The Ninth International Symposium on Operations Research and Its Applications (ISORA'10)
Chengdu-Jiuzhaigou, China, August 19–23, 2010

Perfect Sorting by Reversals and Deletions/Insertions

Hong-Yu Chen[1] Xiang Tan[2,1] Guo-Jun Li[1,*]

[1]School of Mathematics, Shandong University,
Jinan, Shandong, 250100, China
[2]School of Statistics and Mathematics, Shandong University of Finance,
Jinan, Shandong, 250014, China

Abstract Recently, more and more people are interested in the problem of computing a sequence of rearrangement operations to transform one genome into the other such that these operations conserve all common intervals. Bérard et al. show an algorithm to solve the problem in subquadratic time when all operations are reversals. In this paper, we extend the result to allow deletions and a limited form of insertions (which forbids duplications), allowing to compare genomes containing different genes. We provide an exact algorithm for the asymmetric case and a heuristic algorithm for the symmetric case.

Keywords algorithm, perfect sorting, common intervals

1 Introduction

Genome rearrangement problem is to infer a sequence of rearrangement operations which transform one genome into other genome, minimizing the number of rearrangement operations. In [6], Hannenhalli and Pevzner (HP) developed the first polynomial algorithm for sorting by reversals. El-Mabrouk extended that result to allow deletions and a limited form of insertions (which forbids duplications) in [4].

Recently, another combinatorial framework for sorting by reversals was proposed, called perfect sorting by reversals [5]. It is to infer a minimum number of reversals that don't break any common interval of the considered genome. For this problem, the first algorithm proposed has an exponential worst-case running time [5]. Bérard et al. [1] improved it by showing that the problem is fixed parameterized tractable [3] (FPT): i.e., the complexity function is $f(p) \cdot n^{O(1)}$ for some parameter p, where p is the number of prime edges of the strong interval tree of the signed permutation. Recently, they described a more efficient algorithm for the problem in [2]. The worst-case time complexity of their algorithm is parameterized by the maximum prime degree d of the strong interval tree, i.e., $f(d) \cdot n^{O(1)}$, where d is always smaller than or equal to the number of prime edges p. In [7], it was showed that when, for a signed permutation, there exists a parsimonious scenario that is also a perfect scenario, computing such a scenario can be done in polynomial time. In this paper, we will extend the result of [1] to include deletions and insertions of gene segments, allowing to compare genomes containing different genes.

*Corresponding author. E-mail address: guojun@csbl.bmb.uga.edu

Some notation and definitions are introduced in Section 2, then we extend Bérard's method to take into account deletions or insertions in Section 3 and deletions and insertions in Section 4.

2 Preliminaries

A signed permutation on n elements is a permutation on the set of integers $\{1,2,\ldots,n\}$ in which each element has a sign, positive or negative. An *interval* of a signed permutation is a segment of consecutive elements of the permutation. One can define an interval by giving the set of its unsigned elements, called the *content* of the interval. The *reversal* of an interval of a signed permutation reverses the order of the elements of the interval, while changing their signs, that is, a reversal $\rho(i,j)$ rearranges the genes inside the genome and transforms the genome $(x_1\ldots x_{i-1}x_ix_{i+1}\ldots x_jx_{j+1}\ldots x_n)$ into the genome $(x_1\ldots x_{i-1}-x_j\ldots-x_{i+1}-x_ix_{j+1}\ldots x_n)$. We denote $G\cdot\rho$ as the new genome obtained from genome G as a result of a reversal ρ. If P is a permutation, we denote $\overline{P}$ the permutation obtained by reversing the complete permutation P.

Let G and H be two signed permutations. A *scenario* between G and H is a sequence of rearrangement events (reversal, deletion, insertion) that transforms G into H, or G into $\overline{H}$. The *length* of such a scenario is the number of rearrangement events it contains.

Two distinct intervals I and J *commute* if their contents trivially intersect, that is, either $I\subset J$, or $J\subset I$, or $I\cap J=\emptyset$. Else, they *overlap*. We call a reversal $\rho(i,j)$ *breaks* an interval $I=(x_a\ldots x_b)$ if the two intervals $(x_i\ldots x_j)$ and $(x_a\ldots x_b)$ overlap. If there is a reversal $\rho_i(1\leq i\leq t)$ of a reversal sequence $\rho_1,\ldots,\rho_t$ breaks I, we call the reversal sequence $\rho_1,\ldots,\rho_t$ breaks I.

A *common interval* of a permutation G and H is an interval in both G and H. The singletons in both G and H are called *trivial* common intervals.

A scenario S between G and H is called a *perfect scenario* if every element of S does not break any common interval of G and H. A perfect scenario of minimal length is called a *parsimonious perfect scenario*.

Let G and H be two genomes with some genes in common, others specific to each genome, and no gene appearing more than once. Write $\mathscr{A}$ for the set of genes in both G and H, and $\mathscr{A}_G$ and $\mathscr{A}_H$ for those in G and H only respectively.

3 The algorithm

3.1 Background

Given two genomes G and H with $\mathscr{A}_G=\mathscr{A}_H=\emptyset$, the problem of sorting by reversals is to find the minimum number of reversals that transform G into H. It is solved by Hannenhalli and Pevzner [6]. They gave an exact polynomial algorithm, denoted by **HP** algorithm.

For two genomes G and H with $\mathscr{A}_G\neq\emptyset,\mathscr{A}_H=\emptyset$, El-Mabrouk [4] gave a new method to represent G and H by the breakpoint graph $\mathscr{G}(G,H)$, and presented an algorithm that transforms genome G to H, called **Reversal-deletion** algorithm.

Indeed, it is commonly accepted in computational biology, that if a group of homologous genes (that are genes having a common ancestry) is co-localized in two different

species, then those genes were probably together in the common ancestor and were not later separated during evolution. Such conserved clusters of homologous genes are called *common intervals*.

A reversal $\rho(i,j)$ is called *perfect* if it doesn't break any common interval of G. The problem of *perfect sorting by reversals* is to find a reversal sequence $\rho_1,\ldots,\rho_t$ such that $\rho_i(i=1,\ldots,t)$ doesn't break any common interval of G, $G\cdot\rho_1\ldots\rho_t = H$ and t is minimum, where $\mathscr{A}_G = \mathscr{A}_H = \emptyset$. It is solved by Bérard et al. in [1] and [2]. We call their algorithm **BBCP** algorithm.

3.2 Perfect sorting by reversals and deletions (or insertions)

From this section, our goal is to propose algorithms for finding the minimum number of rearrangement operations (reversals, insertions and deletions) necessary to transform genome G into genome H such that they don't break any common interval of G.

First, we consider the case when $\mathscr{A}_G \neq \emptyset$, $\mathscr{A}_H = \emptyset$. It is obvious we only need reversals and deletions to transform G into H, and the genes of $\mathscr{A}_G$ are destined to be deleted.

The following lemma from [5] is fundamental.

Lemma 1. If a sequence of reversals sorts a permutation and does not break an interval I, then there exists a sorting sequence of same size (with the same reversals) in which all the reversals contained in I (they sort I) are before all the other reversals (they sort outside I).

The following theorem is immediately by Lemma 1. It is the basic idea of our algorithms.

Theorem 1. If a sequence of reversals, insertions and deletions sorts a signed permutation and does not break any common interval, then there exists a sorting sequence of same size (with the same reversals, insertions and deletions), in which all the reversals contained in the common intervals are before all the other reversals, insertions and deletions.

Proof. It is obvious that all the reversals contained in the common intervals are before all the other reversals by Lemma 1. Since any insertion or deletion doesn't break any common interval, that is, the insertions and deletions are external to all the common intervals, so the insertions and deletions don't overlap with the reversals contained in the common intervals. Hence all the reversals contained in the common intervals could be located before the insertions and deletions. The result follows. □

Example : Let $G = (1\ -3\ -2\ 7\ 5\ -4\ 6)$, $H = (4\ 5\ 1\ 2\ 3)$

The common intervals of G and H are $\{2,3\}$, $\{1,2,3\}$, $\{4,5\}$ and the singletons $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$ and $\{5\}$.

Reversals $\{1,2,3,4,5,7\}$, $\{1,2,3,6\}$, $\{2,3\}$, $\{5\}$ and deletion $\{6,7\}$ is a perfect scenario that transforms G into H.

Change the reversals' order such that all the reversals contained in the common intervals are before all the other reversals, deletions, that is, the reversals $\{2,3\}$, $\{5\}$, $\{1,2,3,4,5,7\}$, $\{1,2,3,6\}$ and deletion $\{6,7\}$ is also a perfect scenario that transforms G into H.

The general idea of our algorithm is as follows: since the sequence of reversals, insertions and deletions can be rearranged such that the reversals contained in the common

intervals are before all the other operations by Theorem 1, we can sort the common intervals of G first, then transform the obtained permutation G' to the other permutation H by El-Mabrouk's method.

Let $P = \{P_1, P_2, \ldots, P_k\}$ be a subset of common intervals of G and H, where $|P_i| \geq 2$, every other common interval is a subset of P_i $(1 \leq i \leq k)$ and $P_i \cap P_j = \emptyset$ $(1 \leq i, j \leq k)$.

Now we give the following algorithm to solve the problem of perfect sorting by reversals and deletions.

Algorithm 1: Perfect sorting by reversals and deletions
S is an empty scenario.
For each common interval $P_i \subseteq P$, denote S_i be the corresponding
scenario by applying **BBCP** algorithm
Add S_i to S.
Denote the obtained permutation after executing the reversals of each
S_i by G'.
End for
For G' and H, applying **Reversal-deletion** algorithm to transform G'
to H, denote S' be the corresponding scenario
Add S' to S.
End for

Theorem 2. Algorithm 1 performs $RD(G,H) = R(G,G') + RD(G',H)$ reversals and deletions, where $R(G,G')$ is the total number of reversals that sort each P_i of P, and $RD(G',H) = R(\widetilde{G'},H) + D(G',H)$ is the reversals and deletions that transform G' into H. Moreover, $RD(G,H)$ is the minimum number of rearrangement operations necessary to transform G into H and don't break any common interval.

Proof. First, we prove Algorithm 1 is feasible.

Since the reversals contained in each common interval can exist before all the other reversals and deletions by Theorem 1, Algorithm 1 sorts each maximal common interval first. In the breakpoint graph $\mathscr{G}(G',H)$, the vertices corresponding to the genes of each common interval P_i belong to the cycles of size 1 by the definition of breakpoint graph. Since applying **Reversal-deletion** algorithm doesn't affect the cycles of size 1, then the step of transforming G' to H by **Reversal-deletion** algorithm doesn't break any common interval and the deletions don't affect the common interval obviously. Hence Algorithm 1 is feasible.

By **BBCP** algorithm, $R(G,G')$ reversals sorting each P_i of P is minimum. By the **Reversal-deletion** algorithm, $R(\widetilde{G'},H) + D(G',H)$ is minimum. Hence, $RD(G,H) = R(G,G') + RD(G',H)$ is the minimum number of rearrangement operations necessary to transform G into H. The result follows. □

The case $\mathscr{A}_G = \emptyset$, $\mathscr{A}_H \neq \emptyset$, where the set of genes of G is a subset of the set of genes of H, i.e. the problem of transforming G into H with a minimum number of reversals and insertions such that the operations don't break any common interval, can be solved by Algorithm 1, where G takes on the role of H, and vice versa. Each deletion in the H-to-G solution becomes an insertion in the G-to-H. Each reversal in one direction corresponds to a reversal of the same segment in the opposite direction.

3.3 Perfect sorting by reversals, deletions and insertions

First, we give the following definition.

Definition[4]: *the filled genome* of G

Let G and H be two genomes that $\mathscr{A}_G = \emptyset$, $\mathscr{G}_1$ be the graph containing only cycles of size 1 obtained by applying the HP algorithm to the graph $\mathscr{G}(H,G)$. Each indirect black edge of $\mathscr{G}_1$ is labeled by a sequence of genes of $\mathscr{A}_H$. The genome G^c obtained from G as follows: for each pair of adjacent vertices a and b in G, if (a,b) is an indirect black edge of $\mathscr{G}_1$ labeled by the sequence of genes $y_1,\ldots,y_p$ of $\mathscr{A}_H$, then insert this sequence of genes between a and b in G. We call G^c *the filled genome* of G relative to H.

Now, we consider the general case where $\mathscr{A}_G \neq \emptyset$ and $\mathscr{A}_H \neq \emptyset$.

Let $\widetilde{G} = G \setminus \mathscr{A}_G$, $\widetilde{H} = H \setminus \mathscr{A}_H$, $\widetilde{G}^c$ be the filled genome of $\widetilde{G}$ relative to H, a and b be two adjacent vertices in $\widetilde{G}$, separated in G by a segment $X(a)$ of vertices of $\mathscr{A}_G$. If a and b are separated in $\widetilde{G}^c$ by a segment $Y(a)$ of vertices of $\mathscr{A}_H$, then we have the choice to place $X(a)$ before or after $Y(a)$. This choice is made so that we require as few deletions as possible to transform G^c into H. By G^c we denote the genome obtained by appropriately adding the vertices of $\mathscr{A}_G$ to $\widetilde{G}^c$. The graph $\mathscr{G}(G^c,H)$ is obtained from $\mathscr{G}(\widetilde{G}^c,H)$ by transforming certain black edges of $\mathscr{G}(\widetilde{G}^c,H)$ into indirect edges. El-Mabrouk [4] gave the construction of G^c, called **Procedure Construct-G^c**.

Now we give the algorithm of perfect sorting by reversals, deletions and insertions.

Algorithm 2: Perfect sorting by reversals, deletions and insertions.

Step 1. For each common interval $P_i \subseteq P$, $i = 1,\ldots,k$, apply **BBCP** algorithm to sort them respectively.

Denote the obtained permutation from G by G'.

Step 2. Apply the **HP** algorithm to the graph $\mathscr{G}(H,\widetilde{G'})$.

The final graph obtained contains $D(H,\widetilde{G'})$ indirect edges.

Step 3. Do the $D(H,\widetilde{G'})$ insertions deducible from the indirect edges, and transform $\widetilde{G'}$ to $\widetilde{G'}^c$.

Step 4. Fill the genome $\widetilde{G'}^c$ to obtain genome G'^c, using **Procedure Construct-G^c**.

Step 5. Apply **Reversal-deletion** algorithm to the graph $\mathscr{G}(G'^c,H)$. The algorithm does $R(\widetilde{G'},\widetilde{H})$ reversals and $D(G'^c,H)$ deletions.

Lemma 2 [4] The number of rearrangement operations that transform G' to H during the application of the method is $RDI(G',H) = R(\widetilde{G'},\widetilde{H}) + D(H,\widetilde{G'}) + D(G'^c,H)$, where $R(\widetilde{G'},\widetilde{H})$ is the number of reversals, $D(H,\widetilde{G'})$ is the number of insertions and $D(G'^c,H)$ is the number of deletions. Moreover, if $D(G'^c,H) = D(G',\widetilde{H})$, then the total number of rearrangement operations is minimal.

Theorem 3. Algorithm 2 performs $RDI(G,H) = R(G,G') + RDI(G',H)$ reversals, deletions and insertions, where $R(G,G')$ is the total number of reversals that sort each P_i of P, and $RDI(G',H) = R(\widetilde{G'},\widetilde{H}) + D(H,\widetilde{G'}) + D(G'^c,H)$ is the reversals, deletions and insertions that transform G' into H. Moreover, the rearrangement operations necessary to

transform G into H don't break any common interval and if $D(G'^c,H) = D(G',\widetilde{H})$, then it is minimal.

Proof. By Theorem 1, we can sort each common interval first, then transform the obtained genome to H. After Step 1, in the graph $\mathscr{G}(H,\widetilde{G'})$ and $\mathscr{G}(G'^c,H)$, the vertices corresponding to the genes of each common interval P_i belong to the cycles of size 1. With a similar proof to that of Theorem 2, Step 2 to Step 5 doesn't break any common interval. So Algorithm 2 is feasible.

It is obvious Algorithm 2 performs $RDI(G,H) = R(G,G') + R(\widetilde{G'},\widetilde{H}) + D(H,\widetilde{G'}) + D(G'^c,H)$ by the five steps.

By **BBCP** algorithm, $R(G,G')$ reversals that transform G to G' is minimal. By Lemma 2, if $D(G'^c,H) = D(G',\widetilde{H})$, then the total number of rearrangement operations $R(\widetilde{G'},\widetilde{H}) + D(H,\widetilde{G'}) + D(G'^c,H)$ that transform G' to H is minimal. So if $D(G'^c,H) = D(G',\widetilde{H})$, then the rearrangement operations necessary to transform G into H is minimal. The result follows. □

Lemma 3([1]) Assume $m = max(|P_i||i = 1,\ldots,k)$. The transformation of G to G' by **BBCP** algorithm is of time complexity $O(m\sqrt{mlogm})$ if each $T_S(P_i)$ is unambiguous; $O(2^p m\sqrt{mlogm})$, if there is some ambiguous $T_S(P_i)$, where p is the maximum number of unsigned prime vertices of $T_S(P_i)$.

Lemma 4([4]) Assume $n = max(|\mathscr{A}| + |\mathscr{A}_G|, |\mathscr{A}| + |\mathscr{A}_H|)$. The transformation of G' to H by El-Mabrouk's method is of time complexity $O(n^2)$.

It is obvious that $n > m$. By Lemma 3 and Lemma 4, the following theorem is immediately.

Theorem 4. The transformation of G to H by Algorithm 2 is of time complexity $O(n^2)$, when each $T_S(P_i)$ is unambiguous; $O(2^p m\sqrt{mlogm} + n^2)$, when there is some ambiguous $T_S(P_i)$, where p is the maximum number of unsigned prime vertices of $T_S(P_i)$.

4 Conclusion

Our main result in this note is an algorithm that finds the minimum rearrangement operations necessary to transform genome G into H, where G and H have different genes, such that these operations don't break any common interval. Obviously, in such case, insertions or deletions are needed. We combine a strong interval tree with a breakpoint graph to show how to extend the BBCP algorithm to an algorithm that allows deletions and a limited form of insertions.

The next step of this work will be to generalize our approach to handle duplications as well as insertions and present an algorithm for computing (near) minimal edit sequences involving insertions, deletions and reversals.

Acknowledgements

This work was supported by National Natural Science Foundation of China (10971121, 60873207, 60373025), and also partially supported by Graduate Independent Innovation Foundation of Shandong University (11140070613071).

References

[1] S. Bérard, A. Bergeron, C. Chauve, C.Paul, Perfect sorting by reversals is not always difficult, IEEE/ACM Trans. Comput. Biology Bioinform, 4(1) (2007) 4-16.

[2] S. Bérard, C. Chauve, C.Paul, A more efficient algorithm for perfect sorting by reversals, Information Processing Letters, 106 (2008) 90-95.

[3] R.G. Downey, M.R. Fellows, Parameterized Complexity, springer, 1999.

[4] N. El-Mabrouk, Genome rearrangement by reversals and insertions/deletions of contiguous segments, In proceeding 11th Ann. Symp. Combin. Pattern Matching CPM 00, London: springer, 1848 (2000) 222-234.

[5] M. Figeac, J.-S Varré, Sorting by reversals with common intervals, Proceedings of WABI 2004, Lecture Notes in Computer Sciences, Berlin: springer, 3240 (2004) 26-37.

[6] S. Hannenhalli, P.A. Pevzner, 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.

[7] M. -F. Sagot, E. Tannier, Perfect sorting by reversals, Proceedings of COCOON 2005, Lecture Notes in Computer Sciences, Berlin: springer, 3595 (2005) 42-52.