On the Structure of Syntenic Distance[*]

David Liben-Nowell
Department of Computer Science
Carleton College
Northfield, MN 55057 USA
`dlibenno@carleton.edu`

Abstract

This paper examines some of the rich structure of the syntenic distance model of evolutionary distance, introduced by Ferretti, Nadeau, and Sankoff. The syntenic distance between two genomes is the minimum number of fissions, fusions, and translocations required to transform one into the other, ignoring gene order within chromosomes. We prove that the previously unanalyzed algorithm given by Ferretti et al. is a 2-approximation and no better, and that, further, it always outperforms the algorithm presented by DasGupta, Jiang, Kannan, Li, and Sweedyk. We also prove the same results for an improved version of the Ferretti et al. algorithm.

We then prove a number of properties which give insight into the structure of optimal move sequences. We give instances in which any move sequence working solely within connected components is nearly twice optimal, and a general lower bound based on the spread of genes from each chromosome. We then prove a monotonicity property for the syntenic distance, and bound the difficulty of the hardest instance of any size. We discuss the results of implementing these algorithms and testing them on real and simulated synteny data.

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 as the minimum 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 same 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.

Ferretti, Nadeau, and Sankoff [7] propose a somewhat different sort of measure of genetic distance, known as the *syntenic distance*. This model abstracts away from the order of the genes within chromosomes, and handles each chromosome as an unordered set of genes. The transformations are *fusions*, in which two chromosomes join into one, *fissions*, in which one chromosome splits into two, and reciprocal *translocations*, in which two chromosomes exchange subsets of their genes. In

[*]Appears in *Journal of Computational Biology*, Volume 8, Number 1, February 2001, pp. 53–68. A preliminary version of this paper appeared in *Proceedings of the 10th Annual Conference on Combinatorial Pattern Matching (CPM '99)*. Minor changes have been made in this document; the last update was on 4 August 2005. Comments are welcome.

biological practice, the order of genes within chromosomes is often unknown, and this model allows the computation of the distance between species regardless. Additional justification follows from the observation that interchromosomal evolutionary events may occur with far different frequency than intrachromosomal events. (For some discussion of this and related models, see [6] and [13].)

Ferretti et al. [7] 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 algorithm is given. Identifying a performance guarantee for this algorithm has remained an open question since.

DasGupta, Jiang, Kannan, Li, and Sweedyk [5] 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. These results give interesting insight into the rich structure of optimal move sequences. The structural properties aid in reasoning about the syntenic distance, and may lead to improved approximation algorithms.

Using results of [5], we prove a general lower bound for the syntenic distance between two genomes. Intuitively, if for many chromosomes C in one genome, genes from C appear in many of the chromosomes of the other genome, then the distance between the genomes is large. This lower bound may be helpful in developing improved approximation algorithms, since it implies that for the class of instances in which this scattering occurs, previously proposed algorithms are already less than a factor of 2 away from optimal.

We prove a *monotonicity theory* for syntenic distance, showing a natural ordering on the difficulty of problem instances. We define the *syntenic diameter* of n-chromosome species $\mathsf{diameter}(n)$—in the spirit of the reversal and transposition diameters (see [11])—as the maximum number of moves required to solve an instance in which both genomes have at most n chromosomes. Monotonicity identifies a worst instance of this size, and implies that $\mathsf{diameter}(n)$ is exactly the number of moves required to solve this instance. In recent work [9], we have shown that this number is $2n-4$. (We have recently learned that Pisanti and Sagot [12] have given an independent proof of this fact.) Here, we also consider the analogous question for the *linear synteny* problem, a restricted version of the synteny problem defined by DasGupta et al. [5]. Using our lower bound, we prove that the *linear syntenic diameter* $\mathsf{diameter}_{linear}(n)$ is exactly $2n-3$.

Instance-by-instance comparison of two heuristics is a valuable notion that is rarely explored. This type of analysis leads to very strong results in comparing the performance of two approximation algorithms, even those with the same approximation ratio. Using this technique, we analyze the previously unanalyzed approximation algorithm given by Ferretti et al., settling the open question of finding a performance guarantee for this heuristic. We prove that this algorithm is never worse than the approximation algorithm presented by [5], immediately giving a performance guarantee of 2. We further show that there are instances in which the algorithm performs $2-\varepsilon$ away from optimal.

We also consider the algorithm resulting from making the fixes necessary to handle these instances in which the original algorithm performs nearly twice optimal. We prove the same results about this modified heuristic: it is also a 2-approximation that always outperforms the algorithm of DasGupta et al., and there are instances in which it performs a factor of $2-\varepsilon$ 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 $2-\varepsilon$ 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. This raises the new problem of *connected synteny*, in which move sequences are constrained to work only within connected components. The above results indicate that the algorithms of [5] and [7], and the modified version of the latter, are only 2-approximations for connected synteny, as well.

We also discuss an implementation of the syntenic distance model and all of the algorithms discussed above. We analyze the results of running all three algorithms on both randomly generated data and seven sets of real synteny data from the Institut National de la Recherche Agronomique (INRA) Comparative Homology Database.

2 Notational Preliminaries and Previous Heuristics

The syntenic distance model is as follows: a *chromosome* is a subset of a set of n *genes*, and a *genome* is a collection of k chromosomes.[1] A genome can be transformed by any of the following moves (for S, T, U, and V non-empty chromosomes): (1) a *fusion* $(S,T) \longrightarrow U$, where $U = S \cup T$; (2) a *fission* $S \longrightarrow (T,U)$, where $T \cup U = S$; or (3) a *translocation* $(S,T) \longrightarrow (U,V)$, where $U \cup V = S \cup T$. The *syntenic distance* $D(\mathcal{G}_1, \mathcal{G}_2)$ between genomes $\mathcal{G}_1$ and $\mathcal{G}_2$ is then given by the minimum number of moves required to transform $\mathcal{G}_1$ into $\mathcal{G}_2$. In computing this distance, we ignore any genes that appear in only one of the two genomes.

The *compact representation* of an instance of synteny is described by Ferretti et al. [7] and formalized by DasGupta et al. [5]. This representation makes the goal of each instance uniform and thus eases reasoning about move sequences. For an instance $\langle \mathcal{G}_1, \mathcal{G}_2 \rangle$ in which we are attempting to transform genome $\mathcal{G}_1$ into genome $\mathcal{G}_2$, we relabel each gene a contained in a chromosome of $\mathcal{G}_1$ by the indices 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 $\bigcup_{\ell \in S} \{i \mid \ell \in G_i\}$ (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 [7], consider the instance

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

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 $\mathcal{S}$ in this compact notation, let $D(\mathcal{S}) := D(\mathcal{S}, \{\{1\}, \ldots, \{n\}\})$ be the minimum number of moves required to solve $\mathcal{S}$. For an instance $\mathcal{S}$ in compact notation derived from a pair of genomes $\langle \mathcal{G}_1, \mathcal{G}_2 \rangle$, the distance $D(\mathcal{G}_1, \mathcal{G}_2) = D(\mathcal{S})$ [5, 7]. We will write $\mathcal{S}(n,k)$ to denote a instance $\mathcal{S} = S_1, \ldots, S_k$ with n elements (i.e., $\bigcup_i S_i = \{1, \ldots, n\}$) and k sets. Throughout the rest of this paper, we will assume

[1]The only "real" genomes that we consider consist of *disjoint* chromosomes, but for economy of notation we allow non-disjoint chromosomes in the definition. (The compact representation requires non-disjointness.)

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

Priority 1: Any ℓ for which $\mathsf{count}(\ell) = 1$.

Priority 2: Any ℓ for which $\mathsf{count}(\ell) = 2$.

Priority 3: If all $\mathsf{count}(\ell) > 2$, pick ℓ which minimizes $\mathsf{count}(\ell)$ and, if there are several such, which minimizes $\mathsf{count}(\ell')$ 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, $\sum_\ell \mathsf{count}(\ell)$ is minimized.

For the ℓ selected above, do one of the following operations:

Operation 1: If $\mathsf{count}(\ell) = 1$ 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 $\mathsf{count}(\ell) = 1$ and all genes ℓ' syntenic with ℓ appear in $\mathsf{count}(\ell') \geq c_{\min} > 1$ 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 $\mathsf{count}(\ell') = c_{\min}$, and, if there are several, with a maximal number of genes syntenic with ℓ.

Operation 3: If $\mathsf{count}(\ell) > 1$, effect $\mathsf{count}(\ell) - 2$ fusions followed by one translocation (possibly a fusion if no other genes are present in the component), again to produce a separate $\{\ell\}$.

Figure 1: The approximation algorithm $\mathcal{F}$ [7].

that all instances of the synteny problem are given in the compact representation unless otherwise indicated.

We will occasionally make use of the *characteristic matrix* representation of an instance of synteny: an instance $\mathcal{S}(n, k) = S_1, S_2, \ldots, S_k$ is stored as a k-by-n matrix M, where $M_{i,j} = 1$ iff $j \in S_i$.

The *dual* of a synteny instance $\mathcal{S}(n, k) = S_1, S_2, \ldots, S_k$ is the synteny instance $\mathsf{Dual}(\mathcal{S}(k, n)) = S'_1, S'_2, \ldots, S'_n$, where $j \in S'_i$ iff $i \in S_j$. DasGupta et al. [5] prove a duality property: $D(\mathcal{S}(n, k)) = D(\mathsf{Dual}(\mathcal{S}(k, n)))$.

If $S_1 \cap S_2 \neq \emptyset$, we will say that the chromosomes S_1 and S_2 are *connected*, and that both are in the same *(connected) component*.

For a gene ℓ, let $\mathsf{count}(\ell)$ denote the number of chromosomes in which ℓ appears.

Ferretti et al. [7] present the 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 by DasGupta et al. [5]: 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. Thus in an instance with p components, $\mathcal{H}$ requires $n + k - 2p$ moves. 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 $\{1\}, \{1,2\}, \ldots, \{1,2,\ldots,n\}$). To derive the performance guarantee for $\mathcal{H}$, DasGupta et al. prove the following *component bound*: if an instance of synteny $\mathcal{S}(n,k)$ has p components, then $D(\mathcal{S}(n,k)) \geq n - p$.

3 An Analysis of $\mathcal{F}$

In this section, we give an analysis of the heuristic $\mathcal{F}$. We first show that $\mathcal{F}$ is *never* worse than $\mathcal{H}$, and is therefore a 2-approximation. We then show that the factor of 2 is tight by giving a class of instances in which $\mathcal{F}$ performs a factor of 2 away from optimal. (In Section 4, we give a modification of $\mathcal{F}$ that handles this class of instances optimally, and analyze the modified algorithm.)

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

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

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

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.

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 of 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, after rearrangement, 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. □

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 now show the corresponding lower bound for $\mathcal{F}$:

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

Proof. Select any n such that $1/(n-1) \leq \varepsilon$. 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 $\{n\}$. Here is an $n-1$ move sequence solving the instance:

[$n-1$ moves] For $i = 1$ to $n-1$, translocate the ith singleton $\{n\}$ with the remaining elements of the large set to produce the singleton $\{i\}$.

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. This is optimal by the component bound.

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.

[$n-2$ moves] For $i = 1$ to $n-2$, $\mathsf{count}(i) = 1$. The gene $n-1$ is syntenic with i and appears in no other chromosome, so by Operation (1) we fission off the singleton $\{i\}$. This leaves $\{n-1,n\}$ and $n-1$ copies of $\{n\}$.

[1 move] $\mathsf{count}(n-1) = 1$, so select it. By Operation (2), translocate $\{n-1,n\}$ and $\{n\}$ to produce $\{n-1\}$ and $\{n\}$. This leaves $n-1$ copies of $\{n\}$.

[$n-2$ moves] Fuse the $n-1$ copies of $\{n\}$ by Operation (3).

Thus $\mathcal{F}$ requires $n-2$ fissions, 1 translocation, and $n-2$ fusions, or $2n-3$ moves total. □

$\mathcal{F}$ therefore has the same asymptotic behavior as $\mathcal{H}$: it is always within a factor of 2 of optimal, and there are instances in which it performs $2-\varepsilon$ away from optimal.

4 A Possible Improvement to $\mathcal{F}$

Note that the non-optimality of $\mathcal{F}$ on the instance in Lemma 3.3 is only as the result of applications of Operation (1). When the applications of this operation have been completed, the resulting genome is $\{n-1,n\}$ and $n-1$ copies of $\{n\}$. $\mathcal{F}$ takes $n-1$ more moves after this point, which is optimal by the component bound. In other words, 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 applications 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 following 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' syntenic with ℓ such that $\mathsf{count}(\ell') = 1$.

Note that $\mathcal{F}'$ performs optimally on the bad instance for $\mathcal{F}$ in Lemma 3.3: the genes are still selected in the same order, but each of the first $n-1$ fissions becomes a translocation, and the instance is solved after these translocations.

Theorem 4.1 *For any instance $\mathcal{S}(n,k)$ of synteny, $|\mathcal{F}'(\mathcal{S}(n,k))| \leq |\mathcal{H}(\mathcal{S}(n,k))|$.*

Proof. Analogous to the proof of Theorem 3.1. □

Corollary 4.2 *$\mathcal{F}'$ is a 2-approximation.* □

The following lemma shows, however, that $\mathcal{F}'$ is also no better than a 2-approximation.

Lemma 4.3 *For any $\varepsilon > 0$, there exists an instance $\mathcal{S}(n,k)$ with $|\mathcal{F}'(\mathcal{S}(n,k))| \geq (2-\varepsilon)\cdot D(\mathcal{S}(n,k))$.*

Proof. We give an instance $\mathcal{S}(\alpha i+1, \alpha i+i+1)$ with $D(\mathcal{S}(\alpha i+1, \alpha i+i+1)) = \alpha i + i$ and $|\mathcal{F}'(\mathcal{S}(\alpha i+1, \alpha i+i+1))| = 2\alpha i - 1$ for $3 \leq \alpha < i$. Selecting α and i so that $\varepsilon \geq (2i+1)/(\alpha i + i)$ then gives us an instance in which $\mathcal{F}'$ performs $(2-\varepsilon)$ away from optimal.

Consider the instance $\mathcal{S}(\alpha i+1, \alpha i+i+1)$ consisting of the following sets, for $3 \leq \alpha < i$:

- $S = \{1, 2, \ldots, \alpha i + 1\}$
- i copies of $Z = \{i+1, i+2, \ldots, \alpha i + 1\}$
- $X_{j,q} = \{j\}$, for $1 \leq j \leq i$ and $1 \leq q \leq \alpha$.

See Figure 2 for a schematic view of the characteristic matrix of this instance.

Here is a move sequence solving $\mathcal{S}(\alpha i+1, \alpha i+i+1)$ in $\alpha i + i$ moves:

[$i-1$ moves]	Fuse the i copies of Z, leaving S, the $X_{j,q}$s, and a single copy of Z.
[1 move]	Translocate S and Z to produce $\{1, 2, \ldots, \alpha i\}$ and $\{\alpha i + 1\}$.
[$(\alpha - 1)i$ moves]	Translocate $(\alpha - 1)$ of the singletons for each of the genes $1, 2, \ldots, i$ with the set $\{1, 2, \ldots, \alpha i\}$ to produce the singletons $\{i+1\}, \{i+2\}, \ldots, \{\alpha i\}$. This leaves $\{1, 2, \ldots, i\}$ and $\{1\}, \{2\}, \ldots, \{i\}$.
[i moves]	For $j = 1$ to $i-1$, translocate $\{j\}$ with the large set to remove j from it. This leaves two copies of $\{i\}$. Fuse these to solve the instance.

This is optimal by duality and the component bound.

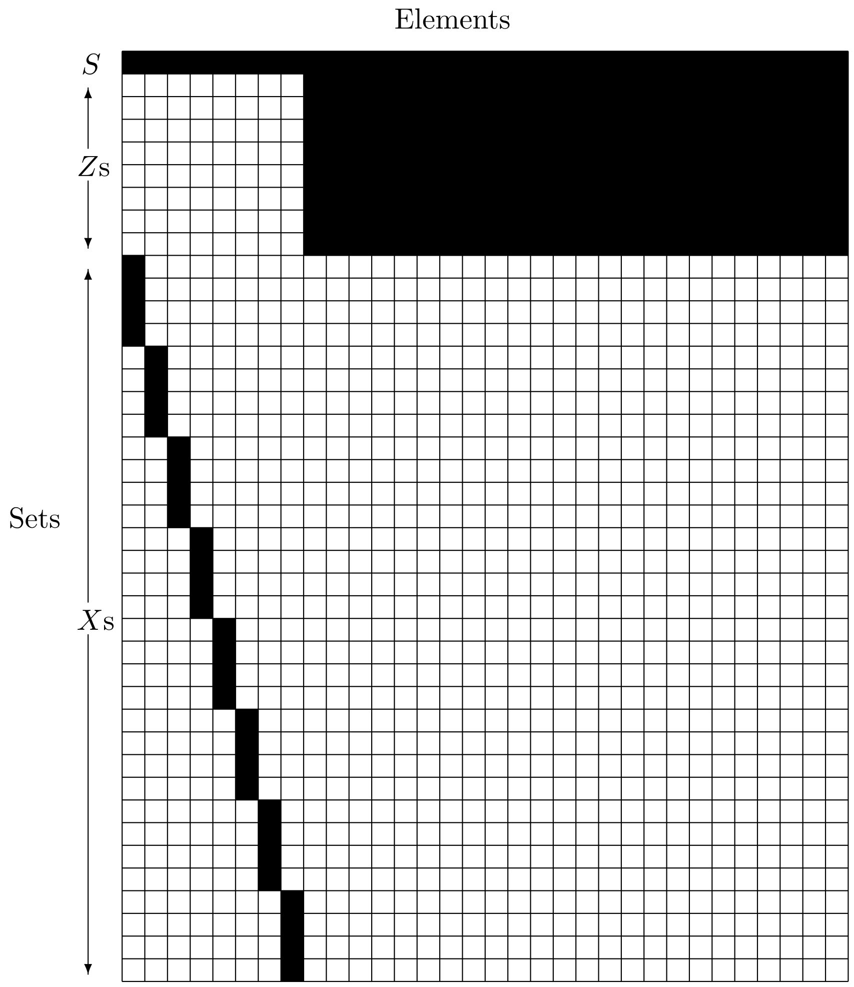

Figure 2: A bad example for $\mathcal{F}$ and $\mathcal{F}'$. Fusing the Zs together allows every succeeding move to be a translocation; greedily working on the singletons produces a move sequence nearly twice as long.

We now examine what $\mathcal{F}'$ does on this instance. Notice that

$$\mathsf{count}(\ell) = \begin{cases} \alpha + 1 & \text{for } \ell \in \{1, 2, \ldots, i\} \\ i + 1 & \text{for } \ell \in \{i+1, i+2, \ldots, \alpha i + 1\}. \end{cases}$$

Since $1, 2, \ldots, i$ are completely symmetric in this instance, without loss of generality we assume that the algorithm picks them in ascending order. Similarly, $i+1, i+2, \ldots, \alpha i + 1$ are symmetric, and we assume without loss of generality that they are selected in ascending order, as well.

[α moves]	$\alpha < i$, so we first select $\ell = 1$. Applying Operation (3), we fuse all singletons $\{1\}$ and then translocate with S to produce $\{1\}$ and $\{2, 3, \ldots, \alpha i + 1\}$.
[α moves]	Select $\ell = 2$. Apply Operation (3) as above to produce $\{2\}$ and $\{3, 4 \ldots, \alpha i + 1\}$.
$\vdots$	$\vdots$
[α moves]	Select $\ell = i$. Apply Operation (3) as above to produce $\{i\}$ and $\{i+1, i+2, \ldots, \alpha i + 1\} = Z$. The only remaining sets are $i + 1$ copies of the set Z.
[i moves]	Select $\ell = i + 1$. Apply Operation (3) to fuse i copies of Z, and then translocate the last two copies to produce $\{i + 1\}$ and $\{i+2, i+3, \ldots, \alpha i + 1\}$.
[$(\alpha-1)i-1$ moves]	Fission the remaining set $\{i+2, i+3, \ldots, \alpha i + 1\}$ into singletons, by Operation (1).

$\mathcal{F}'$ thus has to complete $\alpha i + i + (\alpha - 1)i - 1 = 2\alpha i - 1$ moves to solve this instance. □

Note that $\mathcal{F}$ does poorly on these instances, as well — bad choices of the genes by Priority (C) are sufficient to cause the non-optimality, and $\mathcal{F}$ selects genes in the same way as $\mathcal{F}'$.

5 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 a factor of two away from optimal.

Theorem 5.1 *For any algorithm $\mathcal{A}$ that works only within connected components, and for any $\varepsilon > 0$, there exists an instance $\mathcal{S}(n, k)$ where $|\mathcal{A}(\mathcal{S}(n, k))| \geq (2 - \varepsilon) \cdot D(\mathcal{S}(n, k))$.*

Proof. We construct an instance of synteny $\mathcal{S}(n, n)$ solvable in $n - 1$ moves, but for which $\mathcal{A}$ will require $2n - 4$ moves. Selecting n so that $\varepsilon \geq 2/(n-1)$ yields an instance where $\mathcal{A}$ is $2 - \varepsilon$ away from optimal.

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 $n-1$ moves:

[1 move]	Translocate $\{1, 2, \ldots, n-1\}$ and $\{n\}$ to produce $\{1\}$ and $\{2, 3, \ldots, n-1, n\}$.
[$n-2$ moves]	For $i = 2$ through $n-1$, perform a translocation of the set $\{i, i+1, \ldots, n\}$ and $\{n\}$ to produce $\{i\}$ and $\{i+1, i+2, \ldots, n\}$.

For any algorithm $\mathcal{A}$ working only within components, however, the moves that $\mathcal{A}$ can make are severely limited. Since $\{1, 2, \ldots, n-1\}$ is a component all by itself, there is no choice but to complete $n-2$ fissions. The $n-1$ copies of $\{n\}$ also form an entire component by themselves. 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. □

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 a single component. (This is equivalent to limiting consideration to instances with a single connected component — an optimal move sequence for an instance with multiple components is just the concatenation of optimal move sequences solving each of its components.)

Obviously, the optimal move sequence that works within components is no shorter than the optimal unconstrained move sequence. Because $\mathcal{H}$, $\mathcal{F}$, and $\mathcal{F}'$ all generate move sequences that work within components, these algorithms are also 2-approximations for the connected synteny problem. In each of the examples in which these algorithms are $2 - \varepsilon$ away from optimal, the optimal move sequence works only within components. (In fact, there is only one component in each example.) Thus $\mathcal{H}$, $\mathcal{F}$, and $\mathcal{F}'$ are all 2-approximations for this problem, and no better. Whether it is easier to approximate connected synteny, however, remains an open question.

6 Non-Redundancy and Monotonicity

In this section, we show that, for any instance, there is an optimal move sequence containing no moves that produce 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 $S_1, \ldots, S_k$ is a collection of sets and, for some i, $S_i = \emptyset$, we understand the synteny instance $\mathcal{S}(n, k) = S_1, \ldots, S_k$ to represent the synteny instance $\mathcal{T}(n, k-1) = S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_k$.

Lemma 6.1 *If there is a move sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ solving $S_1, \ldots, S_i \cup \{a\}, \ldots, S_k$ where $a \notin S_i$ (with S_i possibly empty), then there is a move sequence σ' solving $S_1, \ldots, S_i, \ldots, S_k$ in at most m steps.*

Proof. We proceed by induction on m.

For the base case ($m = 1$), 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 $S_1, \ldots, S_k$ 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 $S_1, \ldots, S_k$.

- 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 $S_1, \ldots, S_k$.

  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. (Since a does not appear elsewhere in the genome, a would have to be coupled with some element $b \neq a$.)

  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 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.

For the 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 $\sigma_{2\ldots m}$ 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 $S_r(1 \leq r \leq k, r \neq i), S_{i_1}, S_{i_2}$ 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.

- 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:

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

  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 $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2} \cup \{a\}$ in $m-1$ steps. By the inductive hypothesis, there is a move sequence σ' solving $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2}$ in at most $m-1$ steps. Gluing this together with σ_1' yields a sequence solving $S_1, \ldots, S_k$ in 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 $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_{\ell_1} \cup S_{i_1} \cup \{a\}, S_{\ell_2} \cup S_{i_2} \cup \{a\}$ in $m-1$ moves. By the inductive hypothesis applied to $\sigma_{2\ldots m}$ and $S_{\ell_1} \cup S_{i_1} \cup \{a\}$, we have a move sequence σ' solving $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2} \cup \{a\}$ 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 $S_r(1 \leq r \leq k, r \neq \ell, r \neq i), S_{\ell_1} \cup S_{i_1}, S_{\ell_2} \cup S_{i_2}$ in 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 $S \cap T \neq \emptyset$. (Note that only fissions and translocations can be redundant, because fusions do not create two sets.)

We need the following result on reordering from [5] to prove a theorem on redundancy: 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 $m' \leq m$ moves in which every fusion precedes every translocation precedes every fission, using $m'_1 \leq m_1$ fusions, $m'_2 \leq m_2$ translocations, and $m'_3 \leq m_3$ fissions. (DasGupta et al. actually state 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 6.2 *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 in Lemma 6.1 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. □

The canonicalizing process 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 consecutively applying canonicalization, redundancy elimination, and canonicalization again.

Theorem 6.3 (Monotonicity) *Let $S_1, \ldots, S_k$ and $T_1, \ldots, T_k$ be two collections of sets where, for all $1 \leq i \leq k$, $T_i \subseteq S_i$. Let $n = |\bigcup_i S_i|$ and $n' = |\bigcup_i T_i|$. Let $\mathcal{S}(n,k) = S_1, \ldots, S_k$ and let $\mathcal{T}(n',k) = T_1, \ldots, T_k$. Then $D(\mathcal{S}(n,k)) \geq D(\mathcal{T}(n',k))$.*

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

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

Inductive case ($\delta \geq 1$). Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ 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 6.1, we can convert σ into σ' solving $S_1, S_2, \ldots, S_j - \{a\}, \ldots, S_k$ in at most m steps. This instance is one element "closer" to $\mathcal{T}(n',k)$, so, by the inductive hypothesis, we can solve $\mathcal{T}(n',k)$ in at most m steps. □

7 A Lower Bound on Synteny

In this section, we give a lower bound on syntenic distance when many elements appear in many sets. The intuition for the bound is the following: consider an instance such that many elements appear in many sets in the compact representation. This occurs exactly when, in the non-compact representation, for many chromosomes C in genome $\mathcal{G}_1$, genes from C appear in many of the chromosomes in genome $\mathcal{G}_2$. This can only occur if many evolutionary events "scattered" C from $\mathcal{G}_1$ to $\mathcal{G}_2$. If this occurs for many chromosomes C, then many events must have occurred for many chromosomes, and thus the distance between the genomes must be large.

To formalize and prove this lower bound, we will make use of the following restricted form of the synteny problem, defined by DasGupta et al. [5]. Define the *linear synteny* problem as the synteny problem in which all move sequences are constrained as follows:

- The first $k-1$ moves must be fusions or severely restricted translocations. One of the input sets is initially 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' as output. If there is some element a that 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 each of the next $|\Delta|-1$ moves simply fissions off a singleton $\{a\}$ and produces the new merging set $\Delta' = \Delta - \{a\}$.

Let $\widetilde{D}(\mathcal{S}(n,k))$ be the length of the optimal linear move sequence. Notice that, unlike $\mathcal{F}$ and $\mathcal{F}'$, $\mathcal{H}$ does produce linear move sequences. Note that if a linear move sequence performs m_1 fusions in the first $k-1$ moves, then the move sequence contains $k-m_1-1$ translocations. After the $k-1$ fusions and translocations are complete, there are $n-k+m_1+1$ elements left in the merging set, since exactly one element is eliminated by each translocation. Therefore, $n-k+m_1$ fissions must be performed to eliminate the remaining elements. Thus the length of the linear move sequence is $n+m_1-1$ moves. (Every move either is a fusion or removes one element, and all but the last element must be removed.)

Theorem 7.1 *For any instance of synteny $\mathcal{S}(n,k)$,*

$$\widetilde{D}(\mathcal{S}(n,k)) \quad \geq \quad n-1+\max_{1\leq c\leq k-1}\left[c-\Big|\{\ell \mid \mathsf{count}(\ell)\leq c+1\}\Big|\right].$$

Proof. Consider an arbitrary c between 1 and $k-1$, and consider any linear move sequence solving $\mathcal{S}(n,k)$. In the first c moves, only genes ℓ such that $\mathsf{count}(\ell) \leq c+1$ can be eliminated. (Any ℓ with $\mathsf{count}(\ell) > c+1$ remains present in at least one unused input set, since the first c moves can only merge $c+1$ sets.)

Thus, in the first c moves we have at most $\Big|\{\ell \mid \mathsf{count}(\ell)\leq c+1\}\Big|$ translocations, and therefore at least $c-\Big|\{\ell \mid \mathsf{count}(\ell)\leq c+1\}\Big|$ fusions. Thus the instance requires at least $n-1+c-\Big|\{\ell \mid \mathsf{count}(\ell)\leq c+1\}\Big|$ moves to solve. □

DasGupta et al. prove a very useful fact relating linear synteny to the unconstrained synteny problem: for any instance $\mathcal{S}(n,k)$ of synteny, $\widetilde{D}(\mathcal{S}(n,k)) \leq D(\mathcal{S}(n,k)) + \log_{4/3}(D(\mathcal{S}(n,k)))$. This gives us the following bound on the general synteny problem:

Corollary 7.2 *For any instance of synteny $\mathcal{S}(n,k)$,*

$$D(\mathcal{S}(n,k)) + \log_{4/3}(D(\mathcal{S}(n,k))) \quad \geq \quad n - 1 + \max_{1 \leq c \leq k-1} \left[c - \left| \{\ell \mid \mathsf{count}(\ell) \leq c+1\} \right| \right].$$

□

We refer to this lower bound as the *size bound* on synteny. This bound may help in the development of improved approximation algorithms for the (linear) synteny problem. In particular, for a significant class of instances, $\mathcal{H}$ is better than a factor of 2 away from the optimal linear solution:

Corollary 7.3 *For any instance $\mathcal{S}(n,k)$ of synteny in which $n \geq k$, if there exists some c such that $c - \left|\{\ell \mid \mathsf{count}(\ell) \leq c+1\}\right| \geq \beta n + 1$ for $0 \leq \beta < 1$, then*

$$\frac{|\mathcal{H}(\mathcal{S}(n,k))|}{\widetilde{D}(\mathcal{S}(n,k))} \quad < \quad \frac{2}{1+\beta}.$$

Proof. Suppose that $\mathcal{S}(n,k)$ has p components. Then

$$\frac{|\mathcal{H}(\mathcal{S}(n,k))|}{\widetilde{D}(\mathcal{S}(n,k))} \;\leq\; \frac{n+k-2p}{n-1+\beta n+1} \;\leq\; \frac{2n-2p}{(1+\beta)n} \;<\; \frac{2n}{(1+\beta)n} \;=\; \frac{2}{1+\beta}.$$

□

8 Syntenic Diameter

In this section, we consider the difficulty of the hardest instance of a given size. This corresponds to an important question about the syntenic distance model: how different can two genomes possibly be? This gives a more meaningful interpretation to distances between species, since they can be compared to this maximum distance.

Formally, let the *syntenic diameter* and *linear syntenic diameter of n-chromosome species*, respectively, be

$$\mathsf{diameter}(n) := \max_{\mathcal{S}(n,n)} D(\mathcal{S}(n,n)) \qquad\qquad \mathsf{diameter}_{linear}(n) := \max_{\mathcal{S}(n,n)} \widetilde{D}(\mathcal{S}(n,n)),$$

i.e., the length of the longest optimal move sequence over all instances of up to n elements and n sets. We also define the *complete instance* $\mathcal{K}(n,n)$ of synteny, which consists of n copies of the set $\{1,2,\ldots,n\}$.

Proposition 8.1 $\mathsf{diameter}(n) = D(\mathcal{K}(n,n))$.

Proposition 8.2 $\mathsf{diameter}_{linear}(n) = \widetilde{D}(\mathcal{K}(n,n))$.

Proof. Immediate from monotonicity and linear monotonicity [10]. □

We now turn to the difficulty of solving $\mathcal{K}(n,n)$:

Lemma 8.3 $\widetilde{D}(\mathcal{K}(n,n)) = 2n-3$.

Proof. All genes appear n times, so

$$n - 2 - \left|\{\ell \mid \mathsf{count}(\ell) \leq n-1\}\right| = n - 2.$$

By Theorem 7.1, then, $\widetilde{D}(\mathcal{K}(n,n)) \geq 2n-3$. We easily have that $\widetilde{D}(\mathcal{K}(n,n)) \leq 2n-3$: complete any $n-2$ fusions to leave two copies of $\{1,2,\ldots,n\}$, complete one translocation to eliminate n, say, leaving $\{1,2,\ldots,n-1\}$. The instance is then solved by $n-2$ fissions. This is a linear move sequence of length $2n-3$ solving $\mathcal{K}(n,n)$. □

Corollary 8.4 $\mathsf{diameter}_{linear}(n) = 2n-3$. □

In a previous version of this paper, we conjectured that a linear move sequence of the form described in the proof of Lemma 8.3 was optimal in the general case as well. However, Cormode and Paterson have pointed out a four move sequence to solve $\mathcal{K}(4,4)$. This move sequence was also independently given by Christie [3, p. 15] previously. The same idea as in this sequence yields a sequence of length $2n-4$ solving $\mathcal{K}(n,n)$ for any n, by more efficiently "turning the corner" from fusions to fissions. We can fuse down to 4 sets and then complete four translocations to emit 3 elements. Then we need only $n-4$ fissions to solve the instance, or $2n-4$ moves total [4].

In recent work, we have shown that this move sequence is indeed optimal [9]. (We have recently learned that Pisanti and Sagot [12] have independently proved the same result.) This improves the lower bound of $2n-3-\log_{4/3}(2n-3)$ proved in a previous version of this paper.

9 An Implementation

We have implemented all of the heuristics discussed above in Matlab.[2] The full implementation is approximately 600 lines of code. We represent an instance by its characteristic matrix, which allows us to use Matlab's built-in matrix manipulation operations extensively in our code. In addition to using this implementation to verify the bad examples contained in the proofs of Lemmas 3.3 and 4.3 and Theorem 5.1, we have also performed tests of the algorithms and lower bounds on real and simulated data.

Let $\mathcal{R}(n,n)$ be a random instance where every element ℓ is chosen to be in every set S_i independently with probability 1/2. We have run the three algorithms and computed the two lower bounds on such instances. The results are shown in Figure 3. Repeated runs of this experiment yield extremely similar results. The results for $\mathcal{F}'$ are not shown, because they were almost always identical to the results of $\mathcal{F}$. Note that:

- When n reaches approximately 40 (a large but not unreasonable number of chromosomes), the size bound becomes more informative than the component bound.
- While this is far from a performance guarantee, it is worth observing that the ratio of the performance of $\mathcal{F}$ and the larger lower bound is consistently below 2 throughout this range. (Although this number fluctuates somewhat, it never exceeds 1.75.)

[2]The code is available from the author.

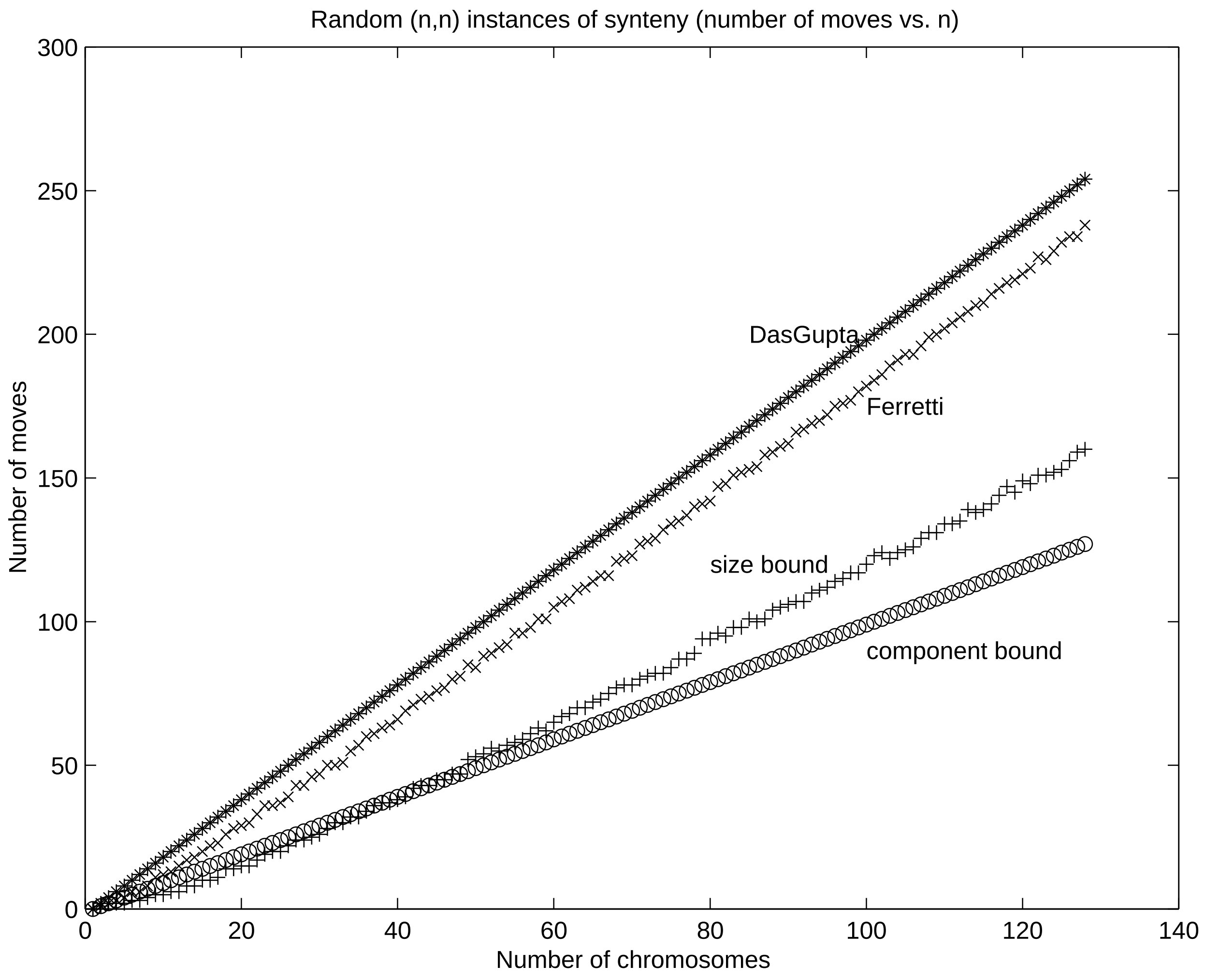

Figure 3: Results on instances $\mathcal{R}(n,n)$.

	Algorithms					Bounds			
Species	$\mathcal{H}$	$\mathcal{F}$	dual	$\mathcal{F}'$	dual	comp	dual	size	dual
Felis catus (cat)	14	11	11	11	11	3	11	10	5
Drosophila melanogaster (fruit fly)	60	43	43	41	41	41	19	11	30
Homo sapiens (human)	47	28	27	28	27	20	27	17	11
Mus musculus (mouse)	49	31	32	31	32	21	28	18	19
Sus scrofa (pig)	30	19	19	19	19	11	19	11	6
Rattus norvegicus (rat)	47	28	28	28	28	19	28	18	12
Ovis aries (sheep)	33	18	18	18	18	15	18	13	10

Table 1: Number of moves for algorithms and bounds: comparisons to *Bos taurus* (cow).

In addition, we have run these algorithms on seven sets of real synteny data, found in the Institut National de la Recherche Agronomique (INRA) Comparative Homology Database. We have compared the genomes of seven common species to entries in the BOVMAP database, which contains known homologies with the cow, *Bos taurus*. The results are summarized in Table 1. We make the following observations based upon the results of these tests:

- In all cases, $\mathcal{F}'$ performed at least as well as $\mathcal{F}$; in the case of *D. melanogaster*, $\mathcal{F}'$ outperformed $\mathcal{F}$ by two moves.
- In all seven instances, the size bound introduced in this paper is less informative than the component bound of [5].
- In most cases (five out of seven), the component bound was actually attained by both $\mathcal{F}$ and $\mathcal{F}'$. In the sixth case, $\mathcal{F}'$ achieved the component bound and $\mathcal{F}$ was within two moves of it.

The last point may raise some question about the validity of the model (that it is too easy to solve too many instances, and thus that the model fails to be informative about relative distances among groups of species), or may indicate that there is simply insufficient synteny data presently available.

10 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?

The lower bound from Theorem 7.1 may be useful in improving the approximation ratio. Limiting consideration to connected synteny may also be fruitful.

An additional avenue of attack for this problem is further consideration of the algorithms that have already been proposed. The results of running $\mathcal{F}$ and $\mathcal{F}'$ on real synteny data (see Section 9) indicate that they very often produce optimal move sequences. This motivates a more formal exploration of when these algorithms do well:

- What are the characteristics of instances of synteny on which $\mathcal{F}$ and $\mathcal{F}'$ perform optimally?

A more general related goal is the design of algorithms optimally solving special cases of the syntenic distance problem. We have begun this investigation [10], but a great amount remains to be done.

11 Acknowledgements

I would like to thank Jon Kleinberg for extensive and fruitful discussions on numerous aspects of this paper and the syntenic distance problem. Thanks also to Anne Bergeron, Chris Jeuell, Manish Sambhu, Jeff Vinocur, and Wes Weimer for comments on previous versions of this paper.

This work was supported in part by a Churchill Scholarship from the Winston Churchill Foundation, an NSF Graduate Research Fellowship, and the ONR Young Investigator Award of Jon Kleinberg. The majority of this work was performed at Cornell University and the University of Cambridge.

References

[1] Vineet Bafna and Pavel A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, April 1996. A previous version appeared in FOCS'93.

[2] Vineet Bafna and Pavel A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, 1998. A previous version appeared in SODA'95.

[3] David Christie. *Genome Rearrangement Problems*. PhD thesis, University of Glasgow, August 1998.

[4] Graham Cormode and Mike Paterson. Personal Communication, July 1999.

[5] Bhaskar DasGupta, Tao Jiang, Sampath Kannan, Ming Li, and Elizabeth Sweedyk. On the complexity and approximation of syntenic distance. *Discrete Applied Mathematics*, 88(1–3):59–82, November 1998. A previous version appeared in RECOMB'97.

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

[7] Vincent Ferretti, Joseph H. Nadeau, and David Sankoff. Original synteny. In Dan Hirschberg and Gene Myers, editors, *Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, pages 159–167, 1996.

[8] Institut National de la Recherche Agronomique. Comparative Homology Database, 1999. `http://locus.jouy.inra.fr/cgi-bin/lgbc/mapping/common/taxonomy.pl`.

[9] Jon Kleinberg and David Liben-Nowell. The syntenic diameter of the space of N-chromosome genomes. In David Sankoff and Joseph H. Nadeau, editors, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families*. Kluwer Academic Press, 2000.

[10] David Liben-Nowell and Jon Kleinberg. Structural properties and tractability results for linear synteny. In Raffaele Giancarlo and David Sankoff, editors, *11th Annual Symposium on Combinatorial Pattern Matching*, pages 248–263, 2000.

[11] Pavel A. Pevzner and Michael S. 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.

[12] Nadia Pisanti and Marie-France Sagot. Further thoughts on the syntenic distance between genomes. *Algorithmica*, 34(2):157–180, 2002.

[13] David Sankoff and Joseph H. Nadeau. Conserved synteny as a measure of genomic distance. *Discrete Applied Mathematics*, 71(1–3):247–257, December 1996.