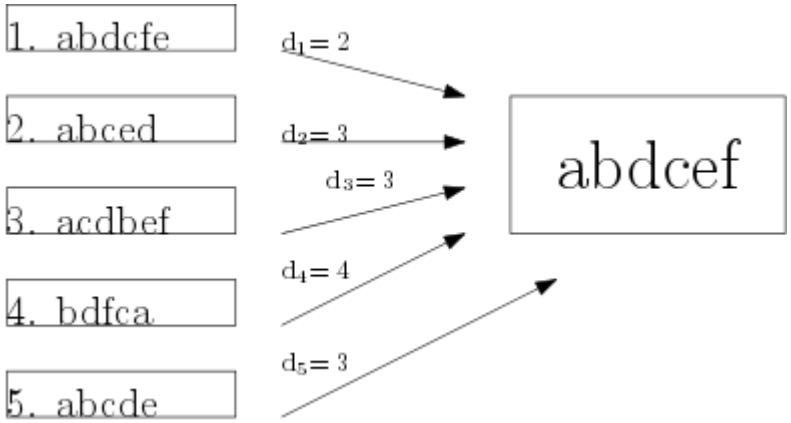

Figure 3.7: Make a worse *new genome* when only insertions and deletions are allowed.

four of the five genomes at the first position, so we put it first in the constructed genome e.t.c..

3. Finally, we select the genes with the bigger counter. If L is the biggest length of the genomes, we would like to create the *new genome* near to this length. Consider as x the length of the constructerd genome. Then it must be $l <$ x $<$ L.

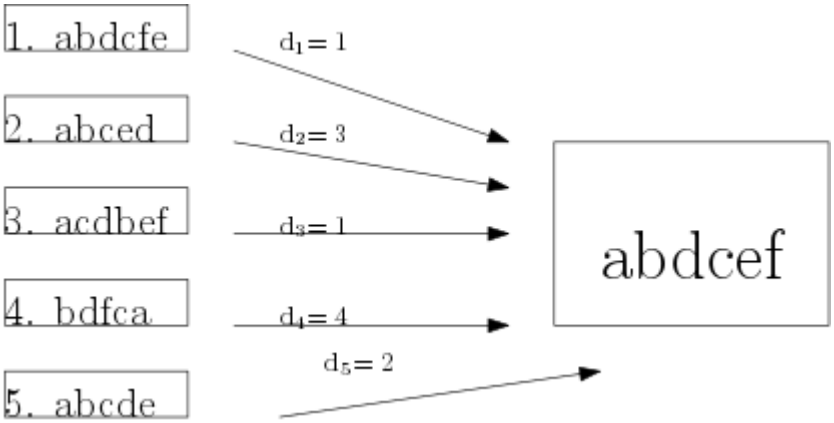

Figure 3.8: Make a *new genome* G*= L when deletions, insertions and reversals are allowed.

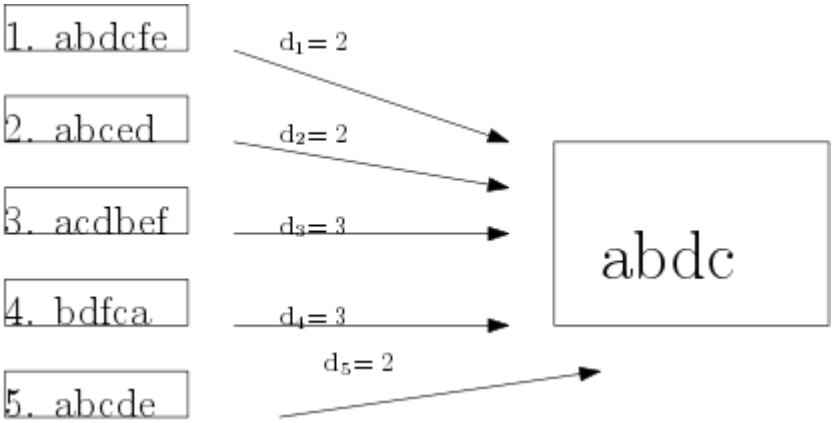

Figure 3.9: Make a *new genome* G*= l when deletions, insertions and reversals are allowed.

Table 3.11 gives us the results. The approximation ratio is 1,4 which is a good approximation. So, in case we can do all the possible operations, insertion, deletion and reversal, we choose to create a new genome which contains all the common genes and these ones that appear to the most of the genomes. Observe that we take better approximation when the sum of the distances $d_1 + d_2 + \ldots d_{n-1} + d_n$ between each genome and the created genome is the smaller one.

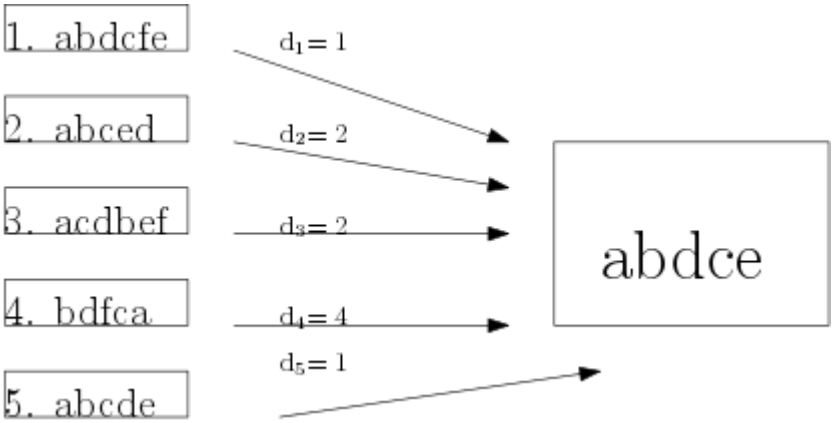

Figure 3.10: Make a *new genome* G*= x when deletions, insertions and reversals are allowed.

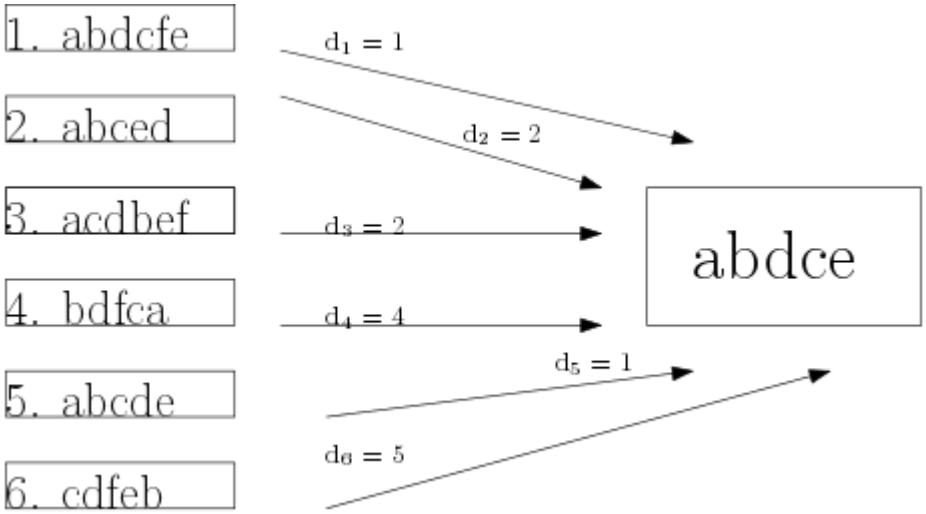

Figure 3.11: Make a *new genome* when deletions, insertions and reversals are allowed following the new algorithm. The distance between G_4 and the new genome is d_4=4 and the distance between G_6 and the new genome is d_6=5.

In case that the distance between some genomes and the new genome is big enough, we propose to create another new genome. We call it 'internal' genome. As we see in Figure 3.11 the distance between G_4 and the new

Table 3.11: Results of all distances when G*=x. Insert-delete-reverse.

d	d'	Approximation ratio
$d(G_1,G_2)=3$	$d'(G_1,G_2)=d_1+d_2=3$	1
$d(G_1,G_3)=2$	$d'(G_1,G_3)=d_1+d_3=3$	3/2
$d(G_1,G_4)=4$	$d'(G_1,G_4)=d_1+d_4=5$	5/4
$d(G_1,G_5)=2$	$d'(G_1,G_5)=d_1+d_5=2$	1
$d(G_2,G_3)=4$	$d'(G_2,G_3)=d_2+d_3=4$	1
$d(G_2,G_4)=4$	$d'(G_2,G_4)=d_2+d_4=6$	6/4
$d(G_2,G_5)=1$	$d'(G_2,G_5)=d_2+d_5=3$	3
$d(G_3,G_4)=3$	$d'(G_3,G_4)=d_3+d_4=6$	2
$d(G_3,G_5)=3$	$d'(G_3,G_5)=d_3+d_5=3$	1
$d(G_4,G_5)=5$	$d'(G_4,G_5)=d_4+d_5=5$	1

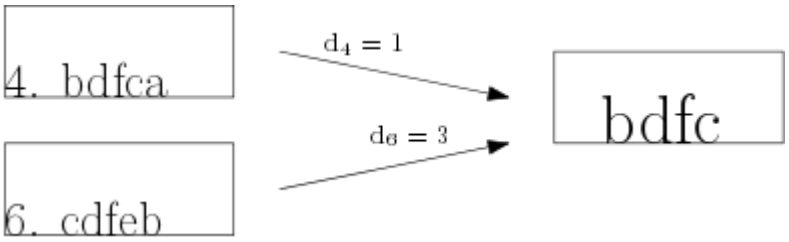

Figure 3.12: Construction of an 'internal' new genome.

genome is d_4=4 and the distance between G_6 and the new genome is d_6=5. So it would be better to create an internal genome only for those genomes that their distances are very big, according to the criteria we have defined above. In that way we might get better approximations of the distances. In Figure 3.12 $d_4 = 1$ and $d_6 = 3$. Finally, the distance between G_4 and G_6 via the internal genome is $d_4 + d_6 = 1+3 = 4$, which equals the optimal!

Let's see a few observations about the complexity of our model:

- The complexity to create the new genome in all cases is O(L), where L= l_1 + l_2 + ...+ l_k, l_i is the length of sequence i.
- The complexity to compute the distance between each one of the initial genomes with the constructed genome is M*p, where p = the complexity of the algorithm we use and M= the number of genomes.
- The complexity to compute the optimal distance (without the new genome, but by comparing each pair of genomes) would be M^2*p.

Chapter 4

Conclusion

The rearrangement problem is difficult to be solved. We would like to find an algorithm that computes the distance between more than two genomes easy and fast. Our method was based on the construction of a new genome. We propose some criteria in order to create a new genome and then we compute all the distances through this genome. Thus we get an estimation of all distances without computing all of them. We can't quarantee a good approximation. After a series of theoritical experiments we can present a few observations about our study:

1. If only reversals are allowed, we achieve a good approximation when all the genomes have a common substring of genes and not only common genes.

2. If only insertions and deletions are allowed, we haven't achieved a good approximation.

3. If all the operations can take place (insertions, deletions and reversals) it is very important to create the correct *new genome*. In order to create the best *new genome* we select all genes that appear to all genomes and these ones that apppear to the most of the genomes. If l is the length of the common substring and L is the biggest length of the genomes (the number of all genes) , then we need a new genome whose length is between l and L. Consider this length as x. Then $l < x <$ L. Also, if the distance between some of the initial genomes and the constucted one is big, we propose to make an 'internal' new genome only for those genomes.

4. Finally, we see that if the sum of distances between the initial genomes and the constructed one- $d_1 + d_2 + \ldots\ d_{n-1} + d_n$ - is the smallest one , it is a first sigth that we are close to a good approximation.

Many models have been studied during the years. We can't prove which model is more efficient but it is an open theme for study and research.

Bibliography

[1] G.A.Watterson.W.J.Ewens. T.E.Hall. A.Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99(1):1–7, November 1982.

[2] Vineet Bafna and Pavel A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, June 1993.

[3] Alberto Caprara. Sorting by reversals is difficult. *RECOMB '97 Proceedings of the first annual international conference on Computational molecular biology*, pages 75–83, January 1997.

[4] Bernard M.E. Moret David A. Bader and Mi Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, July 2004.

[5] Rusu I. Fertin G., Labbare A. *Combinatorics of genome rearrangements*. MIT PRESS, 2009.

[6] Douglas J. Futuyma. *Evolutionary biology*. Sinauer Associates Inc, 2nd edition, 1986.

[7] Cedric Chauve Guillaume Blin, Guillaume Fertin. The breakpoint distance for signed sequences. *1st Conference on Algorithms and Computational Methods for biochemical and Evolutionary Networks (CompBioNets'04)*, 3:3–16, December 2004.

[8] S. Hannenhalli and P.A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, 1995.

[9] Mark Marron Krister M. Swenson. Approximating the true evolutionary distance between two genomes. *Journal Journal of Experimental Algorithmics (JEA)*, 12(3.5), 2008.

[10] Yu Lin Bernard M. E. Moret Krister M. Swenson, Vaibhav Rajan. Sorting signed permutations by inversions in o(nlogn) time. *Research in Computational Molecular Biology*, 5541:386–399, 2009.

[11] BME Moret M Shao, Y Lin. An exact algorithm to compute the dcj distance for genomes with duplicate genes. *RECOMB: International Conference on Research in Computational Molecular Biology*, April 2014. 18th Annual International Conference, RECOMB 2014, Pittsburgh, PA, USA, April 2-5, 2014, Proceedings.

[12] Eric Tannier Marie-France Sagot. Perfect sorting by reversals. *Computing and Combinatorics*, 3595:42–51, 2005.

[13] Shao Mingfu and Moret Bernard M.E. A fast and exact algorithm for the exemplar breakpoint distance. *Computational Biology*, 23(5):337–346, May 2016.

[14] Bernard M. E. Moret Mingfu Shao. On computing breakpoint distances for genomes with duplicate genes. *RECOMB 2016: Research in Computational Molecular Biology*, 9649:189–203, April 2016. Part of the Lecture Notes in Computer Science book series (LNCS).

[15] Eugene W. Myers. Ano(nd) difference algorithm and its variations. *Algorithmica*, 1(1-4):251266, November 1986.

[16] CAREY NESSA. *THE EPIGENETICS REVOLUTION, HOW MODERN BIOLOGY IS REWRITING OUR UNDERSTANDING OF GENETICS, DISEASE AND INHERITANCE*. 2015.

[17] Sridhar Hannenhalli Piotr Berman. Fast sorting by reversal. *Part of the Lecture Notes in Computer Science book series*, 1075:168–185, 1996.

[18] Michael J. Fischer Robert A. Wagner. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, January 1974.

[19] I Rusu A Thvenin S Angibaud, G Fertin. A pseudo-boolean programming approach for computing the breakpoint distance between two

genomes with duplicate genes. *Comparative Genomics*, 4751 of Lecture notes in Computer Science:16–29, September 2007. RECOMB 2007 International Workshop, RECOMB-CG 2007, San Diego, CA, USA, September 16-18, 2007. Proceedings.

[20] David Sankoff. Genome rearrangement with gene families. *Bioinformatics 15*, pages 909–917, November 1999.

[21] J. KececiogluD. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13(1):180–210, February 1995.

[22] Richard Friedberg Sophia Yancopoulos, Oliver Attie. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics 21*, 16:3340–3346, June 2005.

[23] Pavel A. Pevzner Sridhar Hannenhalli. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM (JACM)*, 46(1):1–27, January 1999.

[24] Irena Rusu Sbastien Angibaud, Guillaume Fertin and Stphane Vialette. A pseudo-boolean framework for computing rearrangement distances between genomes with duplicates. *Journal of Computational Biology*, 14(4):379–393, June 2007.

[25] Irena Rusu Annelyse Thvenin Stphane Vialette Sbastien Angibaud, Guillaume Fertin. Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. *Journal of Computational Biology*, 15(8):1093–1115, September 2008.

[26] Kenneth Srensen. Distance measures based on the edit distance for permutation-type representations. *Journal of Heuristics*, 13:35–47, February 2007.

[27] Zheng Fu Peng Nan Yang Zhong Stefano Lonardi Tao Jiang Xin Chen, Jie Zheng. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2(4):302–315, October 2005.

[28] C. Thach Nguyen Y. C. Tay Louxin Zhang. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics*, 21:21712176, May 2005.

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