On the Structure of Syntenic Distance

David Liben-Nowell
Department of Computer Science
Cornell University
Ithaca, NY 14853 USA
dln@cs.cornell.edu

October 8, 1998

Abstract

This paper examines some of the rich structure of the syntenic distance measure of the evolutionary distance between genomes. This model, introduced by Ferretti, Nadeau, and Sankoff, abstracts away from the order of genes, and considers chromosomes as unordered sets of genes. The syntenic distance between two genomes is given by the minimum number of moves (fusing two chromosomes, fissioning one chromosome, or completing a reciprocal translocation between two chromosomes) required to transform one into the other. We consider previously unanalyzed approximation algorithm given by Ferretti et al, and prove that it is in fact a 2-approximation and that, further, it outperforms the algorithm presented by DasGupta et al on all instances. We prove a number of properties which give insight into the structure of optimal move sequences. We prove a monotonicity property for the syntenic distance, and give bounds on the number of moves required to solve the hardest instance of any given size. We then demonstrate that there exist instances in which any move sequence working solely within connected components is $2 - \epsilon$ times longer than optimal, which indicates that all previously proposed approximation algorithms can be no better than 2-approximations.

1 Introduction

Numerous models for measuring the evolutionary distance between two species have been proposed in the past. These models are often based upon high-level (non-point) mutations which rearrange the order of genes within a chromosome. The distance between two genomes (or chromosomes) is defined the number of moves of a certain type required to transform the first into the second. A move for the *reversal distance* [1] is the replacement of a segment of a chromosome by the segment in reversed order. For the *transposition distance* [2], a legal move consists of removing a segment of a chromosome and reinserting it at some other location in the chromosome.

In [5], Ferretti, Nadeau, and Sankoff propose a somewhat different sort of measure of genetic distance, known as *syntenic distance*. This model abstracts away from the order of the genes within each chromosome, and handles each chromosome as an unordered set of genes. The legal moves in this model are *fusions*, in which two chromosomes join into one, *fissions*, in which one chromosome splits into two, and *translocations*, in which two chromosomes exchange sets of genes. In many cases, the order of genes within each chromosome is not known, and this model allows the computation of the distance between the species regardless [4, 7]. Additional justification follows from the observation that interchromosomal evolutionary events occur with relative rarity with respect to intrachromosomal events.

Ferretti et al propose using synteny as a measure of the distance between genomes, and present a heuristic to approximate this distance. Although they give some experimental data on its performance, no formal analysis of this approximation algorithm is given. Identifying a performance guarantee for this algorithm has remained an open question since.

In [3], DasGupta et al show a number of results on the syntenic distance problem. They prove that computing the syntenic distance between genomes is NP-hard, and provide a simple polynomial-time 2-approximation. They also prove a number of other useful structural results.

Our results. As with many NP-complete problems, reasoning about the syntenic distance is difficult. We are able, however, to show some results on the structure of the problem, and analyze previously unanalyzed heuristics, including the original algorithm of Ferretti et al [5]. These results give interesting insight into the rich structure of optimal move sequences. These structural properties aid in reasoning about the syntenic distance, and may lead to improved approximation algorithms.

We prove a *monotonicity theory* for syntenic distance, showing a natural ordering on the difficulty of problem instances. We define the *syntenic diameter of order n* $D_y(n)$ (in the spirit of the reversal and transposition diameters [6]) as the maximum number of moves required to solve an instance of size n. Monotonicity identifies a worst instance of size n, and implies that $D_y(n)$ is exactly the number of moves required to solve this instance. We prove that this particular instance requires between $2n-3$ and $2n-3-\log_c(2n-3)$ moves for some constant c, using results from [3].

We analyze the previously unanalyzed approximation algorithm given by Ferretti, Nadeau, and Sankoff, settling the open question of finding a performance guarantee for this algorithm. Instance-by-instance comparison of two heuristics is a valuable notion that is rarely explored. We prove that this algorithm is *never* worse than the approximation algorithm presented in [3]. We also show that there are instances in which the algorithm performs $2-\epsilon$ away from optimal.

Call the *connected components* of an instance the connected components of the intersection graph of the chromosomes. We prove the surprising result that there are instances in which the optimal move sequence *must* connect two unconnected components, and any move sequence that fails to do so is in fact $2-\epsilon$ away from optimal. This implies that any approximation algorithm that works only with components (as all currently proposed algorithms do) is doomed to be no better than a 2-approximation.

2 Notational Preliminaries and Previous Heuristics

The syntenic distance model works as follows: a *chromosome* is a subset of a set of n *genes*, and a *genome* is a collection of k chromosomes. A genome can be transformed by any of the following moves (for S, T, U, V non-empty sets of genes): (1) a *fusion* $(S, T) \longrightarrow U$, where $S \cup T = U$; (2) a *fission* $S \longrightarrow (T, U)$, where $S = T \cup U$; (3) a *translocation* $(S, T) \longrightarrow (U, V)$, where $S \cup T = U \cup V$.

The *compact representation* of an instance of synteny is described in [5] and formalized in [3]. This representation makes the goal of each instance uniform and thus eases reasoning about move sequences. For an instance in which we are attempting to transform genome $\mathcal{G}_1$ into genome $\mathcal{G}_2$, we relabel each gene a in $\mathcal{G}_1$ by the numbers of the chromosomes of $\mathcal{G}_2$ in which a appears. Formally, we replace each of the k sets S in $\mathcal{G}_1$ with $\bigcup_{s \in S}\{\ell \mid s \in G_\ell\}$ (where $\mathcal{G}_2 = G_1, G_2, \ldots G_n$) and attempt to transform these sets into the collection $\{1\}, \{2\}, \ldots \{n\}$. As an example of the compact representation (given in [5]), consider the instance

$$\begin{array}{rclcl} \mathcal{G}_1 & = & \{x, y\}, & \text{(Chromosome 1)} \\ & & \{p, q, r\}, & \text{(Chromosome 2)} \\ & & \{a, b, c\} & \text{(Chromosome 3)} \end{array} \qquad \begin{array}{rcll} \mathcal{G}_2 & = & \{p, q, x\}, & \text{(Chromosome 1)} \\ & & \{a, b, r, y, z\} & \text{(Chromosome 2).} \end{array}$$

The compact representation of $\mathcal{G}_1$ with respect to $\mathcal{G}_2$ is $\{1,2\}, \{1,2\}, \{2\}$ and the compact representation of $\mathcal{G}_2$ with respect to $\mathcal{G}_1$ is $\{1, 2\}, \{1, 2, 3\}$. For an instance of synteny in this compact notation, we will write $\mathcal{S}(n, k)$ to refer to the instance, where there are n elements and k sets. We denote the number of moves required to solve this instance by $D(\mathcal{S}(n, k))$.

For a synteny instance $\mathcal{S}(n, k)$, we will say that two sets S_1, S_2 are *connected* if $S_1 \cap S_2 \neq \emptyset$, and that both are in the same *component*.

In [5], Ferretti et al present an approximation algorithm, reproduced in Figure 1, which we denote by $\mathcal{F}$. (Two genes are *syntenic* iff they appear in the same chromosome.) Although they provide some empirical evidence on the algorithm's performance, they do not give any formal analysis.

Let $\mathcal{H}$ denote the approximation algorithm defined in [3]: for each connected component containing n_i elements and k_i sets, perform $k_i - 1$ fusions to produce one set with all n_i elements, then $n_i - 1$ fissions to produce the n_i singletons. DasGupta et al show that this algorithm is a 2-approximation, a tight bound (the algorithm performs a factor of 2 away from optimal on the instance $\{1\}, \{1, 2\}, \ldots, \{1, 2 \ldots n\}$).

3 An Analysis of $\mathcal{F}$

In this section, we prove several results about $\mathcal{F}$. We show that (1) $\mathcal{F}$ is a 2-approximation, (2) $\mathcal{F}$ is never worse than $\mathcal{H}$, and (3) there is an instance in which $\mathcal{F}$ is a factor of two away from optimal.

Theorem 3.1 *For an arbitrary instance $\mathcal{S}(n, k)$ of synteny, $|\mathcal{F}(\mathcal{S}(n, k))| \leq |\mathcal{H}(\mathcal{S}(n, k))|$.*

Proof. Suppose that $\mathcal{F}$ generates a move sequence σ on $\mathcal{S}(n, k)$. Suppose that in σ there are m_1 fissions (from Operation (1)), m_2 translocations (from Operations (2) and (3)), and m_3 fusions (from Operation (3)).

Select a gene ℓ to work on, under the following priorities:

Priority (A). any ℓ for which $r(\ell) = 1$.

Priority (B). any ℓ for which $r(\ell) = 2$.

Priority (C). if all $r(\ell) > 2$, pick ℓ which minimizes $r(\ell)$ and, if there are several such, which minimizes $r(\ell')$ for some ℓ' in the chromosome remaining from the last operation involving ℓ. If there are several such, choose ℓ so that after it is operated on, $\sum_\ell r(\ell)$ is minimized.

For the ℓ selected above, do the following operation:

Operation (1). If $r(\ell) = 1$ and some of the genes syntenic with ℓ appear in no other chromosomes, effect a fission to create a separate chromosome $\{\ell\}$.

Operation (2). If $r(\ell) = 1$ and all genes ℓ' syntenic with ℓ appear in $r(\ell') \geq r_{\min} > 1$ chromosomes, effect a translocation to obtain a separate chromosome $\{\ell\}$. The second chromosome involved in the translocation is one that contains some gene ℓ' syntenic with ℓ, with $r(\ell') = r_{\min}$, and, if there are several, with a maximal number of genes syntenic with ℓ.

Operation (3). If $r(\ell) > 1$, effect $r(\ell) - 2$ fusions followed by one translocation[a], again to produce a separate $\{\ell\}$.

[a] This translocation could actually be a fusion if no other genes are present in the component.

Figure 1: The approximation algorithm $\mathcal{F}$. [5]

Every translocation generated by Operation (2) is of the form $(S \cup \{\ell\}, T) \longrightarrow (S \cup T, \{\ell\})$ where S and T both contain some gene ℓ' and $\ell \notin S, T$. Every translocation generated by Operation (3) is of the form $(S \cup \{\ell\}, T \cup \{\ell\}) \longrightarrow (S \cup T, \{\ell\})$ where $\ell \notin S, T$. Note that in either case, at the time that $\{\ell\}$ is produced, it appears nowhere else in the genome (i.e., $r(\ell) = 1$).

We create a new move sequence σ' which differs from σ in that each translocation $(S_1 \cup S_2, T_1 \cup T_2) \longrightarrow (S_1 \cup T_1, S_2 \cup T_2)$ is replaced by the two-move sequence

$$(S_1 \cup S_2, T_1 \cup T_2) \longrightarrow S_1 \cup S_2 \cup T_1 \cup T_2 \longrightarrow (S_1 \cup T_1, S_2 \cup T_2).$$

By the form of the translocations and this translation, we have the following facts:

- Each of the newly-created fusions is within a connected component (the input sets are connected by ℓ' for Operation (2) and ℓ for Operation (3)).

- Each of the newly-created fissions produces a singleton $\{\ell\}$ for a gene ℓ that appears nowhere else in the genome.

Now we examine the fusions and fissions in σ. Each original fusion (from Operation (3)) is also within a component (the two input sets are connected by ℓ), and each fission (in Operation (1)) produces a singleton for a gene that appears nowhere else in the genome. Thus, every fusion in σ' fuses two sets in the same component, and every fission in σ' produces a singleton set with an element that appears nowhere else in the genome.

Clearly we can rearrange σ' to completely solve each component before beginning the next, since there are no intercomponent dependencies. Further, inside each component we can put all the fusions before all the fissions, since the fissions merely remove the last instance of an element from a larger set. In other words, the rearranged σ' does *exactly* what $\mathcal{H}$ does: within each component, it fuses all the sets into one massive set, and then fissions off individual elements one at a time. Note that $|\sigma'| = m_1 + 2 \cdot m_2 + m_3 = m_2 + |\sigma|$, and thus $|\sigma| = |\sigma'| - m_2 = |\mathcal{H}(\mathcal{S}(n, k))| - m_2$. In other words, $\mathcal{F}$ performs m_2 moves better than $\mathcal{H}$ on each input. $\square$

Corollary 3.2 *$\mathcal{F}$ is a 2-approximation.*

Proof. Immediate from Theorem 3.1 and the fact that $\mathcal{H}$ is a 2-approximation. □

We can now show the corresponding lower bound on the approximation ratio with the following lemma.

Lemma 3.3 *For any $\epsilon > 0$, there exists a instance $\mathcal{S}(n,k)$ with $|\mathcal{F}(\mathcal{S}(n,k))| \geq 2 \cdot D(\mathcal{S}(n,k)) - \epsilon$.*

Proof. Select any n such that $1/(n-1) \leq \epsilon$. We give a synteny instance $\mathcal{S}(n,n)$ such that $D(\mathcal{S}(n,n)) = n-1$ and $|\mathcal{F}(\mathcal{S}(n,n)| = 2n-3$. Then the ratio between the result of $\mathcal{F}$ and the optimal is $(2(n-1)-1)/(n-1)$, i.e. only $1/(n-1)$ better than two times optimal.

The instance $\mathcal{S}(n,n)$ consists of $\{1,2,\ldots n\}$ and $n-1$ copies of $\{n\}$. First we claim that the optimal move sequence for this instance takes $n-1$ moves. Here is an $n-1$ move sequence, where $\Delta_1 = \{1,\ldots,n\}$.

$$\text{For } i = 1 \text{ to } n-1, \text{ let } \sigma_i = (\Delta_i, \{n\}) \longrightarrow (\Delta_{i+1}, \{i\}), \text{ where } \Delta_{i+1} = \Delta_i - \{i\}.$$

Each move removes one of the $n-1$ genes appearing only in the large set while absorbing another of the singleton $\{n\}$ sets, so that after $n-1$ of these moves all the ns have been joined.

Now, we examine what $\mathcal{F}$ does on this input. Genes $1,2,\ldots n-1$ are exactly symmetric in this instance, so we assume without loss of generality that $\mathcal{F}$ selects them in ascending order. For the selection of $\ell = 1$ through $n-2$ by Priority (A), there is only one chromosome in which ℓ appears, and there is another chromosome on that gene that does not appear in any other chromosome (specifically, $n-1$). The conditions are met for performing Operation (1), so $\mathcal{F}$ effects $n-2$ fissions, separating off the singletons $\{1\},\{2\},\ldots\{n-2\}$. So after $n-2$ moves, we have

$$\{1\},\{2\},\ldots\{n-2\},\{n-1,n\},\overbrace{\{n\},\{n\},\ldots\{n\}}^{n-1 \text{ times}}.$$

Then, since $n-1$ appears only once, it is selected as the next ℓ by Priority (A). $r(\ell) = 1$, and no other gene syntenic with ℓ appears only on that chromosome, so an Operation (2) translocation is performed with the chromosome with maximum overlap with $\{n-1,n\}$, which must be one of the singletons. The move is thus $(\{n-1,n\},\{n\}) \longrightarrow (\{n-1\},\{n\})$. This yields

$$\{1\},\{2\},\ldots\{n-2\},\{n-1\},\overbrace{\{n\},\{n\},\ldots\{n\}}^{n-1 \text{ times}}.$$

$\mathcal{F}$ then has no choice but to select $\ell = n$ (Priority (C)), and fuse the $n-1$ singletons (Operation (3)). This takes $n-2$ moves, yielding $\{1\},\{2\},\ldots\{n\}$. Thus $\mathcal{F}$ required $n-2$ fissions, 1 translocation, and $n-2$ fusions, or $2n-3$ moves total. □

Note that the non-optimality of $\mathcal{F}$ in the above example is only as the result of applications of Operation (1). When the applications of this operation have been completed, the result is

$$\{1\},\{2\},\ldots\{n-2\},\{n-1,n\},\overbrace{\{n\},\{n\},\ldots\{n\}}^{n-1 \text{ times}}.$$

$\mathcal{F}$ takes $n-1$ more moves after this point. DasGupta et al prove that any instance with n sets and p components requires at least $n-p$ moves [3], and this instance has $2n-2$ sets and $n-1$ components. Thus at least $n-1$ moves are required to solve this instance, and $\mathcal{F}$ is optimal after this point. So the non-optimality of Operation (1) is sufficient to cause $\mathcal{F}$ to be a factor of 2 away from optimal.

The difficulty with $\mathcal{F}$ results from overzealous application of Operation (1) when Operation (2) could do some good. (Notice from Theorem 3.1 that the more translocations $\mathcal{F}$ does, the better its performance.) Call $\mathcal{F}'$ the algorithm resulting from making the obvious fixes to $\mathcal{F}$ to deal with this problem:

- Apply Operation (1) only if *all* of the genes syntenic with ℓ appear in no other chromosomes.
- Apply Operation (2) if *any* gene syntenic with ℓ appears in another chromosome. The second chromosome involved in the translocation is selected as in $\mathcal{F}$, but ignoring those genes ℓ' that are in the same chromosome as ℓ and appear nowhere else in the genome.

A similar proof to Theorem 3.1 yields that $\mathcal{F}'$ is never worse than $\mathcal{H}$ on any instance, and is therefore a 2-approximation. See Section 4 for further analysis of $\mathcal{F}'$.

4 Moves between Connected Components

It seems intuitive that when attacking an instance of synteny consisting of two distinct connected components the optimal move sequence would never fuse these components together. Both $\mathcal{H}$ and $\mathcal{F}$ (and $\mathcal{F}'$) work within connected components, in fact. However, the following theorem shows that this approach is doomed to be no better than a 2-approximation.

Theorem 4.1 *For any algorithm $\mathcal{A}$ attempting to approximate syntenic distance, if $\mathcal{A}$ works only within components then, for any $\epsilon > 0$, there is an instance where $\mathcal{A}$ is $2-\epsilon$ away from optimal.*

Proof. We construct an instance of synteny in which any component-based move sequence will require nearly twice as many moves as the optimal move sequence.

Consider the instance $\mathcal{S}(n, n)$ consisting of $\{1, 2, \ldots (n-1)\}$ and $n-1$ copies of $\{n\}$. First we observe that there is a move sequence solving this instance in n moves:

move 1:	$(\{1 \ldots (n-1)\}, \{n\}) \longrightarrow (\{1 \ldots (n-2), n\}, \{n-1\})$
moves i = 2 through (n − 1):	$(\{1 \ldots (n-i), n\}, \{n\}) \longrightarrow (\{1 \ldots (n-i-1), n\}, \{n-i\})$
move n:	$\{1, n\} \longrightarrow (\{1\}, \{n\})$.

Move 1 joins the two components while producing the singleton $\{n-1\}$, and then the next $n-2$ moves translocate off one of the elements from $\{2 \ldots (n-2)\}$ while absorbing an additional singleton $\{n\}$. The final move fissions the remaining set when we have run out of singletons and have just the doubleton $\{1, n\}$ remaining.

For any algorithm $\mathcal{A}$ working only within components, however, the moves that $\mathcal{A}$ can make are severely limited. Since $\{1 \ldots (n-1)\}$ is in a component all by itself, there is no choice but to complete $n-2$ fissions. Similarly, the $n-1$ copies of $\{n\}$ are an entire component. Thus the only possible moves are to complete $n-2$ fusions to create a unique singleton.

Therefore, $\mathcal{A}$ completes $2n-4$ moves on this instance. Selecting n so that $\epsilon > 4/n$ yields an instance where $\mathcal{A}$ is $2-\epsilon$ away from optimal. □

It is now natural to define the *connected synteny problem*, to find the minimum number of moves required to transform one genome into another with all moves constrained to work only within components. We will use $\widehat{D}(\mathcal{S}(n, k))$ to denote the minimum number of moves within components required to solve a synteny instance $\mathcal{S}(n, k)$. Note that $\mathcal{H}$ and $\mathcal{F}$ are 2-approximations for this problem, as well. This is again a tight bound for both, as the optimal move sequences work only within components for the examples in which these algorithms perform a factor of two away from optimal. (In fact, both examples have only one component.) $\mathcal{F}'$ also works only within components, and is therefore not better than a 2-approximation in the general case. For the connected synteny problem, however, determining the approximation ratio of $\mathcal{F}'$ remains an open question.

5 Non-Redundancy and Monotonicity

In this section, we show that any optimal move sequence never has a move that produces two sets with non-empty intersection. We also prove a monotonicity property for syntenic distance.

We first need to introduce an extension to our notation to handle the case of empty sets as input. If $S_1 \dots S_k$ is a collection of sets and $S_i = \emptyset$, we understand the synteny instance $\mathcal{S}(n, k)$ consisting of $S_1 \dots S_k$ to represent the synteny instance $T(n, k-1)$ consisting of $S_1 \dots S_{i-1}, S_{i+1} \dots S_k$.

Lemma 5.1 *If there is a move sequence $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_m)$ solving $S_1 \dots S_i \cup \{a\} \dots S_k$ where $a \notin S_i$ (with S_i possibly empty), then there exists a move sequence σ' solving $S_1 \dots S_i \dots S_k$ in at most m steps.*

Proof by induction on m.
Base case ($m = 1$). Then σ_1 must solve the instance. We have two cases (a cannot appear in more than one additional set, since otherwise no single move could solve the instance):

- The element a also appears in some set $S_{j \neq i}$.

  σ_1 must take $S_i \cup \{a\}$ and S_j as input, and produce the singleton $\{a\}$ as output. Otherwise, two copies of the gene a remain, or the copy of a is bundled up with some other element(s). This first restriction implies that σ_1 cannot be a fission.

  If σ_1 is the fusion $(S_j, S_i \cup \{a\}) \longrightarrow \{a\}$, it must be the case that $S_j = \{a\}$ and $S_i = \emptyset$. Thus $S_1 \dots S_k$ is already in the target form, and in the new instance we are done without making any move.

  If σ_1 is a translocation, a must occur in only one of the output sets, for otherwise it appears twice and the instance is not solved. Thus $\sigma_1 = (S_i \cup \{a\}, S_j) \longrightarrow (S_i \cup [S_j - \{a\}], \{a\})$. We can replace this by $\sigma_1' = (S_i, S_j) \longrightarrow (S_i \cup [S_j - \{a\}], \{a\})$ to solve the instance $S_1 \dots S_k$.

- a does not appear elsewhere in the genome.

  Then the last move need not involve the singleton $\{a\}$. If it does not, then it must be the case that $S_i = \emptyset$. (Otherwise after the last move of the sequence a is in a non-singleton and the instance has not been solved.) In this case, simply doing the last move will solve $S_1 \dots S_k$.

  If the last move does involve $S_i \cup \{a\}$, it is not a fusion since any fusing would couple a with some other element. (a would have to be coupled with some element $b \neq a$, since a does not appear elsewhere in the genome.)

  If σ_1 is a fission, then it must produce a singleton set $\{a\}$ and some other set not containing a in order to solve in the instance. Since $a \notin S_i$, this means that $\sigma_1 = S_i \cup \{a\} \longrightarrow (\{a\}, S_i)$. If we replace $S_i \cup \{a\}$ by S_i in the instance, the instance is already in the target form and we can skip this move.

  If σ_1 is a translocation, it must be $(S_j, S_i \cup \{a\}) \longrightarrow (U, \{a\})$ for some set U, or else (as with the fusion case) the instance would not be solved. If $a \in U$ then the instance is not solved, since a appears twice. Therefore it must be the case that $U = S_i \cup S_j$. To solve the new instance, we can simply do the fusion $(S_i, S_j) \longrightarrow S_i \cup S_j$ and we are done.

Inductive case ($m \geq 2$). First we handle the cases when σ_1 is any move that does not involve the set $S_i \cup \{a\}$. For ℓ and j distinct from i:

- $\sigma_1 = (S_\ell, S_j) \longrightarrow S_\ell \cup S_j$. Then $\sigma_{2\ldots m}$ solves $S_r(1 \leq r \leq k, r \neq \ell, r \neq j, r \neq i), S_i \cup \{a\}, S_\ell \cup S_j$. By the inductive hypothesis, we have a move sequence σ' solving $S_r(1 \leq r \leq k, r \neq \ell, r \neq j, r \neq i), S_i, S_\ell \cup S_j$ in at most $m-1$ moves.

- $\sigma_1 = S_\ell \longrightarrow (U, V)$. Then $\sigma_{2\ldots m}$ solves $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_i \cup \{a\}, U, V$. By the inductive hypothesis, we have a move sequence σ' solving $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_i, U, V$ in at most $m-1$ moves.

- $\sigma_1 = (S_\ell, S_j) \longrightarrow (U, V)$. Then $\sigma_{2\ldots m}$ solves $S_r(1 \leq r \leq k, r \neq \ell, r \neq j, r \neq i), S_i \cup \{a\}, U, V$. By the inductive hypothesis, we have a move sequence σ' solving $S_r(1 \leq r \leq k, r \neq \ell, r \neq j, r \neq i), S_i, U, V$ in at most $m-1$ moves.

In each case, doing σ_1 and σ' solves $S_1 \ldots S_k$ in at most m moves. We now consider the cases in which σ_1 takes $S_i \cup \{a\}$ as input.

- Suppose σ_1 is a fission, and that $S_i = S_{i_1} \cup S_{i_2}$.

  If $\sigma_1 = S_i \cup \{a\} \longrightarrow (S_{i_1} \cup \{a\}, S_{i_2})$, then $\sigma_{2\ldots m}$ solves the instance $S_r(1 \leq r \leq k, r \neq i), S_{i_1} \cup \{a\}, S_{i_2}$. By the inductive hypothesis, there is a σ' solving $S_r(1 \leq r \leq k, r \neq i), S_{i_1}, S_{i_2}$ in at most $m-1$ steps. Then doing $S_i \longrightarrow (S_{i_1}, S_{i_2})$ followed by σ' solves $S_1 \ldots S_k$ in at most m steps.

  If $\sigma_1 = S_i \cup \{a\} \longrightarrow (S_{i_1} \cup \{a\}, S_{i_2} \cup \{a\})$, then $\sigma_{2\ldots m}$ solves the instance $S_r(1 \leq r \leq k, r \neq i), S_{i_1} \cup \{a\}, S_{i_2} \cup \{a\}$ in $m-1$ moves. By the inductive hypothesis applied to σ and $S_{i_1} \cup \{a\}$, there is a σ' solving $S_r(1 \leq r \leq k, r \neq i), S_{i_1}, S_{i_2} \cup \{a\}$ in at most $m-1$ steps. Applying the inductive hypothesis again, this time to σ' and $S_{i_2} \cup \{a\}$, we have that there is a σ'' solving $S_r(1 \leq r \leq k), S_{i_1}, S_{i_2}$ in at most $m-1$ steps. Then doing $S_i \longrightarrow (S_{i_1}, S_{i_2})$ followed by σ'' solves $S_1 \ldots S_k$ in at most m steps.

- Suppose that σ_1 is the fusion $(S_i \cup \{a\}, S_\ell) \longrightarrow S_i \cup \{a\} \cup S_\ell$. Then $\sigma_{2\ldots m}$ solves the instance $S_r(1 \leq r \leq m, r \neq \ell, r \neq i), S_i \cup \{a\} \cup S_\ell$ in $m-1$ steps.

  If $a \in S_\ell$, then $S_i \cup \{a\} \cup S_\ell = S_i \cup S_\ell$. Thus $\sigma_{2\ldots m}$ solves $S_r(1 \leq r \leq m, r \neq \ell, r \neq i), S_i \cup S_\ell$, and doing $(S_i, S_\ell) \longrightarrow S_i \cup S_\ell$ and $\sigma_{2\ldots m}$ solves $S_1 \ldots S_k$ in m steps.

  If $a \notin S_\ell$, then by the inductive hypothesis, there is a σ' solving $S_r(1 \leq r \leq m, r \neq \ell, r \neq i), S_i \cup S_\ell$ in $m-1$ steps. To solve $S_1 \ldots S_k$, we do the fusion $(S_i, S_\ell) \longrightarrow S_i \cup S_\ell$ and run σ', which requires at most m steps.

- Suppose σ_1 is a translocation using the set $S_i \cup \{a\}$ and S_ℓ, where $S_i = S_{i_1} \cup S_{i_2}$ and $S_\ell = S_{\ell_1} \cup S_{\ell_2}$. Then σ_1 must look like one of the following:

$$
\begin{array}{ll}
(1) & (S_\ell, S_i \cup \{a\}) \longrightarrow (S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2} \cup \{a\}) \\
(2) & (S_\ell, S_i \cup \{a\}) \longrightarrow (S_{\ell_1} \cup S_{i_1} \cup \{a\}, S_{\ell_2} \cup S_{i_2} \cup \{a\}).
\end{array}
$$

  In either case we replace this move by the translocation $\sigma_1' = (S_\ell, S_i) \longrightarrow (S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2})$.

  In case (1), if $a \in S_{\ell_2}$, then $\sigma_{2\ldots m}$ solves $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2}$ in $m-1$ steps, since $S_{\ell_2} \cup S_{i_2} \cup \{a\} = S_{\ell_2} \cup S_{i_2}$. Then we can do σ_1' and $\sigma_{2\ldots m}$ to solve $S_1 \ldots S_k$ in m steps.

If $a \notin S_{\ell_2}$, then $\sigma_{2\ldots m}$ solves $S_r(1 \le r \le k, r \ne \ell, r \ne i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2} \cup \{a\}$ in $m-1$ steps. By the inductive hypothesis, there is a move sequence σ' solving $S_r(1 \le r \le k, r \ne \ell, r \ne i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2}$ in at most $m-1$ steps. Gluing this together with σ'_1 yields a sequence solving $S_1 \ldots S_k$ at most m moves.

In case (2), if $a \in S_{\ell_1}$ then $S_{\ell_1} \cup S_{i_1} \cup \{a\} = S_{\ell_1} \cup S_{i_1}$ and this move is actually $(S_\ell, S_i \cup \{a\}) \longrightarrow (S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2} \cup \{a\})$, which is exactly case (1). Otherwise, $a \notin S_{\ell_1}$. If $a \in S_{\ell_2}$, for exactly the same reason as above (with the roles of S_{ℓ_1} and S_{ℓ_2} reversed), we are again in case (1). Thus the only interesting case is when $a \notin S_{\ell_1}$ and $a \notin S_{\ell_2}$.

In this case, $\sigma_{2\ldots m}$ solves $S_r(1 \le r \le k, r \ne \ell, r \ne i), S_{\ell_1} \cup S_{i_1} \cup \{a\}, S_{\ell_2} \cup S_{i_2} \cup \{a\}$ in $m-1$ moves. By the inductive hypothesis applied to σ and $S_{\ell_1} \cup S_{i_1} \cup \{a\}$, we have a move sequence σ' solving $S_r(1 \le r \le k, r \ne \ell, r \ne i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2} \cup \{a\}$ in at most $m-1$ moves. Applying the inductive hypothesis again, to σ' and $S_{\ell_2} \cup S_{i_2} \cup \{a\}$, we have a move sequence σ'' solving $S_r(1 \le r \le k, r \ne \ell, r \ne i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2}$ in at most $m-1$ moves. This is exactly the result of doing the translocation σ'_1, so doing σ'_1 and σ'' solves $S_1 \ldots S_k$ in at most m moves. □

Define a *redundant* move as any move creating two sets S and T such that $S \cap T \ne \emptyset$. (Note that only fissions and translocations can be redundant, because fusions do not create two sets.)

We need a result on reordering from [3] to prove the following theorem: for $\mathcal{S}(n,k)$ an instance of synteny and $\sigma = (\sigma_1, \ldots, \sigma_m)$ any move solving the instance with m_1 fusions, m_2 translocations, and m_3 fissions, there exists a move sequence σ' solving the instance in $m' \le m$ moves in which every fusion precedes every translocation precedes every fission, using $m'_1 \le m_1$ fusions, $m'_2 \le m_2$ translocations, and $m'_3 \le m_3$ fissions. (DasGupta et al actually prove this lemma for the case where σ is optimal, but the proof extends to a general σ straightforwardly.) We refer to a move sequence in which the fusions precede the translocations precede the fissions as in *canonical order.*

Theorem 5.2 *For any synteny instance $\mathcal{S}(n,k)$, there exists an optimal move sequence making no redundant moves.*

Proof. Let $\sigma = (\sigma_1, \ldots, \sigma_m)$ be a canonically-ordered optimal move sequence solving $\mathcal{S}(n,k)$. There are no redundant fusions at all (by the definition of a redundant move). Any redundant fission must yield two copies of at least one gene a, say $S_1 \cup S_2 \cup \{a\} \longrightarrow (S_1 \cup \{a\}, S_2 \cup \{a\})$. But then there are two copies of the gene a, and since all succeeding moves are also fissions, the number of as can only increase, and therefore the instance will not be solved.

Then the only possible redundant moves are translocations of the form $(T_1 \cup T_2 \cup V, U_1 \cup U_2 \cup W) \longrightarrow (T_1 \cup U_1 \cup V \cup W, T_2 \cup U_2 \cup V \cup W)$ for some non-empty overlap $V \cup W$, with $V \cap (T_1 \cup T_2) = \emptyset$ and $W \cap (U_1 \cup U_2) = \emptyset$. Then by repeatedly applying the transformation described Lemma 5.1 to σ for every element of $V \cup W$, we can solve the instance resulting from replacing this redundant move by the translocation $(T_1 \cup T_2 \cup V, U_1 \cup U_2 \cup W) \longrightarrow (T_1 \cup U_1 \cup V \cup W, T_2 \cup U_2)$ in at most as many moves. Repeating this sequentially for every redundant move in σ yields a move sequence of length at most m with no redundant moves. □

Note that the canonicalizing process from does not create redundancies: with a non-redundant move sequence as input, it produces a non-redundant canonical move sequence as output. Thus we can convert any move sequence σ into a non-redundant canonical move sequence by applying consecutively canonicalization, redundancy elimination, and canonicalization again.

Theorem 5.3 *(Monotonicity) Let $S_1 \ldots S_k$ and $T_1 \ldots T_k$ be two collections of sets where, for all $1 \le i \le k$, $T_i \subseteq S_i$. Let $n = |\bigcup_i S_i|$ and $n' = |\bigcup_i T_i|$. Let $\mathcal{S}(n,k)$ be the instance of synteny* *consisting of $S_1 \dots S_k$ and let $\mathcal{T}(n', k)$ be the instance consisting of $T_1 \dots T_k$. Then $D(\mathcal{S}(n,k)) \geq D(\mathcal{T}(n', k))$.*

Proof by induction on $\delta = \sum_i |S_i - T_i|$.

Base case ($\delta = 0$). Then each $S_i = T_i$, and $\mathcal{T}(n', k) = \mathcal{S}(n, k)$, so their distances are trivially equal.

Inductive case ($\delta \geq 1$). Let $\sigma = (\sigma_1, \sigma_2, \dots \sigma_m)$ be an optimal move sequence solving $\mathcal{S}(n, k)$. Let j be the minimum index such that $S_j \supset T_j$ and let a be any element in $S_j - T_j$. By applying the transformation described in Lemma 5.1, we can convert σ into a σ' solving $S_1, S_2 \dots S_j - \{a\} \dots S_k$ in at most m steps. By the inductive hypothesis, then, since this instance is one element "closer" to $\mathcal{T}(n', k)$, we can solve $\mathcal{T}(n', k)$ in at most m steps. □

6 Syntenic Diameter

We define the *syntenic diameter of order* n as

$$D_y(n) \stackrel{\text{def}}{=} \max_{\mathcal{S}(n,\,n)} D(\mathcal{S}(n, n)),$$

the number of moves required to solve the worst instance of up to n elements and n sets. We also define the *complete n-instance* $\mathcal{K}_n(n, n)$ of synteny, which consists of n copies of the set $\{1, \dots, n\}$. Monotonicity immediately gives us the following, since any instance with n elements and n sets is no harder to solve than $\mathcal{K}_n(n, n)$:

Proposition 6.1 *$D_y(n) = D(\mathcal{K}_n(n, n))$.*

The following restricted form of the synteny problem was defined in [3]. Define the *linear synteny* problem as a the synteny problem in which all move sequences are constrained as follows: The first $k-1$ moves must be fusions or severely restricted translocations, as follows. One of the input sets is designated as the *merging set*. Each of the first $k-1$ moves takes the current merging set Δ as input, along with one unused input set S, and produces a new merging set Δ'. If some element a appears nowhere in the genome except in Δ and S, then the move is the translocation $(\Delta, S) \longrightarrow (\Delta', \{a\})$, where $\Delta' = (\Delta \cup S) - \{a\}$. If there is no such element a, then the move simply fuses the two sets: $(\Delta, S) \longrightarrow \Delta'$, where $\Delta' = \Delta \cup S$. If Δ is the merging set after the $k-1$ fusions and translocations, then the next $|\Delta| - 1$ moves simply fission off a singleton and produce the new merging set. Let $\widetilde{D}(\mathcal{S}(n, k))$ be the length of the optimal linear move sequence. DasGupta et al prove that for any instance $\mathcal{S}(n, k)$ of synteny, $\widetilde{D}(\mathcal{S}(n, k)) \leq D(\mathcal{S}(n, k)) + \log_c D(\mathcal{S}(n, k))$, for some constant c. (In the full version of their paper, DasGupta et al show that we can take $c = 4/3$.)

Lemma 6.2 *$\widetilde{D}(\mathcal{K}_n(n, n)) = 2n - 3$.*

Proof. Consider any linear move sequence solving $\mathcal{K}_n(n, n)$. The first $n-2$ moves can only use $n-1$ of the input sets, since each move after the first can only use one new set. Thus after $n-2$ moves, the nth input set still contains a copy of each element in the instance. Therefore moves 1 through $n-2$ must have been fusions. This leaves merging set $\Delta = \{1, \dots, n\}$ and one input set $\{1, \dots, n\}$. Thus any linear move sequence performs one translocation and $n-2$ fissions to solve the instance. This is $2n-3$ total moves. □

Theorem 6.3 *$2n - 3 \geq D(\mathcal{K}_n(n, n)) \geq 2n - 3 - \log_c(2n - 3)$.*

Proof. Clearly for any synteny instance $D(\mathcal{S}(n,k)) \leq \widetilde{D}(\mathcal{S}(n,k))$. Then we have from the bound on linear synteny proved by DasGupta et al that

$$\begin{aligned}\widetilde{D}(\mathcal{K}_n(n,n)) \;\geq\; D(\mathcal{K}_n(n,n)) &\geq \widetilde{D}(\mathcal{K}_n(n,n)) - \log_c D(\mathcal{K}_n(n,n))) \\ &\geq \widetilde{D}(\mathcal{K}_n(n,n)) - \log_c \widetilde{D}(\mathcal{K}_n(n,n)).\end{aligned}$$

By Lemma 6.2, we have

$$2n-3 \;\geq\; D(\mathcal{K}_n(n,n)) \;\geq\; 2n-3-\log_c(2n-3).$$

□

Note that this is almost tight, with only a $\log_c(2n-3)$ window for the syntenic diameter. This result may help in the development of improved approximations, since, for instances close to $\mathcal{K}_n(n,n)$, this theorem indicates that $\mathcal{H}$ performs very close to optimally.

Conjecture 6.4 $D(\mathcal{K}_n(n,n)) = 2n-3$.

7 Conclusions and Future Work

We have proven a number of interesting structural results for syntenic distance, including monotonicity and the fact that improving the approximation ratio for this problem will require an algorithm that works among components. These results may help in solving the obvious remaining open question:

- Is there an approximation algorithm for syntenic distance that achieves an approximation ratio strictly better than 2?

Other interesting open questions include:

- Can we improve the approximation ratio for connected synteny? In particular, what is the performance of $\mathcal{F}'$?
- Can we improve the bound on $D(\mathcal{K}_n(n,n))$?

Acknowledgements. We thank Jon Kleinberg for extensive and fruitful discussions on numerous aspects of this paper and the syntenic distance problem.

References

[1] Vineet Bafna and Pavel A. Pevzner. Genome rearrangements and sorting by reversals. In *34th IEEE Symposium on Foundations of Computer Science*, pages 148–157, 1993.

[2] Vineet Bafna and Pavel A. Pevzner. Sorting by transpositions. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 614–623, 1995.

[3] Bhaskar DasGupta, Tao Jiang, Sampath Kannan, Ming Li, and Elizabeth Sweedyk. On the complexity and approximation of syntenic distance. In *1st Annual International Conference on Computational Molecular Biology (RECOMB '97)*, pages 99–108, 1997.

[4] Jason Ehrlich, David Sankoff, and Joseph H. Nadeau. Synteny conservation and chromosome rearrangements during mammalian evolution. *Genetics*, 147(1):289–296, September 1997.

[5] Vincent Ferretti, Joseph H. Nadeau, and David Sankoff. Original synteny. In *Combinatorial Pattern Matching, 7th Annual Symposium*, pages 159–167, 1996.

[6] Pavel Pevzner and Michael Waterman. Open combinatorial problems in computational molecular biology. In *Proceedings of the Third Israel Symposium on Theory of Computing and Systems*, pages 158–173, January 1995.

[7] David Sankoff and Joseph H. Nadeau. Conserved synteny as a measure of genomic distance. *Discrete Applied Mathematics*, 71:247–257, 1996.