Computing Genomic Distance Between Signed Genomes Without Co-tail Constaint*

Xiaoli Wang† Tao Li‡ Guojun Li§ Ying Xu¶

Abstract. Genomes rearrangement is an important problem in evolutionary molecular biology. From a computational perspective, the study of evolution based on rearrangements leads to computing the rearrangement's distance problem. Two slightly different $O(n^3)$-time algorithms are given by Hannenhalli and Pevzner [11], Ozery-Flato and Shamir [12], respectively, for computing the minimum number of reversals, translocations, fissions and fusions that would transform one multi-chromosomal genome into another when both have the same set of genes without repeats and the orientation of the genes is known. This minimum number is called the genomic distance between two multi-chromosomal genomes. We give a linear algorithm for computing the genomic distance between signed multi-chromosomal genomes.
Keywords. genomic distance; signed multi-chromosomal genome; translocation; reversal; linear algorithm

1 Introduction

With the advent of large-scale DNA physical mapping and sequencing, studies of genome rearrangements are becoming increasingly important in evolutionary molecular biology. A computational approach to evolutionary studies based on rearrangements was pioneered by Sankoff [1, 2]. Study of genomes evolving by rearrangements involves a combinatorial problem of computing the minimum number t of rearrangement events transforming one genome Π into another genome Γ and finding a shortest sequence of rearrangement events transforming one genome into another. Such problem is known as *genomic sorting problem*. We call t the genomic distance between Π and Γ, and denote it by $d(\Pi, \Gamma)$. The problem of finding t is called the *genomic distance problem*.

The importance of computing the genomic distance, under most of the rearrangement events, have motivated researchers to develop approximation algorithms for genomic distance problems for various types of rearrangements. The uni-chromosomal genomic sorting problem that only allows reversals is called *reversals sorting problem* (SBR) and the genomic distance is then called the *reversal distance*. Kececioglu and Sankoff [3],

*This work was supported by NSFC of China under Grant No.10271065, No.60373025; GJL and YX's work was supported in part by the US Department of Energy's Genomes to Life program (http://doegenomestolife.org/) under project, "Carbon Sequestration in Synechococcus sp.: From Molecular Machines to Hierarchical Modeling" (www.genomes2life.org) and by National Science Foundation (NSF/DBI-0354771,NSF/ITR-IIS-0407204).

†Department of Mathematics, Nanyang Normal University, Henan 473061, P. R. China, and School of Mathematics and Systems Science, Shandong University, Jinan 250100, P. R. China

‡Department of Mathematics, Nanyang Normal University, Henan 473061, P. R. China

§Institute of Software, Chinese Academy of Sciences, Beijing 100080, P. R. China; School of Mathematics and Systems Science, Shandong University, Jinan 250100, P. R. China

¶Dept of Biochemistry and Molecular Biology, University of Georgia, USA

Corresponding author: Xiaoli Wang

Bafna and Pevzner [4] gave approximation algorithms, respectively, for computing rearrangement distance for uni-chromosomal genomes evolving by reversals. Hannenhalli and Pevzner [5] showed that SBR problem is in P, and give an polynomial algorithm for SBR. The algorithm was improved in [6, 7], the best algorithm's running time is $O(n^3)$ [7] by now. In 2001 Bader, Moret and Yan [8] presented a linear algorithm for computing the reversal distance. Kececioglu and Ravi [9] gave a 2-approximation algorithm for the rearrangement distance problem for genomes evolving by translocations and a 1.5 approximation algorithm for the rearrangement distance problem for multi-chromosomal genomes evolving by both the translocations and reversals. Hannenhalli [10] has devised a polynomial algorithm for the genomic sorting problem when only internal translocations are allowed.Hannenhalli and Pevzner studied the genomic sorting problem for reversals and translocations (SBRT) [11]. They presented a duality theorem for SBRT, gave a polynomial algorithm for computing the genomic distance, using reversals, translocations,fissions and fusions, between multi-chromosomal genomes. But for the general case their theorem and algorithm fail. Ozery-Flato and Shamir [12] correct the theorem and present an exactly polynomial algorithm. Both algorithms can be executed in running time $O(n^3)$.

In this paper we are interested in multi-chromosomal genomes evolving by reversals, translocations, fissions and fusions, each gene occurs exactly once in a genome. We give a linear-time algorithm for computing the genomic distance between signed multi-chromosomal genomes. That is, we present a linear-time algorithm that computes the length of a shortest sequence of reversals, translocations, fissions and fusions required to transform one genome into another.

The rest of this paper is split into three sections. In Section 2, we give some preliminary results needed to discuss the genomic distance, and the combinatorial formulation of the problem. In Section 3, we design and analyze an algorithm for computing the genomic distance between signed permutations. Some concluding remarks appear in Section 4.

2 Preliminaries

2.1 Chromosome, genome and transformation

In the model we consider any gene is unique within a genome and is represented by an identification number (positive integer), and an associated sign "+" or "-" reflecting the direction of the gene. A chromosome can be represented as a sequence of genes, and a genome is a set of chromosomes. A chromosome is orientation-less, therefore flipping a chromosome $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ into $-\pi = (-\pi_n, \ldots, -\pi_2, -\pi_1)$ does not affect the chromosome it represents. Hence a chromosome π is said to be *identical* to a chromosome γ iff $\pi = \gamma$ or $\pi = -\gamma$. Two genomes are said to be identical if their sets of chromosomes are the same.

Let π and γ be two chromosomes, $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_m)$. A *reversal* $\rho(\pi, i, j), 1 \leq i \leq j \leq n$, transforms π into $\pi' = (\pi_1, \ldots, \pi_{i-1}, -\pi_j, \ldots, -\pi_i, \pi_{j+1}, \ldots, \pi_n)$. A reversal $\rho(\pi, i, j)$ is said to be internal if $1 < i < j < n$. A translocation is a rearrangement that works on two chromosomes switching their 'tails'. A *translocation* $\rho(\pi, \gamma, i, j), 1 \leq i \leq n+1, 1 \leq j \leq m+1$, exchanges segments of genes between the chromosomes π and γ and transforms them into the chromosomes $(\pi_1, \ldots, \pi_{i-1}, \gamma_j, \ldots, \gamma_m)$ and

$(\gamma_1, \ldots, \gamma_{j-1}, \pi_i, \ldots, \pi_n)$. A *fusion* is a special translocation $\rho(\pi, \gamma, n+1, 1)$, which concatenates the chromosome π and γ resulting in a chromosome $(\pi_1, \ldots, \pi_n, \gamma_1, \ldots, \gamma_m)$ and an empty chromosome $\emptyset$. A *fission* is a translocation $\rho(\pi, \emptyset, i, 1)$, for some $1 < i \leq n$, 'breaks' a chromosome π into two chromosomes $(\pi_1, \ldots, \pi_{i-1})$ and $(\pi_i, \ldots, \pi_n)$. A translocation $\rho(\pi, \gamma, i, j)$ is called *internal* if $\pi, \gamma \neq \emptyset, 1 < i \leq n, 1 < j \leq m$. The muti-chromosomal genomic sorting problem evolving by internal reversals, internal translocations, fissions and fusions is indeed a genomic sorting problem evolving by reversals and translocations (SBRT).

For a chromosome $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$, the numbers $+\pi_1$ and $-\pi_n$ are called the *tails* of π. Notice that flipping a chromosome does not change the set of its tails. We use $T(\Pi)$ to denote the set of tails of all chromosomes in a genome Π. Genomes Π and Γ are called *co-tailed* if $T(\Pi) = T(\Gamma)$. Using an internal reversal or translocation on a genome does not change the set of its tails. Therefore, the problem of sorting a genome Π into a target genome Γ, using internal translocations and reversals, is limited to genomes Π and Γ that are co-tailed.

For any positive integer i, we use $[i, j]$ to denote the set $\{i, i+1, \ldots, j-1, j\}$. Given a graph $G = (V, E)$, for any vertex $v \in V$, we use $N[v]$ to denote the closed neighborhood of v: $N[v] = \{u : u = v \text{ or } uv \in E\}$.

2.2 Reversals on signed permutations (uni-chromosomal genomes)

Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ be two unsigned permutation of elements $\{1, 2, \ldots, n\}$. We extend them by adding $\pi_0 = 0, \pi_{n+1} = n+1$ to π and $\gamma_0 = 0, \gamma_{n+1} = n+1$ to γ. The *breakpoint graph* of π with respect to γ is a graph with $n+2$ vertices $\{\pi_0, \pi_1, \pi_2, \ldots, \pi_n, \pi_{n+1}\}$. We join vertices π_i and π_j by a black (resp. gray) edge iff they are neighbors in π (resp. γ).

Given two signed permutations $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ of elements $\{1, 2, \ldots, n\}$, where $|\pi_i|, |\gamma_i| \in \{1, 2, \ldots, n\}$. We transform them into unsigned permutations $u(\pi)$ and $u(\gamma)$ by substituting the ordered pair $(2x-1, 2x)$ for the positive element x, and ordered pair $(2x, 2x-1)$ for the negative element $-x$. Then by the definition of breakpoint graph of $u(\pi)$ with respect to $u(\gamma)$, there are always two edges (one black and the other gray) connecting the two split vertices $2x-1$ and $2x$ in the breakpoint graph of $u(\pi)$ with respect to $u(\gamma)$. Deleting all such $2n$ edges from the breakpoint graph results in a new graph, denoted by $G(\pi, \gamma)$, which is called the *cycle graph* of the signed permutation π with respect to the target signed permutation γ. In $G(\pi, \gamma)$, two vertices connected by a black(gray) edge are neighbor in $u(\pi)(u(\gamma))$. Every reversal $\rho(\pi, i, j)$ on π can be mimicked by the reversal $\rho(u(\pi), 2i-1, 2j)$ on $u(\pi)$.

For the unsigned permutation $u(\pi) = (\pi_1, \pi_2, \ldots, \pi_{2n})$, associated every gray edge $e = (\pi_i, \pi_j), i < j$, in $G(\pi, \gamma)$ with the *interval* $[e.B, e.E]$, where $e.B = i$, $e.E = j$. The two edges are said to be *overlap* if the intersection of their associated intervals is nonempty but neither properly contains the other. Cycles C_1 and C_2 in $G(\pi, \gamma)$ are *interleaving* if there exist two gray edges $g_1 \in C_1, g_2 \in C_2$ such that g_1 and g_2 are overlap.We call a cycle in $G(\pi, \gamma)$ with k gray (black) edges as *k-cycle*.

Let $b(\pi, \gamma)$ and $c(\pi, \gamma)$ denote the number of black edges and the number of cycles in $G(\pi, \gamma)$ respectively.

Let C_π be the set of cycles in $G(\pi, \gamma)$. Define the interleaving graph $H(\pi, \gamma) = (C_\pi, I_\pi)$ of a signed permutation π with respect to γ of vertex set C_π and edge set $I_\pi = \{(C_1, C_2) :$

C_1, C_2 *are interleaving cycles in* $G(\pi,\gamma)\}$.

The *interval of cycle* C in $G(\pi,\gamma)$ is the interval $[C.B, C.E]$, where $C.B = min\{i : \pi_i \in C\}$ and $C.E = max\{i : \pi_i \in C\}$. The *interval of a connected component* U in $H(\pi,\gamma)$ is the interval $[U.B, U.E]$, where $U.B = min\{C.B : C \in U\}$ and $U.E = max\{C.E : C \in U\}$.

Notice that every reversal acts on two black edges and decreases or increases $c(\pi,\gamma)$ by at most 1. A reversal is said to be *proper* if it increase the number of cycles by exactly 1, and said to be *acting on a gray edge* g if it acts on the two black edges incident to g. We say that a gray edge is *oriented* if a reversal acting on it is proper, and a cycle is *oriented* if it contains an oriented edge. A connected component of $H(\pi,\gamma)$ that contains an oriented cycle is called an *oriented component*, otherwise it is called an *unoriented component*.

The following conclusion is obvious.

Lemma 1 *(1)An edge e is oriented iff* $e.E - e.B$ *is even; (2) The number of oriented edges in an oriented cycle is even.*

Let Ψ be a collection of sets of integers. Define a partial order on Ψ by the rule $\prec$: $U \prec W$ iff $[U.B, U.E] \subset [W.B, W.E]$ for $U, W \in \Psi$. We say a set $U \in \Psi$ *separates* two sets U' and U'' if there exists $u \in U$ such that $U'.E < u < U''.B$. A *hurdle* for Ψ is an element U in Ψ that satisfies either (1) U is the minimal elements in Ψ under relation $\prec$, or (2)U is the maximum element in Ψ under relation $\prec$, and U does not separate any two sets in Ψ. A hurdle satisfying condition (1) is called *minimal hurdle*, and that satisfying condition (2) is called *maximum hurdle*. A hurdle U is called a *superhurdle* if there is a non-hurdle U_0 in Ψ such that U_0 is a hurdle in $\Psi \setminus \{U\}$. Otherwise, a *simple hurdle*.

Let $\overline{U} = \{i : \pi_i \in C, C \in U\}$ be the set of positions of π belonging to cycles of a connected component U in $H(\pi,\gamma)$. $\Delta(\pi,\gamma) = \{\overline{U} : U$ is an unoriented component of $H(\pi,\gamma)\}$. The signed permutation π is a *fortress* with respect to γ if $\Delta(\pi,\gamma)$ has odd number of hurdles, and all of which are superhurdles. Denote by $h(\pi,\gamma)$ the number of hurdles in $\Delta(\pi,\gamma)$. Let $f(\pi,\gamma) = 1$ if π is a fortress with respect to γ, and $f(\pi,\gamma) = 0$ otherwise.

2.3 Transformations on multi-chromosomal genomes

2.3.1 SBRT-limited to internal reversals and translocations

Given two multi-chromosomal genomes $\Pi = \{\pi(1), \pi(2), \ldots, \pi(N)\}$, a *concatenation* of Π is defined by $\pi = \{\overline{\pi(i_1)}, \overline{\pi(i_2)}, \ldots, \overline{\pi(i_N)}\}$, where $(i_1, i_2, \ldots, i_N)$ is a permutation on $\{1, 2, \ldots, N\}$ and $\overline{\pi(i_k)} = \pi(i_k)$ or $-\pi(i_k)(1 \leq k \leq N)$.

Given two multi-chromosomal genomes Π and Γ, the genomic distance between Π and Γ, denoted by $d(\Pi,\Gamma)$, is the minimum number of reversals, translocations, fissions and fusions required to transform Π into Γ.

Let Π and Γ be two co-tailed genomes with n genes, each with N chromosomes. Let π and γ be two arbitrary concatenations of the chromosomes in Π and Γ respectively. Let $G(\pi,\gamma)$ (resp. $H(\pi,\gamma)$) be the cycle (resp. interleaving) graph of π w.r.t. γ. An edge in $G(\pi,\gamma)$ is called *intrachromosomal* if it connects two vertices from the same chromosome in Π and *interchromosomal* otherwise. A cycle in $G(\pi,\gamma)$ is called *interchromosomal* if it contains an interchromosomal edge, and *intrachromosomal* otherwise.

There are $N-1$ black edges between tails of Π and 2 black edges originating from vertices 0 and $2n+1$ in $G(\pi,\gamma)$. These $N+1$ black edges define the concatenation π. Equivalently, there are $N+1$ gray edges that define the concatenation γ. Let $G(\Pi,\Gamma)$ be the graph obtained from by removing the $2(N+1)$black and gray edges that constitute

cycles in $G(\Pi, \Gamma)$. Therefore, $G(\Pi, \Gamma)$ is also composed of cycles. Define the interleaving graph of Π w.r.t. Γ, denoted by $H(\Pi, \Gamma)$, as the subgraph of $H(\pi, \gamma)$ induced by the set of intrachromosomal cycles in $G(\Pi, \Gamma)$ in the same way as previous.

Let $b(\Pi, \Gamma)$ and $c(\Pi, \Gamma)$ be the number of black edges and cycles in $G(\Pi, \Gamma)$ respectively. A component of $H(\Pi, \Gamma)$ is *interchromosomal* if it contains an interchromosomal edge, and *intrachromosomal* otherwise. Consider the set of intrachromosomal unoriented components $IU(\Pi, \Gamma)$ in $H(\Pi, \Gamma)$. Let $\Delta(\Pi, \Gamma) = \{\overline{U} : U \in IU(\Pi, \Gamma)\}$. Then hurdles, superhurdles and fortresses for the collection $\Delta(\Pi, \Gamma)$ are called *knots*, *super-knots*, and *fortresses-of-knots* respectively. Let $k(\Pi, \Gamma)$ be the number of knots in $\Delta(\Pi, \Gamma)$(or $IU(\Pi, \Gamma)$). Define $f(\Pi, \Gamma) = 1$ if $\Delta(\Pi, \Gamma)$ is a fortress-of-knots and $f(\Pi, \Gamma) = 0$ otherwise.

Theorem 1 *[11] For co-tailed genomes Π and Γ, formula (1) gives the genomic distance between Π and Γ, i.e., the minimum number of internal reversals and translocations required to transform Π into Γ,*

$$d(\Pi, \Gamma) = b(\Pi, \Gamma) - c(\Pi, \Gamma) + k(\Pi, \Gamma) + f(\Pi, \Gamma). \qquad (1)$$

2.2.2 SBRT-the general case

In the general case, let $\Pi = \{\pi(1), \pi(2), \ldots, \pi(M)\}$, $\Gamma = \{\sigma(1), \sigma(2), \sigma(N)\}$ be two signed genomes, M and N be the number of chromosomes in Π and Γ respectively, the tail set $T(\Pi)$ of Π contains $2M$ elements, and the tail set of Γ contains $2N$ elements. $d(\Pi, \Gamma) = d(\Gamma, \Pi)$ since every rearrangement is reversible. Therefore, we can assume without loss of generality that $M \leq N$. If $M < N$, we can extend Π with $N - M$ empty chromosomes, so we may assume that Π and Γ have the same number of chromosomes.

Let $\{c_0, c_1, \ldots, c_{2M-1}\}$ be a set of $2M$ distinct positive integers (called caps) that are different from the genes in Π. A capping of Π, denoted by $\hat{\Pi} = \{\hat{\pi}(1), \hat{\pi}(2), \ldots, \hat{\pi}(M)\}$, is a genome obtained from Π by adding caps to the ends of each chromosome, i.e. $\hat{\pi}(k) = \{c_{2(k-1)}, \pi(k)_1, \pi(k)_2, \ldots, \pi(k)_{n_k}, c_{2(k-1)+1}\}, 1 \leq k \leq M$. A capping of Γ can be defined analogously.

Let $\hat{\pi}$ and $\hat{\gamma}$ be two arbitrary concatenations of cappings $\hat{\Pi}$ and $\hat{\Gamma}$. Let $G(\hat{\pi}, \hat{\gamma})$ be the cycle graph of $\hat{\pi}$ w.r.t. $\hat{\gamma}$. Let $G(\hat{\Pi}, \hat{\Gamma})$ be the graph obtained from $G(\hat{\pi}, \hat{\gamma})$ by removing the $2(N + 1)$ gray and black edges which define the concatenations of $\hat{\Pi}$ and $\hat{\Gamma}$. Let $G(\Pi, \Gamma)$ be the graph obtained from $G(\hat{\Pi}, \hat{\Gamma})$ by removing the $2N$ gray edges defined by between tails of Γ and caps which determine the capping $\hat{\Gamma}$ of Γ. $G(\Pi, \Gamma)$ has $4N$ vertices of degree 1 corresponding to the $2N$ caps of Π(called Π-caps) and $2N$ tails of Γ (called Γ-tails). Therefore $G(\Pi, \Gamma)$ is composed of cycles and $2N$ paths, each path starting and ending with a black edge. A path is called a $\Pi\Pi$-path ($\Gamma\Gamma$-path) if it starts and ends with Π-caps (Γ-tails) and a $\Pi\Gamma$-path if it starts with a Π-cap and ends with a Γ-tail.

Let $b(\Pi, \Gamma)$ be the number of black edges in $G(\Pi, \Gamma)$, and $c(\Pi, \Gamma)$ be the overall number of cycles, $\Pi\Pi$-paths and $\Pi\Gamma$-paths in $G(\Pi, \Gamma)$. Clearly, $b(\Pi, \Gamma) = n + N$. Closing a $\Pi\Gamma$-path($\Pi\Pi$-path or $\Gamma\Gamma$-path) results a cycle, the path is *oriented* if the corresponding cycle is oriented, and *unoriented* otherwise. We can easily get the the following lemma from Lemma 1:

Lemma 2 *Let C be a cycle ($\Pi\Pi$-path, $\Gamma\Gamma$-path or $\Pi\Gamma$-path) in $G(\Pi, \Gamma)$. Then it is oriented if and only if there exist an oriented edge in C.*

Interleaving cycles/paths in the graph are defined as in the section 2.2 by making no distinction between cycles and paths in $G(\Pi, \Gamma)$. Define the interleaving graph $H(\Pi, \Gamma)$

of Π w.r.t. Γ from $G(\Pi, \Gamma)$ in the same way as that of defined in subsection 2.2. The intervals of paths/cycles/connected components are defined in an analogous way.

Let $\omega(x) = 1$ if x is an oriented edge(cycle, or path or component), and let $\omega(x) = 0$ if x is an unoriented edge(cycle, or path or component). From definition of ω, Lemma 1 and Lemma 2, we have the following conclusion:

Lemma 3 *Let C be a cycle ($\Pi\Pi$-path, $\Gamma\Gamma$-path or $\Pi\Gamma$-path), D is a connected component in $G(\Pi, \Gamma)$. Then the following formulas hold:*

$$\omega(C) = max\{\omega(e) : e \text{ is a gray edge in } C\}, \omega(D) = max\{\omega(C) : C \text{ is a cycle/path on } D\}$$

An intrachromosomal component in $H(\Pi, \Gamma)$ is called *real* if there are no Π-caps or Γ-tails in its interval. Define $RU(\Pi, \Gamma)$ as the set of real unoriented components in $H(\Pi, \Gamma)$, which contains no 1-cycle. Hurdles, super-hurdles and fortresses for the set are called *real-knots*, *super-real-knots* and *fortresses-of-real-knots*. Let RK be the set of real-knots and denote by $r(\Pi, \Gamma)$ the number of real-knots in $RU(\Pi, \Gamma)$.

Notice that an intrachromosomal component involving only cycles in $G(\Pi, \Gamma)$ is not necessarily real. The intrachromosomal components involving only cycles are therefore called *pseud-real*. That involving a $\Pi\Gamma$-path but no other kinds of paths is called a $\Pi\Gamma$ *-component*, involving a $\Gamma\Gamma$-path but no $\Pi\Pi$- paths a $\Gamma\Gamma$-*component*, and involving a $\Pi\Pi$-path a $\Pi\Pi$-*component*.

Note that an intrachromosomal component whose interval contains a Π-cap must contain the Π-cap. Hence a $\Pi\Gamma$-path cannot be located in the interval of an intrachromosomal pseud-real component, whereas $\Gamma\Gamma$-path can. A $\Pi\Gamma$-component may contain a $\Gamma\Gamma$-path in its interval.

Given a cycle(path) C in $G(\Pi, \Gamma)$, and a component D in $H(\Pi, \Gamma)$. Let $g(C) = 0$ if C is a cycle, $g(C) = 1$ if C is a $\Pi\Gamma$-path, $g(C) = 2$ if C is a $\Gamma\Gamma$-path, and 3 a $\Pi\Pi$-path. Let $g(D) = 0$ if D is pseud-real, $g(D) = 1$ if D is a $\Pi\Gamma$-component, $g(D) = 2$ if D is a $\Gamma\Gamma$-component, and 3 a $\Pi\Pi$-component. The following lemma is obvious.

Lemma 4 $g(D) = max\{g(C) : C \text{ is a cycle or path on } D\}$.

As in section 2.3.1, define $IU(\Pi, \Gamma)$ as the set of intrachromosomal unoriented components in $H(\Pi, \Gamma)$. A component in $IU(\Pi, \Gamma) \setminus RU(\Pi, \Gamma)$ is a *semi-real-knot* if (i) it contains no a $\Gamma\Gamma$-paths in its interval, and (ii) closing all the $\Pi\Gamma$-paths in it creates a minimal real-knot or a simple (not super-real-knot) maximum real-knot. A semi-real note is called *minimal* if it turns into a minimal real-knot when all the $\Pi\Gamma$-paths in it are closed, *maximum* otherwise. Obviously, there is at most one maximum-semi-real-knot in $H(\Pi, \Gamma)$. From the definition of knot, a minimal-semi-real-knot is just a knot in $IU(\Pi, \Gamma)$, but a maximum-semi-real-knot is not always a knot (see fig.1, E is a maximum-semi-real-knot, but not a knot). Let s be the number of semi-real-knots in $IU(\Pi, \Gamma)$ including the maximum-semi-real-knot.

The interleaving graph $H(\Pi, \Gamma)$ is a *weak-fortress-of-real-knots* if (1) the number of real knots in $RU(\Pi, \Gamma)$ is odd, (2) one of the real-knots is the maximum-real-knot, (3) every real-knots but the maximum one is a super-real-knot and (4) the number of semi-real-knots in $H(\Pi, \Gamma)$ is at least one. A *simple component* is a component containing a $\Pi\Gamma$-path that is not a semi-real-knot.

Let $\overline{G}(\Pi, \Gamma)$ be a graph obtained from $G(\Pi, \Gamma)$ by closing all the $\Pi\Gamma$-paths in all the simple components in $H(\Pi, \Gamma)$, $\overline{H}(\Pi, \Gamma)$ a interleaving graph of $\overline{G}(\Pi, \Gamma)$, and $\overline{RU}(\Pi, \Gamma)$

a set of real unoriented components of $\overline{H}(\Pi, \Gamma)$. We define $gr = 1$ if there exists the maximum-real-knot in $\overline{RU}(\Pi, \Gamma)$ and $s > 0$; $gr = 0$ otherwise. We define $fr = 1$ if either (i)$\overline{RU}(\Pi, \Gamma)$ is a fortress-of-real-knots and there is no maximum-semi-real-knot in $\overline{H}(\Pi, \Gamma)$ or (ii)$\overline{RU}(\Pi, \Gamma)$ is a weak-fortress-of-real-knots; and $fr = 0$ otherwise.

Lemma 5 *There exists the maximum real-knot in $RU(\Pi, \Gamma)$ iff there exists the maximum real-knot in $\overline{RU}(\Pi, \Gamma)$; $\overline{H}(\Pi, \Gamma)$ is a weak-fortress of real-knot iff $H(\Pi, \Gamma)$ is a weak-fortress of real-knot.*

Proof: Suppose there exists the maximum real-knot U in $RU(\Pi, \Gamma)$, then there exists no maximum semi-real-knot in $H(\Pi, \Gamma)$. If there is a $\Pi\Gamma$-component W whose interval involving the interval of U, then W is the maximum real-knot in $\overline{RU}(\Pi, \Gamma)$. If there is no such a component, then U is the maximum real-knot in $\overline{RU}(\Pi, \Gamma)$.

Suppose there exists the maximum real-knot U in $\overline{RU}(\Pi, \Gamma)$. If U is not a real-knot in $RU(\Pi, \Gamma)$, then it is not a semi-real-knot in $H(\Pi, \Gamma)$. Hence U is a super real-knot in $\overline{RU}(\Pi, \Gamma)$, the maximum real-knot in $\overline{RU}(\Pi, \Gamma) \setminus \{U\}$ is the maximum real-knot in $RU(\Pi, \Gamma)$.

From the definition of $\overline{H}(\Pi, \Gamma)$, $H(\Pi, \Gamma)$ and $\overline{H}(\Pi, \Gamma)$ have the same set of semi-real-knots, and the same set of minimal real-knots. Hence Lemma 5 holds. □

We know from Lemma 5, $gr = 1$ iff there exists the maximum-real-knot in $RU(\Pi, \Gamma)$ and $s > 0$; $fr = 1$ iff either (i)$\overline{RU}(\Pi, \Gamma)$ is a fortress-of-real-knots and there is no maximum-semi-real-knot in $H(\Pi, \Gamma)$ or (ii) $RU(\Pi, \Gamma)$ is a weak-fortress-of-real-knots.

Theorem 2 *[12] Given two genomes Π and Γ, formula (2) gives the the genomic distance between Π and Γ, i.e., the minimum number of reversals, translocations, fusions and fissions required to transform Π into Γ,*

$$d(\Pi, \Gamma) = b(\Pi, \Gamma) - c(\Pi, \Gamma) + r(\Pi, \Gamma) + \lceil \frac{s(\Pi, \Gamma) - gr(\Pi, \Gamma) + fr(\Pi, \Gamma)}{2} \rceil \quad (2)$$

3 Linear algorithm to compute parameters in formula

3.1 Determining whether a maximum element U in Λ separates the set $\Lambda \setminus \{U\}$

In this subsection, we are going to design an algorithm that will be used to decide whether the maximum element U in a given partial set Λ will separate $\Lambda \setminus \{U\}$.

Let $\prec$ be a partial order on a set P. We say x is covered by y in P, if $x \prec y$ and there is no element z in P, such that, $x \prec z \prec y$. The *cover graph* of P is a directed graph with vertex set P and arc set $\{(x, y) : x, y \in P$ and x is covered by $y\}$.

Given a set Λ of connected components of interleaving graph for signed permutations π and γ, the partial relation is defined as that in section 2.2. Let U be the maximum elements under partial order $\prec$. We want to determine whether U separates the set $\Lambda \setminus \{U\}$. To do so, we will use $D[i]$ to denote the connected component involving position i, and introduce a parameter α to capture the information we need.

Algorithm 1:

Input: a set Λ of connected components in permutation π, the maximum element U in Λ

Output: parameter α

Step 0. Scan permutation π, each position i, such that $D[i] \in \Lambda$ is labelled by $D[i]$.

Step 1. Scan permutation π, to find the minimum number i_0 such that i_0 is labelled and $D[i_0] \in \Lambda \setminus \{U\}$, set $\alpha \leftarrow 0$.

Step 2. Scan the positions in the order $i_0+1, i_0+2, \ldots$, to find the minimum labelled number i_1 such that $D[i_1] = U$, set $\alpha \leftarrow \alpha + 1$.

Step 3. Scan the positions in the order $i_1+1, i_1+2, \ldots$, set $\alpha \leftarrow \alpha + 1$ if there is a position j such that $D[j] \in \Lambda \setminus \{U\}$; and $\alpha \leftarrow \alpha$ otherwise. Stop.

Theorem 3 *Algorithm 1 will be completed in $O(n)$ time, where n is the number of genes in π. The maximum element U separates $\Lambda \setminus \{U\}$ iff $\alpha = 2$. In particular, if Λ is a set of unoriented components, then the maximum element U is the maximum knot in Λ iff $\alpha = 1$.*

3.2 The linear algorithm in general case

Bader et al. [8] give a linear-time algorithm, named BMY here, which computes all the connected components of the interleaving graph of signed permutions. Based on BMY algorithm, we will design a linear algorithm to compute genomic distance between multi-chromosomal genomes, i.e., we need to compute the parameters in formula (2) in linear time.

Given genomes Π and Γ as in subsection 2.3.2. From the definition of $G(\Pi, \Gamma)$, the interleaving graph $H(\Pi, \Gamma)$ is independent of cappings of Π and Γ, and concatenations of their cappings. Assume without loss of generality that $\hat{\Pi} = \{\hat{\pi}(1), \hat{\pi}(2), \ldots, \hat{\pi}(N), \}$, $\hat{\Gamma} = \{\hat{\gamma}(1), \hat{\gamma}(2), \ldots, \hat{\gamma}(N), \}$,$\hat{\pi} = \hat{\pi}(1) + \ldots + \hat{\pi}(N)$,$\hat{\gamma} = \hat{\gamma}(1) + \ldots + \hat{\gamma}(N)$, where $\hat{\pi}(k) = \{c_{2(k-1)}, \pi(k)_1, \ldots, \pi(k)_{n_k}, c_{2(k-1)+1}\}$ and $\hat{\gamma}(k) = \{c_{2(k-1)}, \gamma(k)_1, \ldots, \gamma(k)_{m_k}, c_{2(k-1)+1}\}$,$1 \leq k \leq N$.

By making no distinction between paths and cycles in $G(\Pi, \Gamma)$, we can invoke BMY algorithm to compute all the components of the interleaving graph $H(\Pi, \Gamma)$. At the same time, we get all the informations required to compute the parameters in formula (2): intrachromosomal unoriented components and oriented components, pseud-real components, $\Pi\Gamma$-components, $\Gamma\Gamma$-components and $\Pi\Pi$-components.

Consider the concatenation $\hat{\pi}$ of capping $\hat{\Pi}$. The tails of $\hat{\pi}$ are in position 0 and position $2\sum_{j=1}^{k-1} n_j + 4N + 1$. For every k, the kth chromosome $\hat{\pi}(k)$ in $\hat{\Pi}$ takes $2n_k + 4$ positions: $2\sum_{j=1}^{k-1} n_j + 4(k-1) + 1, \ldots, 2\sum_{j=1}^{k} n_j + 4k$.

Phase 1

For each k, the vertex $v_i (2\sum_{j=1}^{k-1} n_j + 4(k-1) + 2 \leq i \leq 2\sum_{j=1}^{k-1} n_j + 4k - 1)$ in position i in the kth chromosome $\hat{\pi}(k)$ have one neighbor a_i in $\hat{\pi}$ ($\hat{\pi}$-neighbor) and one neighbor b_i in $\hat{\gamma}$($\hat{\gamma}$-neighbor) except for the two tails in $\hat{\pi}$. Scanning $\hat{\pi}$ and $\hat{\gamma}$, we can get the two neighbors. In particular, we suppose that $b_i = -$ (blank) if b_i is a π-cap or γ-tail in $G(\Pi, \Gamma)$. Define $c_i = 1$ if v_i is a Π-cap, $c = 2$ if v_i is a Γ-tail, and $c_i = 0$ otherwise. We label each position i, corresponding to a non-isolated vertice in $G(\Pi, \Gamma)$, with (v_i, a_i, b_i, c_i).

For each i, we use e_i to denote the gray edge involving position i, $C[i]$ to denote the cycle involving position i. Scanning the permutation $\hat{\pi}$ again, we can get all the gray edges, cycles and paths in $G(\Pi, \Gamma)$, we also get the value of parameter $c(\Pi, \Gamma)$ in formula (2). We can get the value $\omega(e_i)$ from $[e_i.B, e_i.E]$, get the value $g(C[i])$ from previous

labels for all positions. From Lemma 3, we can calculate $\omega(C[i])$. Label position i with $(C[i].B, C[i].E, \omega(C[i]), g(C[i]))$.

This phase can be completed in $O(n)$ time.

Phase 2

Now we invoke BMY algorithm to compute all the components of the interleaving graph $H(\Pi, \Gamma)$. For each i, let $D[i]$ be the connected component (may be a 1-cycle or an isolated edge)involving position i in $G[\Pi, \Gamma]$. From Lemma 3 and Lemma 4, we can calculate $\omega(D[i])$ and $g(C[i])$. Label position i with $(D[i].B, D[i].E, \omega(D[i]), g(D[i]))$.

This phase will be completed in $O(n)$ time.

Phase 3.

In order to compute all the reminder parameters in formula (2), we should to find out all the real-knots and semi-real-knots in $IU(\Pi, \Gamma)$, decide whether each real-knot is a super-real-knot and whether there exist the maximum real-knot or maximum semi-real-knot. We should know all real-knots in $\overline{RU}(\Pi, \Gamma)$, decide whether each real-knot is a super-real-knot. First of all,we need to know whether a pseud-real component is a real component. Hence we scan the set Φ_0 of all the unoriented pseud-real components,all the unoriented $\Pi\Gamma$-components, and all the $\Gamma\Gamma$-components, in order to find its subset Φ, which consists of all pseud-real components and $\Pi\Gamma$-components in $IU(\Pi, \Gamma)$, and all intrachromosomal $\Gamma\Gamma$-components in $H(\Pi, \Gamma)$. It is clearly that $\Phi_0 = \{D[i]|g(D[i]) = 0, \omega(D[i]) = 0, D.E \neq D.B + 1\} \cup \{D[i]|g(D[i]) = 1, \omega(D[i]) = 0\} \cup \{D[i]|g(D[i]) = 2\}$.

We can easily get the following Lemma:

Lemma 6 *For each $D \in IU(\Pi, \Gamma)$, D is a real-component iff (1) $D \in \Phi, g(D) = 0, \omega(D) = 0$; and (2)for each $D' \in \Phi$, such that $[D'.B, D'.E] \subseteq [D.B, D.E]$, we have $g(D') = 0$.*

In algorithm 2, we skip all vertices belonging to the interchromosomal component in Φ_0 or belonging to the components not included in Φ_0. The remainder vertices belong to the components in Φ. We want to produce the covering graph Ω on Φ. For each $x \in \Omega$, if it is a real-component($\Pi\Gamma$-component), and there is no another $y \in \Omega$, such y is covered by x, then x is called a *minimal real-component*(*minimal* Π, Γ-*component*). We can define *maximal real-components* and *maximal* $\Pi\Gamma$-*components* in the same way.

For each $D \in \Phi$, we use $\beta(D)$ to denote the number of components in Φ covered by D. For each k, we introduce the following subsets of Φ involving the vertices in the positions of the kth chromosome: we use $\overline{A}_k$ to denote the set of all minimal real-knots which are isolated in Ω; A'_k the set of all maximal real-components that are not minimal in Ω; $\overline{\overline{A}}_k$ the set of all minimal real-knots which are not isolated; $\overline{B}_k$ the set of all minimal semi-real-knots; B'_k the set of all maximal $\Pi\Gamma$-components that are not minimal. Clearly, all minimal real-components are minimal knots, all minimal $\Pi\Gamma$-components are minimal semi-real-knots. Hence we have the following conclusion:

Lemma 7 *The set of all minimal real-knots in $IU(\Pi, \Gamma)$ is $A = \bigcup_{k=1}^{N}(\overline{A}_k \cup \overline{\overline{A}}_k)$. The set of all minimal semi-real-knots in $IU(\Pi, \Gamma)$ is $B = \bigcup_{k=1}^{N} \overline{B}_k$. Hence the number of minimal real-knots is $r' = \sum_{k=1}^{N} |A_k|$, the number of minimal semi-real-knots is $s' = \sum_{k=1}^{N} |\overline{B}_k|$.*

For each i,we use $parent(D[i])$ to denote the element covering $D[i]$ in Ω. We use $g(D[i])$ to denote the current information about $D[i]$: $g(D[i]) = 0$ indicate that there exist only cycles on $D[i]$ and the current scanned subinterval of $D[i]$;$g(D[i]) = 1$ indicate that there exist $\Pi\Gamma$-path but no other kind path; and $g(D[i]) = 2$ indicate that there exist $\Gamma\Gamma$-path

but no ΠΠ-path. By implementing Algorithm 2, for each k, we can get $\overline{A}_k, \overline{\overline{A}}_k, A'_k, \overline{B}_k, \overline{\overline{B}}_k$ and the set A of all minimal real-components.

Algorithm 2
Input: permutation $\hat{\pi}$
Output: parent$D[i]$,$g(D[i])$,$\beta(D[i])$,for each $i.\overline{A}_k, \overline{\overline{A}}_k, A'_k, \overline{B}_k, B'_k$, for each k
Begin

1. scan the permutation, label each position i with $D[i].B$, set up $(D[i].B, D[i].E, \omega(D[i]), g(D[i]))$
2. initialize the parameters: $R \leftarrow -1, stack = \emptyset, r' \leftarrow 0, s' \leftarrow 0$
3. **for** $k \leftarrow 1$ **to** N **do**
   $L \leftarrow R+3, R \leftarrow R+2n_k+4, \overline{A}_k \leftarrow \emptyset, \overline{\overline{A}}_k \leftarrow \emptyset, A'_k \leftarrow \emptyset, \overline{B}_k \leftarrow \emptyset, B'_k \leftarrow \emptyset$
   for$i \leftarrow L$ **to** R **do**
   if $D[i] \notin \Phi_0$ **then** skip i
   else if $D[i].B < L$ or $D[i].E > R$ **then** skip i
   else if $i = D[i].B$ **then** push $D[i]$ and $\beta(D[i]) \leftarrow 0$
   else
   if $i = D[i].E$ **then** pop $D[i]$
   (a) **if** $(stack = \emptyset) \wedge (\beta(D[i]) = 0)$ **then**
      i. **if** $g(D[i]) = 0$ **then** $(\overline{A}_k \leftarrow \overline{A}_k \cup \{D[i]\}, r' \leftarrow r'+1)$
      ii. **else if** $g(D[i]) = 1$ **then** $(\overline{B}_k \leftarrow \overline{B}_k \cup \{D[i]\}, s' \leftarrow s'+1)$
      iii. **else** skip i
   (b) **else if** $stack \neq \emptyset$ **then**
      i. $parent(D[i]) \leftarrow top, g(top) \leftarrow max\{g(top), g(D[i])\}$, $\beta(top) \leftarrow \beta(top)+1$
      ii. **if** $\beta(D[i]) = 0$ **then**
         A. **if** $g(D[i]) \neq 0$ **then** skip i
         B. **else if** $g(top) = 0$ **then** $(\overline{\overline{A}}_k \leftarrow \overline{\overline{A}}_k \cup \{D[i]\}, r' \leftarrow r'+1)$
         C. **else** $(\overline{A}_k \leftarrow \overline{A}_k \cup \{D[i]\}, r' \leftarrow r'+1)$
      iii. **else**
         if $g(D[i]) = 0$ and $g(top) \neq 0$ **then** $A'_k \leftarrow A'_k \cup \{D[i]\}$
         else skip i
   (c) **else**
      i. **if** $g(D[i]) = 0$ **then** $A'_k \leftarrow A'_k \cup \{D[i]\}$
      ii. **else if** $g(D[i]) = 1$ **then** $B'_k \leftarrow B'_k \cup \{D[i]\}$
      iii. **else** skip i
4. **return** parent $D[i]$,$g(D[i])$,$\beta(D[i])$, for each $i.\overline{A}_k, \overline{\overline{A}}_k, A'_k, \overline{B}_k, \overline{\overline{B}}_k$, for each k

End
The validity of Algorithm 2 and the following theorem are clearly.

Theorem 4 *(1)The output of Algorithm 2 satisfies:*

$$g(D) = max\{g(D') : [g(D').B, g(D').E] \subseteq [g(D).B, g(D).E], D' \in \Omega\}.$$

And $D[i]$ is real-component iff $g(D[i]) = 0$, $D[i]$ is a $\Pi\Gamma$ -component iff $g(D[i]) = 1$ and $D[i]$ is a $\Gamma\Gamma$-component iff $g(D[i]) = 2$

We can regard $g(D)$ as the coloring of $D \in \Omega$. Graph Ω_2 obtained by delating all the vertices colored 1 or 2 in Ω is the covering graph of $RU(\Pi, \Gamma)$. And the graph obtained by delating all the vertices colored 2 in Ω is denoted by Ω_1. Let G be the covering graph of $\overline{RU}(\Pi, \Gamma)$, then $\Omega \subseteq G \subseteq \Omega_1$. We can easily get from Lemma 5 that the minimal set of real-knot in $RU(\Pi, \Gamma)$ is still A. We still use $\beta(D)$ to denote the number of children of D in G. For each G, $\Omega \subseteq G \subseteq \Omega_1$, if the number of real knot in G is odd; and for each $D \in A, \beta(parent(D)) = 1$, then we say that G has property (F). It is clear to see the following result.

Theorem 5 *(1)A minimal real-knot D in $RU(\Pi, \Gamma)$ is a super-real-knot iff $\beta(D) = 1$ in Ç"; a minimal real-knot D in $\overline{RU}(\Pi, \Gamma)$ is a super-real-knot iff $\beta(D) = 1$ in the covering graph of $\overline{RU}(\Pi, \Gamma)$.*

(2)If there is no maximum real-knot in $H(\Pi, \Gamma)$, then $\overline{RU}(\Pi, \Gamma)$ is a fortress of real-knots iff $RU(\Pi, \Gamma)$ is a fortress of real-knots.

We use Ω_0 to denote the subgraph of Ω_1 obtained by delating all isolated vertices coloring 1. We want to decide whether there exist the maximum real-knot and maximum semi-real-knot in $H(\Pi, \Gamma)$, whether $\overline{RU}$ is a fortress of real-knots or $H(\Pi, \Gamma)$ is a weak fortress of real-knots. Then we can get all parameters in formula (2).

Case 1. For each k, $A_k = \emptyset$.

In this case: $\Pi = \Gamma$,$d(\Pi, \Gamma) = 0$.

Case 2. There exists at least two k, such that $A_k \neq \emptyset$.

There is no maximum real-knot and no maximum semi-real-knot in $H(\Pi, \Gamma)$, $s = s', r = r', gr = 0$. $H(\Pi, \Gamma)$ is not a weak-fortress-of-real-knots. Ω_0 is the covering graph of $\overline{RU}(\Pi, \Gamma)$. Hence if Ω_0 has property (F), then $\overline{RU}(\Pi, \Gamma)$,$fr = 1$. Otherwise $fr = 0$.

The genomic distance can be computed in this case.

Case 3. There exists exactly one k, such that $A_k \neq \emptyset$.

Subcase 3.1 $A'_k = \emptyset$, i.e., there is no maximal element in Ω_2.

There is no maximum real-knot in $H(\Pi, \Gamma)$, hence $H(\Pi, \Gamma)$ is not a weak-fortress-of-real-knots. Ω_0 is the covering graph of $H(\Pi, \Gamma)$. It is easy to see that $\overline{RU}(\Pi, \Gamma)$ is not a fortress-of-real-knots. $r = r', gr = 0, fr = 0$.

If (1)$B'_k = \{U\}$ and $U = parent(D)$, for any $D \in A_k$, and (2) execute Algorithm 1 on set $N[U]$, we have $\alpha = 1$. Then U is the maximum semi-real-knot in $H(\Pi, \Gamma)$ and $s = s' + 1$.

Otherwise, $s = s'$.

We can compute the genomic distance in this case.

Subcase 3.2 $A'_k = \{U_1\}$.

(1)$A_k \neq \emptyset$, i.e., there exists some minimal real-knots in $H(\Pi, \Gamma)$, which are isolated vertices in Ω_2. There is no maximum real-knot. Hence $r = r', gr = 0$ and $H(\Pi, \Gamma)$ is not a weak-fortress-of-real-knots.

(1.1)If (i).$B'_k = \{U\}$ and $U = parent(D)$, for any $D \in \overline{A_k} \cup \{U_1\}$, and (ii) execute Algorithm 1 on the set $\Lambda = N[U]$,we have $\alpha = 1$.

Then U is a maximum semi-real-knot and $s = s' + 1$. Ω_2 is the covering graph of $\overline{RU}(\Pi, \Gamma)$. Clearly,$\overline{RU}(\Pi, \Gamma)$ is not a fortress of real-knots,$fr = 0$.

(1.2)In all other case, there is no maximum semi-real-knot in $H(\Pi, \Gamma)$,$s = s'$. Ω_0 is the covering graph of $\overline{RU}(\Pi, \Gamma)$. Hence if G_0 has property (F), then $\overline{RU}(\Pi, \Gamma)$ is a fortress of real-knots, $fr = 1$. And $fr = 0$ otherwise.

We can compute the genomic distance in this case.

(2) $\overline{A}_k = \emptyset$, i.e., there exists no isolated minimal real-knot in Ω.

We use Algorithm 1 on the set $\Lambda = N[U_1]$.

(2.1)If $\alpha = 1$, then U_1 is a maximum real-knot in $H(\Pi, \Gamma)$. There is no maximum semi-real-knot in $H(\Pi, \Gamma)$. Hence $r = r' + 1, s = s'$, and Ω_0 is the covering graph of $\overline{RU}(\Pi, \Gamma)$.

(i).If $s > 0$ then $gr = 1$ else $gr = 0$.

(ii). If $s > 0$ and Ω_2 has property (F), then $H(\Pi, \Gamma)$ is a weak-fortress-of-real-knots, $fr = 1$.

(iii). If G_0 has property (F),but (ii) is not hold. Then $H(\Pi, \Gamma)$ is not a weak-fortress-of-real-knots

A. If $B'_k \neq \emptyset$ then $|B'_k| = 1$. Let $B'_k = \{U\}$, then $U = parent(U_1)$, U is the maximum super-real-knot in $\overline{RU}(\Pi, \Gamma)$. $\overline{RU}(\Pi, \Gamma)$ is a fortress-of-real-knots, $fr = 1$.

B.$B'_k = \emptyset$.

a. If U_1 has only one child U_2 in Ω_2, and we get $\alpha = 1$ after executed Algorithm 1 on the set $\Lambda = N[U_2]$.

Then U_2 is a maximum real-knot in $RU(\Pi, \Gamma) \setminus \{U_1\}$. Hence U_1 is a super-real-knot and $fr = 1$.

b. Otherwise, $fr = 0$.

(iii) In all other case, $fr = 0$.

(2.2)If $\alpha = 2$ then U_1 is not a maximum real-knot. $H(\Pi, \Gamma)$ is not a weak-fortress-of-real-knots.$r = r', gr = 0$. $\overline{RU}(\Pi, \Gamma)$ has no maximum real-knot by Lemma 5. Ω_0 is the covering graph of $\overline{RU}(\Pi, \Gamma)$. Therefore, if Ω_0 has property (F), $\overline{RU}(\Pi, \Gamma)$ is a fortress-of-real-knots, $fr = 1$. And $fr = 0$ otherwise.

(i). If $B'_k \neq \emptyset$ then $|B'_k| = 1$. Let $B'_k = \{U\}$, then $U = parent(U_1)$, U is the maximum semi-real-knot in $H(\Pi, \Gamma)$, and $s = s' + 1$.

(ii). If $B'_k = \emptyset$ then there is no maximum semi-real-knot in $H(\Pi, \Gamma)$. $s = s'$.

Subcase 3.3 $|A'_k| \geq 2$. There is no maximum real-knot in $H(\Pi, \Gamma)$, $r = r', gr = 0$ and $H(\Pi, \Gamma)$ is not a weak fortress of real-knot. By Lemma 5, there is no maximum real-knot in $\overline{RU}(\Pi, \Gamma)$.

(1).If (a)$B'_k = \{U\}$ and $U = parent(D)$, for any $D \in \overline{A}_k \cup A'_k$; and (b)we have $\alpha = 1$ after complete execute Algorithm 1 on the set $\Lambda = N[U]$. Then U is a maximum semi-real-knot in $H(\Pi, \Gamma)$, $s = s' + 1$. Ω_2 is the covering graph of $\overline{RU}(\Pi, \Gamma)$.

$\overline{A}_k \neq \emptyset$.$\overline{RU}(\Pi, \Gamma)$ is not a fortress of real-knots, $fr = 0$.

$\overline{A}_k = \emptyset$. If Ω_2 has property (F), then $\overline{RU}(\Pi, \Gamma)$ is a fortress of real-knots, $fr = 1$. $fr = 0$ otherwise.

(2). (1)is not hold, then $s = s'$. Ω_0 is the covering graph of $\overline{RU}(\Pi, \Gamma)$. Therefore, if Ω_0 has property (F), $\overline{RU}(\Pi, \Gamma)$ is a fortress-of-real-knots, $fr = 1$. And $fr = 0$ otherwise.

We can compute the genomic distance in this case.

The above discussion can be described as the following algorithm.

Algorithm 3
Input: Covering graph Ω, parent $D[i]$, for each i. $\overline{A}_k, \overline{\overline{A}}_k, A'_k, \overline{B}_k, B'_k$,$1 \leq k \leq N$
Output: genomic distance d
Begin

1. **if** $\forall k(A_k = \emptyset)$ **then** $r = s = gr = fr = 0$
2. **else if** there exists at least two k, s.t.,$A_k \neq \emptyset$ **then**

(a) $s = s', r = r', gr = 0$

(b) **if** Ω_0 has property (F) **then** $fr = 1$ **else** $fr = 0$

3. **else**

(a) **if** $A'_k = \emptyset$ **then**
(1) $r = r', gr = 0, fr = 0$
(2) **if** $((B'_k = \{U\}) \wedge (U = parent(D), \forall D \in A_k) \wedge$ (the output of Algorithm 1 on $\Lambda = N[U]$ is $\alpha = 1))$
then $s = s' + 1$ **else** $s = s'$

(b) **else if** $A'_k = \{U_1\}$ **then**
(1)**if** $\overline{A}_k \neq \emptyset$ **then**

i. $r = r', gr = 0$

ii. **if** $((B'_k = \{U\}) \wedge (U = parent(D), \forall D \in \overline{A}_k \cup \{U_1\}) \wedge$ (the output of Algorithm on $\Lambda = N[U]$) is $\alpha = 1))$
then $(s = s' + 1, fr = 1)$
else
A. $s = s'$
B. **if** Ω_0 has property (F) **then**$fr = 1$ **else** $fr = 0$

(2)**else** execute Algorithm 1 on $\Lambda = N[U_1]$

i. **if** $\alpha = 1$ **then**
A. $r = r' + 1, s = s'$
B.**if** $s > 0$ **then**
a. $gr = 1$
b. **if**Ω_2 has property (F)) **then** $fr = 1$
else
a. $gr = 0$
b. **if** Ω_0 has property (F) **then**
b1. **if** $B'_k \neq \emptyset$ **then** $fr = 1$
b2. **else if** $(B'_k = \emptyset) \wedge (N(U_1) = \{U_2\}) \wedge$ (the output of Algorithm 1 on $\Lambda = N[U_2]$ is $\alpha = 1)$
b3. **then**$fr = 1$
b4. **else** $fr = 0$

ii. **else**
A.$r = r', gr = 0$
B.**if** Ω_2 has property (F)**then** $fr = 1$ **else** $fr = 0$
C.**if** $B'_k \neq \emptyset$ **then** $s = s' + 1$ **else** $s = s'$

(c) **else**
(1) $r = r'.gr = 0$
(2) **if** $(B'_k = \{U\}) \wedge (U = parent(D), \forall D \in \overline{A}_k \cup A'_k) \wedge$(the output of Algorithm 1 on $\Lambda = N[U]$ is $\alpha = 1)$
then

i. $s = s' + 1$

ii. **if** $(A_k = \emptyset) \wedge (\Omega_2$ has property (F)) **then** $fr = 1$ **else** $fr = 0$

else

i. $s = s'$

ii. **if** Ω_0 has property (F)**then** $fr = 1$ **else** $fr = 0$

4. compute the genomic distance by formula (2) in Theorem 2

End

It is clear to see that Algorithm 3 can be completed in $O(n)$, we can easily get the following result:

Theorem 6 *The genomic distance can be computed in the general case in $O(n)$ time.*

3.3 The special case –cotailed genomes

Similar to section 3.2, we can get the linear time algorithm to compute all the parameters in formula (1), and get the following result. We omit the detail discussion since it is quite simple in this case.

Theorem 7 *The genomic distance can be computed in $O(n)$ time for co-tailed genomes.*

4 Conclusion

In this paper we present a simple, linear-time algorithm to compute the genomic distance between two signed multi-chromosomal genomes. The algorithm improves the complexity of the algorithm, whose running time is , described in [11, 12]. It is important to the study of multi- chromosomal genome evolution.

References

[1] Sankoff D, Edit distance for genomes comparison based on non-local operation. *Proc. 3rd Ann. Symp. Combinatorial Pattern Matching,* LNCS 1992,644:121-135,.

[2] Sankoff D, Leduc G, Antoine N, Paquin B, Lang B and Cedergren R, Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome, *Proc. Nat. Sci. USA* 1992,89:6575-6579.

[3] Kececioglu J and Sankoff D, Exact and approximation algorithms for the reversal distance between two permutation, *Algorithms* 1995,13(1/2): 180-210.

[4] Bafna V and Pevzner P, Genome rearrangements and sorting by reversals, *SIAM J. Comput.*, 25:272-289.1996.

[5] Hannenhalli S and Pevzner P, Transforming cabbage into turnip (Polynomial algorithm for sorting sighed permutations by reversals). *In Proc.27th Annual ACM Symposium on the theory of Computing*, 1995a, 178-189.

[6] Berman P and Hannenhalli S, Fast sorting by reversals, *Proc.7th Annual Symposium on Combinatorial Pattern Matching (CPM'96)*(Berlin), Springer, 1996, LNCS 1075,pp.168-185.

[7] Kaplan H, Shamir R and Tarjan R, Faster and simpler algorithm for sorting signed permutations by reversals, SIAM Journal of Computing 1999,29(3): 880-892

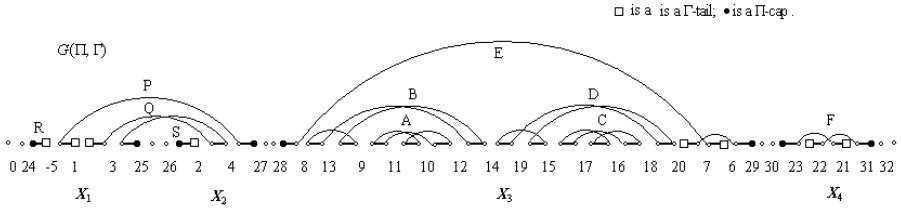
[8] Bader D, Moret B and Yan M, A linear-time algorithm for computing inversion distance between signed permutations with experimental study, *J. comput. Biol.*,2001,8:483-491.

[9] Kececioglu J and Ravi R, Of mice and men: evolutionary distances between genimes under translocation, *Proc. 6th Ann. ACM-SIAM Symp. on Discrete Algorithms*1995: 604 613.

[10] Hannenhali S, Polynomial-time algorithm for computing translocation distance between genomes, *Discrete Applied Mathematics*,1996,71: 137-151.

[11] S. Hannenhali and P. Pevzner, Transforming men into mice-polynomial algorithm for computing genomic distance problem. *Proc. 36th IEEE Symposium on Foundations of Computer Science* , 581-592,1995.

[12] Shamir R and Pzery-Flato M, Two notes on genome rearrangement. *Journal of Bioinformatics and Computational Biology* 2003, 1(1): 71-94



Genomes $\Pi = \{(-5,1,3),(2,4),(8,13,9,11,10,12,14,19,15,17,16,18,20,7,6),(23,22,21)\}$,

$\Gamma = \{(1),(2,3,4,5),(6,7,8,9,10,11,12,13,14,15,16,17,18,19,20),(21,22,23)\}$

Capppings $\hat{\Pi} = \{(24,-5,1,3,25),(26,2,4,27),(28,8,13,9,11,10,12,14,19,15,17,16,18,20,7,6,29),(30,23,22,21,31)\}$,

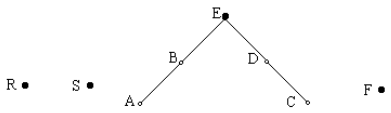
$\hat{\Gamma} = \{(24,1,25),(26,2,3,4,5,27),(28,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,29),(30,21,22,23,31)\}$

Concatenates $\hat{\pi} = (24,-5,1,3,25,26,2,4,27,28,8,13,9,11,10,12,14,19,15,17,16,18,20,7,6,29,30,23,22,21,31)$,

$\hat{\gamma} = (24,1,25,26,2,3,4,5,27,28,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,29,30,21,22,23,31)$

P,Q are inter-chromosomal components; A, B,C,D,E,F,R,S are intra-chromosomal components. E,F,P,Q,R,S are ΠΓ-components; A, B, C, D are pseu-real components, also are real- components. E,F,R,S are ΠΓ-components. Closing the ΠΓ-path in E, it becomes a greatest real-knot in G(Π,Γ). E is a greatest semi-real-knot, but it is not a knot. F,R,S are minimal semi-real-knots. A, C are real-knot, also are super-real-knots.

Covering graph $\Omega = \Omega_1$



R,S,F are minimal semi-real knots, they are are colord by '1'; A,C are minimal real-knots, also are super-real-knots,they are colored by '0'. B,D are not real-knots, they are also colored by '0'. There exists no greatest real-knot in this example.

Picture 1