Advanced Genome Rearrangement

Prepared by: *Dirk Cummings and Matt Carson*

Advanced Genome Rearrangement

Reducing a minimum sort by reversals (MIN-SBR) [7] to a purely graph-theoretic problem using breakpoint graphs, was first introduced by Vincent Bafna and Pavel Pevzner [1]. They were able to show estimating the reversal distances of breakpoints to be very inaccurate. In doing so, they exposed a second "hidden" parameter in which they were able to reveal important links between the maximum cycle decomposition of a graph and the reversal distance.

Bafna and Pevzner's work showed that by considering both the number of breakpoints and the number of alternating cycles (to be explained later), one could constrict the lower bound for the reversal distance. However, finding a maximum cycle decomposition is a difficult problem [5]. To overcome this difficulty, Hannenhalli and Pevzner expose another "hidden" parameter that allows one to compute the reversal distance between permutations in polynomial time. Their algorithm (presented at the end of these notes) is the first polynomial algorithm for a realistic model of genome rearrangements. [1].

Genome Rearrangement Using Graph Theory

The goals of reduction using graph theory are

1. Finding a maximally large family of edge-disjoint cycles, while
2. Minimizing the number of hurdles which this family of cycles defines

Breakpoint Graphs

Instead of representing reversals permutations as just simple arrays, we can connect elements with specified edges and use graph theory to improve evaluation performance.

Given an arbitrary reversal ρ, denote G' as:

$$G' = G(\pi\rho)$$

Let a breakpoint in π be

$$b = b(\pi),$$

and the number of breakpoints in G' be

$$b' = b(\pi\rho)$$

Recall the definition of a breakpoint:

Adjacencies and Breakpoint Let $\pi = \pi_1, \ldots, \pi_n$ be a permutation, $i \sim j$ if $|i - j| = 1$, and π_i, π_{i+1} be consecutive elements of π. The pair is defined as an *adjacency* if $\pi_i \sim \pi_{i+1}$ and a *breakpoint* if $\pi_i \not\sim \pi_{i+1}$ [1].

Cycle A sequence of vertices, $x_1 x_2 \ldots x_m = x_1$ is called a *cycle* in a graph $G(V, E)$ if $(x_i, x_{i+1}) \in E$ for $1 \leq i \leq m - 1$ [1]. The number of cylces in a maximum cycle decomposition of G' is defined as

$$c' = c(\pi\rho)$$

A *cycle* is an edge-coloured graph G is called *alternating* if the colours of every two consecutive edges of this cycle are distinct [1].

When building the graph, we join vertices i and j by a *black edge* if (i, j) is a *breakpoint* of π, or a *gray edge* if (i, j) are *not* consecutive in π [1]. If the verticies do not meet either criteria, they will not be connected by an edge.

The length of a cycle C, denoted by l(C), is the number of of black (or equivalently, gray) edges in it. A cycle C is short if l(C) = 2 and long if l(C) > 2.

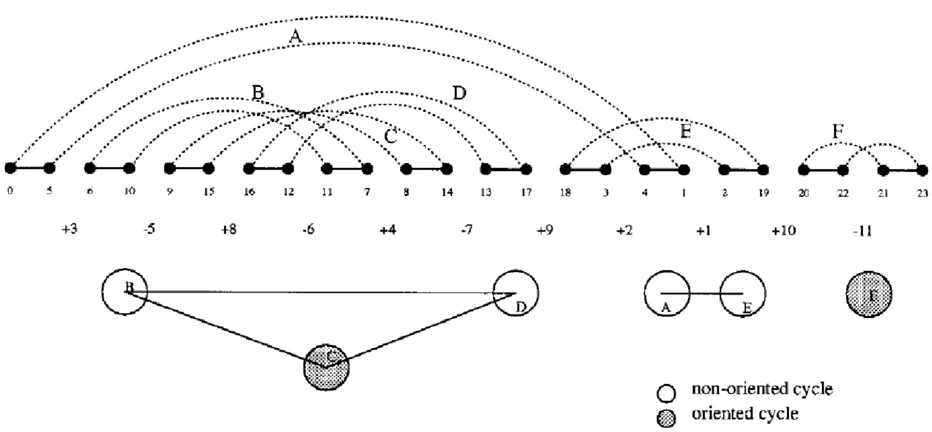

Figure 1: Interleaving breakpoint graph H_π with two oriented and one unoriented components: black edges connect adjacent vertices that are not consecutive, gray edges connect consecutive vertices that are not adjacent. [4]

Oriented and Unoriented Edges A gray edge g is *oriented* if a reversal acting on two black edges incident to g is proper and *unoriented*, otherwise.

Example A gray edge (8, 9) in Figure 1 is oriented (since a reversal acting on black edges (8, 14) and (9, 15) destroys two breakpoints and one cycle) while a gray edge (4, 5) is unoriented. To provide an intuition for the notion of an oriented edge, we state the following lemma:

Lemma 1 *Let (π_i, π_j) be a gray edge incident to black edges (π_k, π_i) and (π_j, π_l). Then (π_i, π_j) is oriented iff $i - k = j - l$* [4].

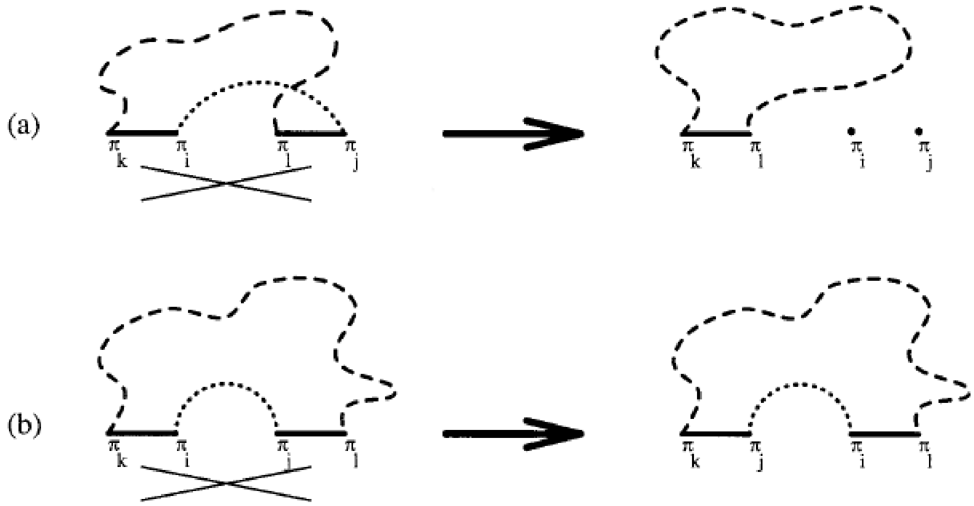

Figure 2: [4]

Proof Notice that $k = i \pm 1$ and $l = j \pm 1$. If $i - k = j - l$, then either $k = i - 1$, $l = j - 1$ or $k = i + 1$, $l = j + 1$ (Figure 2(a)). Clearly $\Delta(b - c) = -1$, hence, the reversal acting on (π_i, π_j) is proper. If $i - k \neq j - l$, then either $k = i - 1$, $l = j + 1$ or $k = i + 1$, $l = j - 1$ (Figure 2(b)). In this case, $\Delta(b) = 0$ and $\Delta(c) = 0$; hence, the reversal acting on (π_i, π_j) is not proper [4].

Oriented and Unoriented Cycles A cycle in $G(\pi)$ is *oriented* if it has an oriented gray edge and *unoriented*, otheriwse.

Example Cycles C and F in Figure 1 are oriented while cycles A, B, D, and E are unoriented. Clearly, there is no proper reversal acting on an unoriented cycle. It is esay to see that a permutation has a proper reversal iff it has an oriented cycle [4].

Interleaving Edges and Cycles

Grady edges (π_i, π_j) and (π_k, π_t) in G(π) are interleaving if the intervals $[i, j]$ and $[k, t]$ overlap, but neither of them contains the other. For example, edges (4,5) and (18,19) in Figure 1 are interleaving while edges (4, 5) and (22, 23) are noninterleaving. Two cycles C_1 and C_2 are interleaving if there exist interleaving gray edges $g_1 \in C_1$ and $g_2 \in C_2$ [4].

Hurdles and Fortresses

In breakpoint graphs, *hurdles* present obstacles in the gnome rearrangement problem. Each additional *hurdle* increases the number of required reversals (the removal of breakpoints) for a given permutation π when transforming into the identity permutation. By considering *hurdles* in our computations we tighten the bound on the reversal distance problem (see Calculating the Reversal Distance).

Hurdle An unoriented cycle component that does *not* seperate two other unoriented cycle components [7].

Simple Hurdle Not a *Super Hurdle* (see Super Hurdle).

Super Hurdle A hurdle which protects a non-hurdle from being a hurdle itself. Deleting a *Super Hurdle* would case the protected non-hurdle to become a hurdle [4].

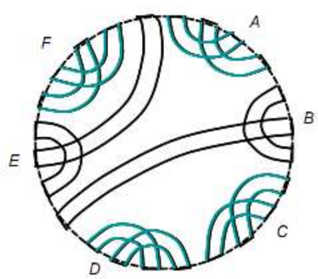

Figure 3: Simple and Super Hurdles [6]. Cycles A and F are both super hurdles becuase they protect cycles B and E (respectively) from becoming hurdles. While cycles D and C are both simple hurdles.

Fortress A breakpoint graph with an odd number of hurdles which are *all Super Hurdles* [4].

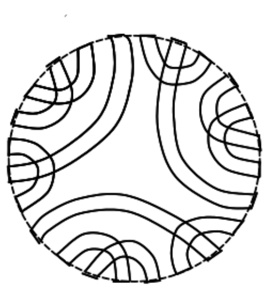

Figure 4: A 3-Fortress [6]

Calculating the Reversal Distance

Theorem 1

Bafna and Pevzner proved that Δ(b-c) $\in${-1, 0, 1}.

(a) If Δ(b - c) = 1, then Δ(b - c + h) $\geq$ -1 since Δh $\geq$ -2 for every reversal ρ

(b) If Δ(b-c) = 0, then ρ acts on a cycle and therefore it affects at most one

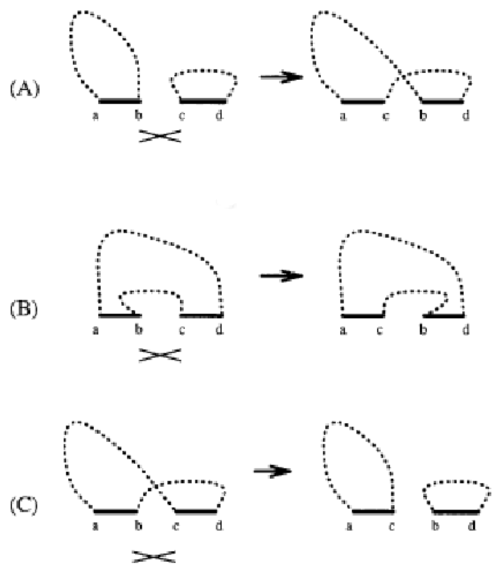

Figure 5: (a) For reversals acting on two cycles, Δ(b - c) = 1. (b) For reversals acting on an unoriented cycles, Δ(b - c) = 0. (c) For reversals acting on an oriented cycles, Δ(b - c) = -1.

hurdle. It implies Δh $\geq$ -1 and Δ(b - c + h) $\geq$ -1

(c) If Δ(b - c) = -1, then ρ acts on an oriented cycle and hence it does not destroy any hurdles in π. Therefore, Δh $\geq$ 0 and Δ(b - c + h) $\equiv$ Δb - Δc + Δh $\geq$ -1

Therefore, for an arbitrary reversal ρ, Δ(b - c + h) $\geq$ - 1 thus implying d(π) $\geq$ b(π) - c(π) + h(π) [4].

Safe Reversals A reversal which does *not* create more hurdles or new unoriented components. A reversal ρ is a safe reversal if

$$\Delta(b - c + h) = -1$$

Equivalent Transformations of Permutations

Previous studies have revealed that the complicated interleaving structure of long cycles in breakpoint graphs poses serious difficulties in analyzing sorting by reversals and transpositions.

To get around this problem, we introduce equivalent transformations of permutations. If a permutation $\pi \equiv \pi(0)$ has a long cycle, transform it into a new permutation $\pi(1)$ by "breaking" this long cycle into two smaller cycles. Continue with $\pi(1)$ in the same manner until you have filtered out all the long cycles [4].

In order to accomplish this task of "breaking" long cycles, we introduce the notion of (g, b)-splits and padding.

Let b = (v_b, w_b) be a black edge and g = (w_g, v_g) be a gray edge belonging

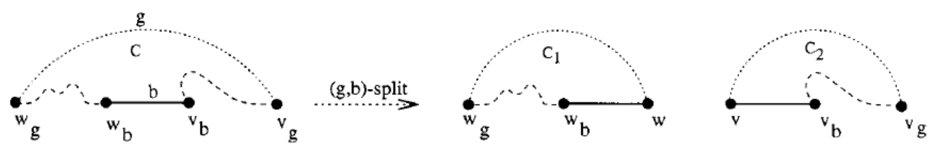

Figure 6: Example of a (g, b)-split.

to a cycle C = ..., v_b, w_b, ..., w_g, v_g, ... in the breakpoint graph G(π) of a permutation π. A (g, b)-split is a new graph G'(π) obtained from G(π) by:

- Removing edges g and b,
- Adding two new vertices v and w,
- Adding two new black edges (v_b, v) and (w, w_b),
- Adding two new gray edges (w_g, w) and (v, v_g) [4]

Figure 6 shows a (g, b)-split transforming a cycle C in G(π) into cycles C_1 and C_2 in G'(π).

By inducing a (g, b)-split into our graph, we have created what is known as (g, b) padding.

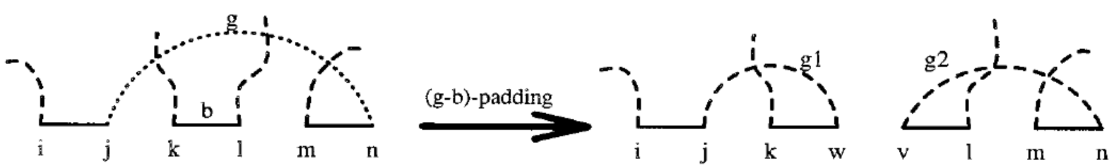

Figure 7: A (g, b)-padding deletes an oriented edge g and adds an oriented edge g_1 and unoriented edge g_2

Let b = (π_{i+1}, π_i) be a black edge and g = (π_j, π_k) be a gray edge belonging to a cycle C = ..., π_{i+1}, π_i, ..., π_j, π_k, ... in the breakpoint graph G(π). Define $\Delta = \pi_j - \pi_k$, and let v = $\pi_j + (\Delta/3)$, let w = $\pi_k + (\Delta/3)$. A (g, b)-padding of $\pi = (\pi_1\pi_2 \ldots \pi_i$ v w $\pi_{i+1} \ldots \pi_n)$ is a permutation of n + 2 elements obtained from π by inserting v and w after the i^{th} element of $\pi(0 \le i \le n)$:

$$\pi' = (\pi_1\pi_2 ... \pi_i v w \pi_{i+1} ... \pi_n).$$

Note that v and w are both consecutive and adjacent in π'. The (g, b)-split of Figure 6 corresponds to (g, b)-padding for g = (w_g, v_g) and b = (v_b, w_b) [4].

Reversal Sort Polynomial Algorithm

ALGORITHM Reversal_Sort(π)
begin
  while π is not sorted
    if π has a long cycle **then**
      select a safe $(g, b)-$padding ρ of π
    else if π has an oriented component **then**
      select a safe reversal ρ in this component
    else if π has an even number of hurdles **then**
      select a safe reversal ρ merging two hurdles in π
    else if π has at least one simple hurdle **then**
      select a safe reversal ρ cutting this hurdle in π
    else if π is a fortress with more than three superhurdles **then**
      select a safe reversal ρ merging two (super)hurdles in π
    else /* π is a 3-fortress */
      select an (un)safe reversal ρ merging two arbitrary (super)hurdles in π
    $\pi \leftarrow \pi\dot{}\rho$
  end while
end

Example Run of Reversal Sort Algorithm

Using the predefined set of permutations in Figure 1, we will trace through the reversal sort algorithm.

Step 1 - π has an oriented component

$[0, 5, 6, 10, 9, \mathbf{15, 16, 12, 11, 7, 8}, 14, 13, 17, 18, 3, 4, 1, 2, 19, 20, 22, 21, 23]$

Cycles :
(0, 1, 4, 5, 0) - Simple Hurdle,
(6, 7, 11, 10, 6) - Simple Hurdle,
(9, 8, 14, 15, 9) - Oriented Cycle,
(18, 19, 2, 3, 18) - Simple Hurdle,
(16, 17, 13, 12, 16) - Simple Hurdle,
(20, 21, 23, 22, 20) - Oriented Cycle

Reversal: **15, 16, 12, 11, 7, 8 → 8, 7, 11, 12, 16, 15**

Step 2 - π has an oriented component

$[0, 5, 6, 10, 9, 8, 7, 11, 12, 16, 15, 14, 13, 17, 18, 3, 4, 1, 2, 19, 20, \mathbf{22, 21}, 23]$

Cycles :
(0, 1, 4, 5, 6) - Simple Hurdle,
(6, 7, 11, 10, 6) - Oriented Cycle,
(12, 13, 17, 16, 12) - Oriented Cycle,
(18, 19, 2, 3, 18) - Simple Hurdle

Reversal: **22, 21 → 21, 22**

Step 3 - π has an oriented component

$[0, 5, 6, \mathbf{10,\ 9,\ 8,\ 7}, 11, 12, 16, 15, 14, 13, 17, 18, 3, 4, 1, 2, 19, 20, 21, 22, 23]$

Cycles :
(0, 1, 4, 5, 0) - Simple Hurdle,
(6, 7, 11, 10, 6) - Oriented Cycle,
(12, 13, 17, 16, 12) - Oriented Cycle,
(18, 19, 2, 3, 18) - Simple Hurdle

Reversal: **10, 9, 8, 7 $\rightarrow$ 7, 8, 9, 10**

Step 4 - π has an oriented component

$[0, 5, 6, 7, 8, 9, 10, 11, 12, \mathbf{16,\ 15,\ 14,\ 13,} 17, 18, 3, 4, 1, 2, 19, 20, 21, 22, 23]$

Cycles :
(0, 1, 4, 5, 0) - Simple Hurdle,
(12, 13, 17, 16, 12) - Oriented Cycle,
(18, 19, 2, 3, 18) - Simple Hurdle

Reversal: **16, 15, 14, 13 $\rightarrow$ 13, 14, 15, 16**

Step 5 - π has an oriented component

$[0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, \mathbf{3,\ 4,\ 1,\ 2,\ 19}, 20, 21, 22, 23]$

Cycles :
(0, 1, 4, 5, 0) - Simple Hurdle,
(18, 19, 2, 3, 18) - Simple Hurdle

Reversal: **3, 4, 1, 2, 19 $\rightarrow$ 19, 2, 1, 4, 3**

Step 6 - π has an even number of hurdles

$[0, \mathbf{5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13,\ 14,\ 15,\ 16,\ 17,\ 18,\ 19,\ 2,\ 1}, 4, 3, 20, 21, 22, 23]$

Cycles :
(0, 1, 4, 5, 0) - Oriented Cycle,
(19, 20, 3, 2, 19) - Simple Hurdle

Reversal: **5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 1 $\rightarrow$
1, 2, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5**

Step 7 - π has an oriented component

$[0, 1, 2, \mathbf{19,\ 18,\ 17,\ 16,\ 15,\ 14,\ 13,\ 12,\ 11,\ 10,\ 9,\ 8,\ 7,\ 6,\ 5,\ 4,\ 3}, 20, 21, 22, 23]$

Cycles :
(2, 3, 20, 19, 2) - Oriented Cycle

Reversal: **19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3 $\rightarrow$**

3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19

Step 8 - π is sorted

$[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]$

References

[1] V. Bafna and P. A. Pevzner, "Genome rearrangements and sorting by reversals," 1993.

[2] S. Borgatti, "Relations, graphs and matrices formal representation of social network data," 2008. [Online]. Available: http://www.analytictech.com/networks/graphs.htm

[3] S. Hannenhalli, C. Chappey, E. V. Koonin, and P. A. Pevzner, "Genome sequence comparison and scenarios for gene rearrangements: A test case," *Genomics*, vol. 30, pp. 299–311, 1995.

[4] S. Hannenhalli and P. Pevzner, "Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals)," in *Journal of the ACM*. ACM Press, 1995, pp. 178–189.

[5] J. Kececioglu and D. Sankoff, "Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement," 1995.

[6] S. Mneimneh, "Computational biology - lecture 16: Genome rearrangements, sorting by reversals," pp. 6–7. [Online]. Available: http://www.cs.hunter.cuny.edu/ saad/courses/compbio/lectures/lecture16.pdf

[7] M. K. Piotr Berman, Sridhar Hannenhalli, "1.375-approximation algorithm for sorting by reversals," vol. 2461, pp. 200–210, 2002.