A New Uniform Translocation Distance

Carlos Martín-Vide[2] and Victor Mitrana[1,2]

[1] Faculty of Mathematics and Computer Science, University of Bucharest
Str. Academiei 14, 70109, Bucharest, Romania
[2] Research Group in Mathematical Linguistics, Rovira i Virgili University
Pça. Imperial Tarraco 1, 43005, Tarragona, Spain
{cmv,vmi}@fll.urv.es

Abstract. A basic problem in the area of combinatorial algorithms for genome evolution is to determine the minimum number of large scale evolutionary events (genome rearrangements) that transform a genome into another. The present paper is a contribution to the algorithmic study of genom evolution by translocations which is an area related to pattern recognition. Furthermore, it may be viewed as a contribution to other areas related to pattern recognition like: error estimation, genetic programming, disease diagnosis. In this paper we consider chromosomes as being linear strings that exchange each other prefixes in the translocation process. A new type of translocation distance between a pair of multi-chromosomal genomes is introduced; we examine the complexity of computing this distance in the case of uniform translocation, that is at each step the strings exchange prefixes of the same length. We present an exact polynomial algorithm based on the "greedy" strategy when the target set is a singleton while a 2-approximation algorithm is provided when considering arbitrary target sets. Some open problems are finally formulated.

1 Introduction

Genomes of arbitrarily complex organisms are organized into chromosomes that contain genes which may be considered as being arranged in linear order. Sequence alignment was actually the first step in molecular evolution studies but in many cases sequence alignment is quite unreliable which makes further evolutionary tree reconstruction almost impossible. It has been often found that the order of genes is much more conserved than the DNA base sequence. In the course of evolution, the genome of an organism mutates not only by processes at the level of individual genes (point mutations: insertion, deletion or substitution of individual bases) but also by some large-scale rearrangements in one evolutionary event. Recently much attention has been given by this phenomenon. One may argue that evolutionary and functional relationships between genes can be captured by taking into considerations only local mutations. However, the analysis of the genomes of some viruses (Epstein-Barr and Herpes simplex viruses, see for instance [7, 13]) have revealed that the evolution of these viruses

A. Fred et al. (Eds.): SSPR&SPR 2004, LNCS 3138, pp. 278–286, 2004.

involved a number of large-scale rearrangements in one evolutionary event. Furthermore, comparing plant and animal mitochondrial DNA, the point mutation is estimated to be 100 times slower in plant than in animal, many genes are nearly identical (more than 99% of them are identical) in related species [17]. These molecules which are almost identical in gene sequence differ fundamentally in gene order. At this level, point mutations are less meaningful compared to arrangements of gene fragments. See also [2, 7], for further discussions on this topic.

Chromosomal rearrangements include pericentric and paracentric inversions, intrachromosomal and interchromosomal transpositions, translocations, etc. For a description of these rearrangements, the reader is referred to [21]. Translocation is the biological process of exchanging material of the end of two chromosomes and could result in a different genotype[12]. Such non-local operations might permit an investigation of the evolutionary history for rather diverged organisms that cannot be identified from the study of point mutations alone [19, 20]. Recent developments in large-scale comparative genetic mapping seem to offer exciting prospects for understanding mammalian genome evolution [4]. A grammatical model based one these non-local operations can be found in [5] and [6]

The aim of this paper is to introduce a new type of translocation distance and to investigate the complexity of computing this distance for uniform (equal-length) translocation. This is a particular type of translocation which takes part just between chromosomes that exchange prefixes of equal length.

Prior work dealing with the combinatorial analysis of genome operations has focused on evolution distance in terms of inversions, transpositions or translocations for chromosomes formed from different markers which correspond to unique segments of DNA. From the formal point of view this means that in traditional genome rearrangement sorting problems the input data consists of permutations of n labels, but this approach cannot capture duplication events. In this paper, we considered chromosome as a nucleotide sequence unlike a unique marker sequence. Perhaps, this approach will not be practical for a while for lack of such data, but we looked to this problem from a mathematical point of view only. Kececioglu and Sankoff ([14, 15]) developed exact and approximation algorithms for two types of inversion distance which was shown to be *NP-complete* [3]. More recently [9] proposes a polynomial algorithm for signed inversion distance. Bafna and Pevzner reported approximation algorithms for transposition distance [1]. Two highly relevant papers which present the first polynomial algorithms for computing translocation distances are [10, 11]. Kececioglu and Ravi [16] discussed exact and approximation algorithms for distance involving translocations alone as well as together with inversions. Some applications of these results to biological data are now underway [2, 8]. The reader is also referred to [18] for a review of open combinatorial problems motivated by genome rearrangements.

Our work differs from the aforementioned approaches in many respects: the strings representing chromosomes may have multiple occurrences of the same symbol, they may have common symbols, the number of copies of all strings in the initial set is assumed arbitrarily large.

2 Preliminaries

Let V be a given alphabet (practically this alphabet is the DNA alphabet $\{A, T, C, G\}$); chromosomes may be viewed as strings over this alphabet. The set of all nonempty strings over V is denoted by V^+. For each string $x \in V^+$, whose length is denoted by $|x|$, $x[i, j]$ delivers the substring of x that starts at position i and ends at position j in x, $1 \leq i \leq j \leq |x|$. Conventionally, $x[i, j]$ is the empty string in all cases $j < i$. For two strings x, y over an alphabet V and two integers $1 \leq i < |x|, 1 \leq j < |y|$, we define the translocation operation

$$(x, y) \vdash_{(i,j)} (z_1, z_2) \text{ iff } x = tu, y = vw, z_1 = tw, z_2 = vu, \text{ and } |t| = i, |v| = j.$$

The pair of natural numbers (i, j) indicates the length of the prefixes they interchange with each other. When we are not interested about the length of these segments, we write simply $\vdash$. Let us note that, from a chromosome and its replica, say xyz, one may get two other chromosomes $xyyz$ and xz. It is worth mentioning here that this type of recombination is known as crossover between "sister" chromatids and it is the main way of producing tandem repeats or block deletions in chromosomes. We extend the translocation operation to a finite set of strings $A \subseteq V^+$ by $TO(A) = \bigcup_{x,y \in A} \{z, w | (x, y) \vdash (z, w)\}$.

Let A be a finite set of strings such that each string of A has arbitrarily many available copies. In other words, A may be viewed as the support of a multiset of strings each of them having arbitrarily many copies. Define iteratively

$$TO_0(A) = A, \quad TO_{k+1}(A) = TO_k(A) \cup TO(TO_k(A)), \quad TO_*(A) = \bigcup_{k \geq 0} TO_k(A).$$

A *translocation sequence* in $TO_*(A)$ is a sequence $S = s_1, s_2, \ldots, s_n$, where for each $1 \leq i \leq n$ $s_i = (x_i, y_i) \vdash_{(k_i, p_i)} (u_i, v_i)$, for some $x_i, y_i, u_i, v_i \in TO_*(A)$ and $1 \leq k_i < |x_i|$, $1 \leq p_i < |y_i|$. Given a translocation sequence S as above and $x \in TO_*(A)$ we define

$$P_i(S, x) = card\{j \leq i | x = x_j \text{ or } x = y_j\} + card\{j \leq i | x_j = y_j = x\},$$

$$F_i(S, x) = \begin{cases} card\{j \leq i | u_j = x_j \text{ or } v_j = y_j\} + card\{j \leq i | u_j = v_j = x\}, & \text{if } x \notin A, \\ \infty, \text{ otherwise.} \end{cases}$$

The *length* of a translocation sequence $S = s_1, s_2, \ldots, s_n$ is denoted by $lg(S)$ and equals n. A translocation sequence S as above is *contiguous* iff the following two conditions are satisfied:

(i) $x_1, y_1 \in A$,

(ii) $F_{i-1}(S, x_i) > P_{i-1}(S, x_i)$, and $F_{i-1}(S, y_i) > P_{i-1}(S, y_i)$, for all $1 \leq i \leq n$.

The second condition is very natural if one considers that the copies of the two strings that exchange prefixes are not available anymore for further translocation steps; it claims that at each translocation step at least one copy for any of the two strings involved in this step is available. By CTS we mean a contiguous translocation sequence. Let B be a finite subset of $TO_*(A)$; a CTS S as above is B-producing if $F_n(S, z) > P_n(S, z)$ for all $z \in B$. In other words, S is B-producing if at the end of all translocation steps form S we have at least one copy

at each string in B. Roughly speaking, the *translocation distance* from A to B ($TD(A,B)$ shortly) is defined as the minimal number of steps strictly necessary to get B starting from A, providing that at each step just one translocation takes place. Formally,

$$TD(A,B) = \min\{lg(S) | S \text{ is a } B- \text{ producing } CTS \text{ in } TO_*(A)\}.$$

Sometimes we refer to B as the target set. In the sequel we are dealing with the complexity of computing the translocation distance defined above for the case of uniform translocation i.e. all strings exchange prefixes of equal length. We distinguish two cases depending on the cardinality of target sets: singleton target sets and arbitrary target sets.

3 Singleton Target Sets

As we said above, by uniform translocation we mean a special type of translocation so that prefixes which are to be exchanged are of the same length. Formally, the translocation operation $\vdash_{(i,j)}$ is said to be uniform iff $i = j$, so that we shall simply write $\vdash_i$.

In the case of uniform translocation with a singleton target set, without loss of generality we may assume that the initial set of strings contains only strings of the same length, that is the length of the target string. The simple proof of this statement is left to the reader. In conclusion, throughout this section the strings in the initial set and the target string will be all of the same length.

Suppose that $A = \{x_1, x_2, \ldots, x_n\}$ and let z be an arbitrary string of length k; the following measure will be very useful in the sequel:

$$MaxSubLen(A,z,p) = \max\{q | \exists\, 1 \le i \le n \text{ such that } x_i[p, p+q-1] = z[p, p+q-1]\}.$$

Note that with uniform translocation, a letter at position i in a string remains at position i after moving to another string. Assume that $z \in TO_*(A)$; define iteratively the set $H(A,z)$ of intervals of natural numbers as follows:

1. $H(A,z) = \{[1, MaxSubLen(A,z,1)]\}$;
2. Take the interval $[i,j]$ having the largest j; if $j = k$, then stop, otherwise put into $H(A,z)$ the new interval $[j+1, j + MaxSubLen(A,z,j+1)]$.

Note that we allow intervals of the form $[i,i]$ for some i to be in $H(A,z)$; moreover, for each $1 \le i \le k$ there are $1 \le p \le q \le k$ (possibly the same) such that $i \in [p,q] \in H(A,z)$.

Lemma 1 *Let S be a z-producing CTS in $TO_*(A)$. Then,*

$$lg(S) \ge card(H(A,z)) - 1.$$

Proof. We prove this assertion by induction on the length k of z. For $k = 1$ the assertion is trivially true because z must be in A, hence $H(A,z)$ contains just one element. Assume that the assertion is true for any string shorter than k. Let us consider a CTS $S = s_1, s_2, \ldots, s_q$ in $TO_*(A)$ producing z. Moreover, we may assume that $s_i = (x_i, y_i) \vdash_{p_i} (u_i, v_i)$, $1 \le i \le q$, and z has been obtained in S at the last step, that is either $u_q = z$ or $v_q = z$. Let

$A' = \{x[MaxSubLen(A,z,1)+1,k] | x \in A\}$, $z' = z[MaxSubLen(A,z,1)+1,k]$. For simplicity denote $r = MaxSubLen(A,z,1)$. Clearly, $H(A',z') = \{[i-r, j-r] | [i,j] \in H(A,z) \setminus \{[1,r]\}\}$, hence $card(H(A',z')) = card(H(A,z)) - 1$. Starting from S we construct a CTS in $TO_*(A')$, producing z' $S' = s'_1, s'_2, \ldots s'_m$ in the way indicated by the following procedure:

```
Procedure Construct_CTS(S,r);
begin
m := 0;
for i := 1 to q begin
    if  (p_i > r)  then
      m := m + 1;  s'_m = (x_i[r+1,k], y_i[r+1,k]) ⊢_{p_i - r} (u_i[r+1,k], v_i[r+1,k]);
    endif;
endfor;
end.
```

Claim 1: *S' is a CTS.*

Proof of the claim. Firstly, we note that for each $1 \leq i \leq q$ so that $p_i \leq r$, the relations $u_i[r+1,k] = y_i[r+1,k]$ and $v_i[r+1,k] = x_i[r+1,k]$ hold. Assume that $p_{i_1}, p_{i_2}, \ldots, p_{i_m}$ are all integers from $\{p_1, p_2, \ldots, p_q\}$ bigger than r. Because all $p_1, p_2, \ldots, p_{i_1-1}$ equal at most r, it follows that both $x_{i_1}[r+1,k], y_{i_1}[r+1,k]$ are in A'. Now, it suffices to prove that for a given $2 \leq j \leq m$, the relations

$F_{j-1}(S', x_{i_j}[r+1,k]) > P_{j-1}(S', x_{i_j}[r+1,k])$,

$F_{j-1}(S', y_{i_j}[r+1,k]) > P_{j-1}(S', y_{i_j}[r+1,k])$,

hold. We shall prove the first relation only. It is not hard to see that

$$F_{j-1}(S', x_{i_j}[r+1,k]) = \sum_{x[r+1,k]=x_{i_j}[r+1,k]} F_{i_j-1}(S,x) - card(X) - card(Y),$$

$$P_{j-1}(S', x_{i_j}[r+1,k]) = \sum_{x[r+1,k]=x_{i_j}[r+1,k]} P_{i_j-1}(S,x) - card(X) - card(Y),$$

where

$$X = \{t \leq i_j - 1 | p_t \leq r,\ u_t[r+1,k] = v_t[r+1,k] = x_{i_j}[r+1,k]\},$$
$$Y = \{t \leq i_j - 1 | p_t \leq r,\ u_t[r+1,k] = x_{i_j}[r+1,k] \text{ or } v_t[r+1,k] = x_{i_j}[r+1,k]\}.$$

In conclusion, as S is a CTS, it follows that $F_{j-1}(S',\ x_{i_j}[r+1,k]) > P_{j-1}(S',\ x_{i_j}[r+1,k])$, and the proof of the claim is complete.

Claim 2: *S' is z'-producing.*

Proof of the claim. More generally, we shall prove by induction on i that S' is producing $u_i[r+1,k]$ and $v_i[r+1,k]$ for all $1 \leq i \leq q$. The assertion is trivially true for $i = 1$. Assume that the assertion is true for all $t \leq i$; we shall prove it for $i+1$. If $u_{i+1}[r+1,k]$ is in A' or $p_{i+1} > r$, we are done. If $p_{i+1} \leq r$, then $u_{i+1}[r+1,k] = y_{i+1}[r+1,k]$; for $y_{i+1} \notin A$ we have $F_i(S, y_{i+1}) > 0$, hence there exists $t \leq i$ such that $u_t = y_{i+1}$ or $v_t = y_{i+1}$. By the induction hypothesis, S' is producing $u_t[r+1,k]$ which concludes the proof of the second claim.

But there exists at least one i such that $p_i \leq r$, it follows that $m \leq q-1$. By the induction hypothesis, $m \geq card(H(A',z')) - 1$, and the proof is complete. □

The next result is a direct consequence of this lemma.

Theorem 1 *Let z be a string of length k and A be a set of cardinality n. There is an exact algorithm that computes TD(A,z) in O(kn) time and O(kn) space.*

Proof. The following algorithm indicates how to construct a CTS $S = s_1, s_2 \ldots, s_m$ in $TO_*(A)$ producing z, when $z \notin A$, whose length is exactly $card(H(A,z)) - 1$.

```
Procedure Uniform_translocation_CTS_construction(A,z);
begin
p := MaxSubLen(A,z,1); let x be a string in A with x[1,p] = z[1,p];
m := 0;
while p < k begin
   r := MaxSubLen(A,z,p+1);
    if r = 0 then THE STRING z CANNOT BE OBTAINED FROM A;
             stop
       else
         let y be a string in A with y[p+1,p+r] = z[p+1,p+r];
         m := m + 1
         s_m = (x,y) ⊢_p (u,v)};
         p := p + r;
         x := u;
    endif
endwhile;
end.
```

It is easy to see that if the algorithm successfully terminates, then either u or v is exactly z, and the length of the CTS determined by the algorithm is exactly $card(H(A,z)) - 1$. By the previous lemma, this in an optimal value. As one can easily see the time complexity of this algorithm is given by the complexity of computing the values $MaxSubLen(A,z,p)$, which is $O(kn)$. Obviously, it requires $O(kn)$ memory. □

4 Arbitrary Target Sets

We shall try to adapt the techniques used in the previous section for arbitrary target sets, too. Let A be a finite set of strings and $z \in TO_*(A)$; denote by

$$MaxPrefLen(A,z) = \begin{cases} |z|, \text{ iff } z \in A, \\ \max(\{q | q < |z|, \text{ there exists } x \in A, |x| > q, \\ \quad \text{so that } x[1,q] = z[1,q]\} \cup \{0\}), \end{cases}$$

$$MaxSufLen(A,z) = \max(\{q | \text{ there exists } x \in A, |x| \geq |z|, \text{ so that } x[|x|-q+1, |x|] = z[|z|-q+1, |z|]\} \cup \{0\}),$$

$$ArbMaxSubLen(A,z,p) = \max(\{q | \text{ there exists } x \in A \text{ and } |x| \geq p+q \text{ such that } x[p, p+q-1] = z[p, p+q-1]\} \cup \{0\}).$$

We define iteratively the set $ArbH(A,z)$ of intervals of natural numbers as follows, provided that all parameters defined above are nonzero:

1. $ArbH(A,z) = \{[1, MaxPrefLen(A,z)]\}$;
2. Take the interval $[i,j]$ having the largest j; if $j = |z|$, then stop. If $j < |z| - MaxSufLen(A,z)$, then put the new interval $[j+1, j+ArbMaxSubLen(A,z,j+1)]$ into $ArbH(A,z)$; otherwise put $[j+1, |z|]$ into $ArbH(A,z)$.

Theorem 2 *1. Let A be a finite set of strings and B be a finite subset of $TO_*(A)$. Then* $\frac{\sum_{z\in B}(card(ArbH(A,z))-1)}{2} \leq TD(A,B) \leq \sum_{z\in B}(card(ArbH(A,z))-1)$.

2. There exist A and $B \subseteq TO_*(A)$ such that $TD(A,B) = \frac{\sum_{z\in B}(card(ArbH(A,z))-1)}{2}$.

3. There exist A and $B \subseteq TO_*(A)$ such that $TD(A,B) = \sum_{z\in B}(card(ArbH(A,z))-1)$.

Proof. 1. We shall prove the first assertion by induction on the length of the longest string in B, say k. The non-trivial relation is

$$\frac{\sum_{z\in B}(card(ArbH(A,z))-1)}{2} \leq TD(A,B). \qquad (*)$$

If $k = 1$, then $B \subseteq A$, hence $card(H(A,z)) = 1$ for all $z \in B$, therefore the relation $(*)$ is satisfied. Assume that the relation $(*)$ holds for any two finite sets X and Y, $Y \subseteq TO_*(X)$, all strings in Y being shorter than k. Assume that $B \setminus A = \{z_1, z_2, \ldots, z_m\}$ and let $S = s_1, s_2, \ldots, s_q$, $s_i = (x_i, y_i) \vdash_{p_i} (u_i, v_i)$, $1 \leq i \leq q$, be a $B \setminus A$-producing CTS in $TO_*(A)$. Note that at least one string in $B \setminus A$ should exist, otherwise the relation $(*)$ being trivially fulfilled.

Consider m new symbols $a_1, a_2, \ldots, a_m$ and construct the sets: $A' = \{x[1,r]a_i x[r+2,|x|] | x \in A, 1 \leq i \leq m\}$, $B' = \{z_i[1,r]a_i z_i[r+2,|z_i|] | 1 \leq i \leq m\}$,, where $r = \min\{p_i | 1 \leq i \leq q\}$. One can construct a B'-producing CTS in $TO_*(A')$ of the same length of S, say S' by applying the next procedure.

```
Procedure Convert(S);
begin
for  j := 1  to  m  begin
   z := z_j; t := q;
    while  z ∉ A  begin
      find the maximal l ≤ t such that u_l = z or v_l = z;
      t := l − 1;
       if u_l = z  then replace u_l by u_l[1,r]a_j u_l[r+2,|u_l|];
          if p_l > r  then z := x_l;
            replace x_l by x_l[1,r]a_j x_l[r+2,|x_l|];
          else z := y_l;
            replace y_l by y_l[1,r]a_j y_l[r+2,|y_l|];
          endif;
       else replace v_l by v_l[1,r]a_j v_l[r+2,|v_l|];
          if p_l ≤ r  then z := x_l;
            replace x_l by x_l[1,r]a_j x_l[r+2,|x_l|];
          else z := y_l;
            replace y_l by y_l[1,r]a_j y_l[r+2,|y_l|];
          endif;
       endif;
    endwhile;
   replace z by z[1,r]a_j z[r+2,|z|];
endfor;
replace the symbol on the position r+1 in all strings in S that
have not been replaced so far by a_1;
end.
```

Now we apply the procedure in the proof of Lemma 1 to the sequence S' for r previously defined. The obtained sequence S'' is a B''-producing CTS in $TO_*(A'')$, where

$A'' = \{a_i x[r+2, |x|] | x \in A, 1 \leq i \leq m\}$, $B'' = \{a_i z_i[r+2, |z_i|] | 1 \leq i \leq m\}$, due to the two claims from the proof of Lemma 2.
For each $1 \leq i \leq m$ $card(ArbH(A'', a_i z_i[r+2, |z_i|]))$ is either $card(ArbH(A, z_i))$ or $card(ArbH(A, z_i)) - 1$. Moreover, for each i such that $card(ArbH(A'', a_i z_i[r+2, |z_i|])) = card(ArbH(A, z_i)) - 1$ there exist at least one step in S' where the strings exchange prefixes of length at most r. It follows that $lg(S'') \leq lg(S') - \lceil t/2 \rceil$, where $t = card(\{i | card(ArbH(A'', a_i z_i[r+2, |z_i|])) = card(ArbH(A, z_i)) - 1\})$. Consequently,

$$lg(S) = lg(S') \geq lg(S'') + \lceil t/2 \rceil \geq \frac{\sum_1^m (card(ArbH(A'', a_i z_i[r+2, |z_i|])) - 1)}{2} + \lceil t/2 \rceil \geq \frac{\sum_1^m (Arbcard(H(A, z_i)) - 1)}{2}.$$

The reader may easily find sets A and B fulfilling the last two assertions. □

An α-approximation algorithm for a minimization problem is a polynomial algorithm that delivers a solution whose value is at most α times the minimum. From the previous theorem we have:

Theorem 3 *There is a 2-approximation algorithm for computing the translocation distance from two sets of strings.*

Proof. Obviously, an algorithm that computes $\sum_{z \in B} (card(ArbH(A, z)) - 1)$ requires $O(n|B|)$, where $n = card(A)$ and $|B|$ is the sum of the lengths of all strings in B. □

5 Conclusion

We have introduced a new translocation distance between two finite sets of strings and proposed an algorithm, based on the "greedy" strategy, for computing this distance when the target set is a singleton. This is a constraint that does not exclude many interesting biological applications. All results presented here are valid for a particular type of translocation, namely the uniform translocation where the strings exchange with each other prefixes of the same length. This restriction might be considered rather far from reality, but, even so, the problem of finding a polynomial algorithm to compute the translocation between two finite sets remains open. The next step is naturally to consider the case of arbitrary translocation which is a more appropriate abstraction of the practical problem. We hope to return to this topic in a forthcoming paper.

References

1. V. Bafna, P.A. Pevzner, Sorting by transpositions. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, 1995.
2. V. Bafna, P.A. Pevzner, Sorting by reversals: genome rearrangements in plant organelles and evolutionary history of X chromosome, *Mol. Biol. Evol.* 12 (1995), 239–246.

3. A. Caprara, Sorting by reversal is difficult. In *Proc. of the First Annual International Conference on Computational Molecular Biology (RECOMB 97)*, ACM New York, 1997, 75–83.
4. N. G. Copeland et al. A genetic linkage map of the mouse: Current applications and future prospects. *Science*, 262(1993), 57–65.
5. J. Dassow, V. Mitrana, A. Salomaa, Context-free evolutionary grammars and the language of nucleic acids. *BioSystems*, 4(1997) 169–177.
6. J. Dassow, V. Mitrana, Operations and grammars suggested by the genome evolution, *Theoretical Computer Science*, 270, 1-2 (2002), 701–738.
7. D.J. McGeoch, Molecular evolution of large DNA viruses of eukaryotes. *Seminars in Virology* 3 (1992) 399–408.
8. S. Hannenhalli et al. Algorithms for genome rearrangements: herpesvirus evolution as a test case. In *Proc. of the 3rd International Conference on Bioinformatics and Complex Genome Analysis*, 1994.
9. S. Hannehalli, P. A. Pevzner, Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversal, *J. of the ACM* 46, 1 (1999) 1–27.
10. S. Hannehalli, P. A. Pevzner, Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proc. of the IEEE 36th Annual Symposium on Foundation of Computer Science*, 1995, 581–592.
11. S. Hannehalli, Polynomial algorithm for computing translocation distance between genomes. In *Combinatorial Pattern Matching, Proc. of the 6th Annual Symposium (CPM'95)*, LNCS, Springer-Verlag, 162–176.
12. D. L. Hartl, D. Freifelder, L. A. Snyder, *Basic Genetics*, Jones and Bartlett Publ., Boston, Portola Valley, 1988.
13. S. Karlin, E. S. Mocarski, G. A. Schachtel. Molecular evolution of herpesviruses: genomic and protein comparisons. *J. of Virology*, 68(1994),1886–1902.
14. J. Kececioglu, D. Sankoff, Exact and approximation algorithms for sorting by reversals, with application to genome rearrangements. In *Proc. of the 4th Symposium on Combinatorial Pattern Matching*, Springer-Verlag, LNCS 684, 1993, 87–105.
15. J. Kececioglu, D. Sankoff, Efficient bounds for oriented chromosome-inversion distance. In *Proc. of the 5th Symposium on Combinatorial Pattern Matching*, Springer-Verlag, LNCS 807, 1994, 307–325.
16. J. Kececioglu, R. Ravi, Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, 1995, 604–613.
17. J.D. Palmer, L.A. Herbon, Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 27(1988), 87–97.
18. P. A. Pevzner, M. S. Waterman, Open combinatorial problems in computational molecular biology. In *Proc. of the 3rd Israel Symposium on Theory of Computing and Systems*, IEEE Computer Computer Society Press, Los Alamitos, California, 1995, 158–163.
19. D. Sankoff, Edit distance for genome comparison based on non-local operations. In *Proc. of the 3rd Symposium on Combinatorial Pattern Matching*, Springer-Verlag, LNCS 644, 121–135, 1992.
20. D. Sankoff et al. Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome, *Proc. Natl. Acad. Sci. USA*, 89(1992),6575–6579.
21. E. Therman, M. Susman, *Human Chromosomes, Structure, Behavior, and Effects.* Springer-Verlag, 1993.