

ขั้นตอนวิธีการจัดเรียงหน่วยพันธุกรรมที่ซ้ำซ้อนหลายครั้งไปยังเป้าหมายใดๆ



นายจักริน สุขสวัสดิ์ชน

สถาบันวิทยบริการ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์


คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2548

ISBN 974-53-2530-9

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

MULTIPLE DUPLICATED GENE REARRANGEMENT ALGORITHM
FOR ARBITRARY TARGET



Mr.Jakkarin Suksawatchon

สถาบันวิทยบริการ

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Science

Department of Mathematics

Faculty of Science

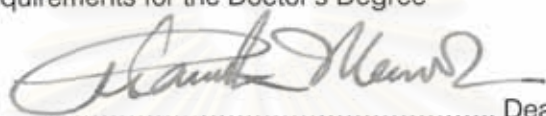
Chulalongkorn University

Academic year 2005


ISBN 974-53-2530-9


Thesis Title MULTIPLE DUPLICATED GENE REARRANGEMENT ALGORITHM
FOR ARBITRARY TARGET
By Mr. Jakkarin Suksawatchon
Field of Study Computer Science
Thesis Advisor Professor Chidchanok Lursinsap, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial
Fulfillment of the Requirements for the Doctor's Degree



..... Dean of the Faculty of Science
(Professor Piamsak Menasveta, Ph.D.)


THESIS COMMITTEE



..... Chairman
(Associate Professor Peraphon Sophatsathit, Ph.D.)


..... Thesis Advisor
(Professor Chidchanok Lursinsap, Ph.D.)


..... Member
(Associate Professor Boonserm Kijisirikul, Ph.D.)


..... Member
(Assistant Professor Paisan Nakmahachalasint, Ph.D.)


..... Member
(Rath Pichyangkura, Ph.D.)


..... Member
(Kanuengnij Kubola, Ph.D.)

จักริน สุขสวัสดิ์ชน : ขั้นตอนวิธีการจัดเรียงหน่วยพันธุกรรมที่ซ้ำซ้อนหลายครั้งไปยัง
เป้าหมายใด ๆ. (MULTIPLE DUPLICATED GENE REARRANGEMENT ALGORITHM
FOR ARBITRARY TARGET) อ. ที่ปรึกษา: ศ. ดร. ชิดชนก เหลือสินทรัพย์,
73 หน้า. ISBN 974-53-2530-9.

การจัดเรียงหน่วยพันธุกรรมสามารถนำมาใช้เป็นเครื่องมือวิเคราะห์ห่างทางวิวัฒนาการระหว่าง
สองสิ่งมีชีวิตใด ๆ โดยใช้ลำดับของยีนในสายพันธุกรรม ซึ่งจุดมุ่งหมายหลักในการศึกษาการจัดเรียง
หน่วยพันธุกรรมนั้นเป็นการหาลำดับในการจัดเรียงหน่วยพันธุกรรมจากจีโนมหนึ่งไปอีกจีโนมหนึ่ง
โดยใช้จำนวนที่น้อยที่สุดของเหตุการณ์ที่ใช้ในการจัดเรียงหน่วยพันธุกรรม เช่น การกลายพันธุ์แบบ
หมุนวน (reversal) , การกลายพันธุ์แบบแทรก (transposition) และการกลายพันธุ์แบบการหมุนวน
และแทรก (transversal) Hannenhalli และ Pevzner ได้นำเสนอแนวคิดของกราฟที่มีจุดเปลี่ยน
(breakpoint graph) มาใช้ในการคำนวณหาระยะห่างทางวิวัฒนาการของการกลายพันธุ์แบบหมุนวน
ซึ่งขั้นตอนวิธีของ Hannenhalli และ Pevzner เป็นการหาจำนวนที่ใช้ในการเรียงลำดับเหตุการณ์ที่
น้อยที่สุด ที่ใช้ในการจัดเรียงหน่วยพันธุกรรมจากจีโนมหนึ่งไปยังอีกจีโนมหนึ่งของสายพันธุกรรมที่มี
การกลายพันธุ์ ซึ่งขั้นตอนวิธีดังกล่าวเหมาะสำหรับสายพันธุกรรมที่ไม่มีการซ้ำซ้อนกันของยีน แต่ใน
ความเป็นจริงพบว่าในสายพันธุกรรมหนึ่งจะมีการซ้ำซ้อนกันของยีนอยู่ประมาณ 40 เปอร์เซ็นต์ จึงทำ
ให้แนวคิดของกราฟที่มีจุดเปลี่ยนไม่เหมาะสมที่จะนำมาประยุกต์ใช้หาระยะห่างทางวิวัฒนาการของ
การกลายพันธุ์แบบหมุนวนสำหรับสายพันธุกรรมที่มีการซ้ำซ้อนกันของยีน ดังนั้นวิทยานิพนธ์นี้จึงเป็น
การขยายแนวคิดของ Hannenhalli และ Pevzner สำหรับสายพันธุกรรมที่มีการซ้ำซ้อนกันของยีน ซึ่ง
เป็นการนำเสนอขั้นตอนวิธีใหม่ที่มีการเรียนรู้ด้วยตัวเอง (heuristic) เพื่อใช้ในการคำนวณหาระยะห่าง
ทางวิวัฒนาการของการกลายพันธุ์แบบหมุนวนสำหรับสายพันธุกรรมที่มีการซ้ำซ้อนกันของยีน โดยใช้
หลักการของไบนารีอินทีเจอร์โปรแกรมมิ่ง (Binary Integer Programming) นอกจากนี้เราสามารถ
นำมาประยุกต์ใช้ในการหาระยะห่างทางวิวัฒนาการของการกลายพันธุ์แบบแทรก และการกลายพันธุ์
แบบการหมุนวนและแทรก ผลการทดลองจากข้อมูลที่สร้างขึ้นเองและข้อมูลจริงในทางชีววิทยา พบว่า
วิธีการที่นำเสนอนี้สามารถคำนวณหาระยะห่างทางวิวัฒนาการของการกลายพันธุ์แบบหมุนวนได้อย่าง
ถูกต้องแม่นยำ

ภาควิชา.....คณิตศาสตร์..... ลายมือชื่อนิสิต..... Jakkarin.....
สาขาวิชา.....วิทยาการคอมพิวเตอร์..... ลายมือชื่ออาจารย์ที่ปรึกษา..... C.kw.....
ปีการศึกษา 2548

4473806023 : MAJOR COMPUTER SCIENCE

KEY WORD: MULTI-GENE FAMILIES / GENOME REARRANGEMENT / GENE DUPLICATION.

JAKKARIN SUKSAWATCHON: MULTIPLE DUPLICATED GENE REARRANGEMENT ALGORITHM FOR ARBITRARY TARGET. THESIS ADVISOR: PROF. CHIDCHANOK LURSINSAP, Ph.D., 73 pp. ISBN 974-53-2530-9.

The genome rearrangement can be used as a measurement of evolutionary distance between two organisms using the gene order of genomic data. The ultimate goal of genome rearrangement studies is finding a series of rearrangement scenarios to transform one genome into another by using the minimum number of the rearrangement events such as reversal, transposition and transversal. Hannenhalli and Pevzner introduced the notion of breakpoint graph to compute the reversal distance. Their algorithm determines the minimum number of reversals required for rearranging a genome to another -- but only in the absence of gene duplicates. However, duplicates often account for 40% of a genome. Therefore, the idea of breakpoint graph is clearly unsuitable for computing the reversal distances for multi-gene families. In this dissertation, we show how to extend Hannenhalli and Pevzner's approach to deal with genomes with multi-gene families. We propose a new heuristic algorithm to compute the reversal distances between two genomes with multi-gene families via binary integer programming. Then, we will be applied for the others two rearrangement distance (transposition and transversal). The experimental results on both synthetic and real biological data demonstrate that the proposed algorithm is able to find the reversal distance with high accuracy.

Department.....**Mathematics**..... Student's signature.....*Jakkarin*.....

Field of study.....**Computer Science**..... Advisor's signature.....*C. Lursinsap*.....

Academic year 2005

Acknowledgements

During my years as a Ph.D. student, I have received a lot of tuition, care and friendship from several people, some of which I wish to thank here.

- First of all I would like to thank the Thai Government who sponsor the research scholarships.
- During my time as a Ph.D.s student, I am grateful to my supervisor, Prof.Chidchanok Lursinsap, to whom with his advice, guidance and care, help me to overcome the necessary difficulties of the process of research and make this dissertation possible.
- I would also like to thank my co-supervisor, Dr.Mikeal Boden at University of Queenlands, Australia, who gives me a wonderful suggestions in Ph.D. research methodologies.
- My thanks also goes to dissertation committee with their advice and guidance, help focus my research activities.
- I would also like to thank all my colleague at the Department of Computer Science, Burapha University, especially Mr.Seree Chinodom, Mr.Jira Jaturanon, Dr.Krisana Chinnasarn, Dr.Suwanna Rasmeequan, Ms.Benchaporn Jantarakongkul, and my friends (Mr.Surachai Wachirahattapong, Mr.Wittawas Puntumjinda) for their wormest care support and being patient during my doubtful stage.
- Finally, my deepest gratitude goes to Suksawatchon's family, for their sponsor, love and care and especially Ms.Ureerat Wattanachon, for her love, wormest care that inspire this research.

Table of Contents

	Page
Thai Abstract	iv
English Abstract	v
Acknowledgements	vi
Table of Contents	vii
List of Tables	ix
List of Figures	x
CHAPTER	
1 INTRODUCTION	1
1.1 Introduction and Problem Review	1
1.2 Research Objectives	5
1.3 Scopes of the Study	5
1.4 Research Plans	6
1.5 Research Advantages	6
2 THEORIES AND LITERATURE REVIEWS	7
2.1 Introductory Biology	7
2.2 The Formal Representation of The Genome	9
2.2.1 Synteny	10
2.2.2 Order and Polarity	11
2.2.3 Linearity <i>versus</i> Circularity	12
2.2.4 Multi-Gene Families	12
2.3 Operations and Distances	13
2.3.1 Alignment Traces	13
2.3.2 Breakpoints	13

CHAPTER	Page
2.4 Edit Distances	15
2.5 Breakpoint Graph : Notations and Definitions	16
2.6 Minimal Reversal Distance	18
2.7 Minimal Transposition Distance	22
2.8 Minimal Transversal Distance	24
2.9 Literature Reviews on Gene Duplication	25
2.9.1 Exemplar Distance	26
2.9.2 The Assignment of Orthologous Genes via Genome Rearrangement	28
3 PROPOSED METHOD	30
3.1 Heuristic Algorithm to Computing the Nearest Minimal Edit Distances for Multi-gene Families via Binary Integer Programming (EDMF)	31
3.2 Observation	42
4 EXPERIMENTAL RESULTS	46
4.1 Minimum Reversal Distance with Multi-Gene Families	46
4.1.1 Synthetic data	46
4.1.2 Real biological data	49
4.2 Minimum Transposition Distance with Multi-Gene Families	51
4.2.1 Synthetic data	51
4.3 Minimum Transversal Distance with Multi-Gene Families	52
4.3.1 Synthetic data	52
5 CONCLUSION	56
References	59
Biography	64

List of Tables

TABLE.		Page
1.1	Prevalence of <i>gene duplication</i> in all three domains of life.	4
3.1	An example of two copies of an element 2 and three copies of an element 3	31
3.2	An example of the renaming results, π_{new} and ϕ_{new} by temporary names x_1, x_2, x_3 and y_1, y_2	31
3.3	The seven possible edge groups ($E_1 - E_7$) and the 24 <i>non-deterministic</i> gray edges ($e_1 - e_{24}$) of the <i>incomplete</i> breakpoint graph (<i>IG</i>).	35
3.4	An example of setting value for l_{ik} and w_k for $24 \leq k \leq 1$ and $7 \leq i \leq 1$	45
4.1	Results of reversal distance from EDMF comparing with SRDD.	50

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

List of Figures

FIGURE	Page
1.1 The example of three rearrangement events (reversal, transposition, and transversal).	3
2.1 The example of DNA : the molecule of life.	8
2.2 The different levels of chromosome structures considered in the genome rearrangement literature.	10
2.3 Breakpoint graph B and overlap graph O for the permutation $\pi = (+5, +1, +3, +2, +4, +6, +11, +7, +9, +8, +10, +12, -13, +14)$ with respect to the identity permutation of size $n = 14$. Connected component e is <i>oriented</i> , a, b, c and d are <i>unoriented</i> , and f is <i>trivial</i> . Unoriented components b and d are hurdles and also superhurdles, but unoriented component a and c are nonhurdles because they separate b and d	17
2.4 The example of sorting by reversals of the permutation $\pi' = (0, 4, 3, 2, 1, 7, 8, 5, 6, 9, 10, 11)$	20
2.5 The example of sorting by transpositions of the permutation $\pi' = (0, 3, 4, 1, 2, 7, 8, 5, 6, 9, 10, 11)$	23
2.6 The example of sorting by transversals of the permutation $\pi' = (0, 8, 7, 6, 5, 1, 2, 3, 4, 9, 10, 11)$	25
2.7 The evolutionary model considered in the exemplar analysis. Using the breakpoint distance as a criterion, the chosen exemplar are the undelined ones.	26
3.1 An example of an <i>incomplete</i> breakpoint graph IG for π'	33

FIGURE	Page
3.2 Cycles in IG named a_1 and a_2 are <i>type A</i> and $b_1, b_2, b_3, b_4, b_5, b_6, b_7$ are <i>type B</i>	38
3.3 An example of three cases for connecting edge e_i . The <i>deterministic</i> gray edge is denoted by a dashed line while the <i>non-deterministic</i> gray edges are denoted by dotted lines. v_i is a vertex in the <i>incomplete</i> graph.	40
4.1 The reversal distance of both SRDD and EDMF comparing with the exact reversal distance from GRAPPA (<i>invdist</i>) for genome length 100 and 10 gene families.	47
4.2 The reversal distance of both SRDD and EDMF comparing with the exact reversal distance from GRAPPA (<i>invdist</i>) for genome length 100 and 20 gene families.	48
4.3 The size distribution of gene families constructed for each pair of genomes. The number of gene families of size one are not shown in the figure for clarity (they are 355, 321, and 348, respectively).	49
4.4 The transposition distance of EDMF comparing with the exact transposition distance from GRAPPA for genome length 100 and 10 gene families.	52
4.5 The transposition distance of EDMF comparing with the exact transposition distance from GRAPPA for genome length 100 and 20 gene families.	53
4.6 The transversal distance of EDMF comparing with the exact transversal distance from GRAPPA for genome length 100 and 10 and 20 gene families.	55

CHAPTER I

INTRODUCTION

1.1 Introduction and Problem Review

In order to understand evolution, we would like to know how all species on earth are related to each other. This problem has been studied for several centuries, using techniques like morphology, anatomy, physiology and paleontology. With the advent of techniques to study the genome of a species, these investigations have entered a new era. Suddenly, there is no shortage of material to study, rather an abundance.

Comparative genomics is the art of extracting reliable genomic data and quantifying it into evolutionary relations. To achieve this, we may use any genomic data available to us. In this dissertation, we will concentrate on gene order data, *i.e.* data regarding the position of the genes in the genome. In a species, there are a large number of genes and the ordering of the genes is hugely important. Since we are interested in the order of genes, we label each gene a unique number. This number can be unsigned. If a label of a gene is signed, for instance -5, it means that this gene is the reverse of another gene, which is labeled as 5.

When comparing genomes in different species, a piece of chromosome in one species can be moved or copied to a different location in another species. Basically to compare two genomes, we often find that these two genomes contain the same set of genes. But the order of the genes is different in different genomes.

For example, it was found that both human X chromosome and mouse X chromosome contain eight genes which are identical. They are labeled as $1, 2, \dots, 8$.

In human, the genes are ordered as

$4, 6, 1, 7, 2, 3, 5, 8$

and in mouse, they are ordered as

$1, 2, 3, 4, 5, 6, 7, 8$.

Similarly, it was found that a set of genes are in cabbage as

$1, -5, 4, -3, 2$

and in turnip, they are ordered as

$1, 2, 3, 4, 5$.

Therefore, we talk about genome rearrangement.

The comparison of two genomes is significant because it provides us some insight as to how far away genetically these species are. If two genomes are similar to each other, they are genetically close; otherwise they are not. The question is how we measure the similarity of two genomes. Essentially, we measure the similarity of two genomes by measuring how easy it is to transform one genome to another by some operations. Here's the ultimate goal of genome rearrangement studies. Therefore discovering what rearrangement events have occurred, and what was their order of occurrence, there is a chance to get a better understanding of the evolutionary process.

The genome rearrangement approaches have been widely studied in the last decade [1, 2, 3, 4, 5, 6, 7, 8, 9]. The major focus has been to infer the most economical scenario of elementary operations transforming one linear order of genes into another. In this context, a *reversal* operation has been the most studied rearrangement event, followed by *transposition* and the *transversal* (inverted transposition) operations. In a reversal, a segment of genes is taken out of the genome and put back in reversed order. In a transposition, a segment of genes is taken out and put back at another place in

the genome, and a transversal is a segment of gene taken out and put back reversed at another place. More formal and precise definitions are given in Chapter II. Three examples describing three operations are as the following figure,

A reversal : Genome X : 3 1 5 2 4 \rightarrow Genome Y : 3 -2 -5 -1 4
 A transposition : Genome X : 3 1 5 2 4 \rightarrow Genome Y : 3 2 1 5 4
 A transversal : Genome X : 3 -1-5 2 4 \rightarrow Genome Y : 3 2 5 1 4

Figure 1.1: The example of three rearrangement events (reversal, transposition, and transversal).

All these studies based on the assumption that *the compared genomes have the same genes, each one appearing exactly once in each genome*. However, this hypothesis may be appropriate for small genomes, *e.g.* viruses and organelles, it is clearly unsuitable for divergent species containing several copies of highly paralogous ¹ and orthologous genes ², scattered across the genome. In this case, it is important to introduce the possibility of having different copies of the same gene, *e.g. gene duplication* (also called *multi-gene families*). These copies may be identical, or found to have a high similarity with BLAST-like search. They may be adjacent on the single chromosome, or dispersed throughout the genome. As an example, Table 1.1 lists the estimate numbers of duplicated genes in completely or nearly completely sequence genomes of representative bacteria, archaeobacteria and eukaryotes. One finds that, in all three domains of life, large proportions of genes were generated by gene duplication [10].

Since we are transforming a sequence of numbers into another sequence, without losing generality, we may always assume that the target sequence is $1, 2, \dots, n$. The similar-

¹Paralogs are genes that were duplicated from a single gene on the same genome.

²Orthologs are genes in different species that evolved from the same gene in the last common ancestor of the species.

Table 1.1: Prevalence of *gene duplication* in all three domains of life.

	Total number of genes	Number of duplicate genes (% of duplicate genes)
Bacteria		
<i>Mycoplasma pneumoniae</i>	677	298 (44)
<i>Helicobacter pylori</i>	1590	266 (17)
<i>Haemophilus influenzae</i>	1709	284 (17)
Archaea		
<i>Archaeoglobus fulgidus</i>	2436	719 (30)
Eukarya		
<i>Saccharomyces cerevisiae</i>	6241	1858 (30)
<i>Caenorhabditis elegans</i>	18424	8971 (49)
<i>Drosophila melanogaster</i>	13601	5536 (41)
<i>Arabidopsis thaliana</i>	25498	16574 (65)
<i>Homo sapiens</i>	40580	15343 (38)

ity between two sequences will be measured by the minimum number of rearrangement operations to transform a sequence into another. Because the target sequence is always $1, 2, \dots, n$, we may view the problem as a sorting problem. But this is not a usual sorting problem which we are familiar with. This sorting problem is to sort a sequence in such a way that the number of operation is minimized. In other words, we are interested in finding algorithm which always sort a sequence with minimum number of operations including with gene duplication.

Therefore, the significant contribution of genome rearrangement problem discussed in this dissertation is *What is the shortest distance between two genomes by mutation events e.g. reversal, transposition, and transversal with multi-gene families?*. This problem is the NP-hard problem [11]. The mutations that we consider in this thesis are primarily **reversals**, but also **transpositions** and **transversal**. The background on genome rearrangement (non-duplicated genes) can be found in [12, 13, 14]. The new heuristic algorithm for computing the nearest edit distances with multi-gene families is proposed in Chapter III. Chapter IV shows the experimental results and the conclusion is explained in the Chapter V.

1.2 Research Objectives

1. To propose a new rearrangement model used to transform one genome into another for each mutation event e.g. *reversal, transposition and transversal*.
2. To find the nearest edit distance (*reversal, transposition, and transversal*) between two genomes with multi-gene families.

1.3 Scopes of the Study

1. The genomes are uni-chromosome (single-chromosome) including gene duplication.
2. The inputs can be positive, negative integers, characters, or symbols.
3. The genome rearrangement events for uni-chromosome, separately tested, are reversal, transposition and transversal.
4. The input genomes do not set the probabilities for each three events that might be occur in the real life.

1.4 Research Plans

1. Study the various algorithms in the genome rearrangement.
2. Study the original theorem that estimated the reversal, transposition and transversal distance.
3. Apply the original theorem to design the new model and to cope with the multigene family problem.
4. Estimate the edit distances from this new model.
5. Conclude the experimental results by comparing the results with those from other methods.

1.5 Research Advantages

It is expected that the new approach and prototype are

1. applicable for genomes that have multi-gene families.
2. used for estimating the evolutionary distance between two any uni-chromosome genome by three mutation events e.g. *reversal, transposition and transversal*.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER II

THEORIES AND LITERATURE REVIEWS

In this chapter, the basic concepts of biology, the formal representation of the genome (synteny, order and polarity, linearity and circularity, and multi-gene families), operations and distances (alignment traces, breakpoint distance and edit distances), and some literature reviews for gene duplication are briefly revised.

2.1 Introductory Biology

This dissertation deals primarily with mathematical aspects of comparative genomics, but it is also important to know how realistic our models are drawn from. We will, therefore, give a short account of the most basic evolutionary biology. Anyone who already knows of genes and genomes should move on. The discovery of the genetic material in living organisms started some 130 years ago and an interesting review of the recent and past development in this area has been given by Aldridge [15]. Less than fifty years ago, Watson and Crick discovered how nature extracts information from the genetic material. The genome consists of one or more (46 chromosomes for humans) chromosomes, each one consisting of two sequences of nucleotides, paired together to form a double helix. This is what we call DNA, which is short for Deoxyribonucleic acid. There are four nucleotides: adenine (**A**), cytosine (**C**), guanine (**G**) and thymine (**T**). These are always paired, A with T and C with G; this means that both sequences contain the same information. An example of DNA [16] is shown in the Fig. 2.1

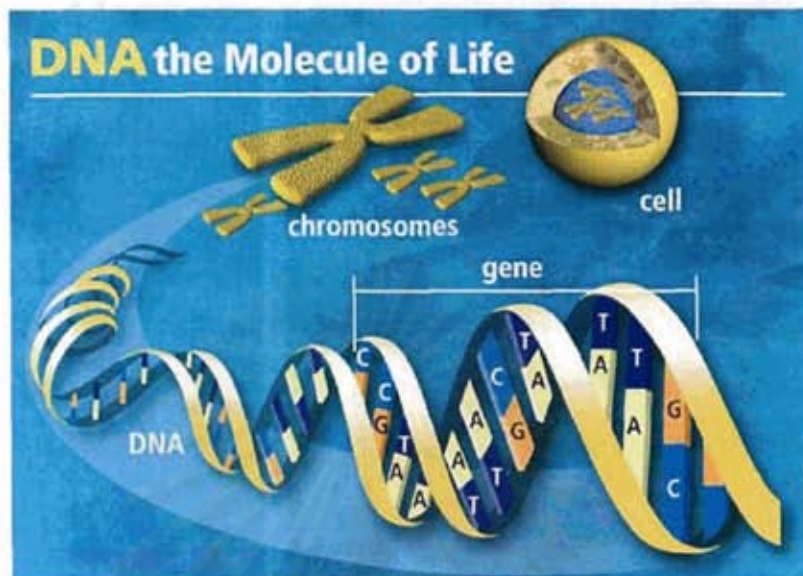


Figure 2.1: The example of DNA : the molecule of life.

The function of the information in these sequences is, as far as we know, mostly as blueprint for proteins. Each triple of nucleotides codes for some amino acid, which proteins are built of. A segment that codes for a protein is called a *gene*. Each gene should be read either from the left or from the right, depending on which of the two nucleotide sequences the gene is positioned at. A *gene* is defined to be a sequence of DNA or bases that code for a specific function/protein. However, a gene can have more than one form or version. So, while there may be a gene for, say, producing hair of a certain colour, that gene will have different *alleles*, such as producing brown hair or blonde hair. A gene is like a variable that can take different values, to use a computational metaphore. It is not known for sure how many genes are capable of having different allelic values or how many different allelic values exist for those genes that can vary. A *genome* is defined to be the complete set of chromosomes inherited from one parent.

By identifying genes that are very similar and code for similar proteins, one finds

that man and mice, for instance, share about 99% of their genes. Thus, on the gene level, the major difference between man and mice is the way these genes are arranged and not the kind of genes that they contain.

The problem of genome rearrangement is a central problem in computational biology. When trying to determine evolutionary distance between two organisms using the genomic data, one wishes to reconstruct the sequence of evolutionary events, which have occurred, transforming one genome into the other. One of most promising ways to trace the evolutionary events is to compare the order of appearance of identical (or orthologous) genes in two different genomes. The study of genome rearrangement began in late 1980's.

2.2 The Formal Representation of The Genome

In contrast to prokaryotes that tend to have single chromosome, often, circular chromosome, the gene in plants, animals, yeasts, and other eukaryotes are partitioned among several chromosomes.

The genome rearrangement approach to comparative genomics focuses on the general structure of a chromosome, rather than on the internal nucleic structure of each gene. This approach assumes that the problems of determining the identity of each gene, and its homologs among a set of genomes, have been solved, so that a gene is simply labeled by a symbol indicating the class of homologs to which it belongs.

Three levels of chromosomal structures have been studied in the literature. The syntenic structure just indicates the content of genes among the set of chromosomes of a genome. Two genes located on the same chromosome are said to be *syntenic* (as shown in the Fig. 2.2). The genome rearrangement approach based on the syntenic structure infers divergence history in terms of inter-chromosome movements such as fusion and

fission. Intra-chromosome movements can be detected only if the order of genes in chromosomes is known. In that case, a chromosome is represented as a linear sequence of genes. In the most realistic version of the rearrangement problem, a sign (+ or -) is associated with each gene representing its transcriptional orientation. This orientation indicates on which of the two complementary DNA strands the gene is located. The distance problems in which this level of structure is known and taken into account are called *signed*, in contrast to this situation where no directional information are used, the *unsigned* case.

Chro1: {a b c d }	a b c d	+a +b -c -d
Chro2: {a b c }	b a b c d e	-b +a -b +c +d
Chro3: {a c d b a }	c a b d c a	+c -a +b +d -c
Synteny sets	Ordered, unsigned case	Ordered, signed case

Figure 2.2: The different levels of chromosome structures considered in the genome rearrangement literature.

2.2.1 Synteney

The genes in plants, animals, yeasts and other eukaryotes are partitioned among a number of chromosomes, generally between 10 and 100 in number, though it can be as low as 2 or 3 [17], or much higher than 100. Two genes located on the same chromosome in a genome are said to be *syntenic* in that genome.

Some genome rearrangements involve parts of one chromosome being relocated to another chromosome. Syntenic structure is generally different between different species and usually identical among all the members of a single species. A few species tolerate population “heterogeneity” involving small differences in syntenic structure, where

heterokaryotypic individuals are not only viable, but fertile [18].

In prokaryotic genomes, comprising both eubacteria and archaeobacteria, the genome typically resides on a single-chromosome. Organelles, such as the mitochondria found in most eukaryotes and the chloroplasts in plants and algae, also have relatively small single-chromosome genomes, containing less than 100 (mitochondria) or 250 (chloroplasts) genes, and are believed to be the highly reduced descendants of prokaryotic endosymbionts.

2.2.2 Order and Polarity

Syntenic structure suffices to initiate the study of genome rearrangements. Two additional levels of chromosomal structure, when they are available, add valuable information about rearrangement. The first is *gene order*. The genes on each chromosome have a linear order that is characteristic of each genome. Note that although our discussion in this dissertation is phrased in terms of the order of genes along a chromosome, the key aspect for mathematical purposes is the order and not the fact that the entities in the order are genes. They could as well be blocks of genes contiguous in the two (or N) species being compared, conserved chromosomal segments in comparative genetic maps or, indeed, the results of any decomposition of the chromosome into disjoint ordered fragments, each identifiable in the two (or in all N) genomes.

The next level of structure is the transcription direction associated with each gene. In the double-stranded DNA of a genome, typically some genes are found on one strand and are read in the direction associated with that strand, while other genes are on the complementary strand which is read in the opposite direction. To capture this distinction in the mathematical notation for a genome, the genes on one strand are designated as of “*positive*” polarity and those on the other as “*negative*.” The latter are written with a

minus sign preceding the gene label, and genomes and genome distance problems where this level of structure is known and taken into account are called “*signed*” in contrast to the situation where no directional information is used, the “*unsigned*” case.

2.2.3 Linearity *versus* Circularity

In eukaryotes such as yeast, amoeba, or humans, the genes on a chromosome are ordered linearly. There is no natural left-to-right order, i.e., there is no structural asymmetry or polarity between one end of a chromosome and the other. In prokaryotes and in organelles, the single chromosome is generally circular. This leads to terminological and notational adjustments, the arbitrariness of left-to-right order becomes the arbitrariness of clockwise versus counterclockwise ordering, and the notion of one gene appearing in the order somewhere before another is no longer meaningful. Most computational problems in genome comparison are no more difficult for circular genomes than linear ones.

2.2.4 Multi-Gene Families

Implicit in the rearrangement literature is that both genomes being compared contain an identical set of genes and the one-to-one homologies (orthologies) between all pairs of corresponding genes in the two genomes have previously been established. While this hypothesis of *unique* genes may be appropriate for some small genomes, e.g. *viruses and mitochondria*, it is clearly unwarranted for divergent species where several copies of the same gene, or several homologous (paralogous) genes may be scattered across a genome.

2.3 Operations and Distances

There are many ways of comparing two linear (or circular) orders on a set of objects. In Subsection 2.3.1, we first discuss one which is not based on any biologically-motivated model. In Subsection 2.3.2 and 2.3.3, we introduce a distance which is motivated by general characteristics of genome rearrangements. In the remainder of this section, we review the many edit distances which are based on particular types of rearrangement.

2.3.1 Alignment Traces

One of the earliest suggestions for comparing genomes was to adapt concepts of alignment in sequence comparison, in particular the notion of the trace of an alignment. In its graphic version, this requires displaying the n genes in each of the two genomes, ordered from left to right, one genome above the other, and connecting each of the n pairs of homologous genes with a line. The number of intersections between pairs of lines is a measure of how much one genome is scrambled with respect to the other [19]. For linear orders, this measure is easily calculated and analytical tests are available for detecting non-random similarities in order; the circular case is much more difficult. The problem has to do with the optimal alignment of the two genomes, where one circular genome is superimposed on the other and rotated in such a way as to minimize the number of intersections between trace lines connecting genes in the two genomes [20].

2.3.2 Breakpoints

Since genome rearrangements generally involve incorrectly repaired breaks between adjacent genes, it seems appropriate to focus on adjacencies when comparing rearranged genomes. For two genomes G and H , we define $b(G, H)$ to be the number of pairs of genes that are adjacent in genome G but not in H . The easily calculated measure b

is and was first defined in the context of genome rearrangements by Watterson et al. [21], but was already implicit much earlier in cytogenetic assessments of chromosomal evolution. For signed genomes, the notion of adjacency requires that the configuration of transcription directions be conserved, so that if genome G contains two genes ordered as $x y$, then these two genes are adjacent in H only if they occur as $x y$ or as $-y -x$.

Example : $G = (-2, -3, +1, +6, -5, -4)$ and $H = (+1, +2, +3, +4, +5, +6)$. The breakpoints of G with respect to H are : $(-2, -3)$, $(-3, +1)$, $(+1, +6)$, $(+6, -5)$, $(-4, +7)$. Note that $(-5, -4)$ is not a breakpoint since $(4, 5)$ appear in H .

Why breakpoint are important? Intuitively, if (x, y) is a breakpoint, then in order to transform G to H , some reversal must separate between x and y . Therefore, the number of breakpoints is an indication to how many reversals are required.

The Breakpoint Distance : the first lower bound

The simplest distance widely used is the breakpoint distance $b(G, H)$. In essence we count the number of adjacent genes in one of the genomes that are not adjacent in the other. It is not hard to see that $b(G, H)$ is a metric on the space of genomes. We use the notation $b(G)$ for the number of breakpoints between G and the identity. The breakpoint distance is widely used by the community. It is easy to compute and we do not have to make any specific assumptions about the underlying model. In fact, it is a decent approximation for many of the other distances we shall look at, although more refined analyses demand more sophisticated distances.

It has been argued by Sankoff and others, see for instance [22], that the success and applicability of the breakpoint metric comes from its being model independent. We would like to offer some words of caution regarding this view. First, it is of course not sufficient for a distance to be model independent. If we let the distance between G and H be zero if $G = H$ and one otherwise we definitely get a model independent distance,

but it will not be of any use to us. Second, the breakpoint distance is by no means model independent. It works about equally well for the common sets of operations, i.e. reversals and transpositions, since these operations change the number of breakpoints by at most two or three, respectively, but if our model would include operations that change the number of breakpoints by far more, the breakpoint distance would not give such good results.

2.4 Edit Distances

One of the most fundamental computational problems in comparative genomics, which must be solved before many higher level problems can be attacked, is to compute the distance between two genomes. The idea is to come up with a measure, based on gene order and gene content, that reflects as closely as possible the evolutionary distance of the given organisms. The challenge is to find a measurement that is biologically meaningful yet efficiently computable.

To be realistic, a measurement should reflect several known mechanisms of genomic rearrangement. In the case of *single-chromosome* genomes (such as those of prokaryotes, chloroplasts, and mitochondria), these mechanisms include the following:

- **Reversal:** A section of a chromosome is excised, reversed in orientation, and re-inserted –Section 2.6.
- **Transposition:** A section of a chromosome is excised and inserted at new position in the chromosome, without changing orientation –Section 2.7.
- **Transversal (Inverted Transposition):** Exactly like transposition, except that the transposed segment changes orientation –Section 2.8.

- **gene duplication:** A section of a chromosome is duplicated, so that multiple copies exist of every gene in that section.

2.5 Breakpoint Graph : Notations and Definitions

Let π and ϕ be signed permutations of size n , such that $\pi = (\pm\pi_1, \pm\pi_2, \dots, \pm\pi_n)$ and $\phi = (\pm\phi_1, \pm\phi_2, \dots, \pm\phi_n)$. Let unsigned permutation $\pi' = (\pi'_0, \pi'_1, \dots, \pi'_{2n}, \pi'_{2n+1})$ be defined such that $\pi'_0 = 0$, $\pi'_{2n+1} = 2n + 1$, and for all i , $1 \leq i \leq n$, $\pi'_{2i} = 2\pi_i$, $\pi'_{2i-1} = 2\pi_i - 1$ (if $\pi_i > 0$) or $\pi'_{2i} = 2|\pi_i| - 1$, $\pi'_{2i-1} = 2|\pi_i|$ (if $\pi_i < 0$). Let the unsigned permutation $\phi' = (\phi'_0, \phi'_1, \dots, \phi'_{2n}, \phi'_{2n+1})$ be defined exactly the same way with respect to ϕ . We say two elements π_i and π_{i+1} are adjacent in π , and we say the corresponding elements π'_{2i} and π'_{2i+1} are adjacent in π' ; similar for ϕ and ϕ' . Bafna and Pevzner [23] introduce the notion of the *breakpoint graph* of a permutation.

The *breakpoint graph* B of π with respect to ϕ be defined as follows:

- B contains a sequence of $2n + 2$ vertices labeled with the element of π' .
- Every two of these vertices that reflect an adjacency in π' are connected with black edge, and every two that reflect an adjacency in ϕ' are connected with gray edge.

Let the *overlap graph* $O = (V, E)$ for B be defined such that there exists a distinct $v_e \in V$ for every gray edge e in B , and two vertices v_e and $v_{e'}$ are connected by an edge $(\{v_e, v_{e'}\} \in E)$ iff gray edges e and e' overlap in B . The example for breakpoint graph and overlap graph are shown in the Fig. 2.3.

A *cycle* in B is a sequence of connected vertices $(v_0, v_1, \dots, v_{2i}, v_{2i+1}, \dots, v_{2n}, v_{2n+1}, v_0)$ where $n \geq 0$ and for all $i, 0 \leq i \leq n$, v_{2i} and v_{2i+1} are connected with black edge, and v_{2i+1} and v_{2i+2} (or v_{2i+1} and v_0 , if $i = n$) are connected with a gray edge. A connected component in O has the usual meaning, and sometimes call simply a *component*.

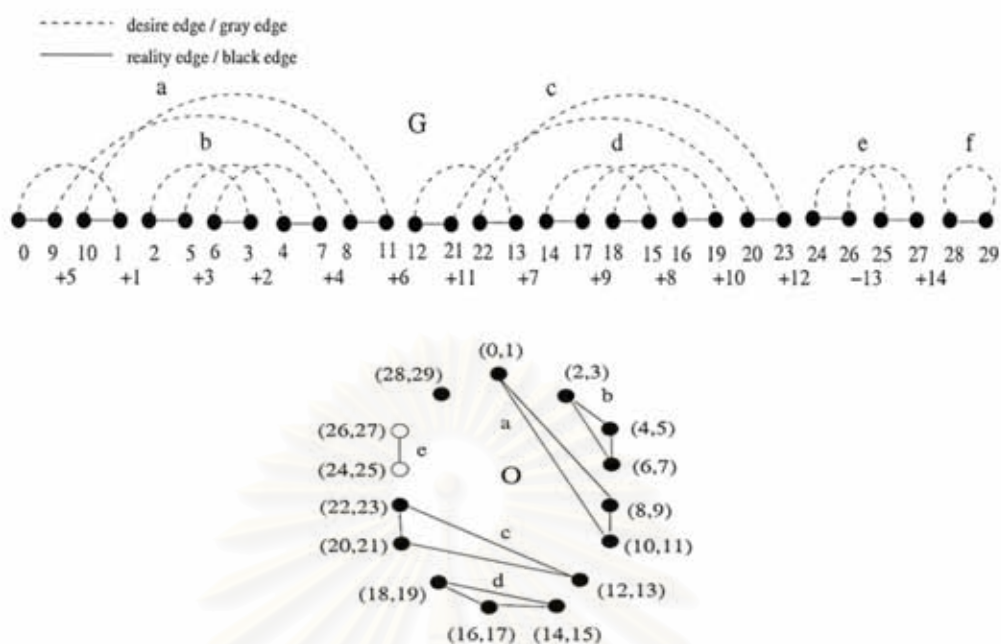


Figure 2.3: Breakpoint graph B and overlap graph O for the permutation $\pi = (+5, +1, +3, +2, +4, +6, +11, +7, +9, +8, +10, +12, -13, +14)$ with respect to the identity permutation of size $n = 14$. Connected component e is *oriented*, a, b, c and d are *unoriented*, and f is *trivial*. Unoriented components b and d are hurdles and also superhurdles, but unoriented component a and c are nonhurdles because they separate b and d .

Every gray edge is said to be *oriented* if it spans an odd number of vertices in B , and *unoriented* otherwise. In the other words, a gray edge is *oriented* if it links two left vertices of two black edges, or two right vertices of two black edges, otherwise it is called *unoriented*. A cycle in B and a connected component in O are each said to be *oriented* if they contain at least one *oriented* gray edge. We call cycle and component *unoriented* if they are not *oriented*, except when they are *trivial*. A trivial cycle consists of a single gray edge and a single black edge, and correspond to an adjacency shared in permutations π and ϕ . A trivial cycle will always create a trivial connected component

—that is, a component consisting of a single, isolated in O — and a trivial component can only arise from a trivial cycle. Note that the gray edges of cycles always belong to the same connected component, so we can say that the cycle belongs to the component.

Every unoriented component can be classified as either a *hurdle* or a *nonhurdle*. A hurdle is an unoriented component that does not separate other unoriented component, and a nonhurdle is one that does. A component b is said to separate two other components c and d if, in a traversal of the vertices of B , it is impossible to pass from a vertex belonging to c to a vertex belonging to d without encountering a vertex belonging to u . A hurdle is called a *superhurdle* if, were it eliminated, a nonhurdle would emerge as a hurdle; otherwise it is called a *simple hurdle*.

2.6 Minimal Reversal Distance

The reversal has generally been considered the most important of the three operations that we usually consider. Some claim that this is because reversals have been more frequently observed [7]. This may be the case, but it seems reasonable to think that their popularity is in part boosted by our ability to treat them mathematically. While transpositions and transversal have been hard to analyse, the following problem has actually been solved. The question is *What is the minimal number of reversals d_{rev} needed to transform a genome π into the identity genome?*

A reversal $\rho_k(i, j)$, for any k and $1 \leq i, j \leq n$, of $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ transforms π into $\rho_k(i, j) = \pi(\pi_1, \dots, -\pi_j, -\pi_{j-1}, \dots, -\pi_i, \dots, \pi_n)$. A reversal distance is the number of reversals $\rho_1(i_1, j_1) \rho_2(i_2, j_2), \dots, \rho_t(i_t, j_t)$ of minimal length t required to transform π to ϕ . The full solution (for signed genomes) was given in 1995 by Hannenhalli and Pevzner [24], preceded by a fairly good approximation in 1996 by Bafna and Pevzner [23].

For the following example, let $\pi = (-2, -1, +4, +3, +5)$ and $\phi = (+1, +2, +3, +4, +5)$ be the permutations. The results are shown the procedure of an optimal sorting of a permutation by reversals and the signed reversal distance $d(\pi, \phi) = 4$.¹

$$\begin{aligned}\pi &= (-2, -1, +4, +3, +5) \\ \rho_1(1, 2) &= (\underline{-2, -1}, +4, +3, +5) \\ \rho_2(3, 3) &= (+1, +2, \underline{+4}, +3, +5) \\ \rho_3(4, 4) &= (+1, +2, -4, \underline{+3}, +5) \\ \rho_4(3, 4) &= (+1, +2, \underline{-4, -3}, +5) \\ \phi &= (+1, +2, +3, +4, +5)\end{aligned}$$

Next, the following example show the steps, for each ρ_t , $1 \leq t \leq 4$, that solved the problem of sorting by reversals by using concept of the breakpoint graph.

A reversal can remove at most two breakpoints in a permutation. Therefore, a simple (*the first*) lower bound on the reversal distance is

$$d_{rev} \geq \frac{b(\pi)}{2} \quad (2.1)$$

where $b(\pi)$ is the number of breakpoint in the permutation π .

The Hannenhalli and Pevzner Theory

Hannenhalli and Pevzner [24] investigate the *breakpoint graph*, as shown in Fig. 2.3. They give an optimal formula for computing the reversal distance of a permutation (based on parameters of the breakpoint graph) and provide a polynomial algorithm for sorting by signed reversals.

¹Note that the problems of *reversal distance* and of *sorting by reversals* are subtly different; it turns out one can compute reversal distance without actually finding a sequence of sorting reversals.

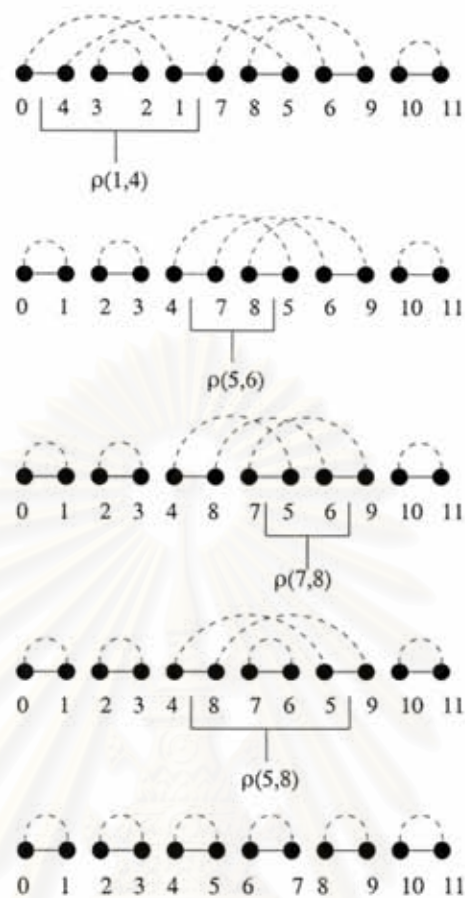


Figure 2.4: The example of sorting by reversals of the permutation $\pi' = (0, 4, 3, 2, 1, 7, 8, 5, 6, 9, 10, 11)$

Let π and ϕ be two genomes defined on the same set of genes, where each gene appears exactly once in each genome. The problem is to find the minimal number of reversal operations necessary to transform π to ϕ . We use d_{rev} to denote the *minimal number of reversal distance* between π and ϕ . The minimal reversal distance necessary to transform π to ϕ of permutation size n is given by this formula :

$$d_{rev} = b(\pi, \phi) - c(\pi, \phi) + h(\pi, \phi) + f(\pi, \phi) \quad (2.2)$$

where $b(\pi, \phi)$ is the number of black edges,

$c(\pi, \phi)$ is the number of cycles in G ,

$h(\pi, \phi)$ its number of hurdles,

$f(\pi, \phi)$ is equal to 1 if fortresses exist in G and zero otherwise.

From Fig. 2.4, the permutation π has 6 black edges, 3 cycles, 1 hurdles, and non-fortress. By using the Equation 2.2, the reversal distance of the permutation is $6 - 3 + 1 + 0 = 4$.

It follows from Caprara [4] that genomes containing hurdles (genomes π with $h(\pi, \phi) + f(\pi, \phi) > 0$) are very rare. *The probability for a given component to be good (oriented component) is greater than its probability to be bad (unoriented component), the number of hurdles is near to 0.* For instance, for genomes of length 8, less than one percent of these contain hurdles, and for genomes of length 100, only one in 105 contains a hurdle. Therefore, the number of cycles is the dominant parameter in the Hannenhalli and Pevzner formula for $d(\pi, \phi)$, if $b(\pi, \phi)$ is considered as a constant. Notice that more cycles mean less reversals.

Many people have looked at the computational aspects of calculating the reversal distance. In the paper by Hannenhalli and Pevzner, both finding this distance and computing a minimal sequence of reversals was done in $O(n^4)$ time, for a genome of length n . This has subsequently been improved by, among others, Berman and Hannenhalli [25], who computed the distance in $O(n\alpha(n))$ time and a minimal sequence in $O(n^2\alpha(n))$ time, where the function $\alpha(n)$ is the inverse of Ackermans function [26]. Next, Kaplan, Shamir and Tarjan [27] gave a minimal sequence in $O(n\alpha(n) + d_{rev}(\pi)n)$ time, where

$d_{rev}(\pi) < n$ is the reversal distance, and Bader, Moret and Yan [8] reduced finding the distance to linear time. Recently, Bergeron [3] and Bergeron, Heber and Stoye [5] have simplified the algorithms, yielding an algorithm for computing the reversal distance without using the breakpoint graph. Finally, Siepel [28] has given an algorithm for finding all optimal sequences of reversals that sort a genome.

For unsigned genomes, calculating the reversal distance is NP-hard, as was shown by Caprara [2]. The proof consists of a series of transformations. For unsigned genomes, the vertices $2k - 1$ and $2k$ in the breakpoint graph are identified for all $k \in n$. It then becomes a problem to compute the maximum number of alternating cycles. Caprara is able to reduce this problem to the reversal distance problem. He then proceeds to show that the alternating cycle problem is NP-hard, which implies that the reversal distance is NP-hard. There exists polynomial time approximations of the reversal distance for unsigned genomes: Christie has given a $3/2$ -approximation [6] and Berman, Hannenhalli and Karpinski [29] have reduced this to $11/8$.

2.7 Minimal Transposition Distance

The question is *What is the minimal number of transpositions d_{trp} needed to transform a genome π into the identity genome?*

A *transposition* $\omega_l(i, j, k)$, for any l and $1 \leq i < j < k \leq n + 1$, of $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ cuts the segment $[i, j - 1]$ and pasting it before the k^{th} position as follows.

$$\omega(i, j, k) = \pi (\pi_1, \dots, \pi_{i-1}, \pi_j, \dots, \pi_{k-1}, \underline{\pi_i, \dots, \pi_{j-1}}, \pi_k, \dots, \pi_n)$$

A *transposition distance* is the number of transposition operations $\omega_1(i_1, j_1, k_1)$, $\omega_2(i_2, j_2, k_2)$, \dots , $\omega_l(i_l, j_l, k_l)$ of minimal length l require to transform π to ϕ . Note that this is a problem on unsigned genomes, since a transposition does not change the sign of any gene.

For the following example, let $\pi = (2, 1, 4, 3, 5)$ and $\phi = (1, 2, 3, 4, 5)$ be the permutations. The results shows the procedure of an optimal sorting of a permutation by transpositions and the transposition distance $d_{trp}(\pi, \phi) = 2$. An example is shown in Fig. 2.5.

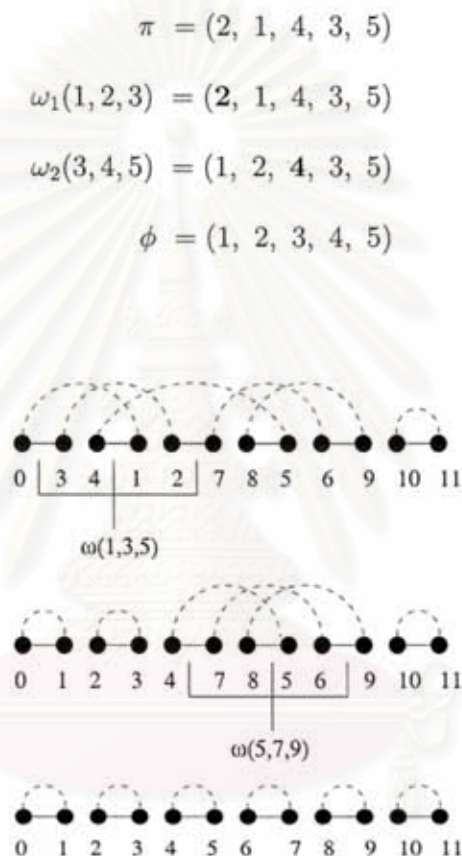


Figure 2.5: The example of sorting by transpositions of the permutation $\pi' = (0, 3, 4, 1, 2, 7, 8, 5, 6, 9, 10, 11)$

Nobody has found a closed formula for this distance. It is easy to give some trivial bounds. For instance, a transposition cannot change the number of breakpoints by more than three. Furthermore, one can always find a transposition reducing the number of

breakpoints by one. Thus, the first lower and upper bound are

$$\frac{b(\pi)}{3} \leq d_{trp}(\pi) \leq b(\pi) \quad (2.3)$$

for all unsigned genomes π .

Research by among others Bafna and Pevzner [30] and Christie [7] introduced the notion of a breakpoint graph of a permutation, and used the breakpoint graph to obtain an improved lower bound as shown in the Eq. 2.4.

$$\frac{n+1-c(\pi)}{2} \leq d_{trp}(\pi) \leq n+1-c(\pi) \quad (2.4)$$

holds for all unsigned genomes length n and $c(\pi)$ is the number of cycles in the breakpoint graph.

2.8 Minimal Transversal Distance

The question is *What is the minimal number of transversals d_{trv} needed to transform a genome π into the identity genome?*

A *transversal* $\alpha_l(i, j, k)$, for any l and $1 \leq i < j < k \leq n+1$, of $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ cuts the segment $[i, j-1]$, pasting and inverting it before the k^{th} position as follows.

$$\alpha(i, j, k) = \pi (\pi_1, \dots, \pi_{i-1}, \pi_j, \dots, \pi_{k-1}, \underline{-\pi_{j-1}, \dots, -\pi_i}, \pi_k, \dots, \pi_n)$$

A *transversal distance* is the number of transversal operations $\alpha_1(i_1, j_1, k_1), \alpha_2(i_2, j_2, k_2), \dots, \alpha_l(i_l, j_l, k_l)$ of minimal length l require to transform π to ϕ . There does not seem to be any similar work done on the transversal distance. However, it seems likely that inequalities similar to those in the transposition case also hold for *transversal*.

Research by among others Q.Gu and S.Peng [31] and Christie [7] introduced the notion of a breakpoint graph of a permutation, and used the breakpoint graph to obtain lower bound for transversal distance. The equation is shown as follows.

$$\frac{n+1-c(\pi)}{2} \leq d_{trv}(\pi) \leq n+1-c(\pi) \quad (2.5)$$

For the following example, let $\pi = (-4, -3, 1, 2, 5)$ and $\phi = (1, 2, 3, 4, 5)$ be the permutations. The results show the procedure of an optimal sorting of a permutation by transversals and the transversal distance (computed from Eq.2.5) to be $d_{trv}(\pi, \phi) = \frac{5+1-4}{2} = 1$. An example is shown in Fig. 2.6.

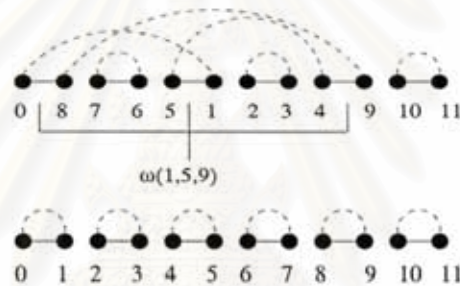


Figure 2.6: The example of sorting by transversals of the permutation $\pi' = (0, 8, 7, 6, 5, 1, 2, 3, 4, 9, 10, 11)$

2.9 Literature Reviews on Gene Duplication

There are many efficient methods to find the shortest distance. However, all of them are applicable to genomes that do not contain gene duplications. In this section, we introduce some approaches that have been developed to account for gene duplicates in the genome rearrangement.

2.9.1 Exemplar Distance

Sankoff [32] has formulated a generalized version of the genome rearrangement problem where each genome may be present in many copies. The idea is to delete, from each gene family, all copies except each of the compared genomes G and H . This preserved copy, called the *exemplar*, represent in the common ancestor of all copies in G and H . The criteria for deleting gene copies is to form two permutations having the minimal distance. Sankoff considers two distance measures: the breakpoint distance and the reversal distance.

The underlying evolutionary model is that the most recent common ancestor F of genomes G and H has single gene copy (Figure 2.7). After divergence, the gene a in F can be duplicated many times in two lineages leading to G and H , and appear anywhere in the genomes. Each genome is then subject to rearrangement events. The key idea is that, after rearrangements, the *true exemplar*, that is the direct descendent of a in G and H , will have been displaced less frequently than the other gene copy. The true exemplar string can thus be identified as those that have been less rearranged with respect to each other than any other pair of reduced genomes.

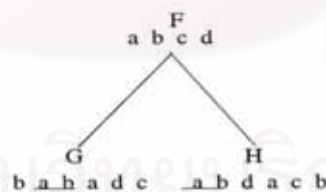


Figure 2.7: The evolutionary model considered in the exemplar analysis. Using the breakpoint distance as a criterion, the chosen exemplar are the undelined ones.

Even though finding the exemplar has been shown NP-hard [33], [32] developed a branch and bound algorithm that has been shown practical enough for simulated

data. The strategy is to begin with empty strings, and to insert successively one pair of homologous genes from each gene families, one after the other. At each step, the chosen pair of exemplars are the one which the least distance increases when inserted into the partial exemplar string already constructed. The gene families are proceeded in increasing order of their sizes: singletons first, then families of size three, four, and so on.

Sankoff considers a branch and bound strategy. At each step, (for each next gene family), all pairs in the family are tested to see how much they increase the distance when the two members are inserted into the partial exemplar strings. The chosen exemplar pair is the one which the least increases distance. A backtracking step from the family currently being considered occurs whenever all its remaining unused pairs have too large test values, that is test values that would increase the distance beyond the current best value.

A natural application of the exemplar approach is to identify orthologies between two genomes containing families of paralogous genes. Unfortunately, as far as we know, the algorithm has only been tested on simulated data.

In [35, 40], a straightforward approach by enumerating all the possible assignments of orthologs between two genomes has been considered. However, this approach is applicable only to genomes with a very small number of duplicated genes, as the number of possible assignments grows exponentially with the number of paralogs. Very recently, X.Chen [11] introduced a new approach to ortholog assignments that considered both sequence similarity and genome rearrangement (Section 2.9.2).

2.9.2 The Assignment of Orthologous Genes via Genome Rearrangement

The assignment of orthologous genes between a pair of genomes is a fundamental and challenging problem in comparative genomics. Existing methods that assign orthologs based on the similarity between DNA or protein sequences may make erroneous assignments when sequence similarity does not clearly delineate the evolutionary relationship among genes of the same families.

X. Chen and et al. [11] propose a new approach for assigning orthologs by taking into account both local mutations and genome rearrangement events. Their method starts by identifying sets of paralogs (gene families) on each genome and the family correspondences between two genomes by using the homology search, i.e., BLAST. The paralogs are then treated as copies of the same genes, and ortholog assignment is formulated as a natural optimization problem of rearranging one genome consisting of a sequence of (possibly duplicated) genes to the other with the smallest number of rearrangement events. This most parsimonious rearrangement process should suggest pairs of orthologous genes in a straightforward way. To simplify the discussion (and as a first attempt), they first consider only inversion events in genome rearrangement. The above optimization problem thus becomes a problem of computing the signed reversal distance with duplicates (SRDD) between two genomes. SRDD is a simple extension of the well-known problem of sorting by reversals [24]. Although the problem of sorting by reversals has been intensively studied in the past decade, SRDD has basically been untouched. They give an efficient and effective heuristic algorithm for solving SRDD, using the techniques of *minimum common partition* of two given genomes (the MCP is solved by the technique of vertex cover.) and *maximum cycle decomposition* on a complete graph (the MCD is solved by the technique of greedy algorithm to find the shortest path among

the paths in the complete graph). The heuristic algorithm for SRDD has been tested on both simulated and real genomic sequence data (from human, mouse, and rat X chromosomes), and compared with the existing algorithm the exemplar algorithm [32] (actually, an iterative version of it). The test results demonstrate that the SRDD in general performs better than the iterated exemplar algorithm in terms of computing the reversal distance and assigning correct orthologs.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER III

PROPOSED METHOD

Three edit distances (reversal, transposition, and transversal) can use the notion of breakpoint graph to compute the distances or generate the steps of sorting. However, the breakpoint graph is used for the genomes that do not have gene duplication. The breakpoint graph does not work for multi-gene families problem. In this chapter, we proposed a new heuristic algorithm to find a canonical permutation of gene duplicates. We extend the concept of the *breakpoint graph* to cope with gene duplicates. The key idea of our method is to generate an *incomplete breakpoint graph* which allows for exploring gene-gene relationships across the two genomes by maximizing the number of graph cycles. The optimization technique *Binary Integer Programming* (BIP) is applied to find the set of gray edges (edges that play central role in the breakpoint graph) that minimize the edit distances.

Once our algorithm is completed, the *incompleted breakpoint graph* will be transform to complete breakpoint graph. The Hannenhalli and Pevzner theorem (Eq.2.2) is applied to compute the reversal distance between two relabelled genomes. Then, Eqs. 2.4 and 2.5 are also applied to compute the transposition and transversal distances, respectively.

3.1 Heuristic Algorithm to Computing the Nearest Minimal Edit Distances for Multi-gene Families via Binary Integer Programming (EDMF)

In case of duplicated genes, let $A = \{g_1, g_2, \dots, g_k\}$ be a set of genes (a uni-chromosome permutation) relating to π of length n in the following way. Each g_i is represented by either a symbol, a character, or a number. For each gene g_i ($1 \leq i \leq k$), let $K(g_i)$ be the number of occurrences of g_i having either $+$ or $-$ sign¹ in π and ϕ such that $K(g_i) \geq 1$. A gene is called a *single* if it is the only member of a gene family in that genome, $K(g_i) = 1$. Otherwise the gene is referred to as a member of a *multi-gene family*. Since the breakpoint graph does not recognise multi-gene families, each multi-gene member element, π_i , must be systematically renamed for the graph to be used to determine the minimum reversal distance.

Table 3.1: An example of two copies of an element 2 and three copies of an element 3.

position	1	2	3	4	5	6	7	8	9	10	11	12
π	+4	+1	+2	+2	+5	+6	+7	+3	+3	+3	-8	-9
ϕ	+1	+2	+2	+5	+4	+6	+3	+3	+3	-8	+7	-9

Table 3.2: An example of the renaming results, π_{new} and ϕ_{new} by temporary names x_1, x_2, x_3 and y_1, y_2 .

position	1	2	3	4	5	6	7	8	9	10	11	12
π_{new}	+5	+1	x_1	x_2	+4	+6	+11	y_1	y_2	y_3	+10	+12
ϕ_{new}	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12

¹a sign $+$ or $-$ is associated with each gene representing its transcription orientation, i.e., on which of the two complementary DNA strands the gene is located.

Consider the example shown in Table 3.1 of the permutation $\pi = (+4, +1, +2, +2, +5, +6, +7, +3, +3, +3, -8, -9)$. Elements π_3 and π_4 contain the same gene represented by number 2 and $\pi_8, \pi_9,$ and π_{10} contain the same gene represented by number 3. These two gene families are rearranged and relocated at new positions in ϕ . To uniquely rename each element, first, all elements in π are renamed according to their positions with respect to ϕ . Symbols x_1, x_2 are introduced as *temporary names* for all duplicated elements, $\pi_3 = \pi_4 = 2$, and symbols y_1, y_2 and y_3 are introduced as *temporary names* for all duplicated elements, $\pi_8 = \pi_9 = \pi_{10} = 3$. The renaming results for both π_{new} and ϕ_{new} (identity) are shown in Table 3.2.

The elements x_1 and x_2 in π can be assigned two possible labels (2 or 3) and, similarly, the elements $y_1, y_2,$ and y_3 in π can be assigned three possible labels (7, 8 or 9). Hence, for this example, there are $2! \times 3!$ possible name assignments as the following $\pi^{(1)} - \pi^{(12)}$.

$$\begin{aligned}
 \pi^{(1)} &= +5 \ +1 \ +2 \ +3 \ +4 \ +6 \ +11 \ +7 \ +8 \ +9 \ +10 \ +12 \\
 \pi^{(2)} &= +5 \ +1 \ +3 \ +2 \ +4 \ +6 \ +11 \ +7 \ +8 \ +9 \ +10 \ +12 \\
 \pi^{(3)} &= +5 \ +1 \ +2 \ +3 \ +4 \ +6 \ +11 \ +7 \ +9 \ +8 \ +10 \ +12 \\
 \pi^{(4)} &= +5 \ +1 \ +3 \ +2 \ +4 \ +6 \ +11 \ +7 \ +9 \ +8 \ +10 \ +12 \\
 \pi^{(5)} &= +5 \ +1 \ +2 \ +3 \ +4 \ +6 \ +11 \ +8 \ +7 \ +9 \ +10 \ +12 \\
 \pi^{(6)} &= +5 \ +1 \ +3 \ +2 \ +4 \ +6 \ +11 \ +8 \ +7 \ +9 \ +10 \ +12 \\
 \pi^{(7)} &= +5 \ +1 \ +2 \ +3 \ +4 \ +6 \ +11 \ +8 \ +9 \ +7 \ +10 \ +12 \\
 \pi^{(8)} &= +5 \ +1 \ +3 \ +2 \ +4 \ +6 \ +11 \ +8 \ +9 \ +7 \ +10 \ +12 \\
 \pi^{(9)} &= +5 \ +1 \ +2 \ +3 \ +4 \ +6 \ +11 \ +9 \ +7 \ +8 \ +10 \ +12 \\
 \pi^{(10)} &= +5 \ +1 \ +3 \ +2 \ +4 \ +6 \ +11 \ +9 \ +7 \ +8 \ +10 \ +12 \\
 \pi^{(11)} &= +5 \ +1 \ +2 \ +3 \ +4 \ +6 \ +11 \ +9 \ +8 \ +7 \ +10 \ +12 \\
 \pi^{(12)} &= +5 \ +1 \ +3 \ +2 \ +4 \ +6 \ +11 \ +9 \ +8 \ +7 \ +10 \ +12
 \end{aligned}$$

Each assignment may result in a different genomic distance. The minimal one can be found by generating and testing all assignments. However, the processing time grows *rapidly* with the number of duplications. Therefore, we turn to an efficient heuristic method for assigning a *final* name for each *temporary* name. The approach can be described in three steps.

1. Creating the *incomplete* breakpoint graph.

The *incomplete* breakpoint graph, IG , is a graph that has the same structure and properties as the breakpoint graph. However, it is not *complete* in the sense that the *final* names of some π_i are known. Consider the example shown in Table 3.2. A *signed* permutation $\pi_{new} = (+5, +1, x_1, x_2, +4, +6, +11, y_1, y_2, y_3, +10, +12)$ becomes an *unsigned* permutation $\pi'_{new} = (0, 9, 10, 1, 2, x_{11}, x_{12}, x_{21}, x_{22}, 7, 8, 11, 12, 21, 22, y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32}, 19, 20, 23, 24, 25)$, where x_1, x_2 are replaced by the *variables*, $(x_{11}, x_{12}), (x_{21}, x_{22})$, and y_1, y_2 and y_3 are replaced by $(y_{11}, y_{12}), (y_{21}, y_{22})$ and (y_{31}, y_{32}) , respectively. The *final* names of x_i , for $1 \leq i \leq 2$ and y_j , for $1 \leq j \leq 3$ are so far unknown. The example of the *incomplete* breakpoint graph of π' is shown in Fig. 3.1.

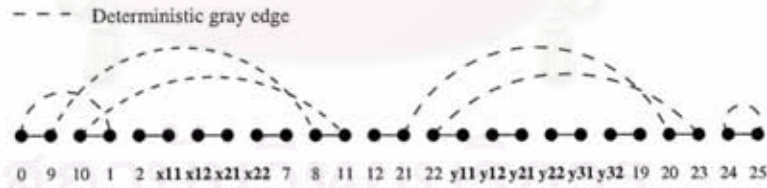


Figure 3.1: An example of an *incomplete* breakpoint graph IG for π' .

With all *variables* in place, all *black* edges can be identified. These black edges are $\{(0, 9), (10, 1), (2, x_{11}), (x_{12}, x_{21}), (x_{22}, 7), (8, 11), (12, 21), (22, y_{11}), (y_{12}, y_{21}), (y_{22}, y_{31}), (y_{32}, 19), (20, 23), (24, 25)\}$. However, only the gray edges not coinciding

with variables can be identified in the first instance. The remaining gray edges are identified during the resolution of the actual names of the variables, specifically $x_{11}, x_{12}, x_{21}, x_{22}, y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32}$. All gray edges are classified into two groups, *deterministic* and *non-deterministic* edges, according to the following characteristics.

- (a) *Deterministic gray edge*: the pair (π'_i, π'_j) in IG , $1 \leq i < j \leq 2n + 2$, is connected by a *deterministic* gray edge if π'_i and π'_j are not *variables*, and $\pi'_i = \pi'_j + 1$ or $\pi'_i = \pi'_j - 1$. For example, the *deterministic* gray edges in Fig. 3.1 are $\{(0, 1), (8, 9), (10, 11), (20, 21), (22, 23), (24, 25)\}$.
- (b) *Non-deterministic gray edge*: an edge connecting a pair (π'_i, π'_j) such that either π'_i or π'_j is a *variable* or both of them are *variables*. For example, edges $(2, x_{11})$ and $(2, x_{21})$ are *non-deterministic* gray edges. Only one gray edge, either $(2, x_{11})$ or $(2, x_{21})$, can be incident with vertex 2. The existence of this gray edge depends upon the correct assignment of vertex 3 to either x_{11} or x_{21} .

The names of x_1, x_2 can be either $+2 : (3, 4)$ or $+3 : (5, 6)$ and the names of y_1, y_2 and y_3 can be either $+7 : (13, 14)$, $+8 : (15, 16)$ or $+9 : (17, 18)$. Hence, the names of x_{i1} must be 3 or 5, for $1 \leq i \leq 2$ and the names of x_{j2} must be 4 or 6, for $1 \leq j \leq 2$. In the same way, the name of y_{m1} , for $1 \leq m \leq 3$, must be 13, 15, or 17 and the name of y_{k2} , for $1 \leq k \leq 3$, must be 14, 16, or 18.

2. Generating all possible solutions for all *non-deterministic* gray edges.

Let E_i be the set of *non-deterministic* gray edges for the corresponding actual vertex pair (a_{i1}, a_{i2}) in π'_{new} . However, the actual name of either a_{i1} or a_{i2} may

Table 3.3: The seven possible edge groups ($E_1 - E_7$) and the 24 *non-deterministic* gray edges ($e_1 - e_{24}$) of the *incomplete* breakpoint graph (IG).

S_1			S_2			
$E_1(2, 3)$	$E_2(4, 5)$	$E_3(6, 7)$	$E_4(12, 13)$	$E_5(14, 15)$	$E_6(16, 17)$	$E_7(18, 19)$
$e_1 : 2-x_{11}$	$e_3 : x_{12}-x_{21}$	$e_5 : x_{12}-7$	$e_7 : 12-y_{11}$	$e_{10} : y_{12}-y_{21}$	$e_{16} : y_{12}-y_{21}$	$e_{22} : y_{12}-19$
$e_2 : 2-x_{21}$	$e_4 : x_{22}-x_{11}$	$e_6 : x_{22}-7$	$e_8 : 12-y_{21}$	$e_{11} : y_{12}-y_{31}$	$e_{17} : y_{12}-y_{31}$	$e_{23} : y_{22}-19$
			$e_9 : 12-y_{31}$	$e_{12} : y_{22}-y_{11}$	$e_{18} : y_{22}-y_{11}$	$e_{24} : y_{32}-19$
				$e_{13} : y_{22}-y_{31}$	$e_{19} : y_{22}-y_{31}$	
				$e_{14} : y_{32}-y_{11}$	$e_{20} : y_{32}-y_{11}$	
				$e_{15} : y_{32}-y_{21}$	$e_{21} : y_{32}-y_{21}$	

exist or may not exist in π'_{new} . For example, both vertices 2 and 3 in π'_{new} must be connected by a gray edge, but vertex 3 does not exist in π'_{new} . The possible vertices that can be assigned vertex 3 are x_{11} and x_{21} . Thus, all *non-deterministic* gray edges in this case are $\{(2, x_{11}), (2, x_{21})\}$. Obviously, only one gray edge is selected from this set. For each E_i , all *non-deterministic* gray edges are subject to the following steps and the example is shown in Table 3.3.

- Let $T = \{T_1, T_2, \dots, T_q\}$ be the set of *temporary* names for the same duplicated gene, V_k be the set of *variables* for *temporary* name T_k , and B_k be the set of *final* names for *temporary* name T_k , $1 \leq k \leq q$. For example, $T_1 = \{x_1, x_2\}$, $V_1 = \{(x_{11}, x_{12}), (x_{21}, x_{22})\}$, $B_1 = \{(3, 4), (5, 6)\}$.
- V_k is split into two sets, *i.e.* V_{1k} and V_{2k} such that V_{1k} contains all the first elements for each order pair of set V_k , and V_{2k} contains all the second elements for each order pair of set V_k . Similarly, the B_k is also split into two sets *i.e.* B_{1k} and B_{2k} . Note that the set V_{1k} corresponds to B_{1k} and the set V_{2k} corresponds

to B_{2k} . For example, $V_{11} = \{x_{11}, x_{21}\}$, $V_{21} = \{x_{21}, x_{22}\}$, $B_{11} = \{3, 5\}$, and $B_{21} = \{4, 6\}$.

- For a vertex pair (a_{i1}, a_{i2}) , the variables a_{i1} and a_{i2} are checked to generate possible *non-deterministic* gray edges *i.e.* $e_1 - e_{24}$ in Table 3.3. There are three possible cases as follows:

1. Variable a_{i1} exists in π'_{new} but a_{i2} does not.

This implies that the name of a_{i1} is known. a_{i2} is assigned to a variable name in set V_{1k} if its actual name is in set B_{1k} . Otherwise, it is assigned to a variable name in set V_{2k} . For example, the vertex pair $(2, 3)$ in E_1 shown in Table 3.3 has two *non-deterministic* gray edges, $\{(2, x_{11}), (2, x_{21})\}$. This is because vertex 2 exists but vertex 3 does not currently exist. All possible solutions of this case are $\max(|V_{1k}|, |V_{2k}|)$.

2. Variable a_{i1} does not exist in π'_{new} but a_{i2} does.

a_{i1} is assigned to a variable name in set V_{1k} if its actual name is in set B_{1k} . Otherwise, it is assigned to a variable in set of V_{2k} . For example, the vertex pair $(18, 19)$ in E_7 (shown in Table 3.3) has three *non-deterministic* gray edges, $\{(y_{12}, 19), (y_{22}, 19), (y_{32}, 19)\}$. All possible solutions of this case are $\max(|V_{1k}|, |V_{2k}|)$.

3. Both a_{i1} and a_{i2} do not exist in π'_{new} .

a_{i1} is assigned to a variable in set V_{1k} if its actual name is in set in B_{1k} . Otherwise, it is assigned to a variable in set V_{2k} . a_{i2} is assigned to a variable in the other set different from the assignment of a_{i1} . For example, consider the vertex pair $(14, 15)$. Both vertices 14 and 15 do not exist in π'_{new} . Vertex 14 must be assigned to a variable in $\{y_{12}, y_{22}, y_{32}\}$ and vertex 15 must be assigned to a variable in $\{y_{11}, y_{21}, y_{31}\}$. Therefore, all

possible gray edges are $\{(y_{12}, y_{11}), (y_{12}, y_{21}), (y_{12}, y_{31}), (y_{22}, y_{11}), (y_{22}, y_{21}), (y_{22}, y_{31}), (y_{32}, y_{11}), (y_{32}, y_{21}), (y_{32}, y_{31})\}$. However, some gray edges, e.g. $(y_{12}, y_{11}), (y_{22}, y_{21}), (y_{32}, y_{31})$, are incorrect because they are generated from the same *temporary* name. Edges $(y_{12}, y_{11}), (y_{22}, y_{21})$, and (y_{32}, y_{31}) are generated from y_1, y_2 , and y_3 , respectively. These incorrect gray edges are discarded during the construction of the *incomplete* breakpoint graph. All possible solutions in this case are $|V_{1k}| \times |V_{2k}| - |t|$, where t is the set of order pairs that generated from the same *temporary* name.

All possible and correct gray edges, labelled by $e_1 - e_{24}$, are shown in Table 3.3. Each gray edge is denoted by a variable e_k . This variable is used in the *integer programming* procedure (described in the following section) as a *decision variable*. The value is set to 1 if its corresponding gray edge is selected, otherwise it is set to 0. The number of all possible solutions – typically very large – depends on the number of duplicated genes. As the time efficiency of integer programming introduces a major bottleneck, we rely on the following observation: Each *temporary* name π_j is decomposed into two variables π'_{j1} and π'_{j2} in π'_{new} and some *non-deterministic* gray edges must be connected to these variables. This indicates that each *temporary* name can generate a set of *non-deterministic* gray edges for connecting only all of its corresponding variables in π'_{new} and the others variables. Based on the fact above, some E_i 's related to the same *temporary* names can be grouped. For example, E_1, E_2 , and E_3 in Table 3.3 can be in the same group since they are related to the *temporary* names x_1 and x_2 . From Table 3.3, all E_i are grouped and named as follows: $S_1 = \{E_1, E_2, E_3\}$ and $S_2 = \{E_4, E_5, E_6, E_7\}$.

3. Formulating the Binary Integer Programming.

For any *incomplete* breakpoint graph, since a *black* edge must be directly connected with a *gray* edge and vice versa, a path can be formed by alternatively traversing these edges. This path can be classified into two types.

- (a) *Type A*: the path of this type forms a cycle and this cycle is named a *complete* cycle.
- (b) *Type B*: any path not in type *A* belongs to this type. A path of this type can form a cycle with itself or with other paths of type *B* only after the *variable* names of some π'_i in π' are assigned the final names and some gray edges are connected. The cycle is named an *incomplete* cycle, and the vertex that is not connected by any gray edge is named a *non-linking* vertex.

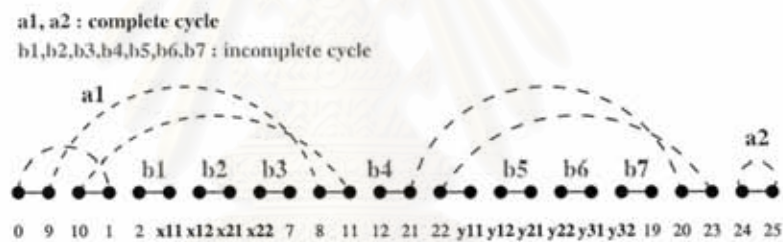


Figure 3.2: Cycles in IG named $a1$ and $a2$ are *type A* and $b1, b2, b3, b4, b5, b6, b7$ are *type B*.

An example of both types is shown in Fig. 3.2. There are two *complete* cycles or *type A* paths.

- 1st *complete* cycle ($a1$): $\{(9, 0), (0, 1), (1, 10), (10, 11), (11, 8), (8, 9)\}$.
- 2nd *complete* cycle ($a2$): $\{(24, 25), (25, 24)\}$.

For *type B* path, there are seven *incomplete* cycles.

- 1st *incomplete* cycle (b1): $\{(2, x_{11})\}$.
- 2nd *incomplete* cycle (b2): $\{(x_{12}, x_{21})\}$.
- 3rd *incomplete* cycle (b3): $\{(x_{22}, 7)\}$.
- 4th *incomplete* cycle (b4): $\{(12, 21), (21, 20), (20, 23), (23, 22), (22, y_{11})\}$.
- 5th *incomplete* cycle (b5): $\{(y_{12}, y_{21})\}$.
- 6th *incomplete* cycle (b6): $\{(y_{22}, y_{31})\}$.
- 7th *incomplete* cycle (b7): $\{(y_{32}, 19)\}$.

While completing the breakpoint graph, the set of gray edges needs to satisfy the following properties: (1) every *non-linking* vertex is incident to exactly one gray edge, and (2) only one gray edge from each edge group (E_p) is selected. Eq. 2.2 provides the essential clue to achieve the minimum number of reversals by minimizing $b(\pi) - c(\pi)$. The selected gray edges should create the *maximum number of cycles*. Because all paths of type *A* are complete cycles, there is no need to consider these *complete* cycles. Only the number of these *complete* cycles is used to estimate the minimum three edit distances. The focus is on finding the maximum number of cycles formed by the paths of type *B*. To find the maximum number of cycles from type *B* paths, the weight w_i for each edge e_i is defined according to the following heuristic rules. Fig. 3.3 illustrates the meaning of each case.

case 1: $w_i = 1$ if e_i forms the *complete* cycle size 1.

case 2: $w_i = 2$ if e_i joins two vertices in the same *incomplete* cycle.

case 3: $w_i = 3$ if e_i joins two vertices in different *incomplete* cycles.

The following notations are used for describing the application of binary integer programming.

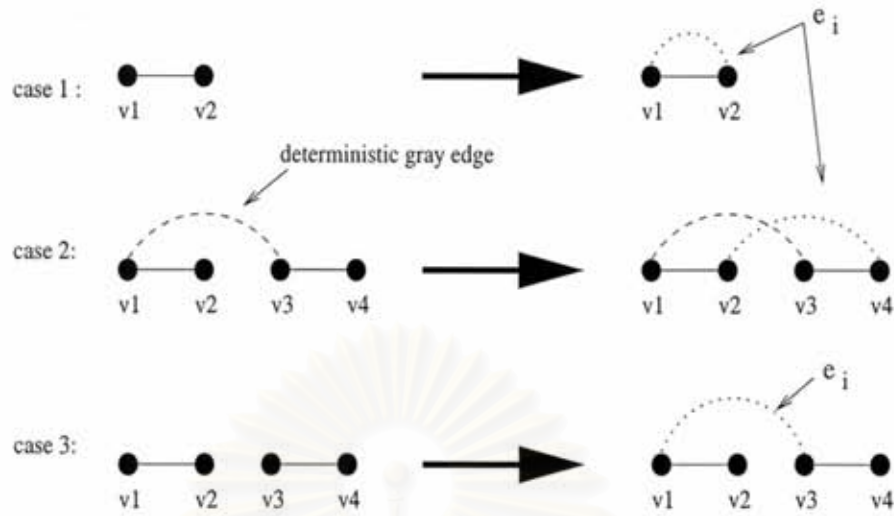


Figure 3.3: An example of three cases for connecting edge e_i . The *deterministic* gray edge is denoted by a dashed line while the *non-deterministic* gray edges are denoted by dotted lines. v_i is a vertex in the *incomplete* graph.

l_{ik} : equals to 1 if edge e_k belongs to an *incomplete* cycle b_i , otherwise 0.

N_B : the number of *incomplete* cycles type B .

N_{B_i} : the number of *variables* for cycle i of type B .

N_g : the number of possible gray edges.

G : the number of possible gray edge groups.

G_p : the number of possible gray edges for group p .

Objective Function:

$$\text{Minimize } \sum_{i=1}^{N_g} w_i \cdot e_i \quad (3.1)$$

Constraints:

$$\sum_{k=1}^{N_g} l_{ik} = N_{B_i} ; \text{ for } i = 1 \text{ to } N_B \quad (3.2)$$

$$\sum_{i=1}^{G_p} e_i = 1 ; \text{ for } p = 1 \text{ to } N_g \quad (3.3)$$

$$e_i - e_j = 0 ; \text{ every pairs } E_k \text{ and } E_{k+1}; \quad (3.4)$$

for $i=1$ to G_p ($p=k$),

and for $j=1$ to G_p ($p=k+1$),

and e_i *implies* e_j .

$$e_i \in 0, 1 \quad (3.5)$$

The objective function 3.1 minimizes the weight of candidate edges (the minimal weight forms the maximum cycles). The constraint 3.2 ensures that the number of selecting edges e_k does not exceed total number of variables for each cycle B_i , constraint 3.3 ensures that only one possible gray edge is selected for each group, and constraint 3.4 ensures that two selected edges are correctly transform from the same temporary name. Once this algorithm is completed, all variables are assigned *final* values. Finally, Eqs. 2.2, 2.4, and 2.5 can be applied to compute the minimal reversal, transposition, and transversal distances between the two re-labelled genomes, respectively. The outline for the EDMF algorithm is shown in the following section and the examples of value setting are shown in Table 3.4.

จุฬาลงกรณ์มหาวิทยาลัย

Algorithm EDMF

Input: (1.) Permutation π and ϕ of length n .

(2.) Set of gene family $g = \{\{g_1\}, \{g_2\}, \dots, \{g_k\}\}$.

Output : The selected gray edges are completed the *incomplete* breakpoint graph.

Begin

1. Rename each family g_i , $K(g_i) = 1$ of π with respect to ϕ , and replace each family g_i , $K(g_i) \geq 2$, of π with respect to ϕ by *temporary* names, $g_{i1}, g_{i2}, \dots, g_{iK(g_i)}$ and keep the mapping position.
2. Transform the signed permutation π to unsigned permutation π' by replacing each actual value π_i by $(2\pi_i - 1, 2\pi_i)$ if π_i is positive and $(2|\pi_i|, 2|\pi_i| - 1)$ if π_i is negative. For name g_{ij} , transforms to variables g_{ij1} and g_{ij2} .
3. Construct an *incomplete* breakpoint graph (IG) for π' .
4. Identify all black edges and *deterministic* gray edges in IG .
5. Identify possible gray edge groups (E) and *non-deterministic* gray edges (e) and the weight setting for each *non-deterministic* gray edge.
6. Formulate the binary integer programming (BIP) from equations 3.1–3.5.
7. Solve the equations 3.1–3.5.

End

3.2 Observation

The outcome from the EDMF algorithm is discussed in this section, which will concern estimation of the number of cycles in *incomplete* breakpoint graph and the edit distances of the EDMF algorithm. A few theoretical group work are establish to formalize some framework for the propose approach.

Lemma 1. *Given an incomplete breakpoint graph IG , the number of possible cycles of type B in case 3, $N_B^{(3)}$, is*

$$N_B^{(3)} \leq \lfloor \frac{n - b_A - b_1 - b_2}{2} \rfloor$$

n is the total number of black edges in IG . b_A is the number of black edges for cycles of type A . b_1 and b_2 are the number of black edges for cycles of type B in cases 1 and 2, respectively.

Proof. The number of black edges for cycles of types A and B in cases 1 and 2 are obviously fixed. Then, the cycles of type B in case 3 must be formed from the remaining black edges not in cases 1 and 2. Note that each cycle of type B in case 3 must use at least two black edges to complete the cycle. Therefore, the number of cycles for type B in case 3 is $\lfloor \frac{n - b_A - b_1 - b_2}{2} \rfloor$. \square

Theorem 1. *Given an incomplete breakpoint graph IG , the number of cycles in the incomplete graph, C , has the following upper bound*

$$C \leq N_A + \sum_{i=1}^3 N_B^{(i)}$$

N_A is the total number of cycles of type A . $N_B^{(i)}$ is the number of cycles of type B in case i , $1 \leq i \leq 3$.

Proof. Every cycle of type A is completed by itself without involving any cycle of type B . Similarly, every cycle of type B needs no cycle of type A as a part of itself. In addition, each case of type B is independent from the other cases of type B . Therefore, counting the total number of cycles of type A and all cases of type B can be considered separately. Without loss of generality, a given *incomplete* graph has cycles of both types. Therefore, the upper bound for the number of cycles is stated as above. \square

Theorem 2. *Given two genomes π and ϕ , the EDMF has the reversal distance with multi-gene families*

$$d_{rev} \geq n - C$$

when the number of hurdles is 0 and the fortress factor is also 0. n is the number of black edges.

Proof. Since the value of C has the maximum value as proved in Theorem 1 and the number of black edges for the given *incomplete graph* is a constant, the minimum reversal distance, d_{rev} , is bounded by $n - C$. \square

Theorem 3. *Given two genomes π and ϕ , the EDMF has the transposition and transversal distance with multi-gene families*

$$d_{trp}, d_{trv} \geq \frac{n + 1 - C}{2}$$

when n is the number of black edges.

Proof. Since the value of C has the maximum value as proved in Theorem 1 the minimum transposition and transversal distance with multi-gene families, d_{tr} , is bounded by $\frac{n+1-C}{2}$. \square

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER IV

EXPERIMENTAL RESULTS

We implemented the EDMF algorithm in MATLAB version 7.0 and tested it for correctness and performance. All tests were performed on Sony laptop with 1.6 MHz Pentium IV processor and 1 GB of RAM, running on Microsoft Windows XP. The optimization toolbox for MATLAB version 7.0 was used for implementing Binary Integer Programming (module *bintprog*). Test data fell into two categories: *synthetic* and *real* biological data. Once the EDMF algorithm is completed, all variables are assigned *final* values (i.e. transformed *incomplete* breakpoint graph into *complete* breakpoint graph). The Eqs. 2.2, 2.4, 2.5 can be applied to compute the reversal, transposition and transversal distance between the two re-labelled genomes, respectively. The experimental results are shown in the following sections.

4.1 Minimum Reversal Distance with Multi-Gene Families

4.1.1 Synthetic data

The synthetic data set for computing the reversal distance with multi-gene families is generated as follows:

1. Start from π with m distinct symbols whose signs are also generated randomly.
2. Randomly generate f families, where each family has random size $2 \leq K(f_i) \leq 4$, recursively combining single gene until the size equals to $K(f_i)$.

To obtain the genome ϕ , we performed t reversals on the genome π . The boundaries of these random are uniformly distributed within the size of genome.

The accuracy of our method was compared to that achieved by the SRDD algorithm. The executable program is provided from the authors [11]. Although SRDD was not originally proposed to compute the reversal distance, but its objective was closely related to this problem. We ran the EDMF and SRDD on 10 random instances. We compared the calculated reversal distance, from the equation 2.2, with the exact minimal reversal distance obtained by the authoritative program GRAPPA version 2.0 [8], module of *invdist*, to compute the reversal distance for all possibilities and found the best one that has the minimal reversal distance.

Fig. 4.1 and 4.2 show the average performance of both algorithms over 10 instances in terms of reversal distance, in comparison with the exact minimal reversal distance as determined by GRAPPA (*invdist*).

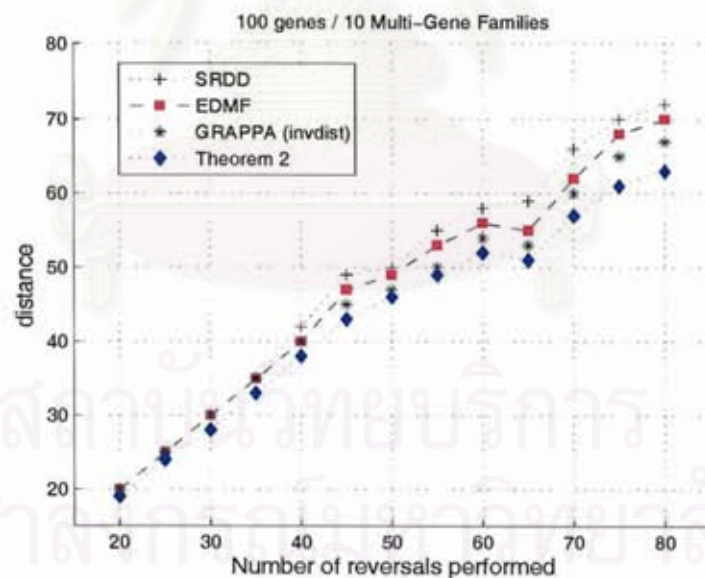


Figure 4.1: The reversal distance of both SRDD and EDMF comparing with the exact reversal distance from GRAPPA (*invdist*) for genome length 100 and 10 gene families.

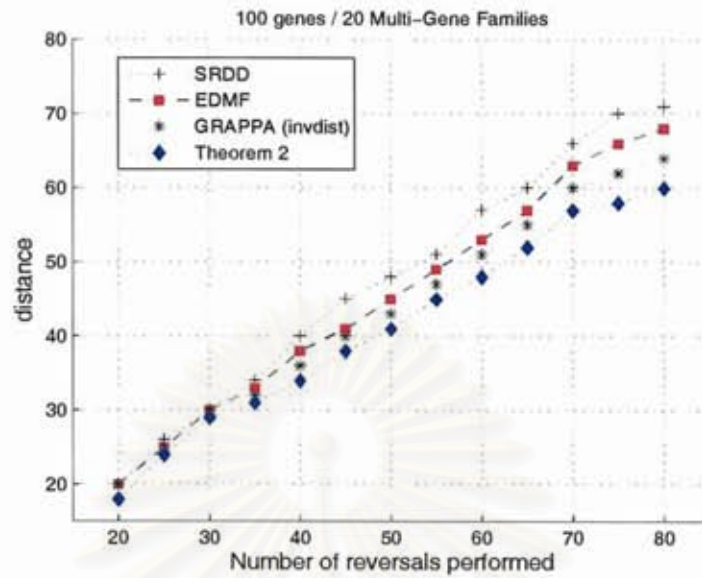


Figure 4.2: The reversal distance of both SRDD and EDMF comparing with the exact reversal distance from GRAPPA (invdist) for genome length 100 and 20 gene families.

On the average, each run of both EDMF and SRDD algorithm takes less than 10 seconds. Our heuristic algorithm consistently produces a closer estimate of the exact minimal reversal distance compared to SRDD, for each $t > 35$. However, both algorithms underestimate the actual reversal distance (t), for each $t > 35$. The last line from both Fig. 4.1 and Fig. 4.2 show the reversal distances that are estimated from theorem 2 (Chapter III). However, all distances are underestimate from EDMF and exact distance, because the assumption to find the number of cycle in type B case 3 from lemma 1 is *the cycle of type B in case 3 that must use at least two black edges to complete the cycle*. Therefore, the estimated cycles from theorem 1 are greater than the exact cycles. That means, the estimated reversal distances from theorem 2 might be less than the exact reversal distances. These statistics indicate that our algorithm is quite reliable in finding the nearest minimal reversal distance with multi-gene families.

4.1.2 Real biological data

We downloaded the X chromosome of human, (Homo Sapien, NCBI build 34, July 2003 UCSC hg 16) mouse (Mus musculus, NCBI build 32, October 2003; UCSC mm4) and rat (Rattus norvegicus, Baylor HGSC v.31, June 2003; UCSC rn3) from the SOAR web page (<http://www.cs.ucr.edu/xinchen/soar.html>). There are 922 genes from human X chromosome, 1030 genes from mouse and 899 genes from rat, respectively. We also used information about gene families from this site. There are 355 families of size one between human-mouse, 321 between human-rat, and 348 between mouse-rat. The size distribution of gene families with more than two members are shown in Fig. 4.4.

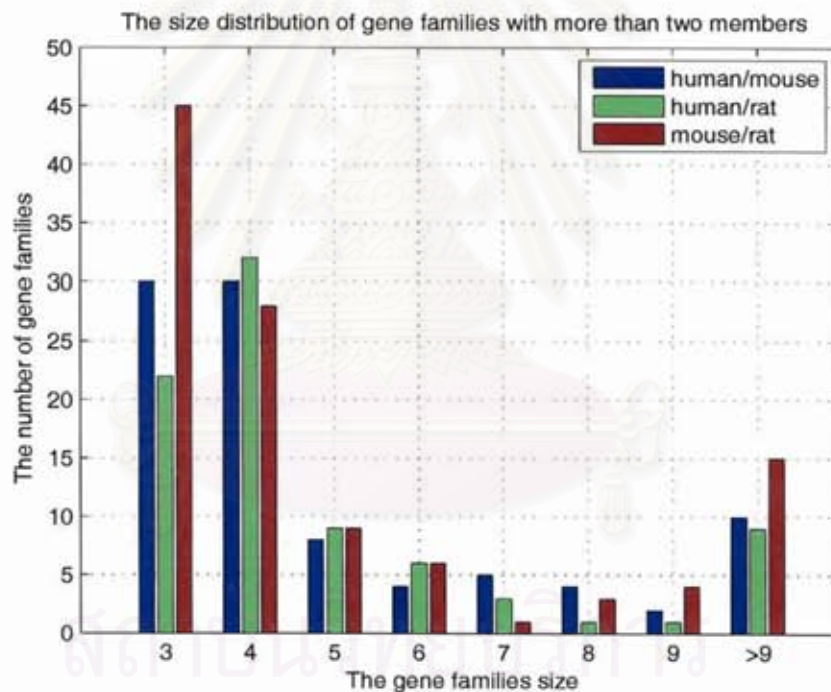


Figure 4.3: The size distribution of gene families constructed for each pair of genomes. The number of gene families of size one are not shown in the figure for clarity (they are 355, 321, and 348, respectively).

By using the SRDD, the reversal distance and breakpoint distance between human-mouse are 124 and 143, between human-rat the distances are 119 and 135, between mouse-rat they are 155 and 188, respectively. We ran EDMF for all genome combinations. The comparative results for all three pairs of genomes are summarized in Table 4.1.

Table 4.1: Results of reversal distance from EDMF comparing with SRDD.

X chromosome of	Reversal Distance		Breakpoint Distance		Lower bound Reversal Distance	Computing time EDMF (minute)
	SRDD	EDMF	SRDD	EDMF		
human-mouse	124	124	143	143	119	18
human-rat	119	119	135	135	108	15
rat-mouse	155	154	188	184	136	20

The results show that SRDD and EDMF determine similar distances for human-mouse and human-rat. We believe that the inconclusive result is due to the low estimated reversal distance between human-mouse and human-rat (34 and 35% of the number of single genes, respectively). According to the synthetic data, both SRDD and EDMF obtain similar minimal distances when $t \leq 35$. However, the EDMF improves slightly on SRDD for the mouse-rat comparison, both in terms of breakpoint and reversal distance. Noteworthy, the estimated reversal distance between mouse and rat is 45% of the number of single genes.

4.2 Minimum Transposition Distance with Multi-Gene Families

4.2.1 Synthetic data

The synthetic data set for computing the transposition distance with multi-gene families is generated as follows:

1. Start from π with m distinct symbols whose signs are plus (+) because the problem of transposition distance is unsigned.
2. Randomly generate f families, where each family has random size $2 \leq K(f_i) \leq 4$, *i.e.* recursively combining single gene until the size equals to $K(f_i)$.

To obtain the genome ϕ , we performed t transpositions on the genome π . The boundaries of these random are uniformly distributed within the size of genome.

Unfortunately, there is no real bioinformatics tool to calculate the transposition distance or sorting by transposition. Therefore, we took an indirect approach to computing the transposition distance by using the module *invdist* of GRAPPA to find the number of cycles for all possibilities. We have also used the Eq. 2.4 to calculate the transposition distance and selected one that had the minimal distance (because the dominant parameter of Eq. 2.4 is the number of cycle from the breakpoint graph).

In order to rigorously test for the correctness of our method, we ran the EDMF on 10 random instances. Fig. 4.4 and 4.5 show the average performance of EDMF algorithm over 10 instances in terms of transposition distance, in comparison with the exact minimal transposition distance as determined by GRAPPA.

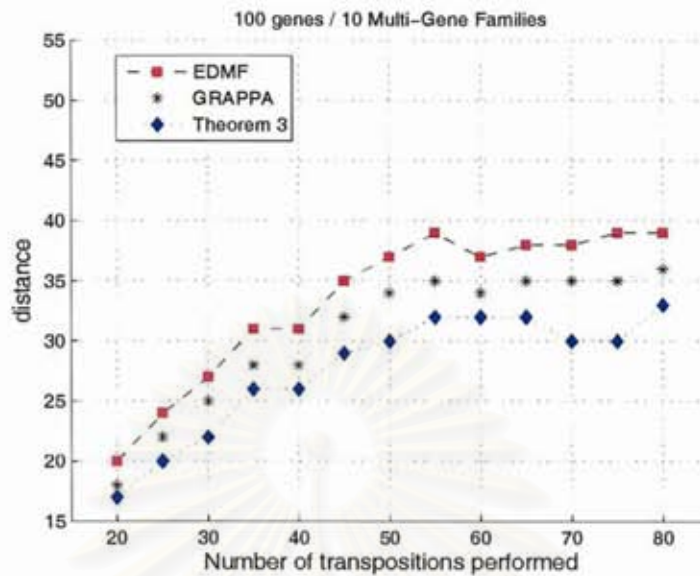


Figure 4.4: The transposition distance of EDMF comparing with the exact transposition distance from GRAPPA for genome length 100 and 10 gene families.

On the average, each run of EDMF takes less than 10 seconds. However, the reason why we do not show the comparisons with other algorithms is because there is no research on transposition distance with multi-gene families. The results only show the calculated transposition that are estimated from EDMF algorithm compare with the exact transposition distance that computed from the GRAPPA program. The last line is the estimated transposition distance from Theorem 3.

4.3 Minimum Transversal Distance with Multi-Gene Families

4.3.1 Synthetic data

The synthetic data set for computing the transversal distance with multi-gene families is generated as follows:

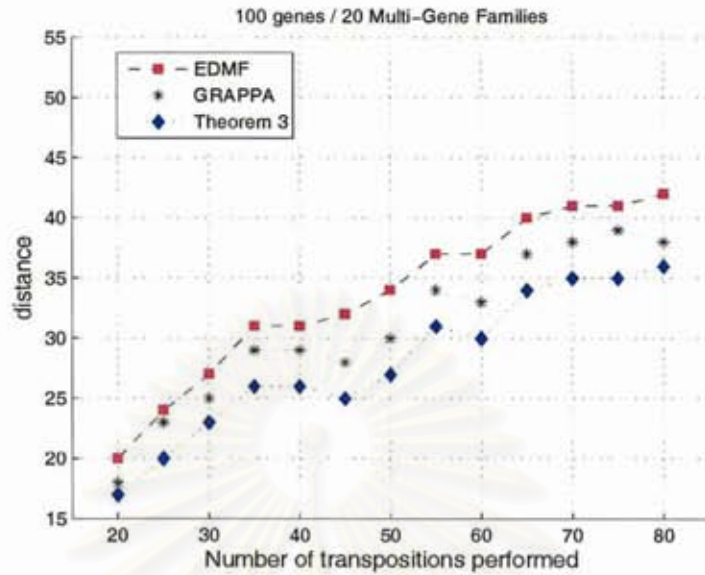


Figure 4.5: The transposition distance of EDMF comparing with the exact transposition distance from GRAPPA for genome length 100 and 20 gene families.

1. Start from π with m distinct symbols whose signs are also generated randomly.
2. Randomly generate f families, where each family has random size $2 \leq K(f_i) \leq 4$, recursively combining single gene until the size equals to $K(f_i)$.

To obtain the genome ϕ , we performed t transversal on the genome π . The boundaries of these random are uniformly distributed within the size of genome.

Unfortunately, there is no real bioinformatics tool to calculate the transversal distance or sorting by transversal. Therefore, we took an indirect approach, *as same as the transposition*, to computing the transversal distance by using the module *invdist* of GRAPPA to find the number of cycles for all possibilities. We have also used the Eq. 2.5 to calculate the transversal distance and selected one that has the minimal distance (because the dominant parameter of Eq. 2.5 is the number of cycle from the breakpoint graph).

In order to rigorously test for the correctness of our method, we ran the EDMF on 10 random instances. Fig. 4.6 shows the average performance of EDMF algorithm over 10 instances in terms of transversal distance, in comparison with the exact minimal transversal distance as determined by GRAPPA.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

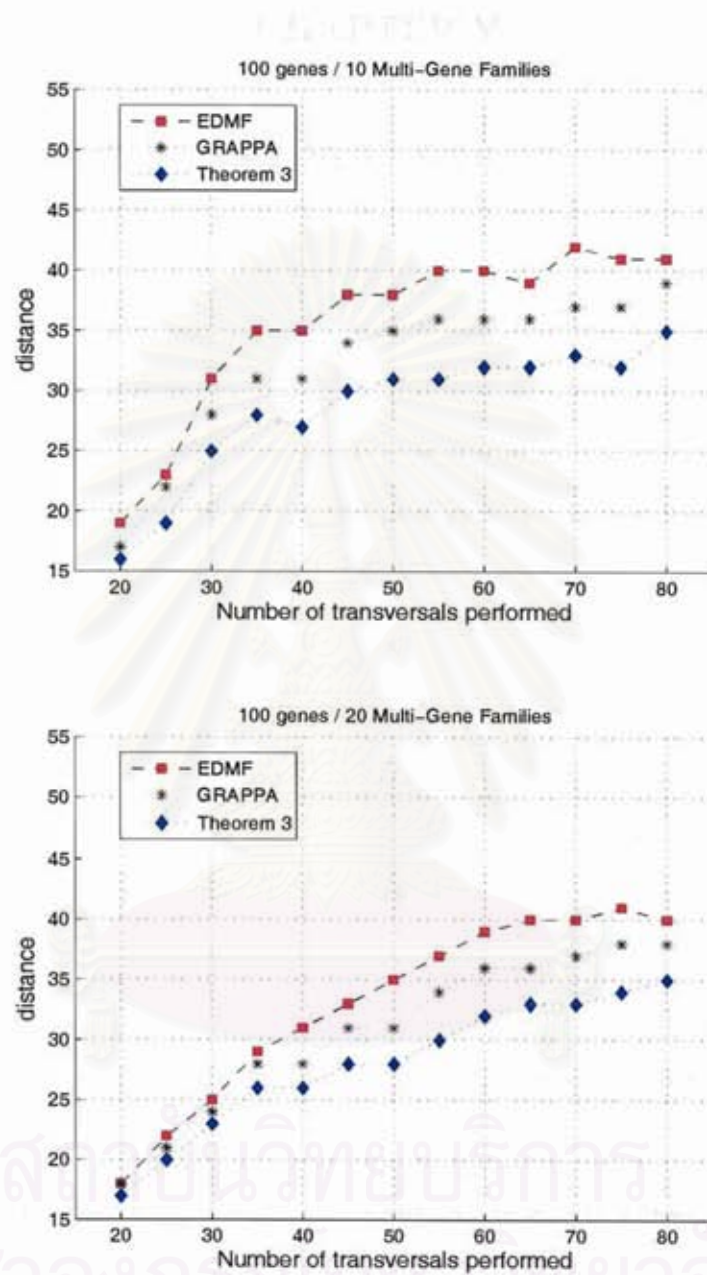


Figure 4.6: The transversal distance of EDMF comparing with the exact transversal distance from GRAPPA for genome length 100 and 10 and 20 gene families.

CHAPTER V

CONCLUSION

The main problem of genome rearrangement concerns the approach to computing the distances (reversal, transposition and transversal) or transforming one genome to another by a minimum number of the operations. There are many efficient methods to find the shortest distance. However, all of them are used for genomes that have not contain gene duplications. In this dissertation, we try to solve the problem of gene duplications. A new heuristic algorithm to transform the duplication genes problem to non-duplication problem and to find the nearest minimal three edit distances (*reversal, transposition, and transversal*) between genomes with multi-gene families is proposed.

We show how to extend the Hannenhalli and Pevzner theorem and concept of the breakpoint graph to genomes with multi-gene families. The approach uses the notion of a breakpoint graph, but readily provides means for exploring possible combinations of duplicate genes across genomes. The exploration is done using binary integer programming optimization based on pre-determined penalties for properties of an *incomplete* version of the breakpoint graph.

We separately show the experimental results for each edit distance as follows:

1. For the reversal distance with multi-gene families, we have tested on synthetic and real data sets (from human, mouse and rat X chromosome) and comparing our results with the existing algorithm, SRDD. The results demonstrate that our approach (EDMF) generally outperforms the SRDD algorithm in term of accuracy

of determining the minimal reversal distance. Moreover, we have compared our results with the exact reversal distance (the distance that finds from all possible permutations, using the GRAPPA, and pick one that obtain the minimal distance). The results show that our reversal distances are closed to the exact distance.

2. For the transposition distance with multi-gene families, we have tested on synthetic data sets and compared the results with the exact transposition distance (the distance that finds from all possible permutations, using the GRAPPA, and pick one that yield the minimal distance). The results show that our transposition distances are closed to the exact transposition distance.
3. For the transversal distance with multi-gene families, we have also tested on synthetic data sets and compared the results with the exact transversal distance (the distance that finds from all possible permutations, using the GRAPPA, and pick one that yields the minimal transversal distance). The results show that our transversal distances are closed to the exact distance.

From the experimental results, we can conclude that the heuristic algorithm to computing the nearest minimal edit distance for multi-gene families via binary integer programming is quite reliable in finding the nearest three edit distances with multi-gene families. The genome rearrangement with multi-gene families problem may have the following further studies:

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

1. To apply with multi-chromosome problem.
2. To apply with the other mutation events such as insertion, deletion, fusion, and fission.
3. To apply with unequal number of gene families between two genomes.
4. To apply with unequal length of two genomes.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

References

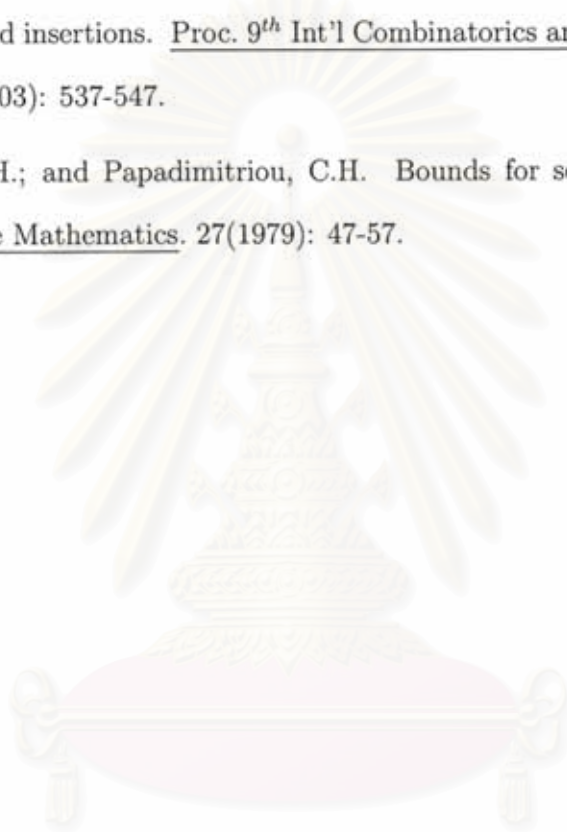
1. Caprara, A. Sorting by reversals is difficult. Proceedings of the First Annual International Conference on Computational Molecular Biology. pp.75-83. New Mexico, 1997.
2. Caprara, A. Sorting permutations by reversals and Eulerian cycledecompositions. SIAM Journal of Discrete Mathematics. 12(1999): 91-110.
3. Bergeron, A. A very elementary presentation of the Hannenhalli-Pevzner theory. Combinatorial Pattern Matching 12th Annual Symposium. pp. 106-117. Israel, 2001.
4. Caprara, A. On the practical solution of reversal median problem. Algorithm in Bioinformatics, Proceeding of WABI 2001. 2149(2001): 238-251.
5. Bergeron, A.; Heber, S.; and Stoye J. Common intervals and sorting by reversals: A marriage of necessity. Bioinformatics. 18(2002): 54-63.
6. Christie, D. A $3/2$ -approximation algorithm for sorting by reversals. Proceedings of ACM-SIAM Symposium on Discrete Algorithms. pp. 244-252. San Francisco, California, 1998.
7. Christie, D. Genome Rearrangement Problems. Doctoral Dissertation, Department of Computer Science, University of Glasgow. 1998.
8. Bader, D.A.; Moret, B.M.E.; and Yan A. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. Algorithms and Data Structures: Seventh International Workshop WADS 2001. pp. 365-376. 2001.
9. Kececioclu, J.; Sankoff D. Exact and approximation algorithms for sorting by

- reversals, with application to genome rearrangement. Algorithmica. 13(1995): 180-210.
10. Zhang, J. Evolution by gene duplication : an update. TRENDS in Ecology and Evolution. 18(2003): 292-298.
 11. Chen, X. et.al, Computing the assignment of orthologous genes via genome rearrangement. Proc. Asia Pacific Bioinformatics Conference (APBC2005). pp. 363-378. 2005.
 12. Setubal, J. and Meidanis, J. Introduction to Computational Biology. PWS Publishing Co., 1997.
 13. Pevzner, P.A. Computational Molecular Biology: An Algorithmic Approach. MIT Press, 2000.
 14. Shamir, R. Algorithms in molecular biology: Lecture notes [Online]. 2001. Available from: <http://www.math.tau.ac.il/rshamir/algmb/algmb00.html>
 15. Aldridge, S. The thread of life: the story of genes and genetic engineering. Cambridge University Press:Cambridge, 1996.
 16. Ph.D. in Toxicology Molecular and Cellular Toxicology (MCT) Concentration [Online]. 2006. Available from http://www.tox.ncsu.edu/graduate/phd_m&c.htm
 17. Jackson, R. New low chromosome number for plants. Science. 126(1957): 1115-1116.
 18. McAllister, B.F. Fixation of chromosomal rearrangements. Comparative Genomics. pp. 19-27. 2000.
 19. Sankoff, D. and Goldstein, M. Probabilistic models of genome shuffling. Bulletin of Mathematical Biology. 51(1989): 117-124.

20. Sankoff, D.; Cedergren, R.; and Abel Y. Genomic divergence through gene re-arrangement. Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences. 183(1990): 428-438.
21. Watterson, G.A.; Ewens, W.J.; and Hall, T.E. The chromosome inversion problem. Journal of Theoretical Biology. 99(1982): 1-7.
22. Sankoff, D.; and Blanchette, M. Multiple genome rearrangement and breakpoint analysis. Journal of Computational Biology. 5(1998): 555-570.
23. Bafna V.; and Pevzner P. Genome rearrangements and sorting by reversals. SIAM Journal of Computing. 25(1996): 272-289.
24. Hannenhalli, S.; and Pevzner P.A. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). Proceedings of the 27th Annual ACM Symposium on the Theory of Computing. pp. 178-189. 1995.
25. Berman, P.; and Hannenhalli, S. Fast sorting by reversal. Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching. pp. 168-185. 1996.
26. Ackermann, W. Zum Hilbertschen Aufbau der reellen Zahlen. Mathematische Annalen. 99(1928): 118-133.
27. Kaplan, H.; Shamir, R.; and Tarjan, R.E. A faster and simpler algorithm for sorting signed permutations by reversals. SIAM Journal of Computing. 29(1999): 880-892.
28. Siepel, A. An algorithm to find all sorting reversals. Proceedings of the Sixth Annual International Conference on Computational Molecular Biology. pp. 281-290. 2002.
29. Berman, P.; Hannenhalli, S.; and Karpinski, M. 1.375-approximation algorithm

- for sorting by reversals. Electronic Colloquium on Computational Complexity. pp. 01-047. 2001.
30. Bafna V.; and Pevzner P. Sorting by transpositions. SIAM Journal of Discrete Mathematics. 11(1998): 224-240.
31. Gu, Q.; Peng, S.; and Sudborough, H. A 2-Approximation Algorithms for Genome Rearrangements by Reversals and Transpositions. Theoretical Computer Science. 210(1999): 327-339.
32. Sankoff, D. Genome rearrangements with gene families, Bioinformatics. 15(1999): 909-917.
33. Bryant, D. The complexity of calculating exemplar distances. Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map alignment and the Evolution of Gene Families. 1(2000): 202-212.
34. Sankoff, D.; Leduc, G.; Antoine, N.; Paquin, B.; and Lang, B.F. Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome. Proceedings of the National Academy of Sciences, 89(1992): 6575-6579.
35. Tang, J.; and Moret, B.M.E. Phylogenetic reconstruction from gene rearrangement data with unequal gene contents, Proc. 8th Workshop on Algorithms and Data Structures, 2748(2003): 37-46.
36. McLysaght, A.; Seoighe, C.; and Wolfe, K.H. High frequency of inversions during eukaryote gene order evolution. Comparative Genomics. Kluwer Academic Press. pp. 47-58. 2000.
37. Li, W.H.; Gu, Z.; Wang, H.; and Nekrutenko, A. Evolutionary analysis of the human genome. Nature. 409(2001):847-849.

38. Lynch, M.; and Conery, J.S. The evolutionary fate and consequences of duplicated genes. Science. 290(200): 1151-1155.
39. Blanchette, M.; Kunisawa, T.; and Sankoff, D. Parametric genome rearrangement. Gene. 172(1996): 11-17.
40. Marron, M.; Swenson, K.M.; and Moret, B.M.E. Genomic Distances under deletions and insertions. Proc. 9th Int'l Combinatorics and Computing Conference. 2697(2003): 537-547.
41. Gates, W.H.; and Papadimitriou, C.H. Bounds for sorting by prefix reversals. Discrete Mathematics. 27(1979): 47-57.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Biography

Name: Mr.Jakkarin SUKSAWATCHON.

Date of Birth: 06th April, 1976.

Educations:

- Ph.D., Program in Computer Science, Department of Mathematics, Chulalongkorn University, Thailand, (June 2001 - March 2006)
- Ph.D. Visiting student, School of Information Technology Electrical and Engineering, University of Queensland, Brisbane, AUSTRALIA, (February 2005 - July 2005).
- M.Sc. Program in Information Technology, King Mongkut's Institute of Technology Ladkrabang, Thailand. (June 1998 - September 2000).
- B.Sc. Program in Computer Science, Burapha University, Chonburi, Thailand. (June 1994 - March 1998).

Publication papers:

- J.Suksawatchon and C.Lursinsap. Symbolic Reversal Algorithm for Estimating Genomic Distance with Multiple Gene Duplication. *Proceedings of the 4th International Conference on Intelligent Technologies*, pp. 108-116, 2002.
- J.Suksawatchon, C.Lursinsap, and M.Boden. Heuristic Algorithm for Computing Reversal Distance with Multi-gene Families via Binary Integer Programming. *Proc. IEEE Symp. on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 187-193, 2005.
- J.Suksawatchon, C.Lursinsap, and M.Boden. Computing the Signed Reversal Distance between Genomes in the Presence of Multi-Gene Families via Binary Integer Programming. *Journal of Bioinformatics and Computational Biology*, Submitted.

Work: Lecturer, Computer Science Department, Faculty of Science, Burapha University.

Scholarship: Commission on Higher Education, Thailand.