On the Complexity and Approximation of Syntenic Distance

Bhaskar DasGupta* Tao Jiang† Sampath Kannan‡ Ming Li§
Elizabeth Sweedyk¶

April 8, 1998

Abstract

The paper studies the computational complexity and approximation algorithms for a new evolutionary distance between multi-chromosomal genomes introduced recently by Ferretti, Nadeau and Sankoff. Here, a chromosome is represented as a set of genes and a genome is a collections of chromosomes. The syntenic distance between two genomes is defined as the minimum number of translocations, fusions and fissions required to transform one genome into the other. We prove that computing the syntenic distance is NP-hard and give a simple approximation algorithm with performance ratio 2. For the case when an upper bound d on the syntenic distance is known, we show that an an optimal syntenic sequence can be found in $O(nk + 2^{O(d^2)})$ time, where n and k are the number of chromosomes in the two given genomes. Next, we show that if the set of operations for transforming a genome is significantly restricted, we can nevertheless find a solution that performs at most $O(\log d)$ additional moves, where d is the number of moves performed by the unrestricted optimum. This result should help in the design of approximation algorithms. Finally, we investigate the median problem: Given three genomes, construct a genome minimizing the total syntenic distance to the three given genomes and compute the corresponding median distance. The problem has application in the inference of phylogenies based on the syntenic distance. We prove that the problem is NP-hard and design a polynomial time approximation algorithm with a performance ratio of $4 + \epsilon$ for any constant $\epsilon > 0$.

1 Introduction

The definition and study of appropriate measures of distance between pairs of species is of great importance in computational biology. Such measures of distance can be used, for example, in

*Department of Computer Science, Rutgers University, Camden, NJ 08102, USA. E-mail:bhaskar@crab.rutgers.edu. Work done while the author was at University of Waterloo and supported by a CGAT (Canadian Genome Analysis and Technology) grant.

†Department of Computer Science, McMaster University, Hamilton, Ontario L8S 4K1, Canada. E-mail: jiang@maccs.mcmaster.ca. Supported in part by NSERC Operating Grant OGP0046613, CGAT and a JSPS fellowship. Work done while visiting at University of Washington and Gunma University.

‡Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA. E-mail: kannan@cis.upenn.edu. Supported in part by NSF Grant CCR 9612829.

§Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Email: mli@math.uwaterloo.ca. Supported by the NSERC operating grant OGP0046506 and a CGAT grant.

¶Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA 19104, USA. E-mail: zzz@saul.cis.upenn.edu. Supported in part by

phylogeny construction and in taxonomic analysis.

As more and more molecular data becomes available methods for defining distances between species have focused on such data. One of the most popular distance measures is the edit distance between homologous DNA or aminoacid sequences obtained from different species. Such measures focus on point mutations and define the distance between two sequences as the minimum number of these moves required to transform one sequence into another. It has been recognized that the edit-distance may underestimate the distance between two sequences because of the possibility that multiple point mutations occurring at the same locus will be accounted for simply as one mutation. The problem is that the probability of a point mutation is not low enough to rule out this possibility.

Recently, there has been a spate of new definitions of distance that try to treat rarer, macrolevel mutations as the basic moves. For example, if we know the order of genes on a chromosome for two different species, we can define the *reversal* distance between the two species to be the number of reversals of portions of the chromosome to transform the gene order in one species to the gene order in the other species. The question of finding the reversal distance was first explored in the computer science context by Kececioglu and Sankoff and by Bafna and Pevzner and there has been significant progress made on this question by Bafna, Hannenhalli, Kececioglu, Pevzner, Ravi, Sankoff and others [2, 3, 8, 11, 12, 13]. Other moves besides reversals have been considered as well. Breaking off a portion of the chromosome and inserting it elsewhere in the chromosome is referred to as a *transposition* and one can similarly define the transposition distance[4]. Similarly allowing two chromosomes (viewed as strings of genes) to exchange suffixes (or sometimes a suffix with a prefix) is known as a *translocation* and this move can also be used to define an appropriate measure of distance between two species for which much of the genome has been mapped [10].

Ferretti et. al.[6] proposed a distance measure that is at an even higher level of abstraction. Here even the order of genes on a particular chromosome of a species is ignored/ presumed to be unknown. It is assumed that the genome of a species is given as a collection of sets. Each set in the collection corresponds to a set of genes that are on one chromosome and different sets in the collection correspond to different chromosomes. In this scenario one can define a move to be either an exchange of genes between two chromosomes, the fission of one chromosome into two, or the fusion of two chromosomes into one. The *syntenic distance* between two species has been defined by Ferretti et. al.[6] to be the minimum number of such moves required to transform the genome of one species into the genome of the other.

Notice that any recombination of two chromosomes is permissible in this model. By contrast, the set of legal translocations (in the translocation distance model) is severely limited by the order of genes on the chromosomes being translocated. Furthermore, the transformation of the first genome into the second genome does not have to produce a specified order of genes in the second genome. The underlying justification of this model is as follows. For many organisms, the information (physical map) which specifies the order of genes within chromosomes is not known, but the distribution of genes among chromosomes is known. Given this incomplete information that is available, it is still important to compute evolutionary trees based on genomic events, which leads to the study of the syntenic distance.

Ferretti et. al.[6] provide a heuristic that attempts to compute the syntenic distance and provide empirical evidence of the value of this distance measure. In this paper we attempt to put the notion of syntenic distance on more formal foundations. To wit, we show the following results.

- The syntenic distance is, in fact, a distance.

- An optimal sequence of moves can be assumed to occur in a canonical order with fusions preceding translocations, preceding fissions.
- The problem of computing the syntenic distance is NP-hard.
- There is an approximation algorithm that achieves a factor of 2 approximation to syntenic distance.
- Computing this distance is fixed parameter tractable.
- When the set of moves is significantly restricted, there is nevertheless an optimal sequence of restricted moves whose length is not much more than the length of the unrestricted optimal sequence.
- The problem of computing the median genome, for a given set of 3 genomes, is NP-hard and admits an approximation with ratio $4 + \epsilon$ for any constant $\epsilon > 0$.

These results will be described in the sections that follow. Let $A^{-1}(m, n)$ denote the *inverse of Ackerman's function* over the two integer variables $m, n \geq 0$ (e.g., see [5, page 452]). $A^{-1}(m, n)$ grows very slowly with m and n.

2 Notation and Preliminaries

For the purpose of this paper, a genome is a collection of k subsets (called synteny sets or chromosomes) of a set of n objects (called genes). A genome mutates by one of three simple *moves*; these are the *translocation, fusion,* and *fission.*

Definition 2.1 *Let S_1, S_2, T_1, T_2 be sets such that at most one is empty and such that $T_1 \cup T_2 = S_1 \cup S_2$.*

(a) *If S_1, S_2, T_1, T_2 are non-empty then $(S_1, S_2) \longrightarrow (T_1, T_2)$ is called a* translocation of S_1 and S_2.

(b) *If S_2 is empty then $S_1 \longrightarrow (T_1, T_2)$ is called a* fission of S_1.

(c) *If T_2 is empty then $(S_1, S_2) \longrightarrow T_1$ is called a* fusion of S_1 and S_2.

Given two genomes $\mathcal{G}_1$ and $\mathcal{G}_2$ over some gene set[1] Σ, the *syntenic distance from $\mathcal{G}_1$ to $\mathcal{G}_2$*, denoted $D(\mathcal{G}_1, \mathcal{G}_2)$, is the minimum number of moves needed to transform $\mathcal{G}_1$ into $\mathcal{G}_2$.

Proposition 2.1 $D(\mathcal{G}_1, \mathcal{G}_2) = D(\mathcal{G}_2, \mathcal{G}_1)$.

Proof: Given an optimal sequence of moves from $\mathcal{G}_1$ to $\mathcal{G}_2$, it is easy to reverse every move (the reverse of a fusion is a fission and vice versa) to get an optimal sequence of moves transforming $\mathcal{G}_2$ to $\mathcal{G}_1$. ■

It follows from Proposition 2.1 that $\mathcal{D}$ defines a metric over the set of genomes over Σ (reflexivity and triangle inequality of $\mathcal{D}$ are obvious).

[1] As explained in [6], if $\mathcal{G}_1$ and $\mathcal{G}_2$ are over two different gene alphabets Σ_1 and Σ_2, we remove all the genes in $\Sigma_1 \oplus \Sigma_2$ from both $\mathcal{G}_1$ and $\mathcal{G}_2$ while computing the syntenic distance, where $\oplus$ is the set symmetric difference operator.

Lemma 2.1 *Let $\mathcal{G}_1, \mathcal{G}_2$ be an instance of syteny. Then there is a sequence of moves $\sigma = (\sigma_1, \ldots, \sigma_m)$ such that $m = D(\mathcal{G}_1, \mathcal{G}_2)$ and every fission occurs after every translocation and fusion.*

Proof: Let $\sigma = (\sigma_1, \sigma_2, \ldots \sigma_m)$ be an optimal move sequence. If every fission occurs after every translocation or fusion we are done; so assume not. Let $i < m$ be the largest index such that σ_i is a fission preceding a translocation or fusion. We give a new optimal sequence $(\sigma_1, \ldots, \sigma_{i-1}, \sigma'_i, \sigma'_{i+1}, \sigma_{i+2}, \ldots, \sigma_m)$ where σ'_i is a translocation or fusion and σ'_{i+1} is a fission. Repeating the argument eventually yields the desired sequence.

Assume that σ_i is the fission $S_1 \cup S_2 \longrightarrow (S_1, S_2)$ and σ_{i+1} is either the fusion $(T_1, T_2) \longrightarrow T_1 \cup T_2$ or the translocation $(T_1, T_2) \longrightarrow (T'_1, T'_2)$. If neither T_1 nor T_2 is created by σ_i we can simply swap σ_i and σ_{i+1} to yield the desired sequence. Thus we need only consider the case where, without loss of generality, $S_1 = T_1$. Then we claim that σ_{i+1} is a translocation; otherwise we could replace σ_i and σ_{i+1} by $(S_1 \cup S_2, T_2) \longrightarrow (S_2, S_1 \cup T_2)$, reducing the number of move by 1, which contradicts the optimality of σ. Finally, since σ_{i+1} is a translocation, we can replace σ_i and σ_{i+1} by $\sigma'_i : (S_1 \cup S_2, T_2) \longrightarrow (T'_1 \cup S_2, T'_2)$ and $\sigma'_{i+1} : T'_1 \cup S_2 \longrightarrow (T'_1, S_2)$ to yield the desired sequence. ■

Note that the number of translocations, fusions and fissions is preserved in construction of the previous proof. Thus we get the following corollary.

Corollary 2.1 *Let $\mathcal{G}_1, \mathcal{G}_2$ be an instance of synteny. If there is an optimal move sequence with m_1 translocations, m_2 fusions, and m_3 fissions, then there is an optimal move sequence with m_1 translocations, m_2 fusions, and m_3 fissions in which all fissions come after all translocations and fusions.*

Lemma 2.2 *Let $\mathcal{G}_1, \mathcal{G}_2$ be an instance of synteny. Then there is a sequence of moves $\sigma = (\sigma_1, \ldots, \sigma_m)$ such that $m = D(\mathcal{G}_1, \mathcal{G}_2)$ and such that all fusions come before all translocations which come before all fissions.*

Proof: Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ be an optimal move sequence. If there are no translocations or fusions we are done. If not, by the Lemma 2.1 we may assume that all translocations and fusions occur before all fissions. Let i be the index of the last non-fission in the sequence and let $\mathcal{G}'$ be the collection of sets after the i-th move. Since σ is optimal $D(\mathcal{G}_1, \mathcal{G}') = i$. Using Proposition 2.1 and Corollary 2.1 (on the problem of transforming $\mathcal{G}'$ to $\mathcal{G}_1$) there is a move sequence that transforms $\mathcal{G}_1$ into $\mathcal{G}'$ consisting solely of fusions and translocations in which the fusions occur first. Concatenating this sequence with $\sigma_{i+1}, ..., \sigma_m$ yields the desired move sequence. ■

Remark 2.1 *The observation in Lemma 2.2, namely that there is always an optimal sequence of moves that consists of fusions followed by translocations followed by fissions, was also made (in the context of genomes where gene order is known) by Kececioglu and Ravi [10].*

2.1 The Compact Representation of Synteny

For our subsequent proofs it is easier to deal with the *compact representation* of the synteny problem as described in [6]. Assume that the genomes $\mathcal{G}_1$ and $\mathcal{G}_2$ contain n and k sets, respectively. Then, the compact representation of $\mathcal{G}_2$ with respect to $\mathcal{G}_1$ is defined as follows: replace the i^{th} set $G_{1,i}$ of $\mathcal{G}_1$ by the set $G'_{1,i} = \{i\}$ for $1 \leq i \leq n$, and for every set $G_{2,j}$ occuring in $\mathcal{G}_2$ for $1 \leq j \leq k$, replace $G_{2,j}$ by the set $G'_{2,j} = \cup_{x \in G_{2,j}} \{\ell \mid x \in G_{1,\ell}\}$. Let $\mathcal{G}'_1$ and $\mathcal{G}'_2$ be the two modified genomes.

Lemma 2.3 $D(\mathcal{G}_1, \mathcal{G}_2) = D(\mathcal{G}'_1, \mathcal{G}'_2)$

Proof: Let ℓ be an element of $G'_{2,j}$ which is a set in $\mathcal{G}'_2$. We can think of this element ℓ as the element (ℓ, j) to remember its origins. We associate the set $G_{1,\ell} \cap G_{2,j}$ with the element (ℓ, j). Now we can simulate a move sequence transforming $\mathcal{G}'_2$ to $\mathcal{G}'_1$ by a move sequence from $\mathcal{G}_2$ to $\mathcal{G}_1$ as follows: Maintain a 1–1 correspondence between the sets in $\mathcal{G}'_2$ and the sets in $\mathcal{G}_2$ at all times. Whenever there is a move involving an element (ℓ, j) simulate this move in the unprimed domain by making the set $G_{1,\ell} \cap G_{2,j}$ go wherever (ℓ, j) goes. Note that if the operation in the primed domain is a translocation or a fission, there might be some ambiguity if ℓ is present in both output sets. In this case, we will let $G_{1,\ell} \cap G_{2,j}$ go to both output sets as well. Since ultimately the move sequence in the primed domain must end with all the ℓ's being unified and isolated from the other elements, the simulation will produce the set $G_{1,\ell}$ in the unprimed domain.

Conversely, we can simulate a move sequence in the unprimed domain as follows: Again we maintain a 1–1 correspondence between the sets in $\mathcal{G}_2$ and the sets in $\mathcal{G}'_2$. Whenever an element $x \in G_{2,j}$ is acted on by some move, we let the element $\ell \in G'_{2,j}$ be acted on in the same manner, where ℓ is such that $x \in G_{2,j} \cap G_{1,\ell}$. The only issue is if there is a fission or a translocation and there are elements x and y in $G_{2,j}$ associated with the same ℓ which get separated by the move. In this case, we let ℓ be present in both output sets. Since x and y must ultimately be united, the two copies of ℓ will ultimately be merged.

This shows that $D(\mathcal{G}_1, \mathcal{G}_2) = D(\mathcal{G}'_1, \mathcal{G}'_2)$.

■

We state an obvious fact about the optimal move sequence. Without loss of generality we can assume that no element is present in both output sets of a translocation or fission in an optimal move sequence.

We can alternatively define the synteny problem using the compact representations of genomes as follows.

Definition 2.2 *Given a collection $\mathcal{S}(n,k)$ of k (not necessarily distinct) sets $S_1, \ldots, S_k$, $S_i \subseteq \{1, 2, \ldots, n\}$, the* synteny problem *is to compute the minimum number of mutations, denoted by $D(\mathcal{S}(n,k))$, to transform $\mathcal{S}$ to the collection $\{\{1\}, \{2\}, \ldots, \{n-1\}, \{n\}\}$.*

The *dual* of $\mathcal{S}(n,k) = (\{a_1, a_2, \ldots, a_n\}; S_1, \ldots, S_k)$ is $\mathcal{S}'(k,n) = S'_1, \ldots, S'_n$, where S_i is a subset of $\{1, \ldots, k\}$ and $j \in S'_i \Leftrightarrow i \in S_j$. The goal in the dual problem is to produce the collection $\{1\}, \ldots, \{k\}$.

Proposition 2.2 follows from Proposition 2.1.

Proposition 2.2 *Let $\mathcal{S}'(k,n)$ be the dual of the synteny problem $\mathcal{S}(n,k)$. Then $D(\mathcal{S}(n,k)) = D(\mathcal{S}'(k,n))$.*

The following proposition gives a time bound for transforming a given instance of the synteny problem to its compact representation.

Proposition 2.3 *Assume that the genomes $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) contain n (resp. k) sets. Also, let t be the total number of elements in either $\mathcal{G}_1$ or $\mathcal{G}_2$ (i.e., $\mathcal{G}_1$ (or $\mathcal{G}_2$) contain (disjoint) subsets of $\{1, 2, \ldots, t\}$). Then, the compact representation of $\mathcal{G}_2$ with respect to $\mathcal{G}_1$ can be computed in $O((t + kn)A^{-1}(t, n))$ time.*

Proof: First, scan each set $G_{1,i}$ of $\mathcal{G}_1$, storing for each element $x \in G_{1,i}$ the index i of the set in $\mathcal{G}_1$ in which it appears. This takes $O(t)$ time. Then, do the following:

For every set $G_{2,j}$ of $\mathcal{G}_2$ do

Perform n MAKESET operations to create n singleton sets $\{1\}, \{2\}, \ldots, \{n\}$ corresponding to the n indices of the n sets in $\mathcal{G}_1$.

Union together the indices of the sets in $\mathcal{G}_1$ in which the elements of $G_{2,j}$ appear by doing $|G_{2,j}|$ SET-UNION operations (one needs to check first, before every SET-UNION, if the two sets on which SET-UNION has to be performed has already been subject to a previous SET-UNION, by doing two FIND-SET operations).

For every element in $G_{2,j}$, do a FIND-SET operation, to find and collect all the indices of sets of $\mathcal{G}_1$ which occur in the same set of $\mathcal{G}_2$, thereby constructing the set $G'_{2,j}$.

For each set $G_{2,j} \in \mathcal{G}_2$, we do at most $4|G_{2,j}| + n$ set operations on sets containing a total of n elements. Using both union by rank and path compression heuristics, the time taken for each set $G_{2,j}$ is $O((4|G_{2,j}| + n)A^{-1}(4|G_{2,j}| + n, n)) = O((|G_{2,j}| + n)A^{-1}(4t + n, n))$ (see [5, page 449]). Hence, the time taken for all the sets in $\mathcal{G}_2$ is at most $O(((\sum_{j=1}^{k} |G_{2,j}|) + kn)A^{-1}(4t + n, n)) = O((t + kn)A^{-1}(4t + n, n)) = O((t + kn)A^{-1}(t, n))$. ■

Henceforth, unless otherwise mentioned, by synteny problem we refer to the compact representation of the synteny problem. By Proposition 2.2, it is sufficient to consider an instance $\mathcal{S}(n, k)$ of the synteny problem with $n \geq k$, since otherwise we can solve the dual problem.

Definition 2.3 *Given an instance $\mathcal{S}(n, k)$ of the synteny problem, the* synteny graph $G(\mathcal{S}(n, k))$ *includes a vertex for each set of $\mathcal{S}(n, k)$. Two vertices are connected by an edge if and only if their corresponding sets in $\mathcal{S}(n, k)$ have a non-empty intersection. If $G(\mathcal{S}(n, k))$ has p connected components (where $1 \leq p \leq n$), we will simply say that $\mathcal{S}(n, k)$ has p components. If $G(\mathcal{S}(n, k))$ is connected we will say that $\mathcal{S}(n, k)$ is connected.*

Proposition 2.4 *Let $\mathcal{S}(n, k)$ be a synteny instance with p components. Then, $D(\mathcal{S}(n, k)) \geq n - p$.*

Proof: Let $\sigma = (\sigma_1, \ldots, \sigma_m)$ be an optimal move sequence for $\mathcal{S}(n, k)$. Let $\mathcal{S}_0 = \mathcal{S}$ and let $\mathcal{S}_i$ be the synteny instance obtained after the first i moves. Obviously, $\mathcal{S}_0$ has p components and $\mathcal{S}_m$ has n components. Now, we show that $\mathcal{S}_{i+1}$ has at most one more component than $\mathcal{S}_i$. This will follow provided we can show that any move can produce at most one more connected component and can be seen as follows:

Let the move be $(S_1, S_2) \longrightarrow (T_1, T_2)$ (where at most one of the sets is empty). This move removes S_1 and S_2 from the vertex set of the associated graph and introduces the vertices T_1 and T_2. (Of course, if any of these sets is empty, there is no corresponding vertex for that set.) If in the new graph we join vertex T_1 and T_2 by an edge, it is clear that the new graph has at most as many components as the old one. (Any path in the old graph passing through S_1 or S_2 can be mapped to a path in the new graph passing through T_1 and/or T_2.) Finally, removing the edge from T_1 to T_2 increases the number of components by at most 1.

Thus, $D(\mathcal{S}(n, k)) = m \geq n - p$. ■

3 NP-hardness of the Synteny Problem

In this section we prove the following theorem.

Theorem 3.1 *Computing the syntenic distance exactly is NP-hard.*

Our reduction will use two problems, the *largest balanced quasi-independent set* (LBQIS) problem and the *largest balanced independent set* (LBIS) problem for bipartite graphs, which are defined as follows:

PROBLEM: Largest balanced independent set (LBIS) problem.
INPUT: A connected bipartite graph $G = (U, V, E)$ with $|U| = |V| = n$ and positive integer k, $1 \leq k \leq n$.
QUESTION: Does there exists $U' \subseteq U$, $V' \subseteq V$, $|U'| = |V'| = k$, such that $(u', v') \notin E$ for any $u' \in U'$ and $v' \in V'$?

PROBLEM: Largest balanced quasi-independent set (LBQIS) problem.
INPUT: A connected bipartite graph $G = (U, V, E)$ with $|U| = |V| = n$ and positive integer k, $1 \leq k \leq n$.
QUESTION: Does there exists $U' \subseteq U$, $V' \subseteq V$, $|U'| = |V'| = k$, such that for some permutation $u'_1, u'_2, \ldots, u'_k$ of the vertices in U' and some permutation $v'_1, v'_2, \ldots, v'_k$ of the vertices in V', $(u'_i, v'_j) \notin E$ for any $1 \leq i \leq k$ and $i > j$?

The LBIS problem is known to be NP-complete[7, page 196][2]. Note that an LBIS of size k is also an LBQIS of size k for a graph G, but the converse is not necessarily true.

First, we prove the following theorem.

Theorem 3.2 *Computing the syntenic distance is NP-hard if the LBQIS problem is NP-hard.*

The proof of the above theorem is as follows. Given an instance (G, k) of the LBQIS problem as mentioned above, we create an instance $\mathcal{S}(2n-k+1, 2n-k+1)$ of the synteny problem containing the following sets (assume that $U = \{u_1, u_2, \ldots, u_n\}$ and $V = \{v_1, v_2, \ldots, v_n\}$):

(a) $S = \{u_1, u_2, \ldots, u_n, a_1, a_2, \ldots, a_{n-k}, b\}$.

(b) $X_i = \{u_j \mid (u_j, v_i) \in E\} \cup \{b\}$ for $1 \leq i \leq n$.

(c) $Y_i = \{b\}$ for $1 \leq i \leq n-k$.

We refer to the elements $u_1, u_2, \ldots, u_n$ (resp. $a_1, a_2, \ldots, a_{n-k}$) as the u-elements (resp. a-elements) and the sets $X_1, X_2, \ldots, X_n$ (resp. $Y_1, Y_2, \ldots, Y_{n-k}$) as the X-sets (resp. Y-sets). For two given sets $\{u'_1, u'_2, \ldots, u'_k\} \subseteq U$ and $\{v'_1, v'_2, \ldots, v'_k\} \subseteq V$, we define the following notations for convenience:

- $P_0 = S$ and $P_i = P_{i-1} - \{a_i\}$) for $1 \leq i \leq n-k$.

[2] In [7, page 196] the largest *balanced complete bipartite subgraph* problem is shown to be NP-complete, which is same as the largest balanced bipartite independent set on the complement of the graph

- $Q_k = P_{n-k}$ and $Q_{i-1} = Q_i - \{u'_i\}$ for $1 \leq i \leq k$.
- $R_0 = Q_0$ and $R_i = R_{i-1} - \{u'_{k+i}\}$ for $1 \leq i \leq n-k$.

Notice that $R_{n-k} = \{b\}$. The following lemma will complete the proof of Theorem 3.2.

Lemma 3.1 *G has a LBQIS of size k if and only if $D(S(2n-k+1, 2n-k+1)) = 2n-k$.*

The proof of the "only if" part of Lemma 3.1 is straightforward. Assume G has a LBQIS (U', V') of size k. Let $U - U' = \{u'_{k+1}, u'_{k+2}, \ldots, u'_n\}$ and $V - V' = \{v'_{k+1}, v'_{k+2}, \ldots, v'_n\}$. An optimal syntenic sequence of $2n-k$ moves consists of the following moves:

- First, for $i = k+1, k+2, \ldots, n$, perform the translocation $(P_{i-k-1}, X_{v'_{k+1}}) \longrightarrow (\{a_{i-k}\}, P_{i-k})$. Notice that after the last move we have created the set $Q_0 = P_{n-k}$.
- Next, for $i = k, k-1, \ldots, 1$, perform the translocation $(Q_i, X_{v'_i}) \longrightarrow (\{u'_i\}, Q_{i-1})$. Notice that after the last move we have the sets $Q_0 = U - U'$, $Y_1, Y_2, \ldots, Y_{n-k}$ still remaining to be processed.
- For $i = 1, 2, \ldots, n-k$, perform the translocation $(R_{i-1}, Y_i) \longrightarrow (\{u'_{k+i}\}, R_i)$.

Before proceeding with the proof of the "if" part of Lemma 3.1, we need a few definitions and results.

Definition 3.1 *A connected instance $\mathcal{S}(n,k)$ of the synteny problem is* exact *if $n = k$ and $D(\mathcal{S}(n,k)) = n-1$.*

Definition 3.2 *Let $\mathcal{S}(n,k)$ be an instance of synteny. A move on $\mathcal{S}$ is called a* splitting move *if it increases the number of components of $\mathcal{S}$ by one and it is called a* non-splitting move *otherwise.*

Definition 3.3 *Let $\mathcal{S}(n,n)$ be a connected instance of synteny. A splitting move on $\mathcal{S}(n,n)$ is called a balanced move if it creates two subproblems $\mathcal{S}_1(n_1,n_1)$ and $\mathcal{S}_2(n_2,n_2)$ for some n_1 and n_2.*

A splitting move must be a translocation or fission since fusions cannot increase the number of components. In the case of a translocation, it must operate on sets in the same connected component. A balanced move must be a translocation since fissions increase the total number of sets.

Lemma 3.2 *Every move in any optimal move sequence for an exact instance of synteny is a balanced move on a connected component.*

Proof: Let $\mathcal{S}(n,n)$ be an exact instance of synteny and let $\sigma = (\sigma_1, \ldots, \sigma_{n-1})$ be an optimal move sequence. Since $\mathcal{S}(n,n)$ is connected, each move of σ must be a splitting move. Assume σ_1 splits $\mathcal{S}(n,n)$ into two subproblems $\mathcal{S}_1(n_1,k_1)$ and $\mathcal{S}_2(n_2,k_2)$, where $n_1 + n_2 = k_1 + k_2 = n$. (Note these problems have disjoint alphabets.) Since each subsequent move must act on a connected component of the current problem, we can partition $(\sigma_2, \ldots, \sigma_{n-1})$ into two subsequences that solve, respectively, $\mathcal{S}_1$ and $\mathcal{S}_2$. By optimality of σ, these subsequences must be optimal move sequences so

$$D(\mathcal{S}_1(n_1,k_1)) + D(\mathcal{S}_2(n_2,k_2)) = D(\mathcal{S}(n,n)) - 1 = n-2.$$

By Proposition 2.4 and the fact that $\mathcal{S}_i(n_i, k_i)$ is connected, $D(\mathcal{S}_i(n_i, k_i)) \geq \max(n_i, k_i) - 1$. Thus $n_i = k_i$. ■

Notice that the instance of the synteny problem created in Theorem 3.2 is exact. Proof of Lemma 3.1 is complete if we can prove the following lemma.

Lemma 3.3 *If $D(S(2n-k+1, 2n-k+1)) = 2n-k$. then G has a LBQIS of size k.*

Proof: Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{2n-k})$ be any optimal move sequence. By Lemma 3.2, every move is a balanced translocation.

First, we claim that, after a possible reordering of the indices of the a-elements, the move σ_j, for $1 \leq j \leq n-k$, must be a translocation $(P_{j-1}, X_\ell) \longrightarrow (\{a_j\}, P_j)$ for some $\ell \in \{v_1, v_2, \ldots, v_n\}$. For contradiction, assume this is not the case and rearrange the indices of the a-elements, if necessary, so that $j \leq n-k$ is the least index such that σ_j violates the condition. Since σ_j must be a splitting translocation, it cannot translocate two X-sets, an X-set with a Y-set, or an Y-set with P_{j-1}. Hence, σ_j must translocate an X-set with P_{j-1}. Let $\sigma_j = (P_{j-1}, X_\ell) \rightarrow (P', P'')$. Assume that $b \in P''$ (and, hence $b \notin P'$). Since σ_j must be a balanced move, P' must contain at most 1 a-element and at most 1 u-element. If P' does not contain any a-element, σ_j does not violate the condition. Otherwise, P' contains exactly one u-element u_t and no a-element. Then, modify σ_j such that the two elements u_t and a_j exchange their places in P' and P'' and then σ_j satisfies our condition.

Hence, after the move σ_{n-k}, we have the set $Q_k = P_{n-k} = \{u_1, u_2, \ldots, u_n, b\}$, the sets $Y_1, Y_2, \ldots, Y_k$ and some k X-sets, say $X_{v_1}, x_{v_2}, \ldots, X_{v_k}$. Then, by essentially the same reasoning as before, after a possible rearrangement of the indices, the move σ_j, for $n-k+1 \leq j \leq n$, must be the translocation $(Q_{n-j+1}, X_{v_{n-j+1}}) \longrightarrow (\{u_{n-j+1}\}, Q_{n-j})$. This implies that $(u_i, v_j) \notin E$ for $1 \leq i \leq k$ and $i > j$. ■

This completes the proof of Theorem 3.2. To complete our proof of Theorem 3.1, it is sufficient to prove the following theorem.

Theorem 3.3 *The LBQIS problem for bipartite graphs is NP-complete.*

We will reduce LBIS to the LBQIS problem. Assume that we are given an instance (G, k) of the LBIS problem, where $G = (U, V, E)$, $U = V = \{1, 2, \ldots, n\}$. We create an instance (G', k') of the LBQIS problem, where $k' = k^2 + k$, $G' = (U', V', E')$, $U' = V' = \{[i, j] \mid 1 \leq i \leq k+1,\ 1 \leq j \leq n\}$, and $E' = E_1 \cup E_2$ consists of the following edges:

$$\begin{aligned} E_1 &= \{([i,k],[j,l]) \mid i < j\} \\ E_2 &= \{([i,k],[j,l]) \mid i \geq j,\ (k,l) \in E\} \end{aligned}$$

Intuitively, we use the amplification technique (see, for example, [14, page 428–429]) and "blow up" the graph G by using $k+1$ copies of it with some additional edges. We will prove the following lemma showing the correctness of our reduction.

Lemma 3.4 *G has an LBIS of size k if and only if G' has an LBQIS of size k'.*

The proof of the "only if" part of Lemma 3.4 is easy. Let $U_1 \subseteq U$ and $V_1 \subseteq V$ be an LBIS of G of size k. Assume, wlog, that $U_1 = V_1 = \{1, 2, \ldots, k\}$. Let $U_1' = V_1'$ be the following permutation of a subset of $k^2 + k$ vertices of G':

$$[1,1], [1,2], \ldots, [1,k], [2,1], [2,2], \ldots, [2,k], \ldots, [k+1,1], [k+1,2], \ldots, [k+1,k]$$

Then, U_1' and V_1' induces an LBQIS of size k' in G'.

The proof of the "if" part of Lemma 3.4 is more involved. Let σ_1 and σ_2 be a permutation of the vertices in U' and V', respectively, which realizes an LBQIS of size $k' = k^2 + k$. One crucial step in the proof is the following lemma which says that σ_1 and σ_2 can be decomposed in $k+1$ modules.

Lemma 3.5 (Rearrangement lemma) *There exist integers $p_1, p_2, \ldots, p_{k+1} \geq 0$, $p_1 + p_2 + \cdots + p_{k+1} = k^2 + k$, such that σ_1 and σ_2 may be assumed to be of the following forms:*

$$\sigma_1 = ([1, x_1^1], \ldots, [1, x_1^{p_1}], [2, x_2^1], \ldots, [2, x_2^{p_2}], \ldots, [k+1, x_{k+1}^1], \ldots, [k+1, x_{k+1}^{p_{k+1}}])$$

$$\sigma_2 = ([1, y_1^1], \ldots, [1, y_1^{p_1}], [2, y_2^1], \ldots, [2, y_2^{p_2}], \ldots, [k+1, y_{k+1}^1], \ldots, [k+1, y_{k+1}^{p_{k+1}}])$$

where $x_i^j, y_i^j \in \{1, 2, \ldots, n\}$ and $p_i = 0$ means that that sequence $[p_i, y_{p_i}^1], [p_i, y_{p_i}^2], \ldots, [p_i, y_{p_i}^{p_i}]$ is absent.

Proof: We may first assume without loss of generality that the permutations are

$$\sigma_1 = ([1, x_1^1], \ldots, [1, x_1^{p_1}], [2, x_2^1], \ldots, [2, x_2^{p_2}], \ldots, [k+1, x_{k+1}^1], \ldots, [k+1, x_{k+1}^{p_{k+1}}])$$

$$\sigma_2 = ([1, y_1^1], \ldots, [1, y_1^{q_1}], [2, y_2^1], \ldots, [2, y_2^{q_2}], \ldots, [k+1, y_{k+1}^1], \ldots, [k+1, y_{k+1}^{q_{k+1}}])$$

Then clearly $q_1 \leq p_1$, $q_1 + q_2 \leq p_1 + p_2$, ..., $q_1 + \ldots + q_k \leq p_1 + \ldots + p_k$ because of the edges in E_1. Since $\sum_{i=1}^{k+1} q_i = \sum_{i=1}^{k+1} p_i = k^2 + k$, $q_{k+1} \geq p_{k+1}$. Define $A = \{i \mid [j, i] \in \sigma_1\}$.

If $|A| \geq q_{k+1}$, then we can modify the suffix $[i, x_i^j], [i, x_i^{j+1}], \ldots, [k+1, x_{k+1}^{p_{k+1}}]$ of σ_1 with length q_{k+1} so that $x_i^j, ..., x_{k+1}^{p_{k+1}}$ are all distinct elements of A. Hence we can replace it with the sequence $[k+1, x_i^j], [k+1, x_i^{j+1}], \ldots, [k+1, x_{k+1}^{p_{k+1}}]$. The proof is then completed by induction.

Otherwise, suppose $|A| < q_{k+1}$. Then $q_{k+1} > p_1, \ldots, p_{k+1}$. In particular, $q_{k+1} > p_1$. Now we can modify the prefix $[1, y_1^1], [1, y_1^2], \ldots, [i, y_i^j]$ of σ_2 with length p_1 so that $y_1^1, \ldots, y_i^j$ are all distinct elements of $\{y_{k+1}^1, \ldots, y_{k+1}^{q_{k+1}}\}$. Hence we can replace it with the sequence $[1, y_1^1], [1, y_1^2], \ldots, [1, y_i^j]$. The proof is again completed by induction. ■

Now, to complete the proof of Lemma 3.4, note that we have the following two cases.

Case 1. There exists $i > j$ such that $p_i \geq k$ and $p_j \geq k$. Then, there is no edge between the vertices $[p_i, x_{p_i}^1], [p_i, x_{p_i}^2], \ldots, [p_i, x_{p_i}^{p_i}]$ and $[p_j, y_{p_j}^1], [p_j, y_{p_j}^2], \ldots, [p_j, y_{p_j}^{p_j}]$. Since $i > j$, by our construction of G', G must have an LBIS of size at least k consisting of the vertices $U_1 = \{x_{p_i}^1, x_{p_i}^2, \ldots, x_{p_i}^{p_i}\} \subseteq U$ and $V_1 = \{y_{p_j}^1, y_{p_j}^2, \ldots, y_{p_j}^{p_j}\} \subseteq V$.

Case 2. There are no such pair of indices as in Case 1. Let $t \geq 2$ be the largest integer such that $p_t \geq k$. Now, we have two cases:

(a) There is no such t. In this case, $p_1 = k^2 + k - (\sum_{i=2}^{k+1} p_i) \geq 2k$.

(b) Otherwise, $p_i < k$ for $i \neq t$ and hence again $p_t \geq 2k$.

Hence, in either case, there exists an index j such that $p_j \geq 2k$. Then, the vertices $U_1 = \{x_j^{p_j}, x_j^{p_j - 1}, \ldots, x_j^{p_j - k + 1}\} \subseteq U$ and $V_1 = \{y_j^1, y_j^2, \ldots, y_j^k\} \subseteq V$ form an LBIS of size k for G. This completes the proof of Lemma 3.4.

4 A Simple Approximation Algorithm for the Synteny Problem

In this section, we describe a polynomial time approximation algorithm for the synteny problem with performance ratio 2.

Theorem 4.1 *Let $\mathcal{S}(n,k)$ be an instance of the synteny problem. Then, it is possible to approximate $D(\mathcal{S}(n,k))$ with a performance ratio of 2 in $O(nkA^{-1}(nk,k))$ time (if the input is not in compact representation, then the time taken is $O(nkA^{-1}(nk,k)+\mu)$, where μ is the time to transform a given instance of the synteny problem to its compact representation as stated in Proposition 2.3).*

Proof: Assume, without loss of generality, that $n \geq k$. Assume $\mathcal{S}(n,k)$ has p components and let n_i (resp. k_i) be the corresponding number of elements (resp. number of sets) in the i^{th} connected component of $G(S)$. Our simple fusion-fission algorithm is as follows. First, find the connected components of $G(S)$. Then, for each connected component, repeatedly use fusion until only one set is remaining and then repeatedly use fission to separate the remaining elements from the set. In all, we perform $\sum_{i=1}^{p}(n_i+k_i-2) = n+k-2p \leq 2n-2p$ moves. By Proposition 2.4, $D(\mathcal{S}(n,k)) \geq n-p$, and hence a performance ratio of 2 is achieved. Note that the approximation algorithm uses no translocations.

Now, we analyze the time complexity of our approximation algorithm. Converting the given instance of the synteny problem to its compact representation, if necessary, takes $O(\mu)$ time. At the end of this preprocessing, the compact representation consists of k sets $S_1, S_2, \ldots, S_k \subseteq \{1,2,\ldots,n\}$, each set being represented by a list of its elements. The remaining time bounds are as follows:

(1) Since each set is represented by a list of its elements, scan the sets $S_1, S_2, \ldots, S_k$, collecting for each element $i \in \{1,2,\ldots,n\}$ the indices of sets, denoted by I_i, in which it appears. This takes $O(kn)$ time. Notice that $|I_i| \leq k$ for all i.

(2) Use k MAKESET operations to create k disjoint sets, say $\{1\},\{2\},\ldots,\{k\}$.

(3) For every element $i \in \{1,2,\ldots,n\}$, do a SET-UNION on the sets that contain the indices in I_i using both the union by rank and path compression heuristics (before doing SET-UNION, one must check by doing two FIND-SET operations if the two sets are not already together).

(4) Finally, do a FIND-SET for every index $1,2,\ldots,k$ to find and collect the indices of the sets in the same connected components, using both the union by rank and path compression heuristics.

We do a total of at most $4k+nk$ set operations on a set of k elements. Hence, the total time to find the connected components is $O((4k+nk)A^{-1}(4k+nk,k)) = O(nkA^{-1}(nk,k))$ (see [5, page 449]). It can be easily seen that the remaining time of our heuristic (for fissions and fusions) is at most $O(nk)$. ■

Remark 4.1 *The performance ratio 2 of the above heuristic is tight. Let the instance $S(n,n)$ consist of the n sets $\{1\},\{1,2\},\{1,2,3\},\ldots,\{1,2,\ldots,n\}$. Then, $D(S(n,n)) = n-1$, whereas our heuristic takes $2n-2$ moves.*

It is possible to use a few less moves (i.e., use $n+k-3p$ moves instead of $n+k-2p$ moves, assuming every component has at least 2 elements and at least 2 sets) if we replace the last fusion in our heuristic by a translocation which separates one of the elements from the rest, but this will not improve the performance ratio asymptotically. However, this shows that at most $2D(\mathcal{S}(n,k))-1$ moves suffice for this slightly modified version of our approximation algorithm.

5 Linear Synteny

The move sequences used in the NP-completeness proof and (without loss of generality) produced by the approximation algorithm have a particular form. There is a merging set Δ that is initially one of the input sets. The first $k-1$ moves are either fusions or very restricted translocations between Δ and an input set. The restriction on translocations is that only translocations that produce a singleton set $\{j\}$ such that j does not occur in any other set are allowed. The remaining moves are fissions on Δ that create singleton sets. In this section we study this restricted problem.

Let $\mathcal{S}(n,k)=\{S_1,\ldots,S_k\}$ be a connected instance of synteny and let π be a permutation of $[1,\ldots,k]$. The *linear move sequence* σ_π for $S(n,k)$ is defined as follows.

1. Let $\Delta_1 = S_{\pi_1}$.
2. For $i=1,2,\ldots,k-1$
   (a) If there is $j\in\Delta_i\cup S_{\pi_{i+1}}$ that is not in $\cup_{\ell=i+2}^{k}S_{\pi_\ell}$ then choose the smallest such j and set $\sigma_i=(\Delta_i,S_{\pi_{i+1}})\longrightarrow(\Delta_{i+1},\{j\})$ where $\Delta_{i+1}=(\Delta_i\cup S_{\pi_{i+1}})-\{j\}$.
   (b) Otherwise $\sigma_i=(\Delta_i,S_{\pi_{i+1}})\longrightarrow\Delta_{i+1}$.
3. For $i=k,\ldots,k+|\Delta_k|-1$, let j be the smallest element in Δ_i and set $\sigma_i=\Delta_i\longrightarrow(\Delta_{i+1},\{j\})$.

If $\mathcal{S}(n,k)$ is not connected, a linear move sequence is a partition of the connected components of $\mathcal{S}(n,k)$ and a linear move sequence for each.[3] We let $\tilde{D}(\mathcal{S}(n,k))$ denote the length of the shortest linear move sequence for $\mathcal{S}(n,k)$.

Since the NP-Completeness proof uses linear move sequences, $\tilde{D}(\mathcal{S}(n,k))$ is hard to compute. Since the approximation algorithm can be easily transformed into a linear move sequence that is no longer (by splitting off singleton sets whenever permitted by the definition of such sequences) and $D(\mathcal{S}(n,k))\le\tilde{D}(\mathcal{S}(n,k))$, this algorithm gives a 2-approximation of $\tilde{D}(\mathcal{S}(n,k))$. Note as well that the optimal move sequence for the example given in Remark 4.1 is a linear move sequence so, as in the general case, the 2-approximation bound is tight for this algorithm.

It remains open whether one can approximate linear synteny by a factor better than 2, but this problem seems easier to analyze than the general synteny problem. The following theorem says, in fact, it suffices to improve the approximation bound for linear synteny since any such algorithm yields a better approximation for the general problem.

Theorem 5.1 *If linear synteny can be approximated within a factor of c in polynomial time then for any $\epsilon>0$, general synteny can be approximated within a factor of $c+\epsilon$ in polynomial time.*

[3] If for example an input is $\{1\},\{2\},\ldots,\{n\}$ then no moves are required in either the original or linear versions of synteny.

In the next section we show that instances of the synteny problem where the distance is a fixed constant can be solved exactly in polynomial time. Thus, in order to prove this theorem we can limit our consideration to instances of synteny where the distance is sufficiently large. Therefore, the theorem follows directly from Lemma 5.1 which is the main content of this section.

Lemma 5.1 *Let $\mathcal{S}(n,k)$ be an instance of synteny. Then*

$$\tilde{D}(\mathcal{S}(n,k)) \leq D(\mathcal{S}(n,k)) + \log_{4/3} D(\mathcal{S}(n,k))).$$

To prove this lemma we need the following definitions.

Definition 5.1 *Let $\mathcal{S}(n,k)$ be an instance of synteny and let σ be an arbitrary move sequence for $\mathcal{S}(n,k)$. The* move digraph *$G_M(\mathcal{S},\sigma)$ contains a vertex for each move in σ. If σ_i creates a set S that is input to σ_j then G_M has an edge from σ_i to σ_j. (Note that we think of each occurrence of a set as being "tagged" by the move that created the set and the move that consumes the set. Thus each occurrence of an intermediate set is associated with exactly one directed edge in G_M.)*

We point out that G_M implies a partial order on the moves in σ and any consistent total order yields a move sequence for $\mathcal{S}(n,k)$. If σ is optimal, each total order yields an optimal move sequence for $\mathcal{S}(n,k)$. Note that G_M is directed, acyclic and each node has in-degree and out-degree at most 2. A directed graph is *weakly connected* if it is connected when its edges are considered in an undirected sense.

Definition 5.2 *Let G be a weakly connected, directed acyclic graph on n nodes. An $f(n)$ directed biseparator of G is a non-empty subset of edges A whose removal partitions G into two weakly connected components G_1 and G_2 such that each has between $f(n)$ and $n - f(n)$ nodes. Further, for every $< u, v > \in A$, $u \in G_1$ and $v \in G_2$.*

The proof of Lemma 5.1 uses the following graph-theoretic lemma whose proof can be found in [9].

Lemma 5.2 *Let G be a weakly connected, directed, acyclic graph on n nodes where the in-degree and out-degree of each node are each at most 2. Then G has a $\frac{n}{4}$ directed biseparator.*

Proof of Lemma 5.1: Because of the form of Lemma 5.2, all logarithms in this proof are to the base 4/3. Let σ be an arbitrary move sequence for $\mathcal{S}(n,k) = \{S_1, \ldots, S_k\}$ of length d. To prove the bound it suffices to prove the case where σ consists solely of translocations. To see this, notice first that by Corollary 2.1 we may assume that fissions occur after all translocations which occur after all fusions. At the end of the fusion/translocation stages, the current sets $T_1, \ldots, T_\ell$ are disjoint. Create a new instance of synteny by renaming each j in the current set T_i as a_i in the original instance. Thus the fusion/translocation stages of σ solves the new problem. Suppose σ_π is a linear move sequence that solves the new problem and has length $\leq d' + \log d'$, where d' is the length of the fusion/translocation stage of σ. Then running σ_π on the original problem requires $d - d'$ additional fissions and has length $\leq d + \log d' \leq d + \log d$. So assume that σ consists of fusions followed by translocations. Consider the synteny instance $T_1, \ldots, T_\ell$ created by the last fusion. Suppose σ_π is a linear move sequence that solves this problem and has length $\leq d' + \log d'$, where d' is the number of translocations. For each T_{π_i}, let π_i' be an arbitrary ordering of the sets that were

fused to create T_{π_i} and let $\pi' = \pi'_1 \cdot \pi'_2 \cdots \pi'_\ell$. Then $\sigma_{\pi'}$ has length at most $d + \log d' \leq d + \log d$. (In other words, we are performing "just-in-time" fusions to create the sets T_i as demanded by the linear move sequence σ_π.) Thus, since prefixes of fusions and suffixes of fissions can be made linear without increasing their length, we will now focus on substrings of moves consisting only of translocations.

Notice in this case that $n = k$. If $d = 1$ then σ is already a linear move sequence so assume $d \geq 2$. First consider the case where $G_M(\mathcal{S}(n,k), \sigma)$ is connected. Note that G_M has d nodes.

By Lemma 5.2, there exists a $d/4$ directed biseparator A of $G_M(\mathcal{S}(n,k), \sigma)$. Let G_1 and G_2 be the two weakly connected components created by removing A. Assume G_i has d_i nodes; note $d_1 + d_2 = d$. We construct two new synteny instances as follows.

$\mathcal{S}_2$: Each edge e of A corresponds to a set T_e that is passed to a node of G_2. The instance $\mathcal{S}_2$ consists of these sets $T_e, e \in A$, plus any of the input sets of $\mathcal{S}(n,k)$ that are input to G_2. Notice that any move sequence implied by G_2 (i.e. consistent total order on its nodes) is a move sequence that solves $\mathcal{S}_2$. Since each move is a translocation, $\mathcal{S}_2$ has the same number of sets as elements; let n_2 be this number.

$\mathcal{S}_1$: Initially let $\mathcal{S}_1$ consist of the input sets of $\mathcal{S}(n,k)$ that are input to nodes of G_1. A move sequence implied by G_1 does not typically solve $\mathcal{S}_1$ because the sets $T_e, e \in A$, may not be disjoint singleton sets. To fix this, let us first rename an element j that occurs in the set S_ℓ of $\mathcal{S}_1$ as $[j, \ell]$. Carry the renaming through the moves of σ. In particular, if the two input sets to a move contain $[j,1]$ and $[j,2]$ respectively, and only one of the output sets of the move contains the element j then associate both $[j,1]$ and $[j,2]$ with this element j. (Recall that we can assume without loss of generality that each element j is present in at most one of the output sets of a translocation.) Then for each T_e create a new *dummy* name a_e. For each $[j,\ell] \in T_e$, rename $[j,\ell]$ as a_e in $\mathcal{S}_1$. Finally, each final set that is not an input to $\mathcal{S}_2$ must consist of elements $[j,\ell]$ where the first component is a fixed j and the second component ranges over all possibilities. Restore the original name, j, for all such pairs. Note that the same element j might end up receiving different names if the various occurrences of j end up in separate sets T_e for $e \in A$. With this renaming a move sequence implied by G_1 solves $\mathcal{S}_1$. As above, $\mathcal{S}_1$ has the same number of sets as elements; let n_1 be this number. Let $W_1 = \{a_e | e \in A\}$.

Inductively assume that there is a linear move sequence σ_i with associated permutation π_i for $\mathcal{S}_i$ of length $\leq d_i + \log_{4/3}(d_i)$. Let π be the order on the sets of $\mathcal{S}$ induced by π_1 followed by π_2. We claim that a modification of σ_π has length at most $d + \max(\log_{4/3}(d_1), \log_{4/3}(d_2)) + 1$. The lemma follows since

$$\log_{4/3}(n) \geq 1 + \log_{4/3}(\frac{3n}{4}).$$

For the accounting we'll modify σ_π slightly to create its singleton sets in a way we can count. In the following $\Delta_{1,j}$, $\Delta_{2,j}$ and Δ_j denote the current merging set of, respectively, σ_{π_1}, σ_{π_2} and σ_π before their j-th move. During the first $n_1 - 1$ merges σ_π matches the moves of σ_{π_1}. By this we mean that when σ_{π_1} creates a singleton $\{j\}$, σ_π creates the same singleton provided $j \notin W_1$. If $j \in W_1$ then σ_π simply performs a fusion at that step. The n_1-th move of σ_π is a fusion of Δ_{n_1} with the first set in the ordering for $\mathcal{S}_2$. In the remaining moves it matches the first $n_2 - 1$ moves of σ_{π_2}, except those involving sets $T_e, e \in A$, during which σ_π makes no move. Finally, the sequence includes whatever fissions are necessary to end up with singleton sets. Let W_2 be the set of singletons

created by σ_{π_2} in translocations with the sets $T_e, e \in A$. Notice that $\Delta_{n_1} = (\Delta_{1,n_1} \cup \bigcup_{e \in A} T_e) - W_1$ and $\Delta_n = (\Delta_{1,n_1} - W_1) \cup \Delta_{2,n_2} \cup W_2$. Let $r_1 = |\Delta_{1,n_1} - W_1|$ and $r_2 = |\Delta_{2,n_2}|$.

As described σ_π is somewhat wasteful. No element of $\Delta_{1,n_1} - W_1$ exists in an $\mathcal{S}_2$ set. If there are f fusions in all but the first $n_1 - 1$ moves of σ_π, we can replace $\min(r_1, f)$ of them by translocations creating singletons from $\Delta_{1,n_1} - W_1$. The sequence σ_{π_2} together with the n_1^{th} move contains r_2 fusions of which $|A| - |W_2|$ involve sets $T_e, e \in A$. The remaining have corresponding fusions in σ_π. Since Δ_{1,n_1} and Δ_{2,n_2} are disjoint, the modified move sequence ends with a set Δ_n where

$$|\Delta_n| = r_1 + r_2 - \min(r_1, r_2 - |A| + |W_2|) + |W_2| = \max(r_1, r_2 - |A| + |W_2|) + |A|.$$

An additional $|\Delta_n| - 1$ fissions are needed. The number of moves we have performed in σ_π is $n - 1$ and taking into account the additional $|\Delta_n| - 1$ fissions we have to perform, we must show:

$$\begin{aligned} d + \log_{4/3}(d) &\geq n + \max(r_1, r_2 - |A| + |W_2|) + |A| - 2 \\ &= n_1 + n_2 + \max(r_1, r_2 - |A| + |W_2|) - 2. \end{aligned}$$

since $n = n_1 + n_2 - ||A||$.

First consider the case $r_1 \geq r_2 + |W_2| - |A|$. Since by induction

$$\begin{aligned} d_1 + \log_{4/3}(d_1) &\geq n_1 - 1 + |\Delta_{1,n_1}| - 1 \\ &\geq n_1 + r_1 - 2, \end{aligned}$$

(as $r_1 = |\Delta_{1,n_1} - W_1|$) and $d_2 \geq n_2 - 1$, we have

$$\begin{aligned} n_1 + n_2 + \max\{r_1, r_2 + |W_2| - |A|\} - 2 &= n_1 + n_2 + r_1 - 2 \\ &\leq d_1 + \log_{4/3}(d_1) + d_2 + 1 \\ &= d + \log_{4/3}(d_1) + 1 \\ &\leq d + \log_{4/3}(3d/4) + 1 \\ &= d + \log_{4/3} d. \end{aligned}$$

So in this case the length of the modified sequence for the overall problem is at most $d + \log_{4/3} d$.

Next, suppose $r_2 - |A| + |W_2| > r_1$. Again by induction $d_2 + \log_{4/3}(d_2) \geq n_2 + r_2 - 2 \geq n_2 + r_2 - (|A| - |W_2|) - 2$ (since $(|A| - |W_2|)$ is non-negative) and $d_1 \geq n_1 - 1$. So we have

$$\begin{aligned} n_1 + n_2 + \max\{r_1, r_2 + |W_2| - |A|\} - 2 &= n_1 + n_2 + r_2 - (|A| - |W_2|) - 2 \\ &\leq d_2 + \log_{4/3}(d_2) + d_1 + 1 \\ &= d + \log_{4/3}(d_2) + 1 \\ &\leq d + \log_{4/3}(3d/4) + 1 \\ &= d + \log_{4/3} d. \end{aligned}$$

■

Finally we consider the situation where G_M has more than one component. We consider the components of G_M in decreasing order of size (number of moves) and assume by Lemma 5.1 that we have found a good linear move sequence for each component. Now when considering the i^{th} component we perform the fusion and translocation moves in the linear move sequence for this component. If there are elements left in the set Δ being carried at this point, we perform a translocation of Δ with the first set in the linear move sequence for the $(i+1)^{th}$ component and produce one of the elements of Δ as a singleton. Whenever we perform a fusion in the future, we can instead make this move a translocation and produce any remaining element of Δ. Thus, it can be seen that the overhead of the entire linear move sequence is at most $\log_{4/3}(n_1)$ where n_1 is the size of the first component.

6 Optimal Syntenic Sequence When the Distance is Bounded

In practice, it may be the case, that the synteny distance between two genomes is bounded, and one is interested in finding the optimal sequence of synteny moves between the two genomes. The following theorem states our result in this regard.

Theorem 6.1 *Let $\mathcal{S}(n,k)$ be an instance of the synteny problem with $D(\mathcal{S}(n,k)) \leq d$. Then, an optimal sequence of synteny moves for $\mathcal{S}(n,k)$ can be computed in $O(nk + 2^{O(d^2)})$ time.*

We need to prove a few results before proving Theorem 6.1. As usual, we may assume $n \geq k$ without any loss of generality.

Lemma 6.1 *If $D(\mathcal{S}(n,k)) \leq \frac{n}{3} - 1$, then $\mathcal{S}(n,k)$ has one connected component containing just the set $\{a_i\}$ for some $1 \leq i \leq n$.*

Proof: Assume $G(S)$ has $p \geq 1$ connected components $C_1, \ldots, C_p$ and let $n_i \geq 1$ (resp. $k_i \geq 1$) be the number of elements (resp. number of sets) in C_i. Suppose $\mathcal{S}(n,k)$ has no connected components consisting of a singleton set, i.e., $n_i + k_i \geq 3$ for all i. Thus, $n + k = \sum_{i=1}^{p}(n_i + k_i) \geq 3p$, implying $p \leq \frac{2n}{3}$. But, by Proposition 2.4, $p \geq n - D(\mathcal{S}(n,k)) \geq \frac{2n}{3} + 1$, a contradiction! ■

Lemma 6.2 *Assume that a given instance $\mathcal{S}(n,k)$ has a connected component containing only the set $\{a_i\}$. Let $T(n-1,k-1)$ be the instance obtained by removing the element a_i and the set $\{a_i\}$ from $\mathcal{S}(n,k)$. Then, $D(\mathcal{S}(n,k)) = D(T(n-1,k-1))$.*

Proof: Obviously, $D(\mathcal{S}(n,k)) \leq D(T(n-1,k-1))$. Conversely, assume that an optimal sequence for $\mathcal{S}(n,k)$ translocates the set $\{a_i\}$ with some other set. We do not perform this translocation, but proceed with the remaining moves assuming that the element a_i is carried to subsequent sets. Finally, we must have a translocation or fission separating a_i from other elements. If it is a translocation, we replace it by a fusion, whereas if it is a fission, we do nothing. So, in fact we save moves by not translocating a_i. Hence, $D(T(n-1,k-1)) \leq D(\mathcal{S}(n,k))$. ■

We now proceed with the proof of Theorem 6.1. By repeatedly applying Lemma 6.2, given an instance $\mathcal{S}(n,k)$, we can derive an instance $T(n',k')$ such that $n' \leq n$, $k' \leq k$, $n' \geq k'$ and $D(\mathcal{S}(n,k)) = D(T(n',k')) \geq \frac{n'}{3}$. Hence, it is sufficient to prove the theorem when $n \geq k$ and $d \geq \frac{n}{3}$. By Lemma 2.2, we know that it is sufficient to do at most α_1 fusions first, at most α_2 translocations next and at most α_3 fissions at the end, for every choice of $\alpha_1, \alpha_2, \alpha_3 \geq 0$ such that $\alpha_1 + \alpha_2 + \alpha_3 \leq d$.

The total number of choices of $\alpha_1, \alpha_2, \alpha_3 \geq 0$ such that $\alpha_1 + \alpha_2 + \alpha_3 \leq d$ is at most

$$\sum_{0 \leq i \leq d} \binom{i+3}{i} = \sum_{0 \leq i \leq d} \binom{i+3}{2} = O(d^3) = 2^{O(\log d)}$$

For every such choice, we count the total number of possible alternative moves we may have to perform. First, we count the total number f of sequences of fusions we need to look at. Clearly,

$$f \leq \alpha_1 \prod_{1 \leq j \leq \alpha_1} \binom{k-j+1}{2} \leq d\binom{k}{2}^{\alpha_1} < d\left(\frac{k^2}{2}\right)^{\alpha_1} \leq d\left(\frac{n^2}{2}\right)^{\alpha_1} \leq d\left(\frac{9d^2}{2}\right)^{d} = 2^{O(d \log d)}$$

Next, we count the total number t of sequences of translocations we need to look at. Since there are at most $2^n - 1$ ways to translocate two sets,

$$t \leq \left(\binom{k}{2} (2^n - 1) \right)^{\alpha_2} \leq \left(\frac{k^2 2^n}{2} \right)^d \leq \left(\frac{n^2 2^n}{2} \right)^d \leq \left(\frac{9d^2 2^{3d}}{2} \right)^d = 2^{O(d^2)}$$

Since all the fissions are done at the end, it suffices to consider a single canonical sequence of doing them. Hence, the total number of sequences of moves to consider is at most

$$2^{O(\log d)} \cdot f \cdot t = 2^{O(d^2)}$$

Finally, our algorithm should do the following for each of the above $2^{O(d^2)}$ move sequences:

(i) Simulate the sequence

(ii) Test if it solves the given input instance

(iii) Retain the shortest solution sequence.

Since any move can be performed in $O(n)$ time and the sequence is of length at most d, the time taken for (i) is $O(nd) = 2^{O(\log d)}$.

Time taken for (ii) is $O(kn) = 2^{O(\log d)}$, since one just needs to check if the final collection of sets is $\{1\}, \{2\}, \ldots, \{n\}$.

Time taken for (iii) can be bounded as follows. The fusions in the sequence can be represented in $O(k)$ space, the translocations can be represented in $O(dn) = O(d^2)$ space, and all the fissions can be represented in only $O(1)$ space, hence the total space taken is $O(d^2)$. Hence, to retain the current best sequence takes only $O(d^2) = 2^{O(\log d)}$ time.

Hence, the total time taken by (i), (ii) and (iii) for a particular sequence is $2^{O(\log d)}$. As a result, the total time for *all* the sequences of moves is $2^{O(d^2)} \cdot 2^{O(\log d)} = 2^{O(d^2)}$.

Finally, we need to consider the time to preprocess the input $\mathcal{S}(n, k)$ to obtain an instance $\mathcal{S}(n', k')$ for which $k' \leq n' \leq 3d$. This just involved deleting all singleton sets containing an element that occurs in no other sets, and can be easily done in $O(nk)$ time.

Combining all the time, the total time taken by our algorithm to compute the optimal synteny sequence is $O(nk + 2^{O(d^2)})$.

7 The Median Problem

The median problem arises in connection with the phylogenetic inference problem[6] and defined as follows. Given three genomes $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$, construct a genome $\mathcal{G}$ such that the *median distance* $\alpha_{\mathcal{G}} = \sum_{i=1}^{3} D(\mathcal{G}, \mathcal{G}_i)$ is minimized and also compute the value of the median distance $\alpha_{\mathcal{G}}$ corresponding to the median $\mathcal{G}$ (while computing the syntenic distance between two genomes, we delete any elements that is not common between the two genomes). Without any additional constraints, this problem is trivial, since we can take $\mathcal{G}$ to be empty (and then $\alpha_{\mathcal{G}} = 0$). In the context of syntenic distance, any one of the following three constraints seems relevant [6]:

(c1) If a gene is present in all the three given genomes, then this gene must be present in $\mathcal{G}$.

(c2) If a gene is present in at least two of the three given genomes, then this gene must be present in $\mathcal{G}$.

(c3) If a gene is present in at least one of the three given genomes, then this gene must be present in $\mathcal{G}$.

Lemma 7.1 *Computing the median distance is NP-hard with any one of the three constraints* **(c1), (c2) or (c3)**.

Proof: We reduce the synteny problem to this problem. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be the two genomes of any instance of the synteny problem and let $d = D(\mathcal{G}_1, \mathcal{G}_2) > 0$. The NP-hardness reduction of Section 3 shows that we may assume that both $\mathcal{G}_1$ and $\mathcal{G}_2$ contain the same set of genes. Let $\mathcal{G}_1$, $\mathcal{G}_1$ and $\mathcal{G}_2$ be the three genomes for the corresponding median problem. Assume that $\mathcal{G}$ is the solution of the median problem (under any one of the constraints). If $\mathcal{G} \neq \mathcal{G}_1$, then $\alpha_{\mathcal{G}} \geq d + D(\mathcal{G}, \mathcal{G}_1) > d$; but if $\mathcal{G} = \mathcal{G}_1$, then $\alpha_{\mathcal{G}} = d$. Hence, $\alpha_{\mathcal{G}}$ is precisely the syntenic distance between $\mathcal{G}_1$ and $\mathcal{G}_2$ and determining $\alpha_{\mathcal{G}}$ determines d also. ■

The next lemma shows that there is an easy polynomial time approximation algorithm for the median problem with performance ratio close to 4. Let n_i (resp. k_i) be the number of elements (resp, number of sets) in the three given genomes $\mathcal{G}_i$. Let $n = \max\{n_1, n_2, n_3\}$, and $k = \max\{k_1, k_2, k_3\}$.

Lemma 7.2 *We can approximate the median distance in* $O(nkA^{-1}(nk, k) + 2^{O(4/\epsilon^2)})$ *time (under any one of the constraints (c1), (c2) or (c3)) with ratio* $4 + \epsilon$ *for any constant* $\epsilon > 0$.

Proof: Let $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ be the three given genomes and $d > 0$ be the optimal median distance.

First, consider the case when $d \leq \frac{2}{\epsilon}$. Then, we can solve the median problem by an exhaustive search as follows:

(I) Arbitrarily pick one of the three given genomes, say $\mathcal{G}_1$, and enumerate all genomes $\mathcal{G}'$ such that $D(\mathcal{G}', \mathcal{G}_1) \leq \frac{2}{\epsilon}$.

The time taken for this can be bounded as follows. First, we need to preprocess $\mathcal{G}_1$ (by repeatedly applying Lemma 6.2 using Lemma 6.1) such that the modified $\mathcal{G}_1$ has k' sets over n' elements with $k' \leq n' \leq 3\frac{2}{\epsilon}$. This preprocessing takes $O(nk)$ time. Now, we generate $\mathcal{G}'$ by applying at most $\frac{2}{\epsilon}$ moves on this preprocessed $\mathcal{G}_1$. ¿From the analysis given in the proof of Theorem 6.1 in Section 6, there are at most $2^{O(4/\epsilon^2)}$ such sequences, hence at most these many $\mathcal{G}'$ will be enumerated. The time to generate each such $\mathcal{G}'$, given the sequence of moves, is $O(n'\frac{2}{\epsilon}) = O(4/\epsilon^2)$.

(II) For each $\mathcal{G}'$ generated in (I), evaluate $\sum_{i=1}^{3} D(\mathcal{G}', \mathcal{G}_i)$, and keep that genome $\mathcal{G}^*$ that gives the minimum sum.

We know that $D(\mathcal{G}', \mathcal{G}_i) \leq \frac{2}{\epsilon}$ for $1 \leq i \leq 3$. Hence, to find $\sum_{i=1}^{3} D(\mathcal{G}', \mathcal{G}_i)$ for each $\mathcal{G}'$, all that we need to do is to start with the genome $\mathcal{G}'$, and find the shortest sequence of moves that transform $\mathcal{G}'$ to $\mathcal{G}_i$ for $1 \leq i \leq 3$, and finally retain the genome that gives the minimum sum. By our analysis in the proof of Theorem 6.1, this can be done in $2^{O(4/\epsilon^2)}$ time.

Thus, if $d \leq \frac{2}{\epsilon}$, the total time taken by our algorithm is:

$$O(nk) + 2^{O(4/\epsilon^2)} \left(O(4/\epsilon^2) + 2^{O(4/\epsilon^2)} \right) = O\left(nk + 2^{O(4/\epsilon^2)} \right)$$

which is polynomial time since $\epsilon > 0$ is a constant.

Next, we need to consider the case when $d > \frac{2}{\epsilon}$. It turns out (as shown below in (1),(2) and (3)) that, for this case, selecting *any* of the 3 given genomes, possibly with a constant number of additional sets, gives a performance ratio of $4+\epsilon$ with any of the constrains (c1), (c2) or (c3). Since we do not know the value of d in advance, the complete approximation algorithm works as follows:

- For the given value of ϵ, enumerate all $\mathcal{G}'$ to find the best $\mathcal{G}^*$ (if any) as describe above. If we find such a $\mathcal{G}^*$ within a median distance of $\frac{2}{\epsilon}$, we select this $\mathcal{G}^*$ as the median. Obviously, we have an optimal solution in this case.
- Otherwise, select any of the three given genomes, possibly with a constant number of additional sets (as described below), as the median. The proof below shows that this approximates the median problem with a performance ratio of $4 + \epsilon$.

Now, we turn our attention to prove that, if $d > \frac{2}{\epsilon}$, selecting any of the 3 given genomes, possibly with a constant number of additional sets, gives a performance ratio of of $4 + \epsilon$ with any of the constrains (c1), (c2) or (c3). Note that Remark 4.1 in Section 4 states that if the optimal syntenic distance between two genomes is d, then $2d - 1$ moves suffice for the approximation algorithm.
(1) First, consider the median problem with constraint (c1). Discard all genes which are not present in all the three genomes[4]. Let $d_1 = D(\mathcal{G}_1, \mathcal{G}_2)$, $d_2 = D(\mathcal{G}_2, \mathcal{G}_3)$ and $d_3 = D(\mathcal{G}_3, \mathcal{G}_1)$, $d = \max\{d_1, d_2, d_3\}$ and $\mathcal{G}$ be any optimal solution of the median problem. Then, $\alpha_{\mathcal{G}} \geq d$. Assume that we have computed approximations d_1', d_2', d_3' of d_1, d_2, d_3, respectively ($d_i' \leq 2d_i - 1$) using the slightly modified approximation algorithm of Section 4 as mentioned in Remark 4.1. This takes $O(nkA^{-1}(nk, k))$ time. We propose any of the given genomes, say $\mathcal{G}_2$, as an approximate median genome and $d_1' + d_3'$ as the approximate value of the median distance. Our approximate median distance is $d_1' + d_2' \leq 4d - 2$, and hence a performance ratio of 4 is achieved.
(2) Next, consider the median problem with constraint (c2). Discard all genes which are present in just one genome. Let $d_1 = D(\mathcal{G}_1, \mathcal{G}_2)$, $d_2 = D(\mathcal{G}_2, \mathcal{G}_3)$ and $d_3 = D(\mathcal{G}_3, \mathcal{G}_1)$, $d = \max\{d_1, d_2, d_3\}$ and $\mathcal{G}$ be any optimal solution of the median problem. Since removal of genes can never increase the synteny distance, $\alpha_{\mathcal{G}} \geq d$. Assume again that we have computed approximations d_1', d_2', d_3' of d_1, d_2, d_3, respectively ($d_i' \leq 2d_i - 1$), in $O(nkA^{-1}(nk, k))$ time. We select any of three given genomes, say $\mathcal{G}_2$, as an approximate median. The problem is that any solution of the constrained median problem may have to contain genes which are not present in $\mathcal{G}_2$. The following are the various cases:

- Genes which are present in all the three genomes. These genes will be used in computation of all the syntenic distances and hence pose no problems.
- Genes which are present in $\mathcal{G}_2$ and one of $\mathcal{G}_1$ or $\mathcal{G}_3$. Following the heuristic presented in Section 4, it is not difficult to see these genes also pose no problems (they will be ignored in the computation of one syntenic distance, but since the heuristic is a simple fusion-fission heuristic, they can always be included and placed in appropriate sets without increasing the syntenic distance).

[4]It is easy to see that removal of genes can never increase the synteny distance

- Genes which are present in $\mathcal{G}_1$ and $\mathcal{G}_3$ but not in $\mathcal{G}_2$. Let X be these sets of genes. Since the heuristic presented in Section 4 is a simple fusion-fission heuristic and since in the compact representation of $\mathcal{G}_2$, X is equivalent to a singleton set, simply including an additional set X in $\mathcal{G}_2$ will increase each of d'_1 and d'_2 by at most one.

Hence, we can construct an approximate median $\mathcal{G}'_2$ with an approximate median distance of at most $(d'_1 + 1) + (d'_2 + 1) \leq 4d$, thus achieving a performance ratio of 4.
(3) Finally, consider the median problem with constraint (c3). We use essentially the same approach as in (2) to select any of the three given genomes, say genome $\mathcal{G}_2$, as our approximate median. We have already described in (2) above how to take care of elements which appear in just two of the three genomes. Hence, we only need to describe how to include genes which appear in exactly one of the genomes $\mathcal{G}_1$ and $\mathcal{G}_3$. Let X be the set of genes which appear only in $\mathcal{G}_1$. Then, we include the set X in our solution. It is easy to see that the simple fusion-fission heuristic of Section 4 will need one additional fission to transform $\mathcal{G}_1$ to $\mathcal{G}_2$. We do the same thing about genes which appear only in $\mathcal{G}_3$. In all, the total approximate median distance of our solution is at most $4d + 2$, which gives a performance ratio of $4 + \frac{2}{d} \leq 4 + \epsilon$. Again, we take $O(nkA^{-1}(nk, k))$ time. ■

Remark 7.1 *[1] After this paper was submitted, Andris Ambainis pointed out to the first author the following improvement in the analysis of Lemma 7.2 that will give an improved performance ratio of $\frac{8}{3} + \epsilon$. Let $d > \frac{2}{\epsilon}$, and for simplicity, consider the median problem with constraint (c1). We use the same notations as in the proof. Compute the approximate distances d'_1, d'_2, d'_3 (approximating the true distances d_1, d_2, d_3, respectively, with performance ratio 2) and select the genome which has the smallest sum of the approximate distances to the two other genomes. Let $\mathcal{G}$ be the true median and a_1, a_2, a_3 be $D(\mathcal{G}, \mathcal{G}_1), D(\mathcal{G}, \mathcal{G}_2), D(\mathcal{G}, \mathcal{G}_3)$, respectively. Assume, without loss of generality, that $a_1 = \min\{a_1, a_2, a_3\}$. Hence, $a_1 \leq \frac{a_1+a_2+a_3}{3}$. Then,*

$$
\begin{array}{lll}
 & \textit{Our approximate median distance} & \\
\leq & d'_1 + d'_3 & \\
\leq & 2(d_1 + d_3) - 2 & \\
\leq & 2(a_1 + a_2) + 2(a_1 + a_3) - 2 & \textit{(by triangle inequality)} \\
\leq & 2(a_1 + a_2 + a_3) + 2a_1 - 2 & \\
\leq & \frac{8}{3}(a_1 + a_2 + a_3) - 2 & \textit{since } a_1 \leq \frac{a_1+a_2+a_3}{3}
\end{array}
$$

A similar argument (together with our inclusion of a few additional sets) gives the same performance ratio for the constraints (c2) or (c3) also.

8 Conclusion

In this paper, we have proved several results concerning the complexities of efficient exact and approximate computations of the syntenic distance between genomes. These results are mainly theoretical. However, the (relatively easy) approximation algorithms for the synteny problem and the median problem are also important as constructive results from the point of view of a practitioner. The following problems still remain open:

- Can we approximate the synteny distance in polynomial time with a ratio better than 2? Does a PTAS for this problem exist? One possible direction of attacking these problems could be to improve the lower bound of Proposition 2.4, especially for small values of p.

- When the synteny distance is bounded, can we improve the time complexity further to compute an optimal move sequence?

9 Acknowledgements

We would like to thank the anonymous reviewers for their detailed and extremely helpful comments that greatly assisted us in revising the paper.

References

[1] A. Ambainis, personal email communication.

[2] V. Bafna and P. Pevzner. Genome Rearrangements and Sorting by Reversals. In *34th IEEE Symp. on Foundations of Computer Science*, 1993, pp. 148-157.

[3] V. Bafna and P. Pevzner. Sorting by Reversals: Genome Rearrangements in Plant Organelles and Evolutionary History of X Chromosome. *Mol. Biol. and Evol.*, 12, 1995, pp. 239-246.

[4] V. Bafna and P. Pevzner. Sorting by Transpositions. In *Proc. of 6th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 1995, pp. 614-623.

[5] T. H. Cormen, C. E. Leiserson and R. L. Rivest. Introduction to Algorithms, The MIT Press, 1990.

[6] V. Ferretti, J.H. Nadeau and D. Sankoff. Original Synteny. In *Proc. of 7th Ann. Symp. on Combinatorial Pattern Matching,* 1996, pp. 159–167.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[8] S. Hannenhalli and P. Pevzner. Transforming Cabbage into Turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. of 27th Ann. ACM Symp. on Theory of Computing*, 1995, pp. 178-189.

[9] S. Kannan and Z Sweedyk. A Separator Theorem for Directed Acyclic Graphs. University of Pennsylvania Tehnical Report MS-CIS-96-20.

[10] J. Kececioglu and R. Ravi. Of Mice and Men: Evolutionary Distances Between Genomes under Translocation. In *Proc. of 6th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 1995, pp. 604-613.

[11] J. Kececioglu and D. Sankoff. Exact and Approximation Algorithms for the Inversion Distance between Two Permutations. In *Proc. of 4th Ann. Symp. on Combinatorial Pattern Matching,* Lecture Notes in Computer Science 684, Springer Verlag, 1993, pp. 87-105.

[12] J. Kececioglu and D. Sankoff. Efficient Bounds for Oriented Chromosome Inversion Distance. In In *Proc. of 5th Ann. Symp. on Combinatorial Pattern Matching,* Lecture Notes in Computer Science 807, Springer Verlag, 1994, pp. 307-325.

[13] J. Kececioglu and D. Sankoff. Exact and Approximation Algorithms for Sorting by Reversals, With Applications to Genome Rearrangements. *Algorithmica*, 13:1/2, 1995, pp. 180-210.

[14] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1982.