TWO NOTES ON GENOME REARRANGEMENT

MICHAL OZERY-FLATO AND RON SHAMIR

Abstract. A central problem in genome rearrangement is finding a most parsimonious rearrangement scenario using certain rearrangement operations. An important problem of this type is sorting a signed genome by reversals and translocations (SBRT). Hannenhalli and Pevzner presented a duality theorem for SBRT which leads to a polynomial time algorithm for sorting a multi-chromosomal genome using a minimum number of reversals and translocations. However, there is one case for which their theorem and algorithm fail. We describe that case and suggest a correction to the theorem and the polynomial algorithm.

The solution of SBRT uses a reduction to the problem of sorting a signed permutation by reversals (SBR). The best extant algorithms for SBR require quadratic time. The common approach to solve SBR is by finding a safe reversal using the overlap graph or the interleaving graph of a permutation. We describe a family of signed permutations which proves a quadratic lower bound on the number of affected vertices in the overlap/interleaving graph during any optimal sorting scenario. This implies, in particular, an $\Omega(n^3)$ lower bound for Bergeron's algorithm.

1. Introduction

Molecular biology has entered an era in which data are available on essentially the entire DNA sequence, known as the *genome*, of human and an increasing number of model organisms. The genome of an organism is organized in a set of chromosomes. Each chromosome can be viewed as an ordered sequences of genes. Each gene is associated with a sign, which represents its direction of transcription. Different organisms are found to be related if they share similar genes, which were inherited from a common ancestor. Point mutations and short insertions and deletions cause local changes during a genes' evolution. In contrast, evolution at the level of the genome proceeds by large scale operations, such as inversions and translocations, which rearrange the order and direction of genes along the genome. Hence, on that level of abstraction, assuming the genomes share the same set of genes with no duplications (paralogs), a genome can be represented by a signed permutation Π (possibly split into contiguous blocks corresponding to chromosomes) and an operation ρ applied to Π generates the genome $\Pi \cdot \rho$.

Given two genomes Π and Γ, which share the same set of genes, the problem of finding a shortest sequence of rearrangement operations allowed by the model, $\rho_1, ..., \rho_t$, such that $\Pi \cdot \rho_1 \cdots \rho_t = \Gamma$, is known as the *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.*

In the model we consider any gene is unique within a genome and is represented by an identification number (positive integer). A chromosome is orientation-less, therefore *flipping* a chromosome $X = (x_1, ..., x_k)$ into $-X = (-x_k, ..., -x_1)$ does not affect the chromosome it represents. Hence, a chromosome X is said to be *identical* to a chromosome Y iff either $X = Y$ or $X = -Y$. Genomes Π and Γ are said to be identical if their sets of chromosomes are the same.

A *reversal* $\rho(X, i, j)$, $1 \leq i \leq j \leq k$, transforms X into $X' = (x_1, ..., x_{i-1}, -x_j, ..., -x_i, x_{j+1}, ..., x_n)$. A reversal $\rho(X, i, j)$ is said to be *internal* if $1 < i \leq j < n$. When Π and Γ are two uni-chromosomal genomes which share the same set of genes, Π can be transformed into Γ using only reversals. The uni-chromosomal genomic sorting problem which allows only reversals is called the *reversal sorting problem* and the genomic distance is then called the *reversal distance*. The reversal sorting problem is equivalent to the problem of sorting a signed permutation by reversals (SBR).

SBR was studied intensively over the last decade. In 1994, Kececioglu and Sankoff [11] gave the first constant factor polynomial approximation algorithm for SBR and conjectured that the problem is NP-hard. SBR was further studied by Bafna and Pevzner [3] who introduced the notion of a breakpoint graph of a permutation. In 1995, Hannenhalli and Pevzner [7] gave a first polynomial algorithm for SBR. They proved a duality theorem which expresses the reversal distance in terms of four combinatorial parameters defined on the breakpoint graph and the interleaving graph of a signed permutation. Based on this theorem, they found an $O(n^4)$ algorithm for SBR. In 1996, Berman and Hannenhalli [5] described a faster implementation for this algorithm which runs in $O(n^2\alpha(n))$ time, where $\alpha(n)$ is the inverse of the Ackerman's function [1]. In 1997, Kaplan, Shamir and Tarjan [9] simplified HP's analysis by using the overlap graph and gave a quadratic time algorithm which has the best known running time so far. In 2001, Bergeron [4] described a very simple algorithm which relies directly on the overlap graph and runs in $O(n^3)$. In Section 5 we give a short review of the different algorithms for SBR and present a family of signed permutations for which every optimal sequence of reversals involves an overall change of a quadratic number of vertices in the overlap graph/interleaving graph. This implies, in particular, a quadratic lower time bound for all three algorithms.

All the algorithms above for SBR also provide the distance as a byproduct. Solving the reversal distance problem directly is in fact simpler. Berman and Hannenhalli [5] showed how to compute the reversal distance in $O(n\alpha(n))$. In 2001, Bader, Moret and Yan [2] presented a linear time algorithm for computing the reversal distance.

A *translocation* is a rearrangement that works on two chromosomes switching their "tails". Let X and Y be two chromosomes $X = (x_1, ..., x_k)$, $Y = (y_1, ..., y_m)$. A translocation

$\rho(X, Y, i, j)$, $1 \leq i \leq k+1$ and $1 \leq j \leq m+1$, exchanges segments of genes *between* the chromosomes X and Y and transforms them into the chromosomes $(x_1, ..., x_{i-1}, y_j, ..., y_m)$ and $(y_1, ..., y_{j-1}, x_i, ..., x_k)$. A translocation $\rho(X, Y, n+1, 1)$ concatenates the chromosomes X and Y resulting in a chromosome $(x_1, ..., x_k, y_1, ..., y_m)$ and an *empty* chromosome $\emptyset$. This special translocation reduces the number of (non-empty) chromosomes and is known in molecular biology as *fusion*. The translocation $\rho(X, \emptyset, i, 1)$ for $1 < i \leq k$ "breaks" a chromosome X into two chromosomes $(x_1, ..., x_{i-1})$ and $(x_i, ..., x_k)$. This translocation increases the number of (non-empty) chromosomes and is known as *fission*. A translocation $\rho(X, Y, i, j)$ is said to be *internal* if $X, Y \neq \emptyset$, $1 < i \leq n$ and $1 < j \leq m$. Fission and fusion are special cases of translocations which are not internal. Hannenhalli [6] has devised a polynomial algorithm for the genomic sorting problem when only internal translocations are allowed.

For a chromosome $X = (x_1, ..., x_k)$, the numbers $+x_1$ and $-x_k$ are called *tails* of X. Note that flipping a chromosome does not change the set of its tails. The set of tails of all the chromosomes in Π is denoted by $\mathcal{T}(\Pi)$. Genomes Π and Γ are *co-tailed* if $\mathcal{T}(\Pi) = \mathcal{T}(\Gamma)$. 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 to a target genome Γ, using internal translocations and/or reversals, is limited to genomes Π and Γ that are co-tailed.

Hannenhalli and Pevzner [8] studied the genomic sorting problem for reversals and translocations (SBRT). Given two genomes Π and Γ, which share the same set of genes, Hannenhalli and Pevzner reduced SBRT on Π and Γ into SBR on π and γ, where π and γ are two permutations for which any optimal sorting by reversals of π into γ mimics an optimal sorting by reversals and translocations of Π into a Γ. Note that the translocation $\rho(X, Y, i, j)$ acting on chromosomes $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_m)$, can be mimicked by the reversal $\rho(X - Y, i, n + (m - j + 1))$ acting on the concatenation of X with $-Y$. Based on the reduction from SBRT to SBR, HP presented a duality theorem for the genomic distance which leads to a polynomial time algorithm for computing a shortest series of rearrangements (reversals and translocations). Tesler recently provided a procedure for ordering the chromosomes (in case the genomes are co-tailed), gave a more symmetric presentation, and improved the efficiency of the HP algorithm [12]. However, there is one case for which the HP theorem, and subsequently the HP algorithm (and also Tesler's), is incorrect. We describe this case in Section 3, and present a revised duality theorem and a corresponding algorithm which handles the problem in Section 4.

2. Preliminaries

In this section we give the basic background needed to discuss SBR and SBRT. The exposition follows closely the HP theory, with slight changes and modifications. The reader is referred to the original papers [7, 9, 8, 12] for a fuller description and illuminating examples demonstrating the definitions.

2.1. **SBR.**

2.1.1. *The breakpoint graph.* Let $\pi = (\pi_1, ..., \pi_n)$ be an (unsigned) permutation of elements $1, ..., n$. Extend π by adding $\pi_0 = 0$ and $\pi_{n+1} = n+1$. Denote $i \sim j$ if $|i-j| = 1$. We call a pair of consecutive elements π_i and π_{i+1}, $0 \leq i \leq n$, of π a *breakpoint* if $\pi_i \nsim \pi_{i+1}$ and an *adjacency* if $\pi_i \sim \pi_{i+1}$. The *breakpoint graph* $G(\pi)$ of π is an edge-colored graph on $n+2$ vertices $\{\pi_0, \pi_1, ..., \pi_n, \pi_{n+1}\}$. We join vertices π_i and π_{i+1} by a black edge for $0 \leq i \leq n$. We join vertices i and $i+1$ by a gray edge for $0 \leq i \leq n$.

We define a one-to-one mapping u from the set of signed permutations of order n into the set of unsigned permutations of order $2n$ as follows. Let $\vec{\pi}$ be a signed permutation of elements $\{1, ..., n\}$. To obtain $u(\vec{\pi})$ we replace each positive element $+x$ in π by $2x-1, 2x$, and each negative element $-x$ by $2x, 2x-1$. Every reversal $\rho(\vec{\pi}, i, j)$, $1 \leq i \leq j \leq n$, on $\vec{\pi}$ can be mimicked by the reversal $\rho(u(\vec{\pi}), 2i-1, 2j)$ on $u(\vec{\pi})$. Denote by $E_{self}(\vec{\pi})$ the set of $2n$ (black and gray) edges where each edge connects two vertices in $G(u(\vec{\pi}))$ that correspond to the same element in $\vec{\pi}$, i.e., $E_{self}(\vec{\pi}) = \{(2i-1, 2i) | 1 \leq i \leq n\}$. For any signed permutation $\vec{\pi}$, let $G(\vec{\pi}) = G(u(\vec{\pi})) \setminus E_{self}(\vec{\pi})$. Observe that in $G(\vec{\pi})$ every vertex is incident to exactly one black edge and one gray edge. Therefore, there is a unique decomposition of $G(\vec{\pi})$ into cycles. Moreover, the edges of each cycle are alternating gray and black.

Denote by $b(\vec{\pi})$ and $c(\vec{\pi})$ the number of black edges and cycles in $G(\vec{\pi})$ respectively. Note that in our definition of the breakpoint graph the number of black edges is fixed ($b = n+1$) and we allow cycles of length 2 (one black edge and one gray edge). For a parameter $\psi(\vec{\pi})$ and an arbitrary reversal ρ on $\vec{\pi}$ define $\Delta\psi(\vec{\pi}, \rho)$ as $\psi(\vec{\pi}\rho) - \psi(\vec{\pi})$. When the reversal ρ and the signed permutation $\vec{\pi}$ are clear from the context we will abbreviate $\Delta\psi(\vec{\pi}, \rho)$ to $\Delta\psi$.

Lemma 2.1. [3] *Every reversal on a signed permutation $\vec{\pi}$ satisfies $|\Delta b - \Delta c| \leq 1$.*

2.1.2. *The interleaving graph and the overlap graph of a signed permutation.* For an (unsigned) permutation we associate every gray edge, (π_i, π_j), in $G(\pi)$ with the interval $[i, j]$. Two gray edges *overlap* if the intersection of their associated intervals is nonempty but neither properly contains the other. Let $\vec{\pi}$ be a signed permutation. Cycles C_1 and C_2 in $G(\vec{\pi})$ are *interleaving* if there exist two gray edges $g_1 \in C_1$ and $g_2 \in C_2$ such that g_1 and g_2 overlap.

Define the *interleaving graph* of $\vec{\pi}$, denoted $INTL(\vec{\pi})$, as follows. The vertex set of $INTL(\vec{\pi})$ is the set of cycles in $G(\vec{\pi})$ and two vertices are connected if their cycles are interleaving. Define the *overlap graph* of $\vec{\pi}$, denoted $OV(\vec{\pi})$, as follows. The vertex set of $OV(\vec{\pi})$ is the set of gray edges in $G(\vec{\pi})$ and two vertices are connected if their gray edges overlap.

Lemma 2.2. [9] *Every connected component of $OV(\vec{\pi})$ corresponds to the set of gray edges of a union of cycles.*

Lemma 2.2 implies that there is a bijection between the connected components of $OV(\vec{\pi})$ and of $INTL(\vec{\pi})$.

Let $\pi = u(\vec{\pi})$ and let M be a connected component of $OV(\vec{\pi})$. Define $\bar{M} = \{i : \pi_i \in g \in M\}$. A component M is associated with the interval $[M_{\min}, M_{\max}]$, where $M_{\min} = \min_{i \in \bar{M}} i$ and $M_{\max} = \max_{i \in \bar{M}} i$.

2.1.3. *Hurdles.* Every reversal on a signed permutation $\vec{\pi}$ decreases $b - c$ by at most 1 (Lemma 2.1). Call a reversal *proper* if $\Delta b - \Delta c = -1$. Every reversal operates on two black edges. We say that a reversal ρ is *acting on* a gray edge g if it operates on the black edges incident to g. A gray edge is *oriented* if a reversal acting on it is proper, otherwise it is *unoriented*. A connected component of $OV(\vec{\pi})$ (equivalently, of $INTL(\vec{\pi})$) that contains an oriented edge is called an *oriented component*, otherwise it is called an *unoriented component*.

Let $\pi = u(\vec{\pi})$ and let $(\pi_{i_1}, \pi_{i_2}, ..., \pi_{i_k})$ be the subsequence of $(0, \pi_1, ..., \pi_{2n}, 2n+1)$ consisting of those elements incident to gray edges that occur in the unoriented components of $OV(\vec{\pi})$ (equivalently, $INTL(\vec{\pi})$). For an unoriented component M let $E(M) \subset \{\pi_{i_1}, \pi_{i_2}, ..., \pi_{i_k}\}$ be the set of elements incident to gray edges in M. Consider the circular order on $\pi_{i_1}, \pi_{i_2}, ..., \pi_{i_k}$ where π_{i_j} follows $\pi_{i_{j-1}}$ for $2 \leq j \leq k$ and π_{i_1} follows π_{i_k}. An unoriented component M is a *hurdle* if the elements in $E(M)$ occur consecutively in the circular order. A hurdle M is called the *greatest hurdle* if the elements in $E(M)$ do <u>not</u> occur consecutively in the linear order; otherwise it is called a *minimal hurdle.* A hurdle is *simple* if, when one deletes it from $OV(\vec{\pi})$, no other unoriented component becomes a hurdle, and it is a *super-hurdle* otherwise. A signed permutation $\vec{\pi}$ is a *fortress* if it has an odd number of hurdles, all of which are super-hurdles. Denote by $h(\vec{\pi})$ the number of hurdles in $OV(\vec{\pi})$. Let $f(\vec{\pi}) = 1$ if $\vec{\pi}$ is a fortress, $f(\vec{\pi}) = 0$ otherwise.

Theorem 2.3. [7] *The minimum number of reversals needed to sort a signed permutation $\vec{\pi}$ is $b(\vec{\pi}) - c(\vec{\pi}) + h(\vec{\pi}) + f(\vec{\pi})$.*

A reversal ρ on a signed permutation $\vec{\pi}$ is called *safe* if $\Delta b - \Delta c + \Delta h = -1$. If $\vec{\pi}$ is not a fortress then every optimal sorting sequence contains only safe reversals. If $\vec{\pi}$ is a fortress then at least one reversal, in any sorting scenario, is unsafe.

2.2. **SBRT limited to internal reversals and translocations.** Let Π and Γ be two co-tailed genomes with n genes. Denote by N the number of chromosomes in Π (or equivalently, Γ). Let π and γ be two arbitrary concatenations of the chromosomes in Π and Γ respectively. Note that there are $2^N N!$ possible concatenations for each genome. Let $G(\pi, \gamma)$ denote the breakpoint 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 *intrachromosomal* if it contains an intrachromosomal edge, and *interchromosomal* otherwise.

$G(\pi, \gamma)$ has $N - 1$ black edges between tails of Π (or equivalently, Γ) and 2 black edges originating from vertices 0 and $2n + 1$. Those $N + 1$ (interchromosomal) black edges define the concatenation π. Equivalently, there are $N + 1$ (interchromosomal) gray edges which define the concatenation γ. Let $G(\Pi, \Gamma)$ be the graph obtained from $G(\pi, \gamma)$ by removing the

$2(N+1)$ black and gray edges which define the concatenations π and γ. Those $2(N+1)$ black and gray edges compose cycles in $G(\pi,\gamma)$. Therefore, $G(\Pi,\Gamma)$ is also composed of cycles. Define the interleaving graph of Π w.r.t. Γ, $INTL(\Pi,\Gamma)$, as the subgraph of $INTL(\pi,\gamma)$ induced by the set of intrachromosomal cycles in $G(\Pi,\Gamma)$.

Let $b(\Pi,\Gamma)$ and $c(\Pi,\Gamma)$ be the number of black edges and cycles in $G(\Pi,\Gamma)$ respectively (notice that $b(\Pi,\Gamma) = n - N$). A component of $G(\Pi,\Gamma)$ (defined by $INTL(\Pi,\Gamma)$) is *interchromosomal* if it contains an interchromosomal edge, and *intrachromosomal* otherwise. Consider the set of intrachromosomal unoriented components $\mathcal{IU}(\Pi,\Gamma)$ in $G(\Pi,\Gamma)$. Hurdles, super-hurdles and fortresses for the set $\mathcal{IU}(\Pi,\Gamma)$ are called *knots*, *super-knots*, and *fortresses-of-knots* respectively. Let $k(\Pi,\Gamma)$ be the number of knots in $G(\Pi,\Gamma)$. Define $f(\Pi,\Gamma) = 1$ if $G(\Pi,\Gamma)$ is a fortress-of-knots and $f(\Pi,\Gamma) = 0$ otherwise.

Theorem 2.4. [8] *For co-tailed genomes,* Π *and* Γ*,* $d(\Pi,\Gamma) = b(\Pi,\Gamma) - c(\Pi,\Gamma) + k(\Pi,\Gamma) + f(\Pi,\Gamma)$*. In addition, there exists an optimal sorting of* Π *into* Γ *where every reversal / translocation is internal.*

Hannenhalli and Pevzner proved that there exist two concatenations, π and γ, of the chromosomes in Π and Γ respectively, such that an optimal sorting of π into γ mimics an optimal sorting of Π into Γ. However, they did not say how to construct such concatenations. In [12], Tesler provided a procedure which constructs two concatenations, π and γ, for which an optimal sorting of π into γ mimics an optimal sorting of Π into Γ with a minimum number of chromosome flips.

2.3. **SBRT - the general case.**

2.3.1. *Capping the chromosomes.* In the general case, genomes Π and Γ are not co-tailed and might have a different number of chromosomes. Let M and N be the number of chromosomes in Π and Γ respectively. $d(\Pi,\Gamma) = d(\Gamma,\Pi)$ since every rearrangement is reversible. Therefore, we can assume w.l.o.g. that $M \leq N$. If $M < N$ we extend Π with $N - M$ empty chromosomes so Π and Γ now have the same number of chromosomes.

Let $\{c_0, ..., c_{2N-1}\}$ be a set of $2N$ distinct positive integers (called *caps*) which are different from the genes in Π (or equivalently, Γ). A *capping* of Π, denoted $\hat{\Pi} = \{\hat{\pi}(1), ..., \hat{\pi}(N)\}$, is a genome obtained from Π by adding caps to the ends of each chromosome, i.e. $\hat{\pi}(i) = (c_{2(i-1)}, \pi(i)_1, ..., \pi(i)_{n_i}, c_{2(i-1)+1})$. Note that every reversal/translocation in Π corresponds to an internal reversal/translocation in $\hat{\Pi}$. A fission in Π, however, requires the existence of an empty chromosome. We shall prove later (in the proof of Theorem 4.1) that there is an optimal sorting of Π into Γ which does not require more than N chromosomes. Every sorting of Π into Γ that uses at most N chromosomes induces a sorting, which includes only internal operations, of $\hat{\Pi}$ into a genome $\hat{\Gamma} = \{\hat{\gamma}(1), ..., \hat{\gamma}(N)\}$, where $\mathcal{T}(\hat{\Gamma}) = \mathcal{T}(\hat{\Pi})$, i.e., $\hat{\gamma}(i) = ((-1)^j c_j, \gamma(i)_1, \ldots, \gamma(i)_{m_i}, (-1)^{k+1} c_k)$, for some $0 \leq j,k \leq 2N-1$. In other words, though Γ contains no capping, the operations on $\hat{\Pi}$ imply some capping on Γ resulting in $\hat{\Gamma}$.

Denote by Γ' the set of $(2N)!$ possible cappings of Γ.

Lemma 2.5. [8] $d(\Pi, \Gamma) = \min_{\hat{\Gamma} \in \Gamma'} b(\hat{\Pi}, \hat{\Gamma}) - c(\hat{\Pi}, \hat{\Gamma}) + k(\hat{\Pi}, \hat{\Gamma}) + f(\hat{\Pi}, \hat{\Gamma})$

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 breakpoint 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 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 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.

Define $b(\Pi, \Gamma)$ as the number of black edges in $G(\Pi, \Gamma)$ and $c(\Pi, \Gamma)$ as the overall number of cycles, $\Pi\Gamma$-paths and $\Pi\Pi$-paths in $G(\Pi, \Gamma)$. Clearly, $c(\hat{\Pi}, \hat{\Gamma}) \leq c(\Pi, \Gamma)$ with $c(\hat{\Pi}, \hat{\Gamma}) = c(\Pi, \Gamma)$ when every cycle in $G(\hat{\Pi}, \hat{\Gamma})$ is composed of one $\Pi\Gamma$-path in $G(\Pi, \Gamma)$ which is "closed" by a gray edge in $G(\hat{\Pi}, \hat{\Gamma})$, or "joined" $\Pi\Pi$-path and $\Gamma\Gamma$-path in $G(\Pi, \Gamma)$. (Note that our definition of c is different from that in [8]).

Lemma 2.6. [8] *For every* $\Pi\Pi$-*path and* $\Gamma\Gamma$-*path in* $G(\Pi, \Gamma)$ *there exists either an interchromosomal or an oriented gray edge which joins these paths into a* $\Pi\Gamma$-*path.*

Lemma 2.7. [8] *For every two unoriented* $\Pi\Gamma$-*paths there exists either an interchromosomal or an oriented gray edge which joins these paths into a* $\Pi\Gamma$-*path.*

Interleaving cycles/paths in the graph $G(\Pi, \Gamma)$ are defined as in Section 2.1.2 by making no distinction between cycles and paths in $G(\Pi, \Gamma)$. Define the interleaving graph of Π w.r.t. Γ as the subgraph of $INTL(\hat{\Pi}, \hat{\Gamma})$ induced by the set of (intrachromosomal) cycles/paths in $G(\Pi, \Gamma)$. We refer to the union of cycles/paths in a connected component of $INTL(\Pi, \Gamma)$ as a component of $G(\Pi, \Gamma)$.

2.3.2. *Real-knots, semi-knots and simple components.* An intrachromosomal component in $G(\Pi, \Gamma)$ is called *real* if there are no Π-caps or Γ-tails in its interval. Define $\mathcal{RU}(\Pi, \Gamma)$ as the set of real unoriented components in $G(\Pi, \Gamma)$. Hurdles, super-hurdles and fortresses for the set $\mathcal{RU}(\Pi, \Gamma)$ 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 or real-knots in $G(\Pi, \Gamma)$.

Genomes Π and Γ are *correlated* if all the real-knots in $G(\Pi, \Gamma)$ are located on the same chromosome and *non-correlated* otherwise. Note that there can be the greatest real-knot in $G(\Pi, \Gamma)$ only if genomes Π and Γ are correlated.

As in Section 2.2, define $\mathcal{IU}(\Pi,\Gamma)$ as the set of intrachromosomal components in $G(\Pi,\Gamma)$. Clearly, $\mathcal{RU}(\Pi,\Gamma)\subseteq\mathcal{IU}(\Pi,\Gamma)$. Let K be the set of knots (i.e. hurdles for the set $\mathcal{IU}$). A knot from the set $K\setminus RK$ is a *semi-knot* if it does not contain a $\Gamma\Gamma$-path in its interval. Note that an intrachromosomal component which contains a Π-cap in its interval contains the Π-cap in it. Also note that an intrachromosomal component that contains a $\Pi\Pi$-path in it must contain a $\Gamma\Gamma$-path in its interval. Denote the number of semi-knots in $G(\Pi,\Gamma)$ by $s(\Pi,\Gamma)$. A component in $G(\Pi,\Gamma)$ containing a $\Pi\Gamma$-path is *simple* if it is not a semi-knot.

2.3.3. *HP's duality theorem for genomic distance.* Let $\bar{G}(\Pi,\Gamma)$ be the graph obtained from $G(\Pi,\Gamma)$ by closing all $\Pi\Gamma$-paths in simple components. rr is defined as the number of real-knots in $\bar{G}(\Pi,\Gamma)$. It can be easily seen that $rr=r$ or $rr=r+1$. The latter happens when: *(i)* $r>0$, *(ii)* all the real-knots are located within one chromosome, *(iii)* there is no greatest real-knot, and *(iv)* there is a simple component in $\mathcal{IU}(\Pi,\Gamma)\setminus\mathcal{RU}(\Pi,\Gamma)$ which contains all the real-knots but no $\Gamma\Gamma$-path within its interval. In this case $G(\Pi,\Gamma)$ must have at least one (minimal) semi-knot (otherwise the component was the greatest knot). Let $gr=1$ if there exists the greatest real-knot in $\bar{G}(\Pi,\Gamma)$ and $s>0$, otherwise $gr=0$. Clearly, if $rr=r+1$ then $gr=1$. A breakpoint graph G is a *weak-fortress-of-real-knots* if:

(1) the number of real-knots in G is odd,
(2) one of the real-knots is the greatest real-knot,
(3) every real-knot but the greatest one is a super-real-knot
(4) $s>0$.

Note that a weak-fortress-of-real-knots turns into a fortress-of-real-knots if we close all the $\Pi\Gamma$-paths in one semi-knot. Let $fr=1$ if $\bar{G}(\Pi,\Gamma)$ is a fortress-of-real-knots or a weak-fortress-of-real-knots, $fr=0$ otherwise. The genomic distance by HP's duality theorem ([8], Theorem 4) is[1]: $b-c+rr+\lceil\frac{s-gr+fr}{2}\rceil$.

3. The case for which the duality theorem fails

The proof of HP's duality theorem relied on a lemma ([8], Lemma 6) which states that there exists an optimal capping $\hat{\Gamma}$ which closes all $\Pi\Gamma$-paths in simple components. This lemma is incorrect. The case for which this lemma is false is described as follows:

(1) Genomes Π and Γ are correlated and
(2) $G(\Pi,\Gamma)$ has an even number of real-knots, all of them are super-real-knots.
(3) $G(\Pi,\Gamma)$ has no greatest real-knot, but
(4) there is a component u, in $\mathcal{IU}(\Pi,\Gamma)\setminus\mathcal{RU}(\Pi,\Gamma)$ which contains all the real-knots and does not contain a $\Gamma\Gamma$-path in its interval.
(5) $G(\Pi,\Gamma)$ has an odd number of semi-knots in other chromosomes, hence u is not a knot and in particular not a semi-knot.

[1] The genomic distance was presented in [8] as $b-c+p+rr+\lceil\frac{s-gr+fr}{2}\rceil$ but we prefer to unite parameters c and p.

An example for this case can be seen in Figure 1. The component F plays the role of u in item 4 above. If we close all the $\Pi\Gamma$-paths in u then we get a new greatest real-knot. In addition, we get a weak-fortress-of-real-knots. Hence, the genomic distance, according to the formula, is $b-c+r+1+\lceil\frac{s-1+1}{2}\rceil = b-c+r+1+\frac{s+1}{2}$. However, if we join two $\Pi\Gamma$-paths, one of which is in u and the other in one of the semi-knots by an oriented or interchromosomal gray edge (Lemma 2.7) then we get that the distance is at most $b-c+1+r+\lceil\frac{s-1+0+0}{2}\rceil = b-c+r+\frac{s+1}{2}$, a contradiction.

It can be proved (see Remark 4.2) that this is the only case for which HP's duality theorem and algorithm fail. The "quick and dirty" way to fix HP's duality theorem and algorithm is to give a special treatment to this case, like changing the definition of a weak-fortress-of-real-knots so that it would not include the case in which the greatest real-knot in $\bar{G}$ is a "simple" component in G. However in this type of correction, the parameters are still defined on $\bar{G}$ which is not a sub-graph of any $G(\hat{\Pi},\hat{\Gamma})$ when $\hat{\Gamma}$ is optimal. In the next section, we suggest a different correction.

4. A revised duality theorem

The main difference in the suggested correction is introducing a definition of *semi-real-knots*, as opposed to semi-knots. A component from the set $\mathcal{IU}(\Pi,\Gamma)\setminus\mathcal{RU}(\Pi,\Gamma)$ is a *semi-real-knot* if *(i)* it does not contain a $\Gamma\Gamma$-path 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) greatest real-knot.

A semi-real-knot is called the *semi-greatest-real-knot* if it turns into a greatest real-knot when all the $\Pi\Gamma$-paths in it are closed; otherwise it is a *minimal* semi-real-knot. Any minimal semi-real-knot is also a minimal semi-knot and vice versa. The difference between the two definitions is when a greatest-knot or a semi-greatest-real-knot are involved. A semi-real-knot is not a semi-knot if it is the semi-greatest-real-knot and there are minimal semi-real-knots (equivalent to minimal semi-knots) in $G(\Pi,\Gamma)$. In Figure 1(a) component F is a semi-real-knot (and the semi-greatest-real-knot) but not a semi-knot. A semi-knot is not a semi-real-knot if it is the greatest knot but there exists the greatest real-knot. See Figure 2 for an example. Figure 3 shows two examples where the two definitions of a semi-knot and a semi-real-knot agree.

We define s' to be the number of semi-real-knots in $G(\Pi,\Gamma)$. A *weak-fortress-of-real-knots* is defined in the same manner as before but s is replaced by s'. A *simple component* is redefined as a component containing a $\Pi\Gamma$-path that is not a semi-real-knot. Note that if $G(\Pi,\Gamma)$ has no greatest real-knot then closing all the $\Pi\Gamma$-paths in a simple component cannot create the greatest real-knot. $\bar{G}(\Pi,\Gamma)$ is obtained from $G(\Pi,\Gamma)$ by closing all the $\Pi\Gamma$-paths in all the simple components in $G(\Pi,\Gamma)$. Obviously $\bar{G}(\Pi,\Gamma)$ has the greatest real-knot iff $G(\Pi,\Gamma)$ has the greatest real-knot. It follows that $\bar{G}(\Pi,\Gamma)$ is a weak-fortress-of-real-knots iff $G(\Pi,\Gamma)$ is a weak-fortress-of-real-knots. We define $gr'=1$ if $G(\Pi,\Gamma)$ has the greatest real-knot and $s' > 0$, $gr' = 0$ otherwise. We define $fr' = 1$ if either *(i)* $\bar{G}(\Pi, \Gamma)$ is a fortress-of-real-knots and there is no semi-greatest-real-knot in $\bar{G}(\Pi, \Gamma)$ (equivalently, $G(\Pi, \Gamma)$), or *(ii)* $\bar{G}(\Pi, \Gamma)$ (equivalently, $G(\Pi, \Gamma)$) is a weak-fortress-of-real-knots, and $fr' = 0$ otherwise. Note that closing all the $\Pi\Gamma$-paths in a simple component does not affect any of the parameters defined in this section.

Theorem 4.1. $d(\Pi, \Gamma) = b(\Pi, \Gamma) - c(\Pi, \Gamma) + r(\Pi, \Gamma) + \lceil \frac{s'(\Pi,\Gamma) - gr'(\Pi,\Gamma) + fr'(\Pi,\Gamma)}{2} \rceil$

Proof. Let $\Psi(\Pi, \Gamma)$ be the right hand side of the expression above. Every capping $\hat{\Gamma}$ can be presented as a sequence of transformations of $G(\Pi, \Gamma)$ into $G(\hat{\Pi}, \hat{\Gamma})$ by consecutively adding $2N$ gray edges to $G(\Pi, \Gamma)$: $G(\Pi, \Gamma) = G_0 \to^{g_1} G_1 \to^{g_2} \cdots \to^{g_{2N}} G_{2N} = G(\hat{\Pi}, \hat{\Gamma})$. For a graph G_i the parameters $b_i = b(G_i) = n + N$, $c_i = c(G_i)$, $r_i = r(G_i)$, $s'_i = s'(G_i)$, $gr'_i = gr'(G_i)$, $fr'_i = fr'(G_i)$ and $\Psi_i = \Psi(G_i)$ are defined in the same way as for the graph $G_0 = G(\Pi, \Gamma)$. For a parameter ϕ define $\Delta\phi_i$ as $\phi_i - \phi_{i-1}$. Denote $\Delta_i = \Psi_i - \Psi_{i-1}$. We first prove that $\Delta_i \geq 0$ for $1 \leq i \leq 2N$, i.e., adding a gray edge does not decrease the parameter Ψ. For a fixed i we ignore the index i, i.e., denote $\Delta = \Delta_i$ and $\Delta\phi = \Delta\phi_i$.

Depending on the edge g_i the following four cases are possible:

Case 1: edge g_i closes a $\Pi\Gamma$-path. Clearly, $\Delta c = 0$. If the semi-greatest-real-knot in G_{i-1} turns into a real-knot in G_i then $\Delta r = 1$, $\Delta s' = -1$, $\Delta gr' \leq 1$, $\Delta fr' \geq 0$ (since $fr'_{i-1} = 0$), so $\Delta \geq 0$.
If a minimal semi-real-knot in G_{i-1} turns into a (minimal) real-knot in G_i then there are three subcases: *(i)* G_{i-1} has the greatest real-knot, *(ii)* G_{i-1} has the semi-greatest-real-knot, and *(iii)* G_{i-1} has neither the greatest real-knot nor the semi-greatest-real-knot. In the first sub-case: $\Delta r = 0$, $\Delta s' = -1$, $\Delta gr' = -1$, $\Delta fr' = 0$ ($fr'_i = 1$ iff $fr'_{i-1} = 1$), so $\Delta \geq 0$. In the second sub-case: $\Delta r = 1$, $\Delta s' = -2$ (the semi-greatest-real-knot in G_{i-1} turns into a simple component in G_i and to a real component in $\bar{G}_i$), $\Delta gr' = 0$, $\Delta fr' \geq 0$ ($fr'_{i-1} = 0$), so $\Delta \geq 0$. In the third sub-case: $\Delta r = 1$, $\Delta s' = -1$, $\Delta gr' = 0$, $\Delta fr' \geq -1$, so again $\Delta \geq 0$.

If g_i closes a $\Pi\Gamma$ path and the number of semi-real-knots is not changed ($s'_i = s'_{i-1}$) then no other parameter is affected and $\Delta = 0$.

Case 2: edge g_i connects two $\Pi\Gamma$-paths. In that case $\Delta c = -1$, $\Delta r = 0$, $\Delta gr' \leq 0$ and at most two semi-real-knots are destroyed. If it destroys less than two semi-real-knots then $\Delta s' \geq -1$, $\Delta fr' \geq -1$, and so $\Delta \geq 0$. If it does destroy two semi-real-knots then $\Delta s' = -2$. If $\Delta fr' = -1$ then $\bar{G}_{i-1}$ is a weak-fortress-of-real-knots ($s'_{i-1} > 0$) but G_i is not since $s'_i = 0$, hence $\Delta gr' = -1$, $\Delta = 0$. If $\Delta fr' = 0$ then $\Delta \geq 0$.

Case 3: edge g_i connects a $\Pi\Gamma$-path with a $\Pi\Pi$-path ($\Gamma\Gamma$-path). $\Delta c = -1$, $\Delta r = 0$, $\Delta s' \geq -1$,$\Delta gr' \leq 0$, $\Delta fr' \geq -1$, hence $\Delta \geq 0$.

Case 4: edge g_i connects a $\Pi\Pi$-path with a $\Gamma\Gamma$-path. $\Delta c = \Delta r = 0$. Clearly, $\Delta = 0$

if no new semi-real-knot is created in G_i. Otherwise, $\Delta s' = 1$ and there are two possible sub-cases: *(i)* there is a new minimal semi-real-knot in G_i, *(ii)* the semi-greatest-real-knot is created in G_i. In the first sub-case $\Delta gr' \leq 1$, $\Delta fr' \geq 0$, $\Delta \geq 0$. In the second sub-case $\Delta gr' = 0$, $\Delta fr' \geq -1$, $\Delta \geq 0$.

After all $2N$ gray edges have been added, $b_{2N} = b(\hat{\Pi}, \hat{\Gamma})$, $c_{2N} = c(\hat{\Pi}, \hat{\Gamma})$, $r_{2N} = k(\hat{\Pi}, \hat{\Gamma})$, $s'_{2N} = 0$, $gr'_{2N} = 0$ and $fr'_{2N} = f(\hat{\Pi}, \hat{\Gamma})$. Hence $d(\hat{\Pi}, \hat{\Gamma}) = b_{2N} - c_{2N} + r_{2N} + fr'_{2N} \geq \Psi(\Pi, \Gamma)$. This completes the proof of the claim.

We shall now prove that there exists a capping $\hat{\Gamma}$ such that $d(\hat{\Pi}, \hat{\Gamma}) = \Psi(\Pi, \Gamma)$ by constructing a sequence of $2N$ gray edges $g_1, ..., g_{2N}$ connecting Π-caps with Γ-tails in $G(\Pi, \Gamma)$ such that $\Delta_i = 0$ for every $1 \leq i \leq 2N$. The algorithm *Genomic_Sort_2* builds this sequence of edges.

algorithm *Genomic_Sort_2*

(1) Construct the graph $G = G(\Pi, \Gamma)$
(2) **while** there is a $\Pi\Pi$-path in G
(3) Find an interchromosomal or an oriented edge joining this $\Pi\Pi$-path with a $\Gamma\Gamma$-path (Lemma 2.6) and add it to G
(4) **while** G has more than two semi-real-knots
(5) Find an interchromosomal or an oriented edge joining $\Pi\Gamma$-paths in any two semi-real-knots (Lemma 2.7) and add it to G
(6) Close all $\Pi\Gamma$-paths in simple components in G
(7) **if** G has two semi-real-knots but is not a fortress-of-real-knots
(8) Find an interchromosomal or an oriented edge joining $\Pi\Gamma$-paths in these semi-real-knots (Lemma 2.7) and add it to G
(9) Close any remaining $\Pi\Gamma$-path in G
(10) Find a capping $\hat{\Gamma}$ defined by the graph $G = G(\hat{\Pi}, \hat{\Gamma})$
(11) Sort genome Π into genome Γ by using the sorting of $\hat{\Pi}$ into genome $\hat{\Gamma}$

If G_{i-1} has a $\Pi\Pi$-path then it has also a $\Gamma\Gamma$-path. Connecting a $\Pi\Pi$-path with a $\Gamma\Gamma$-path via an interchromosomal or oriented edge affects no parameter and $\Delta = 0$ (line 3).

If G_{i-1} has more than two semi-real-knots and two semi-real-knots are destroyed then $\Delta c = -1$, $\Delta r = \Delta gr' = \Delta fr' = 0$, $\Delta s' = -2$, hence $\Delta = 0$ (line 5).

Closing all the $\Pi\Gamma$-paths in simple components does not affect any parameter and hence $\Delta = 0$ (line 6). Line 6 is required in order to calculate if $\bar{G}$ is a fortress-of-real-knots (line 7).
If G_{i-1} is not a fortress-of-real-knots and exactly two semi-real-knots then $-gr'_{i-1} + fr'_{i-1} \leq 0$

($fr'_{i-1} = 1$ iff G_{i-1} is a weak-fortress-of-real-knots). If we connect these two semi-real-knots by an oriented or interchromosomal edge then $\Delta = \Delta(-c + r + \lceil\frac{s'-gr'+fr'}{2}\rceil) = 1 + 0 + (\lceil\frac{0-0+0}{2}\rceil - \lceil\frac{2-gr'_{i-1}+fr'_{i-1}}{2}\rceil) =$ 1-1=0 (line 8).

At the beginning of step 9, G has at most two semi-real-knots. If G has exactly two semi-real-knots then it is a fortress-of-real-knots. The only change in G is the addition of gray edges which close $\Pi\Gamma$-paths. If the number of semi-real-knots is not changed ($s'_i = s'_{i-1}$) then no other parameter is affected and $\Delta = 0$.
Let $G_{start} = G_{i-1}$ be the graph at the beginning of step 9. If G_{start} is a fortress-of-real-knots and exactly two semi-real-knots then there are three possible sub-cases: *(i)* G_{start} has the greatest real-knot, *(ii)* G_{start} has the semi-greatest-real-knot *(iii)* G_{start} has no semi-greatest-real-knot and the two semi-real-knots are minimal. Let G_{finish} be the graph obtained from G_{start} by closing all the $\Pi\Gamma$-paths in the two semi-real-knots in G_{start}. In all the possible sub-cases $s'_{finish} = gr'_{finish} = fr'_{finish} = 0$. In the first sub-case $\Delta r = 1$, $gr'_{start} = 1$, $fr'_{start} = 1$, $\Delta = \Delta r - \lceil\frac{s'_{start}-gr'_{start}+fr'_{start}}{2}\rceil = 1 - \lceil\frac{2-1+1}{2}\rceil = 1 - 1 = 0$. In the second sub-case $\Delta r = 1$, $gr'_{start} = 0$, $fr'_{start} = 0$ (since G_{start} has the semi-greatest-real-knot), $\Delta = \Delta r - \lceil\frac{s'_{start}-gr'_{start}+fr'_{start}}{2}\rceil = 1 - \lceil\frac{2-0+0}{2}\rceil = 1 - 1 = 0$. In the third sub-case $\Delta r = 2$, $gr'_{start} = 0$, $fr'_{start} = 1$ $\Delta = \Delta r - \lceil\frac{s'_{start}-gr'_{start}+fr'_{start}}{2}\rceil = 2 - \lceil\frac{2-0+1}{2}\rceil = 2 - 2 = 0$.
If G_{i-1} has exactly one semi-real-knot then there are three possible sub-cases: *(i)* the semi-real-knot is the semi-greatest-real-knot, *(ii)* the semi-real-knot is a minimal semi-real-knot and G_{i-1} has the greatest real-knot, *(iii)* the semi-real-knot is a minimal semi-real-knot and G_{i-1} has no greatest real-knot. In the first sub-case $\Delta r = 1$, $gr'_{i-1} = gr'_i = 0$, $fr'_{i-1} = fr'_i = 0$, $\Delta = 1 - \lceil\frac{1}{2}\rceil = 0$. In the second sub-case $\Delta r = 0$, $\Delta fr' = 0$ ($fr'_i = 1$ iff $fr'_{i-1} = 1$), $\Delta = \lceil\frac{0-0+fr'}{2}\rceil - \lceil\frac{1-1+fr'}{2}\rceil) = 0$. In the third sub-case $\Delta r = 1$, $fr'_i = 0$, $\Delta = 1 - \lceil\frac{1-0+fr'_{i-1}}{2}\rceil = 1 - 1 = 0$.

□

Note, that as a result of the revised definition of simple components, Lemma 6 in [8] becomes true now.

Remark 4.2. Some of the definitions of the parameters that were used by Hannenhalli and Pevzner in their analysis (see Section 2.3.3) are different than ours (Theorem 4.1). A case by case analysis of the parameters in the two formulas shows that the case studied in Section 3 is the only one for which the formulas differ.

5. On the complexity of SBR

A central question in the study of genome rearrangements is whether one can obtain a sub-quadratic algorithm for SBR. In this section we discuss the three central algorithms currently available for SBR. The first is of Kaplan, Shamir and Tarjan [9] (KST), the second is of Berman and Hannenhalli [5] (BH) and the last is due to Bergeron [4]. All three build heavily on the foundations laid by Hannenhalli and Pevzner [7]. All three algorithms use implicitly or explicitly the overlap graph of the permutation (defined below). The KST and BH algorithms require $O(n^2)$ and Bergeron's algorithm requires $O(n^3)$. We provide here a family of examples where any optimal sequence of reversals on an n-vertex permutation requires $\Theta(n)$ reversals and $\Theta(n)$ modifications in that graph in each reversal. In particular, this implies that Bergeron's algorithm as implemented in [4] requires $\Theta(n^3)$ steps. In addition, the BH and KST algorithms are shown to have a quadratic lower time bound on these families of permutations.

5.1. **The algorithms.** The three algorithms have a common iterative structure. In each iteration they search for a safe reversal and perform it. There are two types of safe reversals: proper safe reversals and hurdle-cutting safe reversals. The latter are performed in a similar manner in all the algorithms, either at the beginning of the sorting or when there are no oriented components. For the rest of this section we will concentrate on permutations which are optimally sorted by proper reversals only, i.e. their overlap graphs (equivalently, interleaving graphs) contain only oriented components. In this case, the time complexity of each algorithm is proportional to the reversal distance times the time complexity of finding and performing a safe reversal.

The time complexity of finding and performing a safe reversal depends on the time complexity of the queries and updates of the data structure used to maintain the current permutation. Each of the three algorithms uses a different data structure to maintain the current permutation. In addition, the characterization of the safe reversal used is different in each algorithm. However, the three algorithms share two key features: *(i)* the search for a safe reversal is done within the set of oriented gray edges and *(ii)* the criterion for a safe reversal relies mainly on the relationships between gray edges in the overlap graph.

Let e be a vertex in $OV(\pi)$. Denote by $r(e)$ the reversal acting on the gray edge corresponding to e. Denote by $N(e)$ the set of neighbors of e in $OV(\pi)$. Denote by $ON(e)$ the set of oriented neighbors of e in $OV(\pi)$. Denote by $UN(e)$ the set of unoriented neighbors of e in $OV(\pi)$. The analysis of any known criterion for a safe reversal is based on the premise that it must not create any new unoriented component in $OV(\pi)$.

Observation 5.1. [9] *Let e be an oriented vertex in $OV(\pi)$. $OV(\pi \cdot r(e))$ can be obtained from $OV(\pi)$ by the following operations: (i) Complement the graph induced by $OV(\pi)$ on $N(e) \cup \{e\}$, (ii) change the orientation of every vertex in $N(e) \cup \{e\}$, and (iii) remove any unoriented single vertices.*

A reversal that operates on an oriented vertex, e, deletes it and at most one more vertex from $OV(\pi \cdot r(e))$. The vertices in the component of e in $OV(\pi)$ which remain in $OV(\pi \cdot r(e))$ may now be part of one or more new components in $OV(\pi \cdot r(e))$. The following observation is implicit in [9].

Observation 5.2. *Let e be an oriented vertex in $OV(\pi)$. Every new component in $OV(\pi \cdot r(e))$ contains a vertex from $N(e)$. Hence, e is safe iff every vertex in $ON(e)$ is contained within an oriented component in $OV(\pi \cdot r(e))$.*

5.1.1. *The KST algorithm.* The KST algorithm uses the notion of *happy cliques*. A *happy clique* is a clique of oriented vertices, C, such that every oriented vertex $e \notin C$ that has a neighbor in C, also has an oriented neighbor that is not connected to any vertex in C.

Theorem 5.3. [9] *Let C be a happy clique and let e be a vertex in C such that $|UN(e)| \geq |UN(e')|$ for every $e' \in C$. Then the reversal $r(e)$ is safe.*

Choosing a vertex from a happy clique guarantees that every vertex in $ON(e) \setminus C$ is contained in an oriented component in $OV(\pi \cdot r(e))$. The second condition ($|UN(e)| \geq |UN(e')|$ for every $e' \in C$) guarantees that every vertex in the happy clique is either contained in an oriented component in $OV(\pi \cdot r(e))$ or deleted from it.

Theorem 5.4. [9] *There is a happy clique in the neighborhood of every oriented vertex e (i.e. $N(e) \cup \{e\}$ contains a happy clique when e is oriented).*

Let $e_1, \ldots, e_k$ be the oriented vertices in $OV(\pi)$ in increasing left endpoint order. The KST algorithm searches for a happy clique by traversing the oriented vertices in $OV(\pi)$ in this order. After traversing $e_1, \ldots, e_i$, $1 \leq i \leq k$ the KST algorithm maintains a happy clique $C_i = \{e_{i_1}, \ldots, e_{i_j}\}$ in the subgraph of $OV(\pi)$ induced by these vertices. The traversal is finished either when $i = k$ or when e_i is to the right of e_{i_j}. After finding a happy clique, C, the KST algorithm searches for a safe reversal within C in time complexity proportional to the size of C plus the number of unoriented vertices in $OV(\pi)$.

The data structure used by the KST algorithm is two arrays containing the sequences $u(\pi)$ and $u(\pi)^{-1}$. A vertex in the overlap graph is represented by one of the endpoints of its gray edge. This data structure allows answering two major questions on the overlap graph in a constant time: *(i)* the orientation of a vertex and *(ii)* whether two given vertices are neighbors. In Section 5.2 we shall see that the number of queries in a KST's search for a safe reversal is $\Theta(n)$. For an oriented vertex, e, the update of the data structure for making a reversal, $r(e)$, is done in time proportional to the length of the interval being reversed, which is at least $|N(e)|$. Recently, Kaplan and Verbin [10] presented various data structures which maintain $u(\pi)$ and $u(\pi)^{-1}$ and allow an update after a reversal in $o(n)$ time. These data structures were used in a heuristic "random-walk" algorithm which picked an oriented reversal randomly and performed it. However, the time complexity of a query in these data structures is $\Theta(\log(n))$.

5.1.2. *The BH algorithm.* The BH algorithm makes a distinction between *long* cycles, which are cycles with more than 2 gray edges, and *short* cycles, which are cycles with exactly 2 gray edges. A permutation is *simple* if every cycle is of size 4. An initialization step in the BH algorithm is producing a simple permutation π' for which every sorting of π' mimics a sorting of π with the same number of reversals. The BH algorithm uses the interleaving graph instead of the overlap graph. However, the overlap graph of a simple permutation can be obtained from the interleaving graph by replacing every oriented vertex by two connected oriented vertices, and every unoriented vertex by two unconnected unoriented vertices. This implies that for simple permutations the interleaving graph and overlap graph are essentially equivalent and hence the interleaving graph can be replaced with the overlap graph and vice versa.

Denote by $V(INTL(\pi))$ and $O(INTL(\pi))$ the set of vertices and oriented vertices in the interleaving graph $INTL(\pi)$ respectively. Let ρ be a reversal on π. Define $Index(\rho)$ as the union of new unoriented components in $INTL(\pi \cdot \rho)$ and $index(\rho) = |V(Index(\rho))|$. The following theorem was proven in [5]. We provide a simpler proof below.

Theorem 5.5. [5] *There exists an oriented vertex, e, for which* $index(r(e)) \leq \frac{|V(INTL(\pi))|}{2}$

Proof. **case 1:** $O(INTL(\pi))$ is a clique. In this case we choose e to be a vertex for which $|UN(e)| \geq |UN(e')|$ for every oriented vertex e'. It can be easily seen that $r(e)$ is a safe reversal and hence $index(r(e)) = 0$.
case 2: $O(INTL(\pi))$ is not a clique. Let e_1 and e_2 be two non adjacent oriented vertices. Let $f \in Index(r(e_1))$. We will now prove that $f \notin Index(r(e_2))$. Look at a shortest path from f to e_1 in $OV(\pi)$: $f = u_1 \to u_2 \to \cdots \to u_t = e_1$. Obviously u_{t-1} and u_t (e_1) are the only oriented vertices in this path in $OV(\pi)$. Since e_1 and e_2 are non adjacent u_t is still oriented in $OV(\pi \cdot r(e_2))$ and the edge $u_{t-1} \to u_t$ also exists. Let i be the minimum index such that u_i is oriented in $OV(\pi \cdot r(e_2))$ $(i \leq t)$. It is easy to see that there is a path from f (u_1) to u_i in $OV(\pi \cdot r(e_2))$and hence $f \notin Index(r(e_2))$. Therefore, $Index(r(e_1)) \cap Index(r(e_2)) = \emptyset$ and $\min(index(r(e_1)), index(r(e_2))) \leq \frac{|V(INTL(\pi))|}{2}$. □

Theorem 5.6. [5] *Let e be an oriented vertex in* $OV(\pi)$ *such that* $Index(r(e)) \neq \emptyset$*. Let K be an unoriented component created in* $OV(\pi \cdot r(e))$*. Let* $e' \in K$ *be an oriented vertex in* $OV(\pi)$ *(*e' *is oriented in* $OV(\pi)$ *and unoriented in* $OV(\pi \cdot r(e))$*). For every vertex f,* $f \in Index(r(e'))$ *iff f belongs to an unoriented component in the subgraph of* $OV(\pi \cdot r(e'))$ *induced by K.*

Theorems 5.5 and 5.6 immediately lead to a recursive algorithm for finding a safe reversal in an oriented component:

Algorithm Find_Safe_Reversal(I **)**

(1) Find an oriented vertex e such that $index(r(e)) \leq \frac{|V_I|}{2}$. (Theorem 5.5)
(2) **If** $r(e)$ is safe (i.e. $index(r(e)) = 0$) **return** $r(e)$.
(3) **Else**
  (a) Let K be an unoriented component in $I \cdot r(e)$.
  (b) Find_Safe_Reversal(K). (Theorem 5.6)

The complexity of Find_Safe_Reversals is based on the complexity of an algorithm that computes the connected components of the overlap graph after a reversal is made. Berman and Hannenhalli have proposed an $O(n\alpha(n))$ algorithm for that task which receives as an input a sequence of gray edges (one from every cycle) ordered by their appearance in a linear scan of the permutation. This implied an overall complexity of $O(n^2\alpha(n))$. Later, Bader et al. presented an improved algorithm for computing the connected components in linear time [2]. Hence the BH algorithm can now be implemented in a quadratic time. Both the original BH algorithm and the improved algorithm of Bader et al. perform a traversal of all the cycles in the breakpoint graph. Hence, the time complexity in both cases is $\Omega(c(\pi)d(\pi))$.

5.1.3. *Bergeron's algorithm.* Bergeron's algorithm is the simplest algorithm of the three, in its implementation as well in the analysis upon it is based.

Observation 5.7. [4] *A vertex has an odd degree iff it is oriented.*

Theorem 5.8. [4] *Let e be an oriented vertex for which $|UN(e)| - |ON(e)|$ is maximal. Then $r(e)$ is a safe reversal.*

The implementation of Bergeron's algorithm is based on a bit matrix which represents the adjacency matrix of the overlap graph, i.e. $M[i,j] = 1$ iff vertices i and j are adjacent. In addition, there are a parity vector and a score vector. The search for a safe reversal is simply choosing the vertex with the highest score. After a reversal is made, the adjacency matrix, parity vector and score vector are kept updated using mostly operations on bit vectors (only the score vector is composed of integers and not bits). The number of operations needed to update the data structures per a reversal $r(e)$ is $\Theta(|N(e)|)$. Hence the time complexity of Bergeron's algorithm is $O(n^3)$ ($O(n)$ reversals, each reversal requires $O(n)$ updates of vectors of size $O(n)$).

Kaplan and Verbin [10] presented an algorithm which uses Bergeron's criterion for a safe reversal but with a different data structure. Their implementation calculates the score vector using a reduction to $\Theta(n)$ queries on a data structure from computational geometry that holds an n by n grid. This data structure is rebuilt on every iteration in $\Theta(n\log(n))$ time and each query takes $\Theta(\log(n))$ time. Hence, the total time complexity of a search for a safe reversal is $\Theta(n\log(n))$.

5.2. **Difficult permutations.** As we have seen, all the above algorithms use the overlap graph as a basis for their analysis. However, only Bergeron's algorithm explicitly maintains data structures for the overlap graph. Every iteration in Bergeron's algorithm involves a number of operations which is proportional to the number of vertices which are actually changed in the overlap graph. The question that immediately rises is: can one obtain a tight lower bound for Bergeron's algorithm? We will now prove that for every n there exists a permutation for which Bergeron's algorithm performs a quadratic number of operations on vectors.

5.2.1. *One cycle permutation.* Let $\pi_k = (2, 4, 6, \ldots, 2k, -1, -3, -5, \ldots, -(2k-1))$. The breakpoint graph of π_k has one oriented cycle and $2k+1$ breakpoints, hence the reversal distance is $2k = n$. See Figure 4(a) for $k = 3$. A *module* X in a graph $G = (V, E)$ is a set of vertices such that if $x, y \in X$ and $z \notin X$ then $(x, z) \in E \Leftrightarrow (y, z) \in E$. A *super clique* (*super* *independent set*) is a clique (independent set) of modules. The overlap graph $OV(\pi_k)$ is a super clique of size $k+1$ consisting of k modules, which are unconnected pairs of oriented vertices, and one singleton module, which is the unoriented vertex $(4k, 4k+1)$ (Figure 4 (b)). All the oriented vertices are symmetric in $OV(\pi_k)$ and by Theorem 5.8 every oriented vertex induces a safe reversal. Let ρ_1 be a reversal upon an oriented vertex in $OV(\pi_k)$. Then $OV(\pi_k \cdot \rho_1)$ is composed of an oriented vertex that is connected to a module M. The module M is a super independent set of size k consisting of $k-1$ modules, which are connected pairs of unoriented vertices, and one singleton module, which is an oriented vertex (Figure 4(c)). There is only one oriented vertex that induces a safe reversal ρ_2 in $OV(\pi_k \cdot \rho_1)$. We perform ρ_2 and get $OV(\pi_k \cdot \rho_1 \cdot \rho_2)$ which is isomorphic to $OV(\pi_{k-1})$ (Figure 4(d)).

Obviously, any safe reversal on π_k affects $\Theta(n)$ vertices in the overlap graph. Hence the update of the data structures in Bergeron's algorithm requires $\Theta(n)$ vectors of size n, and the total running time of the algorithm is $\Theta(n^3)$ time. The KST algorithm for finding a safe reversal on π_k has a time complexity of $\Theta(n)$ since the search for a happy clique in $OV(\pi_k)$ performs a traversal on all the oriented vertices in $OV(\pi_k)$. In addition, the update of the data structure is also $\Theta(n)$ per reversal. Hence the total time complexity of sorting π_k using the KST algorithm is $\Theta(n^2)$.

5.2.2. *A simple permutation.* The BH algorithm is defined on an interleaving graph of a simple permutation (one with no long cycles). Any non-simple permutation is transformed to an equivalent simple permutation. We now present an example with a simple permutation σ_k. Let $\sigma_k = (4k, 2, 4k-2, 4, \ldots, 2k+2, 2k, -(4k+1), -1, -(4k-1), -3, \ldots, -(2k+3), -(2k-1), -(2k+1))$. The breakpoint graph of σ_k has $2k+1$ (short) cycles and $4k+2$ breakpoints, hence the reversal distance is $2k+1 = \frac{n+1}{2}$. For $k=2$, see Figure 5. The interleaving graph of σ_k, $INTL(\sigma_k)$, is super clique of size $k+1$ with k modules, which are unconnected pairs of oriented cycles, and one singleton module, which is an oriented cycle (Figure 5(b)). Each of the vertices in the k pairs induces a safe reversal while the singleton vertex induces an unsafe reversal. Let ρ_1 be a safe reversal induced by one of the $2k$ symmetric vertices. Then $INTL(\sigma_k \cdot \rho_1)$ is composed of an oriented vertex which is connected to a module that is a super independent set consisting of $k-1$ modules, which are connected pairs of oriented vertices, and one singleton module, which is an isolated oriented vertex (Figure 5(c)). Let ρ_2 be the (safe) reversal induced by the single oriented vertex in $INTL(\sigma_k \cdot \rho_1)$. We perform ρ_2 and get $INTL(\sigma_k \cdot \rho_1\rho_2)$ which is isomorphic to $INTL(\sigma_{k-1})$.

It is easy to see that any safe reversal on σ_k affects $\Theta(n)$ vertices in the interleaving graph. Hence any sorting of π_k affects $\Theta(n^2)$ vertices in the interleaving graph. The BH algorithm does not maintain the overlap graph of π_k directly. Instead, in every iteration it performs a traversal on all the the cycles in the breakpoint graph of σ_k. Hence the total time complexity of sorting σ_k using the BH algorithm is $\Theta(n^2)$.

Acknowledgements

We thank Glenn Tesler and two anonymous referees for careful reading of this paper and numerous helpful comments. This work was supported in part by the Israel Science Foundation (grant 309/02).

References

1. W. Ackermann, *Zum hilbertshen aufbau der reelen zahlen*, Math. Ann. **99** (1928), 118–133.
2. D.A. Bader, B. M.E. Moret, and M. Yan, *A linear-time algorithm for computing inversion distance between two signed permutations with an experimental study*, Proceedings of the Seventh International Workshop on Algorithms and Data Structures (WADS 2001), August 2001.
3. V. Bafna and P. A. Pevzner, *Genome rearragements and sorting by reversals*, SIAM Journal on Computing **25** (1996), no. 2, 272–289, A preliminary version appeared in Proc. 34th IEEE Symp. of the Foundations of Computer Science, p. 148–157, 1994.
4. A. Bergeron, *A very elementary presentation of the Hannenhalli-Pevzner theory*, Proc. 12th Annual Symposium on Combinaotrial Pattern Matching (CPM '01), 2001.
5. P. Berman and S. Hannenhalli, *Fast sorting by reversals*, Proc. 7th Annual Symposium on Combinaotrial Pattern Matching (CPM '96) (Berlin), Springer, 1996, LNCS 1075, pp. 168–185.
6. S. Hannenhalli, *Polynomial algorithm for computing translocation distance between genomes*, Discrete Applied Mathematics **71** (1996), 137–151.
7. S. Hannenhalli and P. Pevzner, *Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals)*, Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (Las Vegas, Nevada), 29 May–1 June 1995, pp. 178–189.
8. ———, *Transforming men into mice (polynomial algorithm for genomic distance problems)*, Proc. IEEE Symp. of the Foundations of Computer Science, 1995, pp. 581–592.
9. H. Kaplan, R. Shamir, and R. E. Tarjan, *Faster and simpler algorithm for sorting signed permutations by reversals*, SIAM Journal of Computing **29** (1999), no. 3, 880–892, (Preliminary version in Proceedings of the eighth annual ACM-SIAM Symposium on Discrete Algorithms 1997 (SODA 97), ACM Press, pages 344–351).
10. H. Kaplan and E. Verbin., *Sorting signed permutations by reversals revisited*, Second Haifa Workshop on Interdisciplinary Applications of Graph Theory, Combinatorics and Algorithms, June 17-20, 2002.
11. J. Kececioglu and D. Sankoff, *Efficient bounds for oriented chromosome inversion distance*, Proc. of 5th Ann. Symp. on Combinatorial Pattern Matching, Springer, 1994, LNCS 807, pp. 307–325.
12. G. Tesler, *Efficient algorithms for multichromosomal genome rearrangements*, J. Comp. Sys. Sci. (2002).

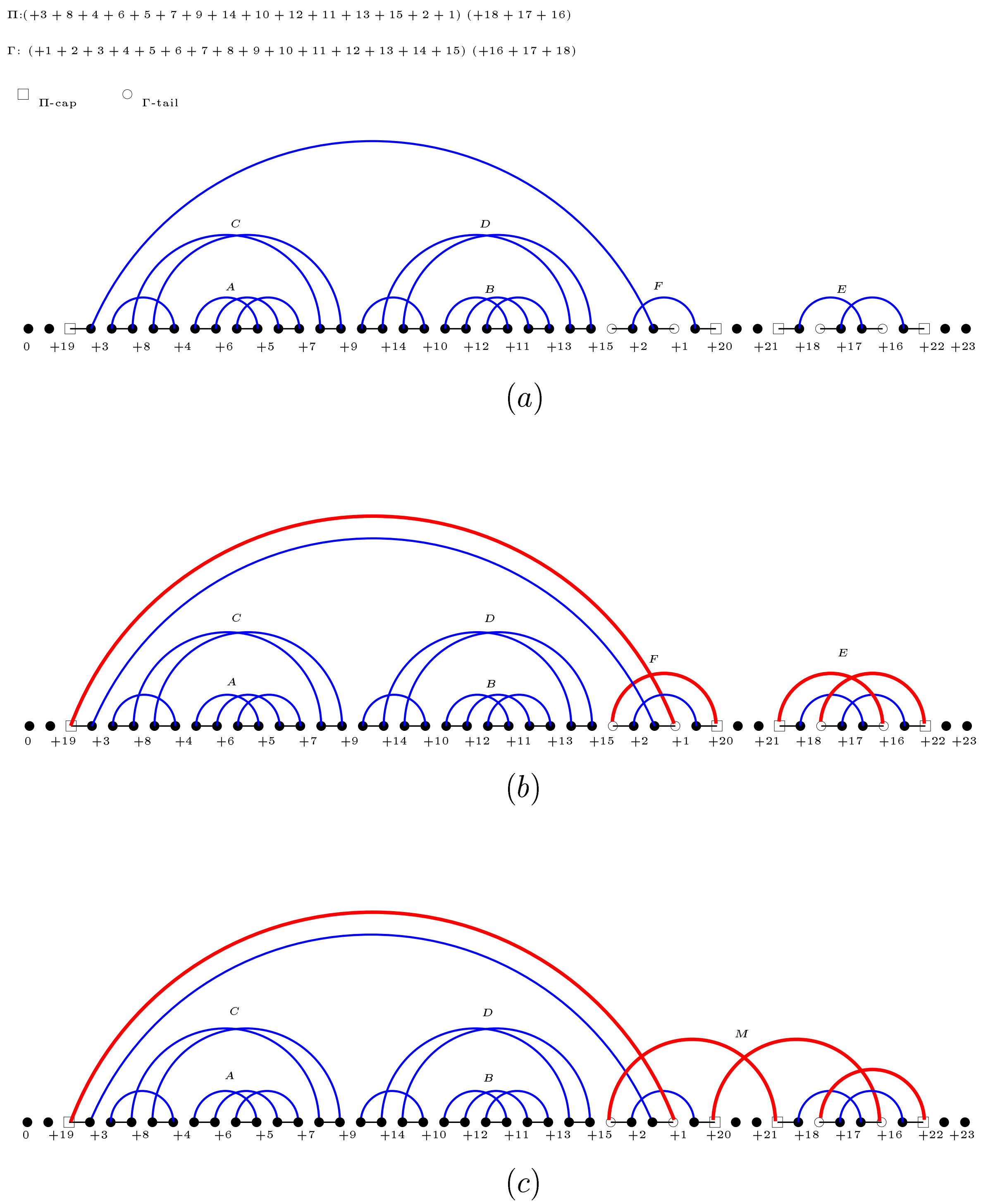


FIGURE 1. An example for which HP's duality theorem fails. (a) $G(\Pi, \Gamma)$ has two super-real-knots, A and B, one semi-knot, E, and one simple component, F. (b) The graph obtained from $G(\Pi, \Gamma)$ after performing the HP algorithm. Component E becomes a super-knot and there is a fortress-of-knots, hence $d(\hat{\Pi}, \hat{\Gamma}) = b - c + k + f = 20 - 8 + 3 + 1 = 16$. (c) Another graph that can be obtained from $G(\Pi, \Gamma)$. Components E and F are united into one interchromosomal component M, so $d(\hat{\Pi}, \hat{\Gamma}) = b - c + k + f = 20 - 7 + 2 + 0 = 15$.

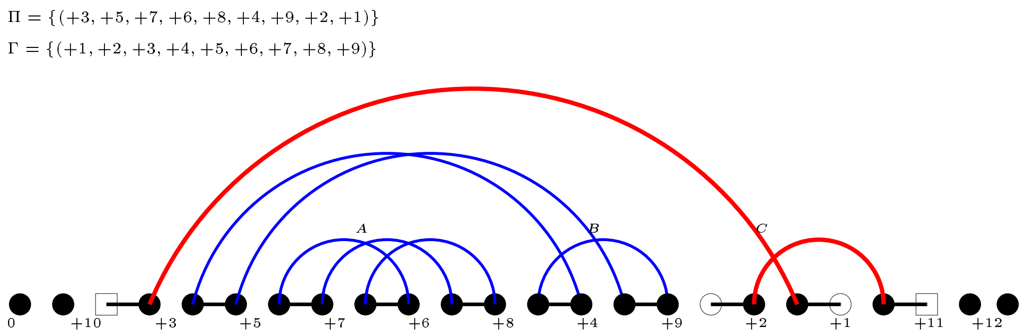


FIGURE 2. An example for a semi-knot which is not a semi-real-knot. There are two real-knots in this example: A (a minimal real-knot) and B (the greatest real-knot). Component C is a knot and in particular a semi-knot. However C is not a semi-real-knot since closing all the $\Pi\Gamma$-paths in it would turn it into a greatest real-knot which is a super-real-knot.

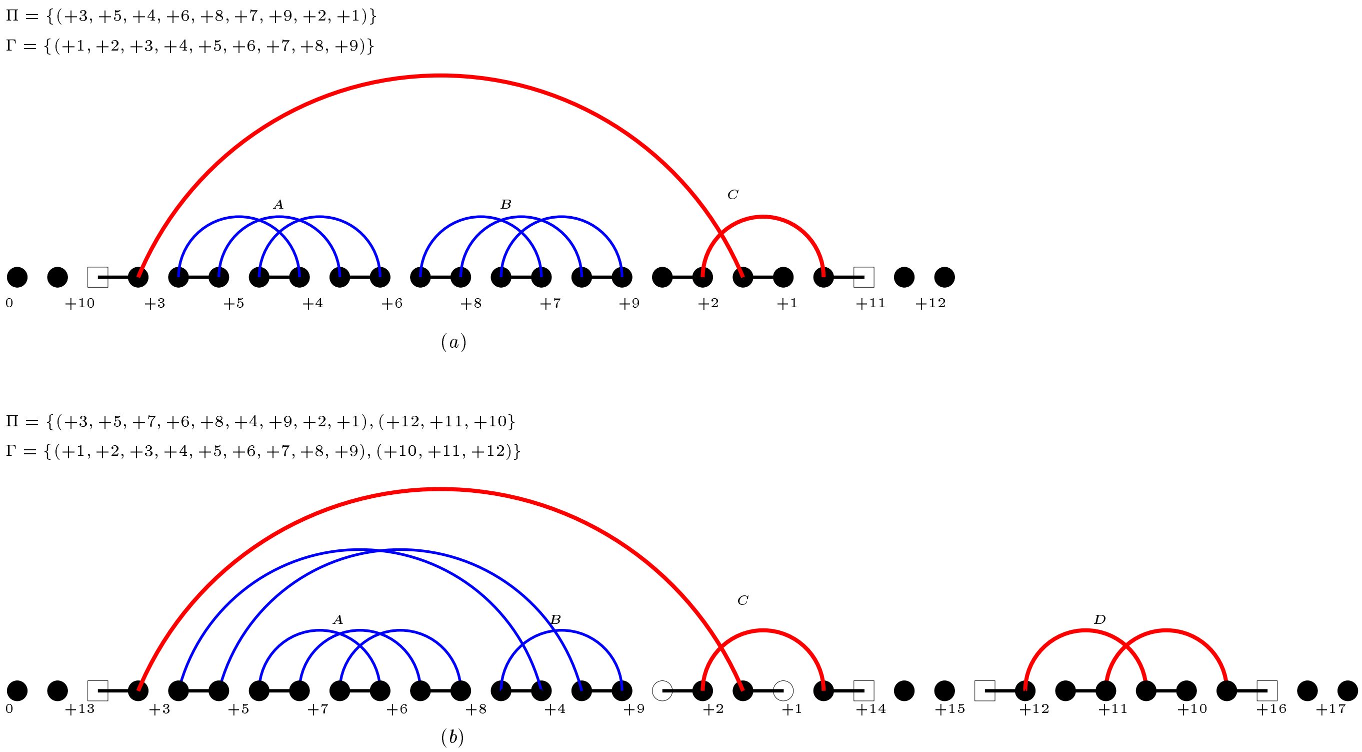
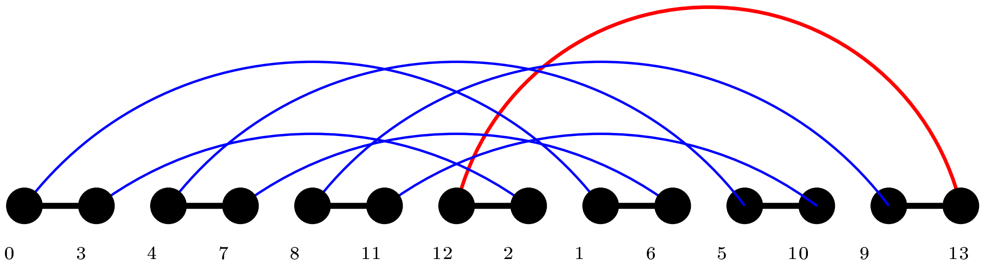
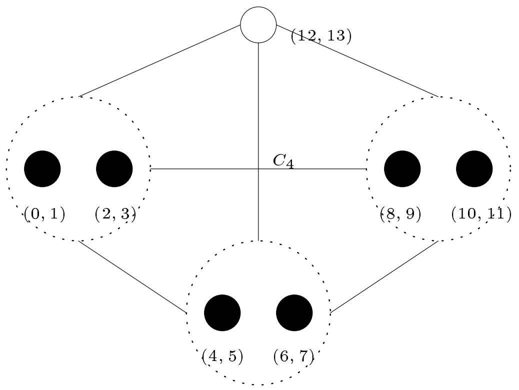


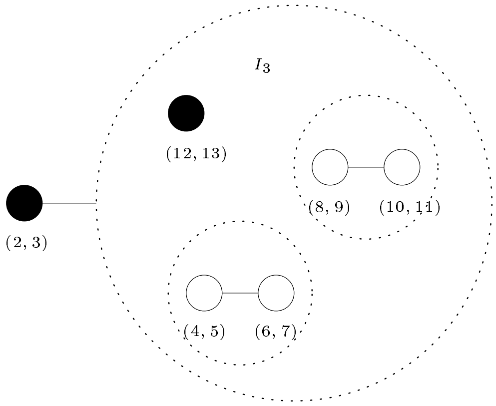
FIGURE 3. Cases where the definitions of semi-knot and semi-real-knot matches. (a) Component C is the greatest knot and in particular a semi-knot. In addition, closing all the $\Pi\Gamma$-paths in it would turn it into a simple greatest real-knot. Hence C is also a semi-real-knot (and a semi-greatest-real-knot). (b) Component C is not a knot and hence it is not a semi-knot. On the other hand, closing all the $\Pi\Gamma$-paths in C would turn it into a greatest real-knot which is a super-real-knot since after deleting it component B would become a (greatest) real-knot. Hence C is not a semi-real-knot either.



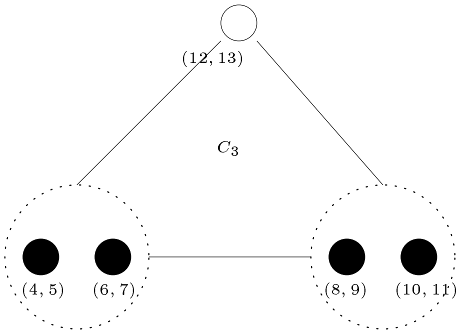
(a) The breakpoint graph of π_3



(b) The overlap graph of π_3

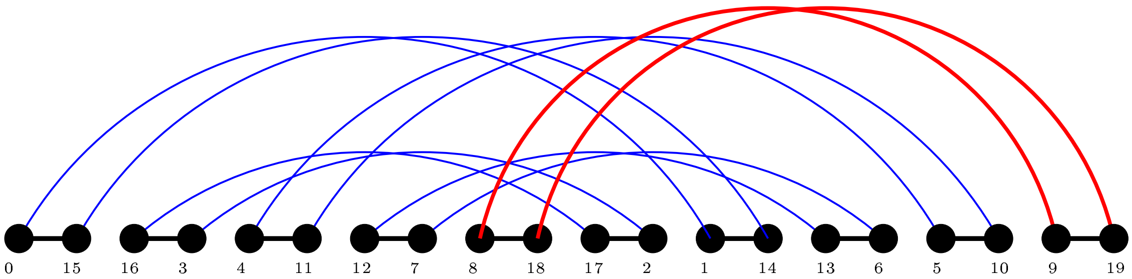


(c) The overlap graph of $\pi_3 \cdot r(0,1)$

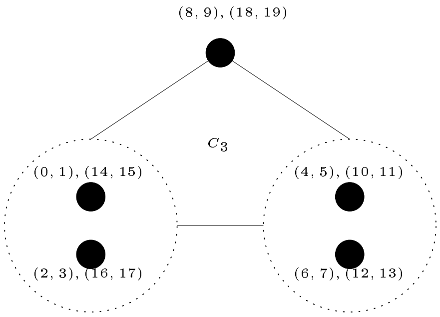


(d) The overlap graph of $\pi_3 \cdot r(0,1) \cdot r(2,3)$

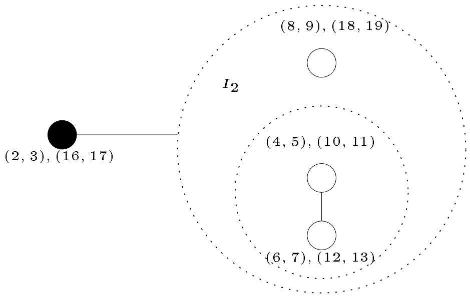
FIGURE 4. One cycle permutation. Each dotted circle denotes a module, i.e., a set of vertices with the same set of neighbors. The edges between circles connect every pair of members in the respective circles.



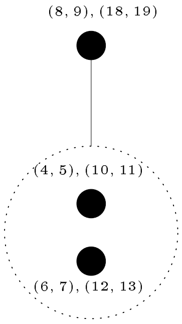
(a) The breakpoint graph of σ_2



(b) The interleaving graph of σ_2



(c) The interleaving graph of $\sigma_2 \cdot r(0, 1)$



(d) The interleaving graph of $\sigma_2 \cdot r(0, 1) \cdot r(2, 3)$

FIGURE 5. A simple permutation