RECOVERY OF ANCESTRAL TETRAPLOIDS

Nadia El-Mabrouk

Methods to recover an ancestral pre-duplicated genome giving rise to a minimal number of genomic rearrangements. Rearrangement operations considered are reversals and translocations. We obtain exact algorithms running in polynomial time.

1 Introduction

Originally a duplicated genome contains two identical copies of each chromosome, but through reversals or other intrachromosomal movements, the gene orders in each copy of a chromosome change independently, and through reciprocal translocation, the identical synteny of the two copies is disrupted. Eventually, all that can be detected are several chromosome segments, each of which appears twice in the genome, containing a series of paralogous genes in parallel orders. The problem considered here is to reconstruct some or most of the original gene order at the time of genome duplication, based on traces conserved in the ordering of those duplicate genes still identifiable, i.e., still recognizably homologous despite evolutionary divergence in nucleotide sequence. The complexity of this problem and the choice of methods depend on the type of rearrangement operations considered and the kind of biological data available. We previously proposed a suite of "Genome halving" problems [2] and offered an algorithm for one of them involving (set-theoretical) relations of synteny only. We then considered the problem of ordered and oriented genes. In the case of circular genomes, we presented a polynomial algorithm that minimizes (within 2 reversals) the number of reversals required to recover a potential ancestral genome at the moment of genome duplication [3]. In the case of multichromosomal genomes and when rearrangement operations considered are translocations, we also found a polynomial-time algorithm, exact in all but certain identifiable cases [1]. The present paper builds on a this preliminary result. We first show how to obtain a polynomial algorithm, exact in all cases, for translocations. We then generalize the method to find an exact polynomial solution, where both reversals and translocations are allowed.

2 Formalizing the problem

A **string** is a sequence of signed ($+$ or $-$) terms (**genes**) from a set $\mathcal{B}$. A **rearranged duplicated genome** G is a collection of non-null strings, $C_1, \cdots, C_{2N}$, (**chromosomes**), such that each gene in $\mathcal{B}$ is present exactly twice, i.e., once in each of two different chromosomes, or twice in a single chromosome.

In this abstract, all genomes are multichromosomal, and rearrangement operations considered are reversals and translocations. A reversal is an operation transforming a substring $X = x_1 x_2 \cdots x_r$ of a chromosome into its **reverse** $(-X) = -x_r - x_{r-1} \cdots - x_1$. Let X_1, X_2, Y_1 and Y_2 be non-null strings. A **reciprocal translocation** between two chromosomes $X = X_1X_2$ and $Y = Y_1Y_2$ is of form $X_1X_2, Y_1Y_2 \longrightarrow X_1Y_2, Y_1X_2$ (prefix-prefix) or of form $X_1X_2, -Y_2 - Y_1 \longrightarrow X_1 - Y_1, -Y_2X_2$ (prefix-suffix) (see Figure 1).

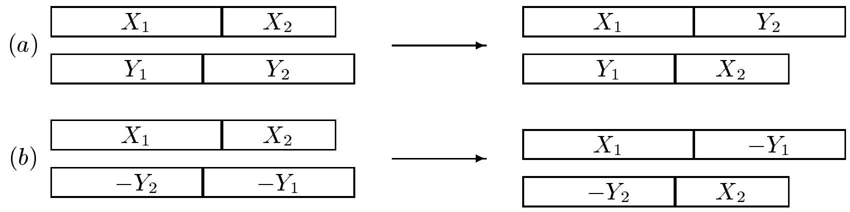

Figure 1: Reciprocal translocation between two chromosomes X_1X_2 and Y_1Y_2; (a): prefix-prefix translocation; (b): prefix-suffix translocation.

The problem is to calculate the minimal number of rearrangement operations required to transform a given rearranged duplicated genome G into some **perfect duplicated genome** H (or just **duplicated genome**) to be found, consisting of $K_1, \cdots, K_{2M}$ chromosomes, where for each $i \in \{1, \cdots, 2M\}$, we have $K_i = K_j$ for exactly one $j \in \{1, \cdots, 2M\} \backslash \{i\}$.

3 The Hannenhalli-Pevzner theory

Given two genomes H_1 and H_2 containing the same gene set $\mathcal{B}$, where each gene appears exactly once in each genome, the genome rearrangement problem is to find the minimum number of rearrangement operations necessary to transform H_1 into H_2 (or H_2 into H_1). Hannenhalli and Pevzner (HP) designed polynomial algorithms for the reversals-only version of the problem (in the case of single-chromosome genomes) [5], the translocations-only version [4], and the version with both reversals and translocations [6] (the latter two for multichromosomal genomes). All these algorithms require that the genomes H_1 and H_2 share the same set of chromosomal endpoints, but this constraint can be avoided by adding dummy endpoints, if necessary. The algorithms all depend on a bicoloured unoriented graph $\mathcal{G}_{12}$ constructed from H_1 and H_2: If gene x of H_1 has positive sign, replace it by the pair $x^t x^h$, and if it is negative, by $x^h x^t$. Then the vertices of $\mathcal{G}_{12}$ are just the x^t and the x^h for all x in $\mathcal{B}$. Any two vertices which are adjacent in some chromosome in H_1, other than x^t and x^h from the same x, are connected by a black edge, and any two adjacent in H_2, by a gray edge. In the case of a single chromosome, the black edges may be displayed linearly according to the order of the genes in the chromosome. For a genome containing N chromosomes, N such linear orders are required. Furthermore, in the model allowing both reversals and translocations, the N orders are concatenated in each of the two genomes, so that we are again left with a single linear order. Now, each vertex is incident to exactly one black and one gray edge, so that there is a unique decomposition of $\mathcal{G}_{12}$ into c_{12} disjoint cycles of alternating edge colours. By the **size of a cycle** we mean the number of black edges it contains. Note that $c_{21} = c_{12} = c$ is maximized when $H_1 = H_2$, in which case each cycle has one black edge and one gray edge.

A rearrangement operation ρ, either a reversal or a translocation, is determined by the two points where it "cuts" the current genome, which correspond to two black edges $e = (a, b)$ and $f = (c, d)$. We say that ρ **is determined by the two black edges** e and f. Rearrangement operations may change the number of cycles, so that minimizing the number of operations can be seen in terms of increasing the number of cycles as fast as possible. Let $\mathcal{G}$ be a cycle graph, ρ a rearrangement operation, and $\Delta(c)$ the difference between the number of cycles before and after applying the operation ρ. Hannenhalli and Pevzner showed that $\Delta(c)$ may take on values 1, 0 or -1, in which cases they called ρ **proper, improper** or **bad**, respectively. Rughly, an operation determined by two black edges in two different cycles will be bad, while one acting on two black edges within the same cycle may be proper or improper, depending on the type of cycle and the type of edges considered. Two cycles, say Cycles 1 and 2, all of whose black edges are related by the same linear order (i.e. are on the same line), and containing gray edges that "cross", e.g., gene i linked to gene j by a black edge (i.e. in H_1) in Cycle 1, gene k linked to gene t by a black edge in Cycle 2, but ordered i, k, j, t in H_2, are connected. A component of $\mathcal{G}_{12}$ is a subset of the cycles (not consisting of a single cycle of size 1), built recursively from one cycle whose edges are all related by the same linear order, at each step adding all the remaining cycles connected to any of those already in the construction. A component is termed **good** if it can be transformed to a set of cycles of size 1 by a series of proper operations, and **bad** otherwise. The HP formulae for all three models may be summarized as follows:

$$RO(G, H) = b(\mathcal{G}_{12}) - c(\mathcal{G}_{12}) + m(\mathcal{G}_{12}) + f(\mathcal{G}_{12}) \quad \textbf{(1)}$$

where $D(G, H)$ is the minimum number of rearrangement operations, $b(\mathcal{G}_{12})$ is the number of black edges in $\mathcal{G}_{12}$, $c(\mathcal{G}_{12})$ is the number of cycles, $m(\mathcal{G}_{12})$ is the number of bad components, and $f(\mathcal{G}_{12})$ is a correction of size 0, 1 or 2.

Bad components are called *subpermutations* in the translocations-only model, *hurdles* in the reversals-only model, and *knots* in the combined model. For more details on formula (1) and bad components, see [5, 6], or the clear summary for reversals only in [7].

4 Preliminaries

To make use of the Hannenhalli graph structure for the genome halving problem, we first introduce, arbitrarily, a distinction within each pair of identical blocks in the rearranged duplicated genome G, labeling one occurrence x_1 and the other x_2, for all x in $\mathcal{B}$. Next, to each chromosome C_i, we add new initial and final terms $+O_{i1}$ and $+O_{i2}$. This releases the erstwhile initial and final genes on each chromosome from their constraint in the Hannenhalli formulation and ensures that all translocations, including those which reduce (by **fusion**, e.g. null $X_1 Y_2$) or augment (by **fission**, e.g. null $X_1 X_2$) the number of chromosomes in the genome, can be treated as reciprocal translocations. This allows us to treat the case of genomes with an odd number of chromosomes, by adding one dummy chromosome, consisting of just one initial and one final O, to obtain $2N$ chromosomes. Moreover, in each chromosome, each x_j (except the O_{ij}) is replaced by x_j^t and x_j^h as in the Hannenhalli construction. Define: $O = \{O_{i1}, O_{i2}\}_{i=1,\cdots,2N}$, $V = \{x_j^s\}^{s\in\{h,t\}}_{\substack{x\in\mathcal{B}\\ j=1,2}}$, $\mathbf{V} = O \cup V$. We use the notation $\overline{1} = 2$, $\overline{2} = 1$, $\widetilde{t} = h$, $\widetilde{h} = t$. For $u = x_j^s \in V$, its **counterpart**, denoted $\overline{u}$, is $x_{\overline{j}}^s$, and its **obverse**, denoted $\widetilde{u}$, is $x_j^{\widetilde{s}}$. The **partial graph** $\mathcal{G}(\mathbf{V}, A)$ associated with G, has the edge set A of (black) undirected edges linking adjacent terms (other than obverses) in G (see example, Figure 2).

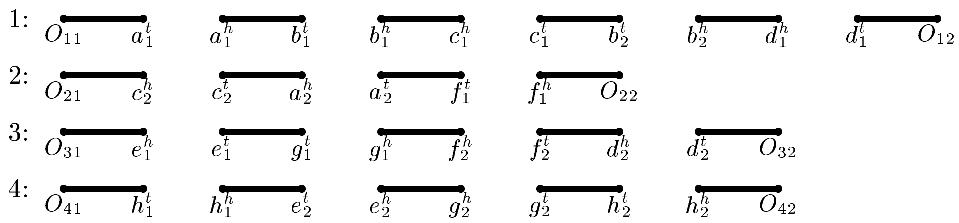

Figure 2: The partial graph $\mathcal{G}(\mathbf{V}, A)$ corresponding to the rearranged duplicated genome G made up of the four chromosomes: 1: $+a\ +b\ -c\ +b\ -d$; 2: $-c\ -a\ +f$; 3: $-e\ +g\ -f\ -d$; 4: $+h\ +e\ -g\ +h$. To differentiate the two occurrences of each block x, one is subscripted "1", its counterpart "2".

The goal is to add to this partial graph a set D of gray undirected edges corresponding to some perfect duplicated genome H, so that every vertex in $\mathbf{V}$ is incident to exactly one black edge and one gray edge. The graph $\mathcal{G}_D(\mathbf{V}, A, D)$ thus produced is called a **completed graph** of $\mathcal{G}(\mathbf{V}, A)$. Lemma 1 clarifies the constraints that gray edges should satisfy to represent a duplicated genome.

Lemma 1 *D contains no edge of form $(u, \overline{u})$, for any $u \in V$. Moreover, suppose $(u, v) \in D$ and $v \in V$. If $u \in V$ then $(\overline{u}, \overline{v}) \in D$. Otherwise ($u \in O$), $\overline{v}$ is also linked by a gray edge to some element of O.*

Despite some differences, methods required to solve each of the rearrangement problems treated in this paper (translocations or translocations plus reversals) are based on a similar cycle graph and have similar fundamental steps. The first step of the method is to find an edge set D that maximizes the number of cycles $c(\mathcal{G}_D)$ of a completed graph $\mathcal{G}(\mathbf{V}, A, D)$; such a graph is called a **maximal completed graph**. A decomposition of $\mathcal{G}(\mathbf{V}, A)$ into a particular set of subgraphs provides an upper bound for $c(\mathcal{G}_D)$. In Section 5, we sketch the properties of a suitable decomposition as previously detailed in [1]. Section 6 describes an algorithm for constructing a maximal completed graph.

The second step of the general method consists in minimizing the number of bad components. Section 7 describes a general approach for both the translocations and the translocations plus reversals problems. We take account of the differences between the two rearrangement problems in Sections 8 and 9.

5 Upper bound on the number of cycles

Definition 1 *Let $e = (u, v) \in A$. Define A_e recursively by: $(u, v) \in A_e$; If $(x, y) \in A_e$ and $x \notin O$, then the edge of A adjacent to $\overline{x}$ is also in A_e (the same holds for y). Let $\mathbf{V}_e$ be the subset of $\mathbf{V}$ made up of vertices incident to the edges in A_e. Then $\mathcal{G}_e(\mathbf{V}_e, A_e)$ is the* **natural graph** *(of* **size** *$|A_e|$) of $\mathcal{G}(\mathbf{V}, A)$ generated by e. Note that if $f \in A_e$, then $A_f = A_e$*

Let GCE be the set of natural graphs of even size, and GCO the set of natural graphs of odd size of $\mathcal{G}(\mathbf{V}, A)$. We further subdivide GCO into $\mathcal{CO}_+$ and $\mathcal{CO}_-$ according to whether the natural graphs include vertices in O or not. Graphs in GCO are amalgamated as follows: arbitrarily choose pairs of graphs in $\mathcal{CO}_+$ to amalgamate; arbitrarily choose pairs of the remaining graphs to amalgamate. Let $\mathcal{CE} = GCE \cup \mathcal{CO}_+$. Graphs of $\mathcal{CE} \cup \mathcal{CO}_-$ are called **supernatural graphs**.

In the ensuing discussion, we start with a decomposition of $\mathcal{G}(\mathbf{V}, A)$ into a set $\mathcal{SS}$ of supernatural graphs. The number of graphs of $\mathcal{CE}$ is the same for any such decomposition. We denote $\alpha(G) = |\mathcal{CE}|$ (see, for an example, Figure 3). The following theorem is proved in [1].

Theorem 1 *Let $\mathcal{G}_D(\mathbf{V}, A, D)$ be a completed graph of the partial graph $\mathcal{G}(\mathbf{V}, A)$. Let $c(\mathcal{G}_D)$ be the number of cycles of $\mathcal{G}_D(\mathbf{V}, A, D)$. Then, $c(\mathcal{G}_D) \leq \frac{|A|}{2} + \alpha(G)$.*

6 Maximizing the number of cycles

The main characteristic of the algorithm is that each gray edge is constructed so that it links two vertices of the same supernatural graph. At each step, we denote by F the set of fragments of the genome H resulting from the preceding steps. At the outset, F is made up of the **unitary fragments**, which include not only $x^t x^h$, for all $x \in \mathcal{B}$, but also the $2N$ elements of O. As the construction proceeds, whenever a gray edge (x, y) is created, the fragment containing x and the one containing y are joined together. An **intermediate fragment** is unitary,

$$\begin{array}{lll}
\mathcal{S}_1{:}O_{11} & \text{———} & a_1^t \\
f_1^t & \text{———} & a_2^t \\
f_2^t & \text{———} & d_2^h \\
b_2^h & \text{———} & d_1^h \\
b_1^h & \text{———} & c_1^h \\
O_{21} & \text{———} & c_2^h
\end{array}
\quad
\begin{array}{lll}
\mathcal{S}_2{:}a_1^h & \text{———} & b_1^t \\
a_2^h & \text{———} & c_2^t \\
c_1^t & \text{———} & b_2^t \\
 & & \\
\mathcal{S}_3{:}d_1^t & \text{———} & O_{12} \\
d_2^t & \text{———} & O_{32}
\end{array}
\quad
\begin{array}{lll}
\mathcal{S}_4{:}f_1^h & \text{———} & O_{22} \\
f_2^h & \text{———} & g_1^h \\
e_2^h & \text{———} & g_2^h \\
e_1^h & \text{———} & O_{31}
\end{array}
\quad
\begin{array}{lll}
\mathcal{S}_5{:}e_1^t & \text{———} & g_1^t \\
e_2^t & \text{———} & h_1^h \\
h_2^t & \text{———} & g_2^t \\
h_1^t & \text{———} & O_{41} \\
O_{42} & \text{———} & h_2^h
\end{array}$$

Figure 3: Natural graphs of the partial graph $\mathcal{G}(\mathbf{V}, A)$ of Figure 2. $\mathcal{S}_2$ and $\mathcal{S}_5$ are amalgamated to form a supernatural graph.

or has a unique endpoint in O. A **terminal fragment** has both its endpoints in O. The algorithm starts with two counterpart vertices (u and $\overline{u}$) in O, and extends them in a symmetric way, until two terminal fragments of form X and $\overline{X}$ are obtained. It then proceeds with two other vertices of O, and so on, until $2N$ terminal fragments, corresponding to the $2N$ chromosomes of a perfect duplicated genome, are obtained.

Let x, y be two distinct vertices, not yet linked by gray edges, such that $x \in V$ and x is one end of the current intermediate fragment. To be able to construct the edge (x, y), we must have the following three conditions satisfied: 1. $y \neq \overline{\widetilde{x}}$; 2. $y \neq \overline{x}$; 3. If $y \in O$ and F contains at least one unitary fragment in V, then F must contain at least four unitary fragments in O (including y).

As the algorithm proceeds by extending an intermediate fragment, when a vertex $x \in V$ is considered, a gray edge adjacent to $\widetilde{x}$ (and thus also to $\overline{\widetilde{x}}$) is already constructed. Therefore, Condition 1 is always satisfied. So that a pair of vertices satisfying the above conditions can always be found, we must avoid, in constructing a gray edge (x, y), where y is the vertex being inserted in the intermediate fragment, provoking an **impossible configuration** characterized by one of the following three properties. A configuration which avoids all of these properties is termed **possible**.

Property I. The only remaining unitary fragments are in V (i.e. none of them is in O).

Property II. The supernatural graph containing $\widetilde{y}$ contains two unitary fragments in O, and two unitary fragments in V, while F contains other fragments in V, but no other fragments in O.

Property III. The second endpoint of the path containing $z = \widetilde{y}$ is $\overline{z}$. Moreover, the supernatural graph containing this path contains another path C_2 with endpoints t and $\overline{t}$, where (z, t) creates an impossible situation in terms of Property I or II.

A pair of vertices (x, y) is said **impossible** if it creates an impossible configuration, and **possible** otherwise. If (x, y) is possible, then so are $(x, \overline{y})$, $(\overline{x}, y)$

and $(\overline{x}, \overline{y})$. Let $\mathcal{G}$ be a partially completed graph obtained at some step of the algorithm. Let x be a vertex of $\mathbf{V}$ not yet linked by a gray edge. Then x is one of the two endpoints of a path C (made up of a succession of black and gray edges) of $\mathcal{G}$, completely contained in the supernatural graph of x. Call the second endpoint of this path x^c. Note that, in general, $\overline{x}^c$ is different from $\overline{x^c}$. Figure 4 contains an algorithm for constructing a completed graph $\mathcal{G}_D(\mathbf{V}, A, D)$. Note that we do not explicitly repeat the fact, each time a gray edge (x, y) is created, this implies the creation of $(\overline{x}, \overline{y})$. (See the example in Figure 5.)

Algorithm dedouble:
While $\mathcal{G}$ contains unitary fragments do
  Choose a vertex x corresponding to a unitary fragment of O;
  Do
    If (x, x^c) is possible
      Construct the edge (x, x^c)
      Set $x = \widetilde{x^c}$;
    Else
      If x and $\overline{x}$ belong to two different paths of $\mathcal{G}$
        Construct the edge $(\overline{x}, \overline{x}^c)$
        Set $x = \widetilde{\overline{x}^c}$;
      Else, the supernatural graph $\mathcal{G}'$ containing x belongs to $\mathcal{CO}_-$. Let $\mathcal{G}'_1$ and $\mathcal{G}'_2$ be the two natural graphs amalgamated to form $\mathcal{G}'$, with $x \in \mathcal{G}'_1$.
        If there exist only two vertices y and $\overline{y}$ in $\mathcal{G}'_2$ not yet linked by gray edges
          Construct the edge (x, y);
          Set $x = \widetilde{y}$;
        Else, there must be vertices y, $\overline{y}$, z, $\overline{z}$ of $\mathcal{G}'_2$ not yet linked by gray edges
          If (x, y) is possible
            Construct the edge (x, y);
            Set $x = \widetilde{y}$;
          Else
            Construct the edge (x, z);
            Set $x = \widetilde{z}$;
  While $x \notin O$

Figure 4: Algorithm for constructing a maximal completed graph.

Theorem 2 *The algorithm dedouble constructs a maximal completed graph $\mathcal{G}_D(\mathbf{V}, A, D)$, containing $c(\mathcal{G}) = \alpha(G) + \frac{|A|}{2}$ cycles.*

(A full version of this paper will contain complete proofs.)

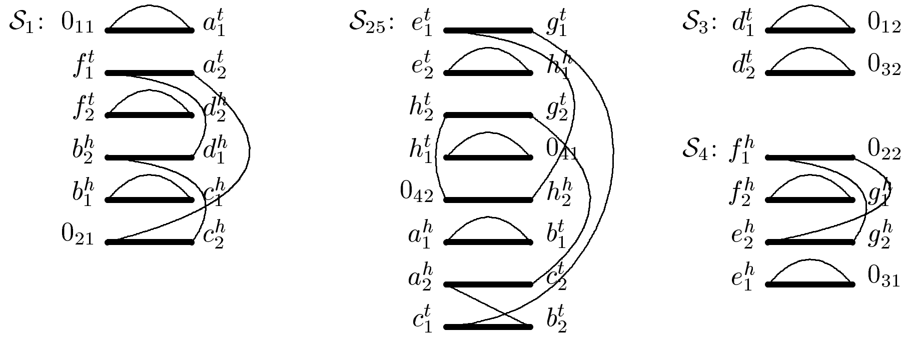

Figure 5: The completed graph $\mathcal{G}_D(\mathbf{V}, A, D)$ obtained by algorithm *dedouble* applied to the supernatural graphs of Figure 3. The corresponding duplicated genome H is made up of the four chromosomes: 1: $0_{11} + a_1 \ + b_1 \ - c_1 \ + g_1 \ - f_2 \ - d_2 \ 0_{32}$; 2: $0_{21} + a_2 \ + b_2 \ - c_2 \ + g_2 \ - f_1 \ - d_1 \ 0_{12}$; 3: $0_{22} \ - e_2 - h_1 \ 0_{41}$; 4: $0_{31} \ - e_1 - h_2 \ 0_{42}$.

7 Bad components

Now, we turn our attention to minimizing the number of bad components of a completed graph. Although the notion of bad components for rearrangement by translocations differs from that of rearrangement by translocations plus reversals, they are both related to the notion of subpermutation. Given two genomes H_1 and H_2 containing the same gene set, where each gene appears exactly once in each genome, a **subpermutation** (SP) of H_1 and H_2 (or, alternately, of the cycle graph $\mathcal{G}_{12}$ associated with H_1 and H_2) are subsequences $S = u_1 u_2 \cdots u_{p-1} u_p$ of a chromosome X of H_1 and $T = u_1 v_2 \cdots v_{p-1} u_p$ of a chromosome Y of H_2, such that for some permutation P, we have $T = P(S)$ with $v_2 \neq u_2$ and $v_{p-1} \neq u_{p-1}$. A **minimal subpermutation** (minSP) is an SP not containing any other SP, and a **maximal subpermutation** (maxSP) is an SP not included in any other SP.

In a cycle graph $\mathcal{G}_{12}$ associated with H_1 and H_2, a component C can be identified as a fragment $I = [u_l, u_r]$ where u_l and u_r are the endpoints of C, such that no gray edge links a vertex of I to a vertex outside of I, and such that at least one cycle of I is of size > 1. We call I **the interval of the component** C. We also identify a **minimal component** as a component whose interval contains no other component. There is a bijection between the SPs of $\mathcal{G}_{12}$ and the components of $\mathcal{G}_{12}$. Specifically, let S be an SP of $\mathcal{G}_{12}$, $\Pi = \{\pi_1, \cdots \pi_p\}$ the set of components of $\mathcal{G}_{12}$ containing the vertices of S, and for all i, $\mathbf{V_i}$ the set of vertices of π_i. Then:

- S_i is an SP contained in S (inner SP of S, possibly S itself) if and only if S_i corresponds to an interval of a component π_i of Π. We call this component **the component of the SP** S_i.
- S_i is a minimal inner SP of S if and only if S_i corresponds to an interval of a minimal component of Π.

(See the example in Figure 6.)

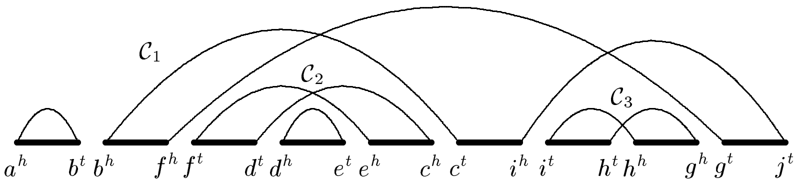

Figure 6: A subgraph of an HP graph corresponding to the fragments F_1 of genome H_1 and F_2 of genome H_2: $F_1 = +a+b-f+d+e-c-i+h-g+j$; $F_2 = +a+b+c+d+e+f+g+h+i+j$. $\mathcal{C}_1$ is the component of the SP $S_1 = +b-f+d+e-c-i+h-g+j$; $\mathcal{C}_2$ is the component of the SP $S_2 = -f+d+e-c$; $\mathcal{C}_3$ is the component of the SP $S_3 = -i+h-g$. $\mathcal{C}_2$ and $\mathcal{C}_3$ correspond to two minSP.

For the problem of rearrangement by translocations, all minSP are bad components of the HP graph. More precisely, if $s(\mathcal{G}_{12})$ is the number of minSP of $\mathcal{G}_{12}$, then, in the HP formula (1), $m(\mathcal{G}_{12}) = s(\mathcal{G}_{12})$. On the other hand, for the problem of rearrangement by reversals plus translocations, only certain intrachromosomal minSPs and maxSPs, **knots** [6], correspond to bad components.

Returning to our genome halving problem, we want to determine the minimum number of such bad SP in a completed graph of $\mathcal{G}(\mathbf{V}, A)$. For this, we introduce a new notion.

Definition 2 *Let $S = x_1 x_2 \cdots x_{n-1} x_n$ be a subsequence of a chromosome of G. S is a* **local SP** *of G if S is a* **real local SP** *or a* **potential local SP***, namely:*

- *S is a real local SP of G if $\{x_1, \cdots, x_n\} \cap O = \emptyset$, and there exists another subsequence of a chromosome of G of form $\overline{S} = \overline{u_1} P(\overline{u_2}, \cdots, \overline{u_{p-1}}) \overline{u_p}$, where P is a permutation other than the identity.*

- *S is a potential local SP if either:* **i.** *$\{x_1, x_n\} \subset O$, and there exists a chromosome of G containing a subsequence $\overline{S} = O_1 P(\overline{x_{i+1}}, \cdots, \overline{x_{j-1}}) O_2$, where P is a permutation other than the identity and $\{O_1, O_2\} \in O$, or* **ii.** *$x_1 \in O$, there exists a chromosome containing a subsequence $\overline{S} = O' P(\overline{x_2}, \cdots, \overline{x_{n-1}}) x_n$, where P is a permutation other than the identity and $O' \in O$. An analogous condition holds for x_n.*

We call $\overline{S}$ the **complementary sequence** *of S. We say that a local SP (real or potential) S is* **minimal** *if it does not contain any subsequence corresponding to another local SP.*

We now show how to correct *dedouble* so that when applied to a genome containing no local SP, it produces a maximal completed graph containing no SP. Let $\mathcal{G}_D(\mathbf{V}, A, D)$ be a maximal completed graph produced by *dedouble* and $S = x_1 \cdots x_n$ an SP of $\mathcal{G}_D(\mathbf{V}, A, D)$. The following procedure applies to the SP S.

Procedure spoil($x_1 \cdots x_n$)

Remove all the edges of D adjacent to the vertices of $\{x_1, \cdots, x_n, \overline{x_1}, \cdots, \overline{x_n}\}$;
Construct edges (x_k, x_{k+1}) and $(\overline{x_k}, \overline{x_{k+1}})$ for all k, $1 \leq k < n$.

Lemma 2 *Suppose that the completed graph $\mathcal{G}_D(\mathbf{V}, A, D)$ produced by* dedouble *contains an SP $S = x_1 x_2 \cdots x_{n-1} x_n$. If S is not a local SP of G, then* spoil-SP *($x_1 \cdots x_n$) gives rise to a completed graph $\mathcal{G}_{D'}(\mathbf{V}, A, D')$ containing at least the same number of cycles as $\mathcal{G}_D(\mathbf{V}, A, D)$, and one less SP.*

Consider the maximal completed graph $\mathcal{G}_D(\mathbf{V}, A, D)$ produced by algorithm *dedouble*. Let $\mathcal{S}$ be the set of SP in $\mathcal{G}_D(\mathbf{V}, A, D)$ that do not correspond to local SP of G, and **spoil** the procedure that applies *spoil-SP* to each $S \in \mathcal{S}$.

Theorem 3 *For G a genome with no local SP, the completed graph produced by* dedouble *followed by* spoil*, is maximal and contains no SP.*

In the remainder of this paper it will be implicit that the procedure *spoil* is incorporated at the end of the algorithm *dedouble*. We turn next to the case where G contains local SP.

Lemma 3 *Suppose G contains a real local SP $S = x_1 \cdots x_n$. Then any maximal completed graph $\mathcal{G}_D(\mathbf{V}, A, D)$ will contain a SP with the set of nodes $\{x_1, \cdots, x_n\}$.*

Potential SP give rise to additional problems. We must minimize the number of such potential SP that become SP of the final completed graph.

The following sections detail the specific modifications of the general method necessary, first for the case of rearrangement by translocations, then for rearrangement by reversals plus translocations.

8 Reconstructing a duplicated genome with translocations

A potential SP of form $S = O_1 x_1 \cdots x_n O_2$ (or $S = O_1 x_1 \cdots x_n$) is **even** (EPS) if both natural graphs (or the only natural graph, respectively) containing vertices in O and also vertices in $\{x_1, \cdots x_n, \overline{x_1}, \cdots, \overline{x_n}\}$, are graphs of GCE, i.e. even size graphs. Otherwise, S is an **odd potential SP** (OPS). We denote by $\mathcal{EPS}$ the set of graphs, and pairs of graphs, corresponding to all EPS.

To avoid transforming an OPS into an SP in a completed graph, the choice of odd-size natural graphs to amalgamate (in forming the supernatural graphs) is no longer arbitrary. A particular choice procedure has to be introduced, and another procedure for amalgamating graphs corresponding to EPS. In the latter case, however, we amalgamate even-size natural graphs, with the consequence that algorithm *dedouble*, applied to the resulting supernatural graphs, generates a non-maximal completed graph. That gives rise to additional difficulties. The details of these choice procedures and how potential SP are resolved are too lengthy to include in this communication, as is the evaluation of parameters in the following theorem.

Theorem 4 *The minimal number of translocations required to transform G into a duplicated genome is:*

$$T(G) = \frac{|A|}{2} - \alpha(G) + sp(G) + \psi(G) + s(G) + sr(G) + f(G),$$

where $s(G)$ is the number of real minSP of G; $sp(G)$ is the number of supernatural graphs obtained after amalgamating natural graphs of $\mathcal{EPS}$; and parameters $\psi(G)$, $sr(G)$ and $f(G)$ depend exclusively on G and take values 0, 1 or 2.

Moreover, when dedouble *is applied to the appropriate supernatural graphs, it generates a duplicated genome H such that $RO(G, H) = T(G)$.*

9 Reconstructing a duplicated genome using reversals and translocations

In the HP approach, the problem of rearrangement by reversals plus translocations is reduced to a problem of rearrangement by reversals only (a multichromosomal genome is converted to a single chromosome genome). Here, an SP of the final completed graph does not necessarily correspond to a bad component. We need some additional notions introduced by HP. An **oriented gray edge** links the vertex at the left (or right) end of a black edge to the vertex at the left (or right, respectively) end of another black edge, while an **unoriented gray edge** links two different sides of two black edges. A cycle C is **oriented** if it contains at least one oriented gray edge. Otherwise it is **unoriented**. An **oriented component** has at least one oriented cycle and thus at least one oriented gray edge. Otherwise it is an **unoriented component**. We say that component U **separates** two components U' and U'' if any edge we tried to draw from a vertex of U' to one of U'' would cut a gray edge of U. A **knot** is an (intrachromosomal) unoriented component which does not separate any pair of unoriented components. HP introduce the difference between two kinds of knots: **real-knots** and **semi-knots**.

In order to determine the components which may form knots, we introduce another definition. Let $\mathcal{U} = \{u_1, \cdots, u_p\}$ be a subset of $\mathcal{B}$, and $\overline{\mathcal{U}} = \{\overline{u_1}, \cdots, \overline{u_p}\}$. We say that $\mathcal{U}$ is **unoriented** if genes u_i and $\overline{u_i}$ either have the same sign in G, or opposite signs, for all i. Consider now a component π_i of Π where $\mathbf{V_i}$ is its vertex set. $\mathbf{V_i}$ is oriented if and only if the set of genes corresponding to its vertices is oriented.

Lemma 4 *π_i is oriented if and only if $\mathbf{V_i}$ is.*

We say that a **real local SP is unoriented** if the set of vertices in its component is unoriented. Real-knots produced by *dedouble* then correspond to the real minimal unoriented local SP, and to at most one other SP, which we will not describe here. All these SP are called **bad real SP**. On the other hand, semi-knots are associated to potential SP of G. The goal is to minimize the number of such potential SP that become bad SP of the completed graph. To do so, there is a procedure (omitted here) for choosing which graphs corresponding to OPS and EPS should be amalgamated.

Theorem 5 *The minimal number of reversals plus translocations necessary to transform G into a duplicated genome is:*

$$RT(G) = \frac{1}{2}|A| - \alpha(G) + brs(G) + \left\lceil \frac{s(G) - gr(G) + fr(G)}{2} \right\rceil,$$

where $brs(G)$ is the number of bad real SP of G; $s(G)$ is the number of local SP that become semi-knots; $fr(G)$ and $gr(G)$ take values 0 or 1. All these parameters depend exclusively on G.

Moreover, the algorithm dedouble *applied to the appropriate supernatural graphs (constructed by the correct choice procedure), generates a duplicated genome H such that $RO(G, H) = RT(G)$.*

10 Analysing the yeast genome

Analysing duplicated genes in the *Saccharomyces cerevisiae* genome led Wolfe and Shields [8] to propose that yeast is a degenerate tetraploid resulting from a genome duplication that would have occured one hundred million years ago. They identified 55 duplicated regions, representing 50% of the genome.

Applying our algorithm for genome halving by translocations only to the Wolfe and Shields data (where each duplicated region is treated as a single "gene" for our purposes), we obtain a completed graph $\mathcal{G}(V, A, D)$ containing $c = 81$ cycles. Since G does not contain any local SP, the algorithm does not generate any bad component. As the yeast genome has 16 chromosomes, the minimal number of translocations required to transform it into a duplicated genome is $T = 2|\mathcal{B}| + |\mathcal{O}| - 2N - c = 110 + 32 - 16 - 81 = 45$. On the other hand, as the yeast genome does not contain any real or potential local SP, our method for halving by reversals plus translocations does not involve any reversals, and so the same 45 translocations are required in this analysis as well.

Acknowledgments

Research supported in part by the Natural Sciences and Engineering Research Council of Canada. The author is a Scholar in the Evolutionary Biology Program of the Canadian Institute for Advanced Research.

References

[1] El-Mabrouk, N., Bryant, B., Sankoff, D.:Reconstructing the Pre-Doubling Genome. Proceedings of the third annual international conference on computational molecular biology (RECOMB'99). (1999) ACM press. 154-163,

[2] El-Mabrouk, N., Nadeau, J.H., Sankoff, D.: Genome halving. Combinatorial Pattern Matching. Ninth Annual Symposium (M. Farach-Colton, ed.). Lecture Notes in Computer Science **1448** (1998) Springer Verlag, 235–250.

[3] El-Mabrouk, N., Sankoff, D.: On the reconstruction of ancient doubled circular genomes using minimum reversal. Genome Informatics, Series No. 10, 83-93, Universal Academy Press, Tokyo, December 1999.

[4] Hannenhalli, S.: Polynomial-time algorithm for computing translocation distance between genomes. In Combinatorial Pattern Matching. Sixth Annual Symposium (Z. Galil and E. Ukkonen, ed.) Lecture Notes in Computer Science **937** (1995) Springer-Verlag, 162–176.

[5] Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip. (polynomial algorithm for sorting signed permutations by reversals). In: Proceedings of the 27th Annual ACM-SIAM Symposium on the Theory of Computing (1995) 178–189.

[6] Hannenhalli, S., Pevzner, P.A.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science (1995) 581–592.

[7] Setubal, J., Meidanis, J.: Introduction to Computational Molecular Biology. PWS Publishing Company (1997).

[8] Wolfe, K.H., Shields, D.C.: Molecular evidence for an ancient duplication of the entire yeast genome. Nature **387** (1997) 708–713

Département d'informatique et de recherche opérationnelle, Université de Montréal, CP 6128 succursale Centre-Ville, Montréal, Québec H3C 3J7.
E-mail: mabrouk@iro.umontreal.ca