On the Rank-Distance Median of 3 Permutations

Leonid Chindelevitch[1] and João Meidanis[2]

[1] Simon Fraser University, Burnaby, Canada, leonid@sfu.ca
[2] University of Ottawa, Ottawa, Canada, on leave from
University of Campinas, Campinas, Brazil, meidanis@ic.unicamp.br

Abstract. Recently, Pereira Zanetti, Biller and Meidanis [9] have proposed a new definition of a distance for which the computational complexity of the median problem is currently unknown. In this formulation, each genome is represented as a permutation on n elements that is the product of disjoint cycles of length 1 (*telomeres*) and length 2 (*adjacencies*). The permutations are converted into their matrix representation, and the *rank distance* d is used to define the median.
In their paper, the authors provide an $O(n^3)$ algorithm for determining three candidate medians, prove the tight approximation ratio $\frac{4}{3}$, and provide a sufficient condition for their candidates to be true medians. They also conduct some experiments that suggest that their method is accurate on simulated and real data.
In this paper, we extend their results and provide the following:

- 2 invariants characterizing the problem of finding the median of 3 matrices
- a strengthening of one of the results in the work of Pereira Zanetti et al.
- a sufficient condition for optimality that can be checked in $O(n)$ time
- a faster, $O(n^2)$ algorithm for determining the median under this condition
- a new heuristic algorithm for this problem based on compressed sensing

1 Introduction

The genome median problem asks, given as input three genomes G_1, G_2, G_3 and a distance (metric) d, to find the *median* genome G minimizing $\sum_{i=1}^{3} d(G, G_i)$. This problem has been extensively studied due to its many applications in phylogenetics and ancestral genome reconstruction [1–3], and is NP-hard for all but a few known distances d [4–7]. Many approximation algorithms have been developed for this problem [4, 8], and some heuristic approaches have also been successfully applied to this problem [3].

Recently, Pereira Zanetti, Biller and Meidanis [9] have proposed a new definition of a distance for which the computational complexity of the median problem is currently unknown. In this formulation, each genome is represented as a permutation on n elements that is the product of disjoint cycles of length

1 (*telomeres*) and length 2 (*adjacencies*). The permutations are converted into their matrix representation, and the *rank distance* d is used to define the median.

In their paper, the authors provide an $O(n^3)$ algorithm for determining three candidate medians, prove the tight approximation ratio $\frac{4}{3}$, and provide a sufficient condition for their candidates to be true medians. They also conduct some experiments that suggest that their method is accurate on simulated data.

In this paper, we extend their results and provide the following:

- 2 invariants characterizing the problem of finding the median of 3 matrices
- a strengthening of one of the results in the work of Pereira Zanetti et al.
- a sufficient condition for optimality that can be checked in $O(n)$ time
- a faster, $O(n^2)$ algorithm for determining the median under this condition
- a new heuristic algorithm for this problem based on compressed sensing

Our results suggest that the problem is easy in a certain regime (determined by the values of the invariants). However, on the negative side, we show that the fraction of random instances in the favorable regime tends to 0 as the genome size grows. While making substantial progress towards a solution, our work therefore leaves open the main problem of determining the complexity of the median of 3 genomic permutation matrices with respect to the rank distance.

2 Definitions and Invariants

Let $\mathbb{F}$ be a field (here, we use $\mathbb{F} = \mathbb{R}$, the field of real numbers, although the problem can also be posed in finite fields). Let $n \in \mathbb{N}$ be an integer and let $\mathbb{F}^{n \times n}$ be the ring of $n \times n$ matrices with entries in $\mathbb{F}$.

2.1 Genomic Matrices

Following [9], we say that a matrix $M \in \mathbb{F}^{n \times n}$ is *genomic* if the following conditions hold:

$$M_{i,j} \in \{0,1\} \; \forall\, 1 \leq i,j \leq n; \; M^T = M^{-1} = M.$$

Note that, unlike [9], we do not require n to be even for M to be genomic.

The first of the conditions above means that M is a $0-1$ matrix, and the condition $M^T = M^{-1}$ means that M is an *orthogonal* matrix; together, they imply that M is a *permutation* matrix, which has a single 1 in each row and each column. It therefore defines a permutation π via the relationship

$$\pi(i) = j \iff M_{i,j} = 1.$$

The condition $M^{-1} = M$ ensures that $M^2 = I$; in this case, the corresponding permutation π is also its own inverse, so it must have order 2 and hence is a product of disjoint cycles of length 1 and 2. The cycles of length 1 correspond to *telomeres* while the cycles of length 2 correspond to *adjacencies*. The correspondence between a genome G and a genomic matrix M is defined by

$$M_{i,j} = 1 \iff (i,j) \text{ is an adjacency in } G, \text{ or } i = j \text{ and } i \text{ is a telomere in } G.$$

There is also a graphical way of depicting one or more genomes. When we use the rank distance, we assume all genomes have the same genes, and the rows and columns of the matrices correspond in fact to the extremities of these common genes [9]. Consider a graph on n vertices, each vertex associated to one of these gene extremities in a one-to-one fashion. Then we can draw the adjacencies of genomes as edges in this graph, using a different color for each genome. If we draw just one genome, we have a *matching* on this graph, and vice-versa, that is, each matching defines a unique genome. Telomeres correspond to unsaturated vertices in this matching. If we draw two genomes, since each one is a matching, we end up with a collection of paths and cycles. We can draw any number of genomes in this way. We call such a graph a *multi-genome breakpoint graph*, in analogy to the breakpoint graph used to study pairs of genomes. Notice, however, that our multi-genome breakpoint graphs do not contain caps to close linear chromosomes. Our representation is cap-free.

2.2 Rank Distance

The rank distance $d(\cdot,\cdot)$ [10] is defined on $\mathbb{F}^{n\times n}$ via

$$d(A,B) = r(A-B),$$

where $r(X)$ is the rank of the matrix X. The fact that it is a metric follows from the following lemma, which also establishes necessary conditions for equality in the triangle inequality that we are going to use later on.

Lemma 2.1. *The rank distance d is a metric. If $d(A,B)+d(B,C)=d(A,C)$, then there exists a projection matrix P such that $B = A + P(C-A)$.*

Proof. Clearly, $r(X) \geq 0 \; \forall \, X$, and $r(A-B) = 0 \iff A-B=0 \iff A=B$.
The symmetry follows from $d(A,B) = r(A-B) = r(B-A) = d(B,A)$.
The triangle inequality follows from the fact that the rank of X is the vector space dimension of its *image* $\mathrm{Im}(X) := \{Xv | v \in \mathbb{F}^n\}$, so it suffices to show that

$$\dim(\mathrm{Im}(X)) + \dim(\mathrm{Im}(Y)) \geq \dim(\mathrm{Im}(X+Y))$$

and substitute $X = B-A$ and $Y = C-B$ to conclude that

$$d(A,B) + d(B,C) \geq d(A,C) \; \forall \, A,B,C \in \mathbb{F}^{n\times n}.$$

Now, it is easy to see that

$$\begin{aligned}\mathrm{Im}(X+Y) &= \{(X+Y)v | v \in \mathbb{F}^n\} = \{(Xv + Yv | v \in \mathbb{F}^n\} \\ &\subseteq \{Xv | v \in \mathbb{F}^n\} + \{Yv | v \in \mathbb{F}^n\} = \mathrm{Im}(X) + \mathrm{Im}(Y),\end{aligned}$$

where the addition in the second line is the addition of vector spaces. Therefore,

$$\begin{aligned}&r(X+Y) = \dim(\mathrm{Im}(X+Y)) \leq \dim(\mathrm{Im}(X) + \mathrm{Im}(Y)) = -\dim(\mathrm{Im}(X) \cap \mathrm{Im}(Y)) \\ &+ \dim(\mathrm{Im}(X)) + \dim(\mathrm{Im}(Y)) \leq \dim(\mathrm{Im}(X)) + \dim(\mathrm{Im}(Y)) = r(X) + r(Y),\end{aligned}$$

where the equality follows from properties of vector space addition [13]. In particular, equality happens if and only if both of the following conditions hold:

$$\dim(\operatorname{Im}(X+Y)) = \dim(\operatorname{Im}(X)+\operatorname{Im}(Y)) \iff \operatorname{Im}(X+Y) = \operatorname{Im}(X)+\operatorname{Im}(Y);$$
$$\dim(\operatorname{Im}(X)\cap\operatorname{Im}(Y)) = 0 \iff \operatorname{Im}(X)\cap\operatorname{Im}(Y) = \{0\}.$$

These two conditions are equivalent to the vector space $\operatorname{Im}(X+Y)$ being the *direct sum* of $\operatorname{Im}(X)$ and $\operatorname{Im}(Y)$, written as $\operatorname{Im}(X+Y) = \operatorname{Im}(X)\oplus\operatorname{Im}(Y)$.

To finish the proof, let us pick a basis $\mathbb{B}_X$ for $\operatorname{Im}(X)$ and a basis $\mathbb{B}_Y$ for $\operatorname{Im}(Y)$. From the directness of the sum, it follows that every vector $u \in \operatorname{Im}(X+Y)$ can be uniquely written as $u = v + w$, with $v \in \operatorname{Im}(X)$ and $w \in \operatorname{Im}(Y)$. Since this is true for any vector in $\operatorname{Im}(X+Y)$, this is in particular true for the n columns of $X+Y$, say $c_1,\dots,c_n$.

But we also have

$$c_i = x_i + y_i \;\forall\, 1 \le i \le n, \tag{1}$$

where x_i (respectively y_i) is the i-th column of X (respectively Y). Therefore, since $x_i \in \operatorname{Im}(X)$ and $y_i \in \operatorname{Im}(Y)$, equation (1) gives the desired decomposition for each c_i. Let $\mathbb{B} := \mathbb{B}_X \cup \mathbb{B}_Y$ be the basis of $\operatorname{Im}(X+Y)$ obtained by joining the bases $\mathbb{B}_X$ and $\mathbb{B}_Y$, and let $\mathbb{B}^+ := \mathbb{B}\cup\mathbb{B}'$ be an extension of the basis $\mathbb{B}$ to a basis of the entire space $\mathbb{F}^n$. Let P be the matrix of the projection operator onto X with respect to the basis $\mathbb{B}^+$, with all the elements in $\mathbb{B}' = \mathbb{B}^+ - \mathbb{B}$ being projected onto the 0 vector. Then

$$Pc_i = x_i \;\forall\, 1 \le i \le n,$$

so that $P(X+Y) = X$. By substituting $X = B - A$ and $Y = C - B$, we see that the desired equality is only possible if $B = A + P(C-A)$, as required.

We also state a helpful

Corollary 2.1. *If $x \in \mathbb{F}^n$ is a vector such that $Ax = Cx$ and $d(A,B) + d(B,C) = d(A,C)$, then $Bx = Ax$ as well. In particular, if A and C have all row sums equal to 1, so does B.*

Proof. The distance condition implies that $B = A + P(C-A)$ for some P, so we can compute $Bx = Ax + P(C-A)x = Ax + P(Cx - Ax) = x$, so $Bx = Ax$ as claimed. The second statement follows from the first one by taking $x = e$, the vector of n 1's, since Ae is the vector containing the row sums of A.

The rank distance is equivalent to the Cayley distance between permutations. We will develop this connection further in Section 3.

2.3 Biological Motivation

Although the rank distance has been known and studied for a long time, its use on genome matrices brings an additional motivation for its study, based on biological grounds. Indeed, the most frequent genome rearrangements occurring in genome evolution, such as inversions, transpositions, translocations, fissions, and fusions, correspond to genomes with very low rank distances. This suggests that the rank distance may be a good indicator of the amount of evolution that separates two genome matrices. Table 1 shows the rank distance associated to some of the most common genome rearrangement operations. Notice that the term "transposition" has a different meaning in biology than it does in permutation group theory. In this section we are using the biological meaning, but the rest of the paper follows the mathematical meaning.

Operation	Rank distance
Inversion	2
Transposition	4
Translocation	2
Fission	1
Fusion	1

Table 1. Frequent rearrangement operations and rank distance between genomes that differ by the corresponding operation.

2.4 Relationship with DCJ and SCJ

In addition, the rank distance is closely related to other distances that have traditionally been used in genome studies, such as the double-cut-and-join (DCJ) distance [11] and the single-cut-or-join (SCJ) distance [7]. If we use the multi-genome breakpoint graph for two genomes A and B, we can derive formulas for the DCJ, SCJ, and rank distances based on graph elements, as follows.

The SCJ distance is defined as the number of adjacencies that belong to exactly one of the two genomes. Therefore, we can compute the SCJ distance by counting all adjacencies (edges) in the graph and subtracting the number of common adjacencies. A common adjacency will appear in the graph as a 2-cycle, that is, a cycle composed of two parallel edges. The formula for the SCJ distance is then:

$$d_{SCJ}(A,B) = \#\text{edges} - 2C_2,$$

where C_2 is the number of 2-cycles in the graph.

The multi-genome breakpoint graph for A and B is a collection of paths and cycles. It can be shown that each path contributes its number of edges to the rank distance, and each cycle contributes its number of edges minus 2 to the rank distance. Therefore, the rank distance is:

$$d(A,B) = \#\text{edges} - 2C,$$

where C is the number of cycles of any length in the graph. From these equations it is easy to derive the following relationship between the SCJ and rank distances:

$$d(A, B) \leq d_{SCJ}(A, B) \leq 2d(A, B).$$

These inequalities are tight, as witnessed by cases in which the graph has no cycles (for the leftmost inequality), and graphs composed solely of 4-cycles (for the rightmost inequality).

With respect to the DCJ distance, it can be shown that

$$d_{DCJ}(A, B) = \frac{1}{2}\#\text{edges} - C + \frac{1}{2}P_{odd},$$

where P_{odd} is the number of paths of odd length (number of edges) in the graph. From these equations it is easy to derive the following relationship between the DCJ and rank distances:

$$d(A, B) \leq 2d_{DCJ}(A, B) \leq 2d(A, B).$$

These inequalities are tight as well, as witnessed by graphs with no paths (leftmost inequality), and by graphs composed solely of paths of length 1 (rightmost inequality).

2.5 Median of 3 Matrices

Given three matrices A, B, C, the median M is defined as a global minimizer of the *score* function

$$d(M; A, B, C) := d(A, M) + d(B, M) + d(C, M). \tag{2}$$

Since d is a metric, we can use symmetry and the triangle inequality to see that the score has a simple lower bound:

$$\begin{aligned} d(M; A, B, C) &= \frac{d(A, M) + d(M, B)}{2} + \frac{d(B, M) + d(M, C)}{2} + \frac{d(C, M) + d(M, A)}{2} \\ &\geq \frac{d(A, B)}{2} + \frac{d(B, C)}{2} + \frac{d(C, A)}{2} = \frac{1}{2}[d(A, B) + d(B, C) + d(C, A)], \end{aligned}$$

with equality if and only if

$$d(X, M) + d(M, Y) = d(X, Y) \text{ for any distinct } X, Y \in \{A, B, C\}. \tag{3}$$

2.6 The First Invariant

We now define the first invariant of the median-of-three problem via

$$\beta(A, B, C) := \frac{1}{2}[d(A, B) + d(B, C) + d(C, A)].$$

It is easy to see that this is indeed an invariant in the sense that it does not change under permuting of the three matrices, or permuting the rows or the columns of all the matrices in the same way. Namely,

$$\beta(A,B,C)=\beta(A,C,B)=\beta(B,C,A)=\beta(B,A,C)=\beta(C,B,A)=\beta(C,A,B),$$

and, for any $n \times n$ permutation matrices P and Q,

$$\beta(A,B,C)=\beta(PAQ^T,PBQ^T,PCQ^T).$$

The fact that d is a metric allows us to establish a first (trivial) approximation algorithm with an approximation ratio of $\frac{4}{3}$ [1] - namely, pick the matrix among A,B,C with the smallest score. The approximation ratio follows from

$$\min_{X\in\{A,B,C\}} d(X;A,B,C) \le \frac{1}{3}\sum_{X\in\{A,B,C\}} d(X;A,B,C)$$

$$=\frac{1}{3}[d(B,A)+d(C,A)+d(A,B)+d(C,B)+d(A,C)+d(B,C)]=\frac{4}{3}\beta(A,B,C).$$

We note that any matrix with score β is necessarily a median, and that for any matrix M that attains this score, we necessarily have

$$d(A,M)=\beta-d(B,C);\ d(B,M)=\beta-d(A,C);\ d(C,M)=\beta-d(A,B).$$

However, in the general case, it is not possible to attain the lower bound β. For instance, if

$$A=\begin{pmatrix}-1 & 0\\ 0 & -1\end{pmatrix}, B=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}, C=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix},$$

then $d(A,B)=d(B,C)=d(C,A)=2$, so $\beta=3$, but it is easy to check that no matrix is simultaneously at unit rank distance from all three of these matrices, and the minimum score is 4 (attained by, for instance, any diagonal matrix with diagonal entries in $\{-1,0,1\}$).

2.7 The Second Invariant

The second invariant, which was already identified in [9] as playing an important role in the median problem, is

$$\alpha(A,B,C):=\dim(V_1),\ \text{where } V_1:=\{x\in\mathbb{F}^n|Ax=Bx=Cx\}. \tag{4}$$

Once again, it is easy to check that it is invariant under permutations of the three matrices, or permutations of the rows or the columns of all the matrices.

3 Permutation Matrices

Let us now consider the special case of A,B,C being permutation matrices. While, as our example showed, the lower bound $\beta(A,B,C)$ for the score cannot always be attained, there is a possibility that the lower bound can always be attained for permutation matrices, meaning that the equality conditions in equation (3) can always be satisfied. The present paper is motivated by an attempt to establish whether this is indeed the case.

3.1 Integrality of the First Invariant

Let us denote by S_n the group of permutations on n elements. Pereira Zanetti et al [9] have already shown that, for any two permutations σ and τ in S_n, the *transposition distance* $d_T(\sigma, \tau)$, also known as Cayley distance [14] and counting the minimum number of transpositions (switches) needed to transform σ into τ, equals $d(S, T)$, where S and T are the permutation matrices corresponding to σ and τ, respectively, and d is the rank distance.

Let us begin by showing that, if A, B, C are permutation matrices, $\beta(A, B, C)$ is always an integer; this is also not the case in general, as can be seen in the one-dimensional example of three different scalars, for which $\beta(A, B, C) = 3/2$.

To this end, we recall that any permutation $\tau \in S_n$ can be written as a product of disjoint cycles, uniquely up to order of the cycles and order of the elements within the cycle, provided that fixed points are represented by cycles of length 1. We define the *cycle counter* $c(\tau)$ as the number of disjoint cycles in a disjoint cycle representation of τ. For instance, if $\tau = (12)(34)(5)$, then $c(\tau) = 3$.

Lemma 3.1. *If A, B are permutation matrices corresponding to permutations ρ, σ, respectively, then $d(A, B) = n - c(\rho^{-1}\sigma)$.*

Proof. It was already shown in [9] that $d(A, B) = d_T(\rho, \sigma)$, where d_T is the transposition distance. Since d_T is left invariant [14], we have

$$d_T(\rho, \sigma) = d_T(\rho^{-1}\rho, \rho^{-1}\sigma) = d_T(e, \rho^{-1}\sigma).$$

It remains to show that the minimum number of transpositions needed to transform a permutation τ into the identity is $n - c(\tau)$. This follows from the facts that a k-cycle needs exactly $k - 1$ transpositions to transform into the identity, the total length of all the cycles (including the fixed points) is n, and the optimal set of transpositions affects one cycle at a time.

Corollary 3.1. *If A, B are permutation matrices corresponding to permutations ρ, σ, respectively, then the nullspace of $A - B$ is spanned by the indicator vectors of the cycles of $\rho^{-1}\sigma$ (each taking value 1 on the cycle and 0 outside it).*

This corollary, which follows from Lemma 3.1 and the rank-nullity theorem, could also have been deduced directly from the following

Remark 3.1. If the permutation matrix A corresponds to ρ, $Ax = [x_{\rho(1)}, \ldots, x_{\rho(n)}]^T$.

Lemma 3.2. *If A, B, C are permutation matrices, $\beta(A, B, C)$ is an integer.*

Proof. Let ρ, σ, τ be the permutations corresponding to A, B, C, respectively. From the foregoing discussion, $d_T(\rho, \sigma)$ is the smallest number of transpositions needed to transform ρ into σ. Note that transforming ρ into σ and then σ into τ also transforms σ into τ. In addition, it is known that the number of transpositions needed to transform one permutation into another has a fixed parity that only depends on the *signs* of the permutations [14]. Therefore

$$\begin{aligned} d(A, B) + d(B, C) + d(C, A) = d_T(\rho, \sigma) + d_T(\sigma, \tau) + d_T(\rho, \tau) \equiv \\ \equiv d_T(\rho, \sigma) + d_T(\sigma, \tau) + [d_T(\rho, \sigma) + d_T(\sigma, \tau)] \mod 2 \equiv 0 \mod 2, \end{aligned}$$

so that $\beta(A, B, C)$ is indeed an integer.

An alternative proof can be obtained by noting that $(-1)^{d_T(\rho,\sigma)} \equiv \det(A^{-1}B)$ mod 2, where A, B are the permutation matrices for ρ, σ respectively; therefore

$$\begin{aligned}
&(-1)^{d_T(\rho,\sigma)+d_T(\sigma,\tau)+d_T(\tau,\rho)} \equiv \det(A^{-1}B)\det(B^{-1}C)\det(C^{-1}A) = \\
&= \det((A^{-1}B)(B^{-1}C)(C^{-1}A)) = \det(A^{-1}(BB^{-1})(CC^{-1})A) = \\
&= \det(I) = 1 \mod 2 \implies d_T(\rho,\sigma) + d_T(\sigma,\tau) + d_T(\tau,\rho) \equiv 0 \mod 2.
\end{aligned}$$

3.2 Fast Computation of the Invariants

We now show how to compute the invariants α and β in $O(n)$ time given the three permutations ρ, σ, τ represented by A, B, C, respectively. For β, it suffices to use the identity

$$\beta(A, B, C) = \frac{1}{2}\big(3n - [c(\rho^{-1}\sigma) + c(\sigma^{-1}\tau) + c(\tau^{-1}\rho)]\big),$$

obtained using Lemma 3.1 and the definition of β. Permutations can be multiplied and inverted in $O(n)$ time using standard algorithms, and their cycles can be counted in $O(n)$ by using their graph representation (with a directed edge from i to $\pi(i)$ for every $1 \leq i \leq n$) and identifying the weakly connected components. Therefore, the computation of $\beta(A, B, C)$ takes $O(n)$ time overall.

For α, we note that, by Corollary 3.1,

$$\begin{aligned}
x \in V_1 \iff Ax = Bx = Cx \iff Ax = Bx \text{ and } Bx = Cx \iff \\
\iff x \text{ is constant on the cycles of } \rho^{-1}\sigma \text{ and on the cycles of } \sigma^{-1}\tau.
\end{aligned}$$

The computation of $\rho^{-1}\sigma$ and $\sigma^{-1}\tau$ is the same as the one performed for computing β, and the dimension of V_1 is then equal to the number of weakly connected components of the union of their graph representations described above.

Indeed, if C_1, C_2 are 2 disjoint weakly connected components of the graph representation of $\rho^{-1}\sigma$ and there is an edge between C_1 and C_2 in the graph representation of $\sigma^{-1}\tau$, then any vector $x \in V_1$ must be constant on $C_1 \cup C_2$. By iterating this reasoning, we conclude that α is precisely the number of weakly connected components of the union of the graph representations of $\rho^{-1}\sigma$ and $\sigma^{-1}\tau$. Each graph can be computed in $O(n)$ time, and so can their union and its connected components, so computing α also requires $O(n)$ time overall.

3.3 Subspace Dimensions in Terms of Invariants

Pereira Zanetti et al [9] showed how to decompose the space $\mathbb{F}^n$ into a direct sum of five subspaces, and expressed their median candidates using the projection operators of these subspaces. We now show how to express the dimensions of these subspaces using the invariants α and β, and deduce a sufficient condition for their median candidate to be a true median. Readers familiar with this paper will recognize our use of the dot notation for partitions introduced there (e.g.,

$.AB.C$). However, all subspaces needed here are defined here as well, so the exact meaning of this notation is not relevant in our context.

The subspace $V_1 = V(.A.B.C.)$ is defined in equation (4), and is the subspace of all vectors on which all three matrices agree. Its dimension is α, by definition.

The subspace $V_2 := V(.AB.C.) \cap V_1^{\perp}$ is defined via V_1 and the subspace

$$V(.AB.C) := \{x \in \mathbb{F}^n | Ax = Bx\}.$$

The dimension of $V(.AB.C)$ is precisely $c(\rho^{-1}\sigma)$, where ρ and σ are the permutations corresponding to A and B, respectively. This follows from Corollary 3.1 which tells us that

$$Ax = Bx \iff A^{-1}Bx = x \iff x \text{ is constant on every cycle of } \rho^{-1}\sigma.$$

Since $V_1 \subseteq V(.AB.C)$, it follows that $V(.AB.C.)^{\perp} \subseteq V_1^{\perp}$ and

$$\begin{aligned} \dim(V_2) = \dim(V(.AB.C.) \cap V_1^{\perp}) = \dim((V(.AB.C.)^{\perp} + V_1)^{\perp}) = \\ = n - \dim((V(.AB.C.)^{\perp} + V_1) = n - \dim(V(.AB.C.)^{\perp}) - \dim(V_1) = \\ = n - (n - c(\rho^{-1}\sigma)) - \alpha = c(\rho^{-1}\sigma) - \alpha, \end{aligned}$$

where the dimension of the sum of vector subspaces in the second line splits into the sum of the individual dimensions due to their trivial intersection.

We can apply a similar reasoning to the subspaces $V_3 := V(.A.BC.) \cap V_1^{\perp}$ and $V_4 := V(.AC.B) \cap V_1^{\perp}$ to get

$$\dim(V_2) = c(\rho^{-1}\sigma) - \alpha;\ \dim(V_3) = c(\sigma^{-1}\tau) - \alpha;\ \dim(V_4) = c(\tau^{-1}\rho) - \alpha.$$

Since V_5 is the last term in the direct sum decomposition of $\mathbb{F}^n$, we get that

$$\begin{aligned} \dim(V_5) = n - \sum_{i=1}^{4} \dim(V_i) = n + 2\alpha - (c(\rho^{-1}\sigma) + c(\sigma^{-1}\tau) + c(\tau^{-1}\rho)) = \\ = n + 2\alpha(A,B,C) - (3n - 2\beta(A,B,C)) = 2(\alpha(A,B,C) + \beta(A,B,C) - n). \end{aligned}$$

From this, we immediately deduce the following

Corollary 3.2. *$\alpha(A,B,C) + \beta(A,B,C) \geq n$ for permutation matrices A, B, C, with equality if and only if $V_5 = \{0\}$.*

In addition, we can now combine this with the expression in [9] for the score of their median candidates M_A, M_B, M_C to deduce that

$$\begin{aligned} d(M_A; A, B, C) = 2\dim(V_5) + \dim(V_2) + \dim(V_3) + \dim(V_4) = n - \dim(V_1) + \\ + \dim(V_5) = n - \alpha + 2(\alpha + \beta - n) = \beta + (\alpha + \beta - n) = \beta + \frac{1}{2}\dim(V_5). \end{aligned}$$

As expected, the median M_A achieves the lower bound if and only if $V_5 = \{0\}$.

3.4 A New Algorithm

Next we introduce a new algorithm that can be used to identify another candidate median, that in general differs from the candidates M_A, M_B, M_C identified in previous work [9]. Although we are not able to prove any approximation ratio on its performance, it does allow us to establish the uniqueness of the median in the special case of equality in Corollary 3.2, and gain insight into why this case is special, in addition to obtaining a faster $O(n^2)$ running time for it.

To this end, let us consider the necessary conditions from Lemma 2.1 that must be satisfied in order for the matrix M to attain the lower bound β:

$$M = A + S(B - A) = B + T(C - B) = C + U(A - C), \tag{5}$$

where S, T, U are some projection matrices. This triple equality follows from the facts that equation (3) must be satisfied for each pair among A, B and C.

Let us count the independent equations and non-redundant unknowns in this system. We note that it suffices to consider the equivalent system

$$\begin{aligned} A + S(B - A) = B + T(C - B) &\iff A - B = T(C - B) - S(B - A); \\ B + T(C - B) = C + U(A - C) &\iff B - C = U(A - C) - T(C - B), \end{aligned} \tag{6}$$

since the third equality automatically follows from the first two. Furthermore, we will not enforce the condition of S, T, U being projection matrices, since a projection matrix is defined by $P^2 = P$ and this results in a set of conditions quadratic in the entries of P.

Consider the effect of multiplying a matrix S by a permutation matrix A corresponding to the permutation ρ. It is easy to see that this results in permuting the columns of S according to ρ, so that, denoting by s_i the i-th column of S, we get $SA = [s_{\rho(1)} s_{\rho(2)} \dots s_{\rho(n)}]$. Therefore, the i-th column of $S(B - A)$ will simply be the difference between $s_{\rho(i)}$ and $s_{\sigma(i)}$. For each cycle C of $\rho^{-1}\sigma$, the corresponding columns of $S(B - A)$ will add up to 0.

Thus, changing variables to the "difference variables" of the form

$$s'_i := s_{\rho(C[i])} - s_{\sigma(C[i])}, \tag{7}$$

where $C[i]$ denotes the i-th element in the cycle C, we can see that $S(B-A)$ will have precisely $n - c(\rho^{-1}\sigma)$ linearly independent columns, and one column per cycle will be linearly dependent on the others in the same cycle. By applying the same argument to $T(C - B)$ and $U(A - C)$, we get a grand total of

$$n - c(\rho^{-1}\sigma) + n - c(\sigma^{-1}\tau) + n - c(\tau^{-1}\rho) = 2\beta(A, B, C)$$

linearly independent (non-redundant) columns, each containing n free variables.

We note that the system of equations (6), rewritten with respect to the non-redundant difference variables, splits into n independent linear systems, one per row, with identical left-hand sides and only differing by their right-hand sides:

$$\sum_{i=1}^{d(A,B)} p_i s_i + \sum_{i=1}^{d(B,C)} q_i t_i = a_{ji} - b_{ji} \; ; \; \sum_{i=1}^{d(C,A)} r_i u_i - \sum_{i=1}^{d(B,C)} q_i t_i = b_{ji} - c_{ji} \; , \tag{8}$$

where the p_i, q_i, r_i are coefficients that only depend on the column (variable) index i, but not on the row j.

Next we count the linearly independent equations within each system. In the first half of the system (with right-hand sides coming from the j-th row of $A - B$), linear dependence between the left-hand sides of the equations can only arise from vectors y such that

$$T(C - B)y = 0 = S(B - A)y \ \forall \ S, T,$$

since S and T represent the variables in the system and those variables are distinct. Similarly, in the second half of the system, they can only arise from vectors y such that

$$U(A - C)y = 0 = T(C - B)y \ \forall \ T, U,$$

since T and U represent the variables in the system and those variables are distinct. But then, for any such vector y it must be the case that

$$Ay = By = Cy \iff y \in V_1,$$

meaning that there are exactly $\alpha(A, B, C)$ dependence relationships between the left-hand sides in each of the half-systems since that is the dimension of the susbspace V_1.

Furthermore, all such dependence relationships lead to the tautology $0 = 0$ rather than the contradiction $0 = 1$, because every row of $A - B$ and $B - C$ is orthogonal to any $y \in V_1$, by definition of V_1. Lastly, the two half-systems are linearly independent from one another since the condition on their linear dependence is subsumed by the condition of the linear dependence within the half-systems; more precisely, since the t-variables in the first half-system appear with coefficients that are exactly the negative of the coefficients in the second half-system, a linear dependence relationship between the half-systems would have to arise from a vector y such that

$$U(A - C)y = 0 = S(B - A)y \ \forall \ S, U \iff Ay = By = Cy \iff y \in V_1.$$

It follows that in after eliminating the redundancies in each half-system, exactly β variables and $n - \alpha$ equations remain. Since $\alpha + \beta \geq n$ the system is always under-constrained except in the special case of $\alpha + \beta = n$, in which it has a unique solution (since the remaining equations are linearly independent).

In the special case when $\alpha + \beta = n$, we can furthermore choose to eliminate precisely those redundant equations that contain the "composite" difference variables of the form $-s_1 - s_2 - \cdots - s_k$, corresponding to a cycle of length $k + 1$ in the appropriate permutation. In this way, the remaining equations will only have two active variables (with non-zero coefficients) each, so the algorithm by Aspvall and Shiloach [15] can be applied to solve the resulting system in $O(n)$ time. It follows that the algorithm will only require $O(n^2)$ time to find the median in the special case.

Although the system is under-constrained outside of the special case, we can use ideas from the field of *compressed sensing* [16] to find a solution that is likely to be sparse, and hence hopefully low rank. Namely, we seek the solution containing as many zeros as possible. While this is a hard problem in general, many instances are solvable by using the L_1 norm minimization, which can be achieved by using linear programming. The appropriate linear program becomes

$$\min \sum_i y_i \text{ subject to } -y_i \leq x_i \leq y_i \;\forall\, i \text{ and } Ux = b,$$

where $Ux = b$ is the original system of equations, and the y_i serve as the absolute values of the x_i whose sum is to be minimized. Such linear programs can be readily solved using existing off-the-shelf solvers such as lpsolve [17] or CPLEX [18].

3.5 An Example of the Algorithm

To clarify the procedure, we now illustrate the running of our algorithm on $\rho = (12)(34), \sigma = (13)(24), \tau = (14)(23)$, with $n = 4$. First, we note that the product of any two of these equals the third, and they are each their own inverse. Hence we can compute

$$\beta(A,B,C) = \frac{1}{2}\big(3n - [c(\rho^{-1}\sigma) + c(\sigma^{-1}\tau) + c(\tau^{-1}\rho)]\big) = \frac{1}{2}[12 - 3 \cdot 2] = 3$$

and

$$\alpha(A,B,C) = 1$$

since the union of the graphs of $\rho^{-1}\sigma = \tau$ and $\sigma^{-1}\tau = \rho$ has a unique weakly connected component. Therefore, $\alpha(A,B,C) + \beta(A,B,C) = n$ and, as we will prove below, the algorithm will in fact produce the unique median of A, B, C.

We start by forming the equations in system (6):

$$\begin{aligned} A - B &= [t_4 - t_3, t_3 - t_4, t_2 - t_1, t_1 - t_2] \qquad -[s_3 - s_2, s_4 - s_1, s_1 - s_4, s_2 - s_3] \\ B - C &= [u_2 - u_4, u_1 - u_3, u_4 - u_2, u_3 - u_1] \qquad -[t_4 - t_1, t_3 - t_4, t_2 - t_1, t_1 - t_2]. \end{aligned}$$

We now define the "difference variables"

$$s'_1 := s_3 - s_2; s'_2 := s_4 - s_1; t'_1 := t_4 - t_3; t'_2 := t_2 - t_1; u'_1 := u_2 - u_4; u'_2 := u_1 - u_3,$$

and express our system in terms of those:

$$\begin{aligned} A - B &= [t'_1, -t'_1, t'_2, -t'_2] - [s'_1, s'_2, -s'_2, -s'_1] \\ B - C &= [u'_1, u'_2, -u'_1, -u'_2] - [t'_1, -t'_1, t'_2, -t'_2]. \end{aligned}$$

For convenience we will use the same name (without row superscripts) for the variables in each row, to emphasize that the system of equations for each row has the same left-hand side. For row j it has $2n = 8$ equations that read:

$$t'_1 - s'_1 = A_{j1} - B_{j1} \tag{9.1}$$
$$-t'_1 - s'_2 = A_{j2} - B_{j2} \tag{9.2}$$
$$t'_2 + s'_2 = A_{j3} - B_{j3} \tag{9.3}$$
$$-t'_2 + s'_1 = A_{j4} - B_{j4} \tag{9.4}$$
$$u'_1 - t'_1 = B_{j1} - C_{j1} \tag{9.5}$$
$$u'_2 + t'_1 = B_{j2} - C_{j2} \tag{9.6}$$
$$-u'_1 - t'_2 = B_{j3} - C_{j3} \tag{9.7}$$
$$-u'_2 + t'_2 = B_{j4} - C_{j4} \tag{9.8}$$

As expected from our counting argument above, the only linear dependencies here are that the first 4 equations add up to 0 and the second 4 equations add up to 0 (both their left-hand sides and their right-hand sides), so after eliminating, say, equations (9.4) and (9.8), we end up with a consistent and non-redundant system of $2(n - \alpha) = 6$ equations in $2\beta = 6$ unknowns, which therefore has a unique solution. We illustrate this system for the first row, $j = 1$, since it looks identical for all the other rows except for changes in its right-hand side.

$$t'_1 - s'_1 = 0 \tag{10.1}$$
$$-t'_1 - s'_2 = 1 \tag{10.2}$$
$$t'_2 + s'_2 = -1 \tag{10.3}$$
$$u'_1 - t'_1 = 0 \tag{10.4}$$
$$u'_2 + t'_1 = 0 \tag{10.5}$$
$$-u'_1 - t'_2 = 1 \tag{10.6}$$

Since each equation has exactly two variables, we can use the method of [15] to solve them in $O(n)$ time, which means that the total time to solve the system for all the n rows is $O(n^2)$. In the case of this system, we see that the solution for the first row is

$$s'_1 = -\frac{1}{2}, s'_2 = -\frac{1}{2}, t'_1 = -\frac{1}{2}, t'_2 = -\frac{1}{2}, u'_1 = -\frac{1}{2}, u'_2 = \frac{1}{2}.$$

After solving the system for all the other rows in the same way, we conclude that the unique median of A, B, C in this case is $\frac{1}{2}J - I$, where $J = ee^T$ is the matrix of all 1's and I is the identity matrix. In particular, this example shows that the unique median of three genomic matrices may not itself be genomic.

3.6 Example of the Compressed Sensing Approach

We now consider a different example, one which does not fall into the special case $\alpha + \beta = n$. We take $n = 3$ and $\rho = (12), \sigma = (13), \tau = (23)$. In this case, we have $d(A,B) = d(B,C) = d(C,A) = 2$ so $\beta(A,B,C) = 3$ and $\alpha(A,B,C) = 1$. We know that the system of equations (6) will be under-constrained in this case. We start by forming this system of equations:

$$A - B = [t_1 - t_3, t_3 - t_2, t_2 - t_1] - [s_3 - s_2, s_2 - s_1, s_1 - s_3]$$
$$B - C = [u_2 - u_1 u_1 - u_3 u_3 - u_2] - [t_1 - t_3, t_3 - t_2, t_2 - t_1]$$

and introduce the difference variables

$$s'_1 := s_3 - s_2, s'_2 := s_2 - s_1, t'_1 := t_1 - t_3, t'_2 := t_3 - t_2, u'_1 := u_2 - u_1, u'_2 := u_1 - u_3$$

to rewrite it as 3 systems (one for each row) of the form

$$t'_1 - s'_1 = A_{1j} - B_{1j} \quad (11.1)$$
$$t'_2 - s'_2 = A_{2j} - B_{2j} \quad (11.2)$$
$$-(t'_1 + t'_2) + (s'_1 + s'_2) = A_{3j} - B_{3j} \quad (11.3)$$
$$u'_1 - t'_1 = B_{1j} - C_{1j} \quad (11.4)$$
$$u'_2 - t'_2 = B_{2j} - C_{2j} \quad (11.5)$$
$$-(u'_1 + u'_2) + (t'_1 + t'_2) = B_{3j} - C_{3j} \quad (11.6)$$

As in the previous example, we can eliminate the redundant equations (11.3) and (11.6) from each system, leaving us with a total of 4 equations in 6 variables.

Let us now show the compressed sensing formulation for this system for row $j = 1$. We get the linear program

minimize $y_1 + y_2 + y_3 + y_4 + y_5 + y_6$
subject to
$-[y_1, y_2, y_3, y_4, y_5, y_6] \leq [s'_1, s'_2, t'_1, t'_2, u'_1, u'_2] \leq [y_1, y_2, y_3, y_4, y_5, y_6]$
$t'_1 - s'_1 = 0; \quad t'_2 - s'_2 = 1; \quad u'_1 - t'_1 = -1; \quad u'_2 - t'_2 = 0.$

The optimal solution to this linear program is

$$s'_1 = 0, s'_2 = -1, t'_1 = 0, t'_2 = 0, u'_1 = -1, u'_2 = 0.$$

By repeating this for the other two rows we obtain the solution $M = [0\ 0\ e]$, which unfortunately yields a suboptimal score of 6, whereas the optimal solutions, given by the identity matrix, either of the 3-cycles (123) or (132), or a subset of the affine combinations of those matrices, yield a score of $\beta = 3$. This shows that the compressed sensing approach is not guaranteed to be optimal, or even better than the algorithm that picks the best "corner" option among A, B, C.

3.7 Proof of Uniqueness For the Special Case

We now prove that if $\alpha(A,B,C) + \beta(A,B,C) = n$, then there is a unique median, and both the $O(n^3)$ algorithm by Pereira Zanetti et al [9] as well as our $O(n^2)$ algorithm proposed here correctly identify it.

Theorem 3.1. *Suppose $\alpha(A, B, C) + \beta(A, B, C) = n$. Then there is a unique median minimizing $d(M; A, B, C)$, found by our algorithm and the one in [9].*

Proof. By the calculations in [9] recapitulated in Corollary 3.2, the median M_A achieves the lower bound β. Furthermore, by the calculations in the analysis of the system (6), we see that there exists a unique matrix M that simultaneously satisfies the necessary conditions for attaining the lower bound β. Since M_A attains this lower bound, M_A also satisfies these necessary conditions; by uniqueness, $M_A = M$, so our algorithm also finds a median, and it is unique.

3.8 Rarity of the Special Case

We now use some asymptotic results from analytic combinatorics to show that the probability of three random genomic matrices satisfying the optimality conditions in Corollary 3.2 tends to 0 as n increases. Recall that an *involution* is a permutation that is its own inverse; this is precisely the class of permutations defined by genomic matrices. We begin by restating, without proof, the following result from [19]:

Theorem 3.2. *If σ and τ are random involutions, then the mean number of cycles of $\sigma\tau$ is $\sqrt{n} + \frac{1}{2}\log n + O(1)$. If σ and τ are constrained to be fixed-point free, then the distribution of the number of cycles of $\sigma\tau$ is asymptotically normal with mean $\log n$ and variance $2\log n$.*

Now we can immediately conclude the following

Corollary 3.3. *If A, B, C are the genomic matrices corresponding to random involutions (respectively random involutions with no fixed points, i.e. telomeres), then $\beta \sim \frac{3}{2}(n - \sqrt{n})$ or $\beta \sim \frac{3}{2}(n - \log n)$, respectively. In particular, the probability of these matrices satisfying the optimality conditions in Corollary 3.2 tends to 0 as n increases.*

4 Experimental Results

We tested our algorithm on two datasets - first, a simulated one obtained by applying rearrangement operations to a starting genome, and second, a real one obtained by taking three genomes at a time from a family of plants. In this section, we describe the performance of our algorithm as well as our observations.

4.1 Implementation

For the implementation, we use the R statistical computing language [20] as well as the CPLEX linear programming solver [18], with which we interface via the command line. Specifically, our program first computes the invariants α and β, and then branches into either an exact solution if $\alpha + \beta = n$, or the compressed sensing heuristic if not. In the latter case, R writes the linear program for finding the solution of system (6) of minimum L_1 norm into a file, the CPLEX solver processes this file, and R parses the solution to obtain the median candidate.

We use the igraph package [21] to quickly compute the invariants as well as the cycle decompositions of the permutations involved in the system 6. The cycle decompositions allow us to decide which variables to include in the system 6, and which equations to exclude to make it non-redundant. The resulting system always has $2(n-\alpha)$ equations in 2β variables. In order to try to make the system as sparse as possible even when $\alpha+\beta>n$, we make the variable corresponding to the last (highest) element of each cycle non-basic by expressing it in terms of the others, as per equation (7). Furthermore, we eliminate the equation corresponding to the last (highest) element of each connected component in the union graph defining α; since each such connected component is a disjoint union of cycles (of each of the three permutations $\rho^{-1}\sigma, \sigma^{-1}\tau, \tau^{-1}\rho$), this guarantees that fewer composite variables remain present, leading to a sparser system (6).

In the special case $\alpha+\beta=n$, we use the *solve* function from the *Matrix* package [22], and do not implement the linear-time algorithm by Aspvall and Shiloach [15]. Therefore, strictly speaking our implementation is currently not guaranteed to run in $O(n^2)$ time despite having a sparse coefficient matrix with all non-zero coefficients being ± 1. However, since *solve* is able to take advantage of the sparsity of the system, the special case runs extremely efficiently even for the largest input size, $n=500$.

4.2 Numerical Stability

Computing the score of the median candidates, as defined in equation (2), requires a rank computation, which is known to be numerically challenging [23]. In fact, Zanetti Pereira et al report that despite all their median candidates being expected to have the same score, this is not true in practice for random permutation inputs in about 10% of the cases when using GNU Octave or MATLAB [9].

In order to circumvent this challenge we adopt several measures. First, we use the combinatorial expression from Corollary 3.2 for the score of the median candidates proposed in [9] in all our comparisons, which does not require any rank computation, only a graph-based analysis of the underlying permutations. Second, to score the candidate median produced by the algorithm presented here, we use the *rankMatrix* function with the *QR* decomposition method from the *Matrix* package [22], with a tolerance of $\epsilon=10^{-12}$. Lastly, we round any entry of the median that happens to be within ϵ of an integer (0 or ± 1) to this integer. While we cannot be completely sure that this bypasses all numerical stability issues, we observe that in all instances with $\alpha+\beta=n$, on which our algorithm should produce a median achieving the lower bound β, this is indeed the case.

4.3 Simulated Dataset

The simulated dataset consists of a collection of 540 genomic inputs, ranging in size from 6 to 250 genes (i.e. $n = 12$ to $n = 500$), a subset of the simulated dataset used by Pereira Zanetti et al [9]. We generate the simulated instances as follows. We start with a unichromosomal linear genome with n genes and apply a random number between $\frac{rn}{2}$ and $\frac{3rn}{2}$ DCJ operations [11] to obtain each of the three input genomes, where r is a fraction between 0 and 1 (we refer to r as the *rearrangement rate*). After that we cut any circular chromosomes so that the resulting instance has three multi-chromosomal linear genomes. We use values of r ranging between 0.05 and 0.3, as higher values of r may require a distance correction and lead parsimony-based methods to produce incorrect results [24]. For each setting of n and r we generate 10 instances, and report the averages.

First, we observe that the exact $O(n^2)$ algorithm for the case $\alpha + \beta = n$ is extremely fast, requiring less than 45 seconds in total for all the 473 inputs (or 87.5%) that belong to it, i.e. less than 0.1 seconds per instance on average. The compressed sensing algorithm is somewhat slower, requiring a total of 105 seconds for the 67 inputs that it ran on, for an average of just over 1.5 seconds per instance. However, the time for all but the largest instances is in fact dominated by writing and reading the linear program file, not the actual solution. For instance, reading the file and solving the linear program each take CPLEX around 1.5 seconds when $n = 500$. In short, producing the median candidate using our method is extremely efficient relative to both the $O(n^3)$ computation proposed by Pereira Zanetti et al [9] as well as the exact and heuristic methods they compared it to.

Second, we observe that the vast majority of the inputs produce median candidates that are genomic matrices. More specifically, only 12 out of 540 outputs contain fractional values (and all of these are actually optimal as they fall into the case $\alpha + \beta = n$); these fractional values are $\pm\frac{1}{2}, \pm\frac{1}{3}, \frac{2}{3}, \pm\frac{1}{4}$ and $\frac{3}{4}$. The remaining 528 out of 540 outputs contain only integer values, among which 5 contain a -1, and it is a single -1 in all cases (none of these are optimal in the sense of attaining the lower bound β). The other 523 are binary (have all entries in $\{0, 1\}$), and of those, 34 are not permutation matrices; as expected from Corollary 2.1 they all contain a single 1 per row, but they each contain multiple 1's in 1 or 2 columns (and none of these are optimal). Of the final 489, 3 are permutation matrices that are not involutions (i.e. genomic matrices), and interestingly, all of these are optimal and are only found by our algorithm, not the one in [9]. The final 486 are genomic matrices, and are optimal in all except 7 cases; in those 7 cases, both our algorithm and the one in [9] are off by 1 from the optimal bound β. The final 479 outputs are both genomic and optimal.

Third, we observe that our algorithm produces strictly more optimal solutions than the one in [9], namely, 493 instead of 473 - this is reassuring as it shows that our algorithm can also be optimal in cases where the original one fails (the other way around is not possible due to Theorem 3.1). However, in the non-optimal cases, the approximation ratio of the original algorithm tends to be lower; this occurs in 39 cases, and 8 other cases result in ties. This is described in table 2.

n, r	0.05	0.1	0.15	0.2	0.25	0.3
12	0 (0)	3.3 (1.7)	0 (0)	2 (2.9)	3.64 (1.8)	9.3 (5.1)
16	0 (0)	0 (0)	0 (0)	0 (0)	3 (2.3)	0.7 (1.4)
20	0 (0)	0 (0)	0 (0)	2.3 (1.5)	2.1 (2.8)	6.4 (2.9)
30	3.3 (1.1)	2.2 (1.1)	2 (0.7)	1.9 (1)	1.1 (0.9)	3.7 (2.3)
50	0 (0)	0 (0)	1.1 (0.4)	0.9 (0.3)	1.6 (1)	1.8 (1.5)
100	0 (0)	0.8 (0.3)	0 (0)	0.8 (0.4)	0 (0.2)	0.8 (0.3)
200	0 (0)	0 (0)	0 (0)	0.3 (0.2)	0.1 (0.2)	0.2 (0.3)
300	0 (0)	0 (0)	0.1 (0.1)	0 (0.1)	0.4 (0.2)	0.1 (0.1)
500	0 (0)	0 (0)	0.1 (0.1)	0 (0)	0 (0.1)	0.1 (0.1)

Table 2. Average percentage excess over the lower bound β; the first number denotes our algorithm, while the second one (in brackets) represents the algorithm by Pereira Zanetti et al [9]

As can be seen from this table, both algorithms tend to be very close to the optimal, but there is no consistent winner between them; therefore, it might make sense to pick the better-scoring candidate among their respective outputs when the best median candidate is desired.

4.4 Real Dataset

The real dataset consists of a set of 12 Campanulaceæ chloroplast genomes as well as the Tobacco choloplast genome. We create all possible triples of inputs from this dataset, for a total of 286 input samples; each input had 105 genes, or $n = 210$ extremities.

The total time required for processing all the samples was 75 seconds, or less than 0.3 seconds per sample on average, which is consistent with the running times we obtained on simulated data.

Among the 286 test cases, 103 had some fractional output values. A total of 2448 entries among them, or 0.05% of the total, were fractions, and they included $\pm\frac{1}{2}, \pm\frac{1}{4}, \pm\frac{1}{5}, \frac{2}{5}, \frac{3}{5}$ and $\frac{3}{4}$. Just over half of them, 52 out of 103, had a score that attained the lower bound β.

Of the remaining 183 median candidates, 3 had a single -1 value in the output and were not optimal. Another 15 were binary but not permutation matrices (most with multiple 1s in 1 or 2 columns, and one occurrence in which there were multiple 1s in 3, 4 and 5 columns, respectively), and those were also not optimal. The remaining 165 were genomic matrices (there were no non-genomic permutation matrices), and all of these were optimal.

On real data, our algorithm again outperformed the one in [9] in terms of the number of optimal medians (those with score β) found - 217 vs. 189 out of 286; however, it did not perform as well in terms of the average ratio between the obtained score and the lower bound β - the average was 3% above β for us vs. 2% above β for the original algorithm. Our algorithm had a higher score more often, 57 vs. 32 out of 286 times, with the remaining 197 being ties. Once again, the choice of algorithm depends on the user's preference for a higher chance of getting an exact median vs. a better approximation ratio, and the optimal method seems to be to pick the best-scoring output among the two algorithms.

5 Conclusion and Future Work

In this paper we introduced a new algorithm for the median-of-three problem based on a necessary condition for attaining the lower bound, and used it to prove the uniqueness of the median in a favorable regime. In addition, we tested this algorithm on both simulated and real genomes, and saw that while it was more likely to find an exact median than the one by Pereira Zanetti et al [9], it generally had a worse approximation ratio than theirs did.

There are several remaining open questions, which we list here.

– What is the approximation ratio of the compressed sensing algorithm here?
– Are there optimality conditions less restrictive than the one we found?
– Are there triples of permutations for which the bound β is unattainable?
– What is the complexity of finding the median of 3 genomic matrices?

6 Acknowledgments

The authors would like to thank Cedric Chauve and Pedro Feijão for helpful discussions. LC would like to acknowledge financial support from NSERC, CIHR, Genome Canada and the Sloan Foundation. JM would like to acknowledge financial support from NSERC.

References

1. D. Sankoff, M. Blanchette. Multiple genome rearrangement and breakpoint phylogeny. J Comput Biol 5(3):555–570, 1998.
2. B. M. Moret, L. S. Wang, T. Warnow, S. K. Wyman. New approaches for reconstructing phylogenies from gene order data. Bioinform 17:165-173, 2001.
3. G. Bourque, P. A. Pevzner. Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Res 12(1):26–36, 2002.
4. A. Caprara The reversal median problem. INFORMS J Comput 15(1):93-113, 2003.
5. G. Fertin, A. Labarre, I. Rusu, E. Tannier, S. Vialette. Combinatorics of genome rearrangements. MIT Press, Cambridge, MA, 2009.
6. E. Tannier, C. Zheng, D. Sankoff. Multichromosomal median and halving problems under different genomic distances. BMC Bioinform, 10:120, 2009.
7. P. Feijao, J. Meidanis. SCJ: a breakpoint-like distance that simplifies several rearrangement problems. Trans Comput Biol Bioinform, 8:1318-1329, 2011.
8. I. Pe'er, R. Shamir. Approximation algorithms for the median problem in the breakpoint model. Comparative genomics. Springer, Berlin, 225-241, 2000.
9. J. P. Pereira Zanetti, P. Biller, J. Meidanis. Median Approximations for Genomes Modeled as Matrices. Bulletin of Math Biology, 78(4), 2016.

10. P. Delsarte. Bilinear forms over a finite field, with applications to coding theory. Journal of Combinatorial Theory A, 25(3):226-241, 1978.
11. S. Yancopoulos, O. Attie, R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinform 21:3340-3346, 2005.
12. P. Feijao, J. Meidanis. Extending the Algebraic Formalism for Genome Rearrangements to Include Linear Chromosomes. Trans Comput Biol Bioinform, 10:819-831, 2012.
13. Steven Roman. Advanced Linear Algebra. Graduate Texts in Mathematics. Springer-Verlag, New York, NY, 2008.
14. V. Arvind, P. S. Joglekar. Algorithmic problems for metrics on permutation groups. Lecture Notes in Computer Science, 4910:136-147, 2008.
15. B. Aspvall, Y. Shiloach. A fast algorithm for solving systems of linear equatlons with two variables per equation. Linear Algebra and Its Applications, 34:117-124, 1980.
16. D. L. Donoho. Compressed sensing. IEEE Transactions on Information Theory, 52(4):1289-1306, 2006.
17. LPsolve Team. lp_solve 5.5. Accessed on October 31, 2017. Web page *http://lpsolve.sourceforge.net/*.
18. IBM. CPLEX Optimizer. Accessed on October 31, 2017. Web page *www-01.ibm.com/software/commerce/optimization/cplex-optimizer/*.
19. Michael Lugo. The cycle structure of compositions of random involutions. 2009. Available at *https://arxiv.org/abs/0911.3604*.
20. R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Available at *http://www.R-project.org/*.
21. G. Csardi, T. Nepusz. The igraph software package for complex network research. InterJournal, Complex Systems 1695, 2006. Available at *http://igraph.org*.
22. D. Bates and M. Maechler. Matrix: Sparse and Dense Matrix Classes and Methods. R package version 1.2-10. Available at *http://CRAN.R-project.org/package=Matrix*.
23. L. N. Trefethen and D. Bau, III. Numerical Linear Algebra. SIAM: Society for Industrial and Applied Mathematics; first edition (1997).
24. P. Biller, L. Guéguen, E. Tannier. Moments of genome evolution by double cut-and-join. BMC Bioinformatics, 16(Suppl 14):S7, 2015.