The emperor has no caps!
A comparison of DCJ and Algebraic distances

Joao Meidanis and Sophia Yancopoulos

Abstract In this chapter we investigate the **DCJ** and **algebraic** distances and how they are found. We introduce a new graphical method to determine the *permutation cycles*, which embody the composition permutation for the genome transformation in the algebraic method. This graphical method helps tie the two approaches together. We discuss how in the usual approaches, the two methods differ only in the distance component due to the *even paths* in the adjacency graph of Bergeron, Mixtacki, and Stoye; these involve operations changing the type and number of chromosomes, such as fission, fusion, altering chromosome type from circular to linear and vice versa. Besides discussing each distance individually, we compare their internal mechanisms. So for example, *without caps* the algebraic approach has a different weighting scheme for even paths than the standard DCJ. However, when caps are added, the two have the *same* weighting scheme. Another convention which can be done in multiple ways is the method of *closure*. We discuss how to implement the *original closure rule* by which we arrive at the expected weighting scheme for the DCJ. Instead, by a new *alternative closure rule* which we introduce, the distance diverts to the algebraic distance. Finally, we note that although the Bergeron, Mixtacki, and Stoye way of computing the DCJ via the adjacency graph does away with "fictitious" caps and nulls, vestiges of *fictitious operations* remain, as the resulting weighting scheme is equivalent to that of the conventional DCJ.

Joao Meidanis
Scylla Bioinformatics and University of Campinas, Brazil, e-mail: meidanis@scylla.com.br

Sophia Yancopoulos
The Feinstein Institute for Medical Research, USA, e-mail: sopheetsa@aol.com

> So now the Emperor walked under his high canopy in the midst of the procession, through the streets of his capital; and all the people standing by, and those at the windows, cried out, "Oh! How beautiful are our Emperor's new clothes! What a magnificent train there is to the mantle; and how gracefully the scarf hangs!" in short, no one would allow that he could not see these much-admired clothes; because, in doing so, he would have declared himself either a simpleton or unfit for his office. Certainly, none of the Emperor's various suits, had ever made so great an impression, as these invisible ones.
> — Hans Christian Andersen, *The Emperor's New Clothes, Tales.*

1 Introduction

The early history of genome rearrangements harks back to a groundbreaking paper analyzing rearrangement scenarios in the fruit fly by Dobzhansky and Sturtevant in 1938 [1]. The onset of whole genome sequencing technologies made such studies truly viable. Hot in pursuit of these developments, the field really took off when David Sankoff and collaborators ushered in a new era of computationally based gene order comparisons [2, 3, 4]. In a prescient paper, David Sankoff made a brilliant intuitive leap to go from using edit distance based on the sequence level, to "non-local" large-scale genome rearrangement operations [5].

The language and ideas of permutations have frequently been inextricably linked with genome rearrangement studies [6, 7], however much of the actual mechanics of permutations has taken a back seat in theoretical developments, subsumed by a graphical formalism that contains permutations implicitly, from the breakpoint graph introduced by Bafna and Pevzner (1993) [8] to the more recent adjacency graph of Bergeron *et al* (2006) [9].

The recent reinjection of a more explicit algebraic formalism into the parlance of genome rearrangements has reinvigorated the discourse and challenged some of the basic underlying assumptions. Feijao and Meidanis's *Adjacency Algebraic Method* [16], a "hybrid" approach which combines the algebraic method with that of the adjacency formalism inherent in the use of the adjacency graph, arrives at some refreshingly unexpected results; these include a new weighting scheme for previously considered operations, particularly linear fissions and fusions, and the circularization or linearization of chromosomes.

In this chapter we focus on understanding the assumptions built into the new (adjacency) algebraic formalism, comparing with those underlying the standard DCJ approach. We explore how fundamental differences in the two approaches ultimately lead to differences in the distance and weighting schemes.

We begin with an introduction to permutations and genome rearrangements, followed by a brisk tour of classical models and operations. We introduce the DCJ and

the essentials of algebraic rearrangement theory. We proceed to examine transformations involving circular genomes, fundamentally suited to the algebraic approach.

We continue by presenting the DCJ as originally conceived, using caps and nulls to effectively "circularize" all genomes. We transition to the algebraic method via two new approaches, a capping scheme for the algebraic method which results in the same distance as the standard DCJ and a new closure scheme that results in the new distance implied by the algebraic method. These differences based on the closure scenario allows us to realize the deep dependence of the capping and closure schemes with the resulting distance and weights of operations.

We move on to examine the Adjacency Algebraic Theory and how it can be applied to linear chromosomes and transformations. We discuss a decomposition of the *Adjacency Graph* (AG) into components, and how these are independent and contribute independently to the distance. The essential components of the Adjacency Graph are *cycles*, and *even* and *odd paths*. We examine how these contribute to the genomic distance by the Adjacency Algebraic Theory, and contrast these contributions with the corresponding contributions for the DCJ.

Having developed the essential formalism for both methods, we go on to understand the consequences of the formalism on operations for different transformations as well as on the weighting scheme. We examine issues that arise from the introduction of "fictitious elements" (caps and nulls) including the possible artificial operations that may result as a consequence. We compare weighting schemes to see what consequences there are for these operations, and speculate on possible alternative weighting schemes as well as generalizations. We explore the implications including the correspondence to biology and conclude with open questions.

Figure 1 shows an overview of the results of this chapter.

1.1 Permutations and the genome rearrangement problem

Mathematically, a **permutation** is a bijective function, or a **bijection**, taking a set A to itself. Bijections can be represented as directed graphs where all vertices have indegree 1 and outdegree 1. Such graphs are collections of directed cycles that we call **permutation cycles** in this text.

Permutations are functions and can be composed as such. The composition of two permutations σ and π, where π is applied before σ, is indicated as a product and denoted by $\sigma\pi$ (Fig. 2). In general $\sigma\pi \neq \pi\sigma$. However, when two permutations are *disjoint*, that is they don't have any elements in common, they commute and $\sigma\pi = \pi\sigma$. The **support** of a permutation π is the set of elements that it moves, that is, $\pi(x) \neq x$. **Disjoint permutations** are permutations with disjoint supports.

In comparative genomics, evolutionary mutational processes causing large-scale rearrangements involve a shuffling of *syntenic segments* [10]. Researches have modeled this shuffling using permutations in a variety of ways.

For instance, in the paper by Bafna and Pevzner on the transposition problem [8], they define a genome as being a function $\pi : A \mapsto A$, where $A = \{1,2,\ldots,n\}$ is

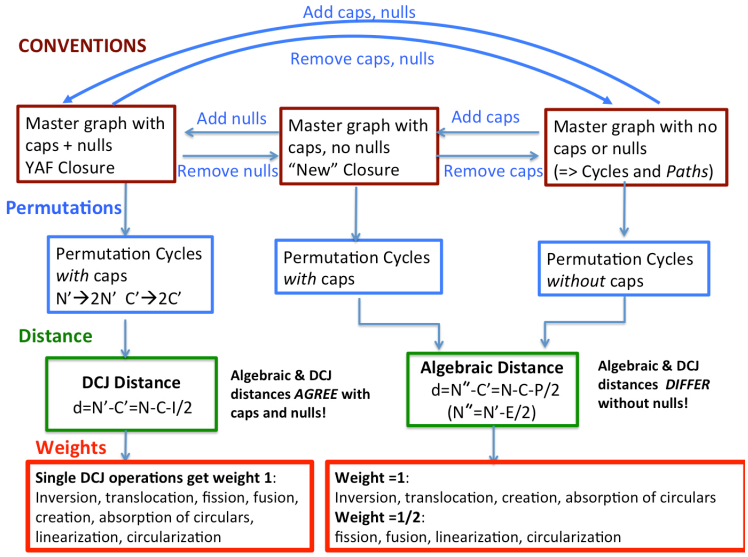

Fig. 1 Chapter Overview Fundamental assumptions affect distance and weighting schemes in the standard DCJ and Algebraic methods.

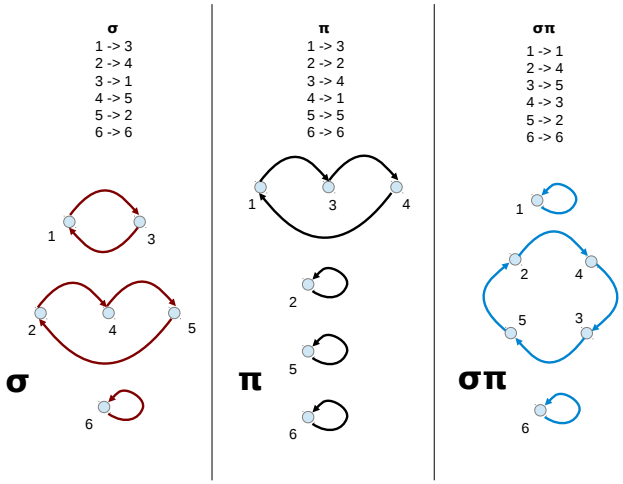

Fig. 2 Two permutations and their product. On top, we have the textual description of the process. Below this, we show the directed graphs representing each permutation, with σ in red, π in black, and the product $\sigma\pi$ in blue.

used to indicate both *chromosome positions* in the domain of π as well as *genes* in the image of π. Transpositions are also modeled as permutations by Bafna and Pevzner. Several other rearragement problems have been modeled via permutations, with adaptations for signed genes and multichromosomal settings [6, 7].

Meidanis and Dias [19] also use permutations, but in a different way. For them, a genome is a function $\pi : G \mapsto G$, where G is the set of *genes and their reverse complements*. No chromosomal positions are explicitly involved. The function π indicates which gene follows which other gene in the genome (they restrict their study to circular genomes, where every gene in a chromosome has a successor). Feijao and Meidanis [16] introduced yet another way of coding a genome as a permutation, by using the function π to indicate *adjacencies* between *gene ends*.

1.2 Genomes and the Chromosomal Notation

In this chapter, we deal with multichromosomal, signed genomes. We consider genomes that are collections of strictly linear or circular chromosomes, or a combination of both, made up of genes from a fixed set G. No gene repetitions are allowed, but all genes are always used in a pairwise comparison. We will represent linear chromosomes as lists of genes comprised in brackets, e.g., $[1,2,3]$, and circular chromosomes as lists of genes in parenthesis, e.g., $(1,2,3)$. Gene orientation is indicated by a sign. So, for instance, $\pi = \{[1,2],(-3,-4,5)\}$ is a two-chromosome genome, with one linear and one circular chromosome. When there is no risk of confusion, we will drop the curly brackets, writing just $\pi = [1,2],(-3,-4,5)$.

Notice that this representation is not unique. For linear chromosomes, their reverse complement indicates the same chromosome, e.g., $[1,2] = [-2,-1]$. For circular chromosomes, apart from the reverse complement, we also arbitrarily choose a gene to start, as there is no preferred starting point, e.g, $(-3,-4,5) = (-4,5,-3) = (5,-3,-4)$.

Observe that there is nothing in this notation that requires genes to be identified by integers. We can use letters, as in $[a,-c,-b,d]$, or even the very names used in biology to denote genes, such as *dnaA*, *cox1*, *adh*, or larger, contiguous regions, e.g., *MHC*, and so on. Nevertheless, to keep compatibility with the majority of theoretical papers on rearrangements, we will stick to genes as integers here.

We typically denote the total number of genes in genome π by N_π, or just N depending on context. We will also often see a gene a in terms of its two **gene ends** or **extremities**, with the **tail** denoted a_t, and its **head**, denoted by a_h, where a gene is typically oriented from tail to the head.

1.3 Genome rearrangement operations and models

Given two genomes of equal gene content and a set of allowed operations, the most basic question is to find the **distance** between these genomes, defined as the smallest number of allowed operations that will transform one genome into the other. In its full generality such a scheme allows for **weights** assigned to the operation, and the distance then becomes the total weight of a minimum-weight series of allowed operations that transforms one genome into the other. An example of such an operation is a **reversal** where a stretch of contiguous genes in a chromosome gets inverted, and their signs flipped. For instance, genomes $\pi = [1,2,3,4]$ and $\sigma = [1,-3,-2,4]$ differ by a reversal. Transforming one genome to another by a series of reversals is a problem that has been well studied [6, 7, 8].

Translocations involve multiple linear chromosomes and result in swapping chromosomal ends between two chromosomes. For instance, genomes

$$\pi = \{[1,-2,-3],[-4,5,6]\}$$

and

$$\sigma = \{[1,-2,5,6],[-4,-3]$$

differ by a translocation. Formally, they are similar to reversals if one considers concatenated chromosomes [11].

A **transposition** is defined as an operation that swaps two adjacent substrings in a permutation. Genomes

$$\pi = (1,-2,-3,-4,5,-6)$$

and

$$\sigma = (1,-4,5,-2,-3,-6)$$

differ by a transposition. Sorting scenarios with transpositions were introduced by Bafna and Pevzner [12]. This is more difficult than the reversal distance problem, and there were multiple improvements in approximation methods for this problem.

Christie introduced **block interchanges** [13], which swap any two non-intersecting substrings, a natural generalization of transpositions. Genomes

$$\pi = [1,-2,-3,-4,5,-6]$$

and

$$\sigma = [5,-6,-3,-4,1,-2]$$

differ by a block interchange.

It is reasonable for biologically realistic models of genome rearrangements to include more than one kind of operation, however in trying to consider generalized reversals and transpositions together in the menu of operations, researchers were somewhat baffled how to weight transpositions relative to reversals and translocations, which occur more frequently. In an intriguing paper, Blanchette et al

(1996) [14] allowed the weight of transpositions to vary relative to a weight of 1 for inversions in order to deduce the best weight. The authors noted there was a trade-off between inversions and transpositions, although some transpositions do not seem to be replaceable by inversions even with high values of the weighting function for transpositions. Using a greedy algorithm for genome rearrangements, they concluded that a weight just over 2 for transpositions and inverted transpositions was best able to optimize the rearrangement distance score for bacterial and mitochondrial genomes.

Recently, Bader and Ohlebusch [15], provided an algorithm for sorting by weighted reversals, transpositions and inverted transpositions using realistic weights.

Other operations are possible, for instance, chromosome fissions and fusions, excisions of linear or circular pieces from linear or circular chromosomes, linearization of a circular chromosome, circularization of a linear chromosome, and so forth. Examples of some of them appear in Table 1.

Table 1 Examples of rearrangement operations.

Operation	Example
Linear Reversal	$[1,2,3] \mapsto [1,-2,3]$
Circular Reversal	$(1,2,3) \mapsto (1,-2,3)$
Linear Translocation	$[1,2],[3,4] \mapsto [1,4],[3,2]$
Circular fission	$(1,2) \mapsto (1),(2)$

1.4 The Basic DCJ

The DCJ, or **double cut and join** [20, 9], is a universal operation capable of modeling a number of genome rearrangement operations, including inversions, translocations, fissions, fusions, and the creation and absorption of circular chromosomes. The corresponding DCJ distance was successful since its first appearance, having spurred a plethora of theoretical papers, and found its way in the implementation of several genome comparison systems. Perhaps the main reasons for its success are that, on the one hand it is easily computable by both humans (simple theory) and machines (low computational complexity), and, on the other hand, it models most operations observed to occur in real genomes, with weights that are considered reasonable.

The DCJ paradigm permits generalizations which allow insertions, deletions and duplications. Here we only entertain DCJ scenarios with equal gene content.

1.5 Algebraic Rearrangement Theory

Meidanis and Dias [19] used permutations to represent genomes, assigning a cycle to each chromosomal strand. The permutation formalism is particularly suited for *circular genomes* as these genomes automatically contain cycles by virtue of the circularity of their chromosomes; permutations, when drawn as directed graphs, also result in a collection of cycles. One interesting aspect of this approach is that the usual rearragement operations become permutations with small support.

Recently, Feijao and Meidanis [16] studied a novel way of representing genomes as permutations, focusing on the *adjacencies between gene ends* rather than trying to model each genome strand as a permutation cycle of chromosomes. This allows both linear as well as circular chromosomes to be modeled. They also discovered a formula relating their *adjacency algebraic* representation to the *chromosomal algebraic* representation of Meidanis and Dias. By going backwards from the adjacency to the chromosomal representation, using this formula, they were able unify both theories, thus extending the chromosomal theory to encompass linear chromosomes as well. The result was a new rearrangement distance between two signed, multichromosomal genomes. The new distance is easy to compute and its value on random genomes correlates well with the DCJ distance [16].

In Fig. 3 we see the chromosomal and adjacency representations of a genome consisting of a single circular chromosome containing 3 genes.

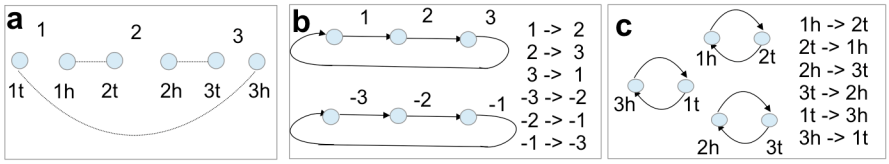

Fig. 3 Algebraic representation of a circular genome. a) Genome representation with adjacencies. b) Algebraic chromosomal representation. c) Algebraic adjacency representation.

1.6 Linear chromosomes and "Fictitious" Elements (caps)

Capping a linear chromosome is a technique that has been fruitful in many situations. For instance, in studying the unichromosomal transposition distance, Bafna and Pevzner extend their linear genomes π with $\pi(0) = 0$ and $\pi(n+1) = n+1$ [8]. The extra elements 0 and $n+1$ connected to the extremities of the linear chromosome are called **caps** and are useful in simplifying notation and arguments, avoiding awkward special cases. Caps have been used extensively in the multichromoso-

mal context as well [7], and in studies with other operations, such as reversals and translocations.

The "basic" DCJ [20] is also heavily rooted in the use of caps. In a way, capping a linear chromosome is a device which transforms it into a circular one. Circular chromosomes seem to be easier to deal with particularly, as we shall see, in the context of permutations, and this is the rationale behind the use of caps.

However, a question lingers on. Does the introduction of caps somehow affect the genomic distance being computed? We show that the addition of caps may affect the resulting distance. As there is some flexibility in the ways of circularizing the genomes via caps and closure scheme, according to the choices made, one may get different distances. We will see that the DCJ and algebraic distances are two facets of this phenomenon.

2 Transformations involving circular genomes

We now consider distance scenarios involving circular genomes. These are particularly suited to methods involving permutations. In Fig. 3 we saw the chromosomal representation for a circular chromosome contains complete cycles.

The goal of this section is to show that for circular genome transformations, the DCJ distance is equivalent to the algebraic distance. Although both distances have been covered in the literature, we offer here a self-contained treatment for the benefit of newcomers. We begin by introducing the main graphs mentioned in the literature, namely, the *breakpoint graph*, the *adjacency graph*, as well as the *master graph*, from which the previous two can be derived. These are important for visualizing the transformation and computing the genomic distance. We move on to some examples of circular operations, that is, simple operations that take a circular genome and transform it into another circular genome.

We note that even though such transformations may involve temporary states containing linear chromosomes, these are ultimately either circularized or absorbed into circular chromosomes so that the initial and final genomes are all circular. It is interesting to observe that temporary linear chromosomes can result from single-step operations such as single cuts in a circular. Operations such as *single cuts* or *single joins* have been considered by others, including by Feijao and Meidanis [17]. This highlights the importance of the particular model used to effect the transformation.

Sections 2.3 and 2.5 show how to compute DCJ and algebraic distances, respectively, and Section 2.6 contains a proof of the main result.

2.1 The master, breakpoint and adjacency graphs

The **master graph**, introduced by Friedberg, Darling, and Yancopoulos in 2008 [18], is a graph that specifies both genomes, and also connects between them. The initial genome, which we call π, is usually represented at the top (but as the diagram is completely symmetric it could also be done the other way, and be on the bottom). The gene extremities in this genome are linked by adjacencies which we color black here. These are *undirected* edges. The target genome which we call σ, is at the bottom and its gene extremities are linked by adjacencies that are colored red and are also undirected. Finally we use "green edges" to link corresponding gene ends in the two genomes. These are the "bridges" between the initial and target genomes. The red and black edges run horizontally, while the green edges connect the two genomes vertically. An example is shown in Fig. 4 in the next section. It is interesting to note that the master graph is in some sense the "precursor" graph for other important graphs. To see this:

- contract along the green edges and curve the black edges in the master graph to get the **breakpoint graph** for the *inverse transformation*
- for the *forward transformation breakpoint graph*, contract along the green edges; curve the *red* edges in the master graph, and invert the whole thing vertically
- to get the **adjacency graph** for the transformation, contract both red and black edges in the master graph to become vertices.

In their paper on polynomially sorting by reversals, Hannenhalli and Pevzner define a breakpoint graph slightly differently from the one defined here [6]. The reason is that the HP graph is for a linear problem and uses caps, while the master graph described above has no caps. However we will see in Section 3, which treats linear chromosomes, there is a capped version of the master graph, for which the previous bulleted items above still apply. The next section shows examples of these graphs.

2.2 Examples of transformations in circular genomes

In this section we consider two examples of transformations in circular genomes. First we consider an internal inversion and then a case of a fission.

2.2.1 An inversion in a circular from $\pi = (1,2,3)$ to $\sigma = (1,-2,3)$.

We construct the master graph and the other graphs as described in Section 2.1.

The master graph MG(π,σ) is shown in Fig. 4a. Below it is the breakpoint graph (BPG) for the inverse transformation. Note it is the *inverse* transformation as the genome at the bottom is the target genome, with adjacencies in red. The genome at

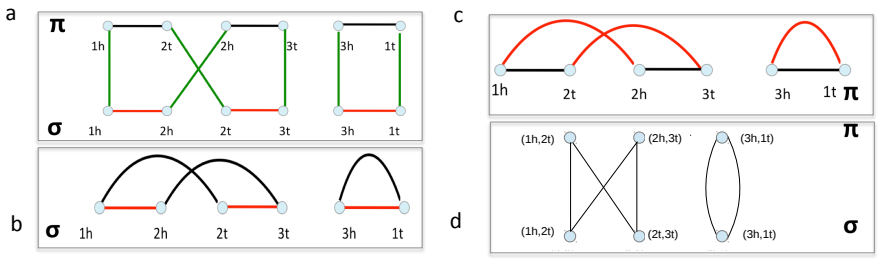

Fig. 4 Inversion in circular. a)Master graph (MG) b) & c) Breakpoint Graphs (BPG) for inverse and forward transformation d) Adjacency Graph (AG)

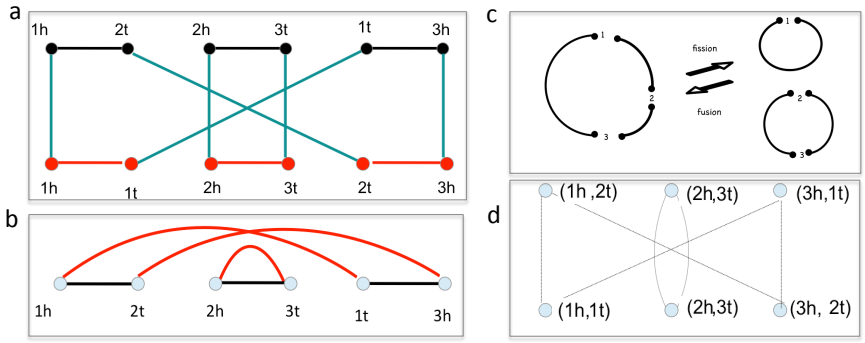

Fig. 5 The a) MG b) BPG c) Genome Graph and d) AG for a **circular fission**

top, is the current or initial genome. The BPG for the forward transition is shown in Fig. 4c arrived at by inverting the MG so the target genome is at the bottom. Its adjacencies are represented by black lines. As previously, the green and red lines are then curved and colored red; these "desire lines" connect gene ends in the target genome. The corresponding adjacency graph (AG), is shown in Fig. 4d. As before, we arrive at the AG by contracting red and black edges in the master graph.

Essentially, forward and inverse BPGs are contained in the MG. With practice, we can see them directly. The forward transformation BPG is up side down! The AG can also be visualized by mentally performing the procedure in the previous section.

2.2.2 Circular fission from $\pi = (1,2,3)$ to $\sigma = (1),(2,3)$.

We show the master, breakpoint, genome, and adjacency graphs for the *circular fission* in Fig. 5(a-d). Comparing these diagrams with those for the inversion, we see they are topologically similar in their adjacency graphs and both contain one 2-cycle and one 1-cycle.

A word about **cycle notation**: in a departure from the usual convention in the genome rearrangement literature, we call a cycle in the adjacency graph containing j adjacencies in only one genome a *j-cycle*. Usually it is called a *2j-cycle*. In the AG, *2-cycles* look like *bow ties*.

2.3 How to compute the DCJ distance from the master graph

For circular genomes, computing the DCJ distance from the master graph is a simple matter. All we have to do is compute

$$d_{DCJ}(\pi,\sigma) = N - C,$$

where $N = N_\pi = N_\sigma$ is the number of adjacencies (as well as the number of genes) per genome, and C is the number of cycles in the master graph. To see this, we note the master graph (or equivalently, the BPG or AG) for a transformation involving only circular genomes consists completely of cycles and each cycle can be resolved independently.

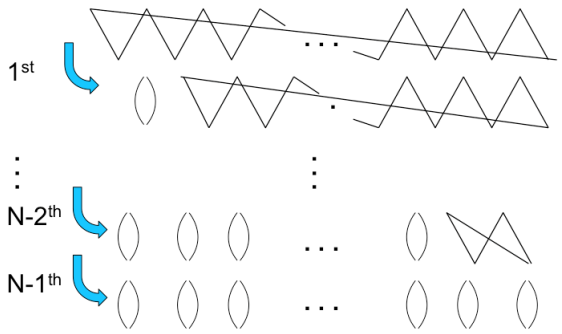

Fig. 6 Resolving a k-cycle by DCJs takes $k-1$ DCJs.

To resolve a k-cycle such as at the top of Fig. 6 by DCJs, we perform one DCJ at a time. A single DCJ can increase or decrease the cycle count by one, or in the case of an "improper" rejoining keep the cycle count unchanged [20]. If we perform two cuts (in "consecutive" adjacencies) we can rejoin so that a target adjacency is achieved. This forms a 1-cycle consisting of the target adjacency, and a remaining cycle with one less adjacency. Proceeding until the remaining cycle is a 2-cycle, the final 2-cycle is resolved by a single DCJ, transforming it into two 1-cycles. As there are k adjacencies, the final graph has k 1-cycles. Each subsequent DCJ produces a new 1-cycle except the last, which produces two. Since a single DCJ can at most augment the total number of cycles by one, we see that it takes a minimum

of $k-1$ DCJs to resolve a k-cycle. A non-minimal path can take more steps by performing DCJs which decrease rather than increase the number of cycles. If the graph is composed of cycles with sizes (counted as the number of adjacencies in each cycle) $k_1, k_2, \ldots$, to find the total distance, we sum on cycles:

$$d_{DCJ}(\pi, \sigma) = \sum_j (k_j - 1) = N - C \tag{1}$$

since the total number of adjacencies N is equal to the adjacencies summed over all cycles, $N = \sum_j k_j$, and similarly, the total number of cycles is just $C = \sum_j 1$.

2.4 Finding the permutation cycles from the master graph

The algebraic (permutation) approach is considered more *symbolic* whereas genome rearrangement approaches from HP [6, 7], to the DCJ resort to graphical representations. The master graph offers an excellent way to connect the DCJ approach with the Algebraic Adjacency Method. In Fig. 7, which revisits the inversion in a circular, we see that we can go directly from the adjacencies in the master graph in (a) to the 2-cycles in the *algebraic adjacency* representation in (b). Hence, the top and bottom of the master graph represent the initial and target genome in terms of their adjacencies, and equivalently their 2-cycle algebraic adjacency representation.

To see how to arrive at the product permutation $\sigma\pi$ from the initial to the target genome, consider the red path in Fig. 7d, where we trace going from a gene end in π (e.g., 1h, which connects to 2t in π) to the connecting gene end in σ (i.e., 2t connects to 3t in σ). Hence, the product permutation, where π is performed first and then σ, results in a transition from 1h to 3t, shown in the $\sigma\pi$ table in (d) and also outlined in the red *3-step* path in (a), leading to the *blue line*. Superimposing all these blue lines on the master graph ((a) and (c)), will ultimately trace out the product permutation cycles (Fig. 7e).

To find the product permutation cycles for transformations involving arbitrary circular genomes, we generalize the procedure performed above. We proceed as follows to generate the $\sigma\pi$ **permutation cycles**:

1. construct the master graph (MG) of σ and π
2. for the blue lines, at each extremity in π walk 3 steps starting on a black line
3. dissolve red and black "adjacency" lines connecting gene ends in adjacencies
4. contract along "green" lines; the blue lines trace the permutation cycles.

2.5 Computing the algebraic distance for circular genomes

An important concept introduced within the algebraic theory is the **norm** of a permutation, which measures the "complexity" of this permutation and equals $n - p$,

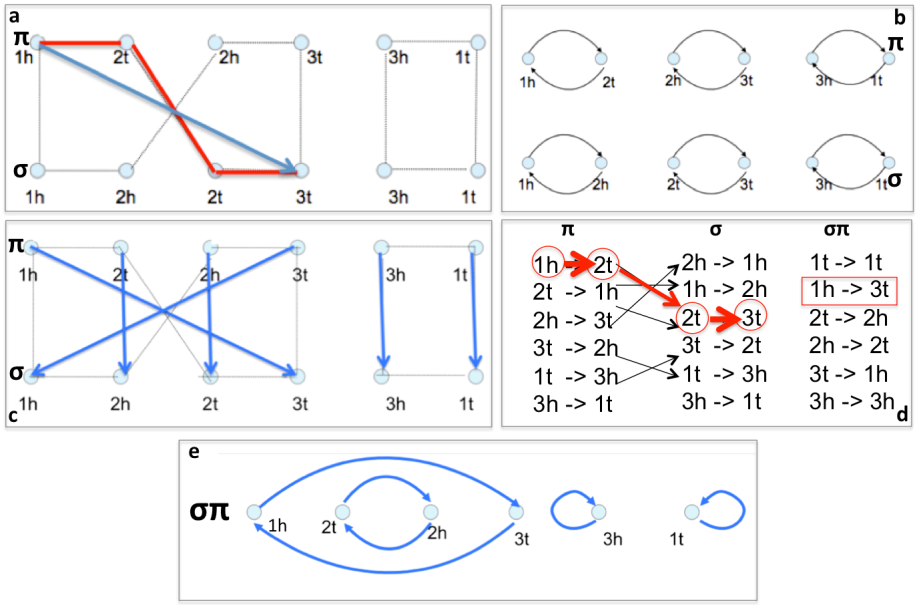

Fig. 7 From master graph to permutation cycles

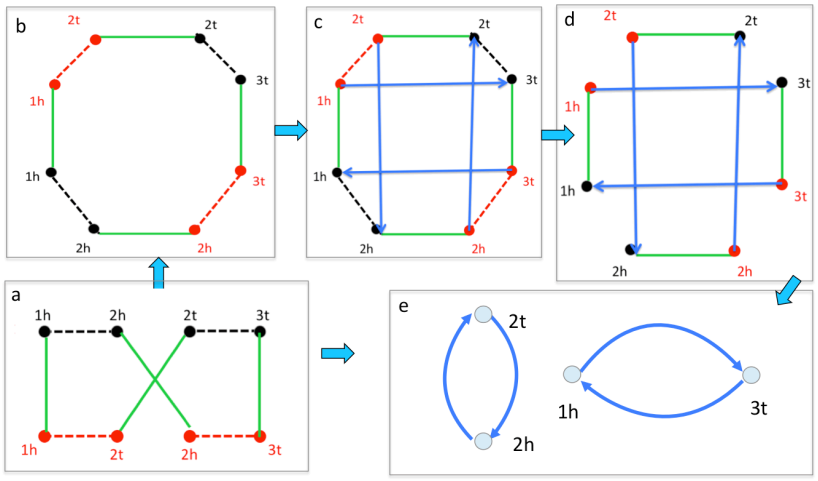

Fig. 8 Cycle doubling from master graph to permutation cycles

where n is the number of vertices and p the number of permutation cycles in the corresponding directed graph. We note the DCJ distance is of the same form.

In order for d_{DCJ} and d_{alg} to coincide, the algebraic distance is defined to be the norm of the composition permutation *divided by two* or they would differ by a factor of two. For circular genomes, the master graph for the transformation resolves completely into cycles which double in the permutation representation.

To visualize the cycle doubling from the master graph to the permutation cycles, it is helpful to look at Fig. 8b, where the 2-cycle from the previous example (also shown here in Fig. 8a) is "opened up". Adding blue lines as in Fig. 8c one blue line emanates from each gene end in π and bypasses 3 "BGR" (black-green-red) segments, ending at a gene end in σ. Traversing the next green line returns us to π and after N traversals alternating between blue and green lines, we return to our starting point.

As the red and black adjacency lines are dissolved, separating gene ends in adjacencies, and the green lines contracted, the number of permutation cycles are double those in the master graph. The number of gene ends ($2N$) is twice the number of genes (N). Hence the norm of the composition permutation is $2N-2C$ and the corresponding distance is half, or $N-C$ which agrees with the DCJ distance.

2.6 Comparing methods for circular chromosomes and proof

We have seen that for circular genomes the following formula can be used to compute the DCJ distance:

$$d_{DCJ}(\pi,\sigma) = N - C, \tag{2}$$

where $N = N_\pi = N_\sigma$ is the number of adjacencies per genome, and C is the number of cycles in the master graph.

On the other hand, we can compute the algebraic distance directly from the permutation cycles by taking the norm (the difference between the number of vertices and the number of cycles) and dividing by 2, which is equivalent to:

$$d_{alg}(\pi,\sigma) = \frac{n-p}{2} = \frac{2N-p}{2} = N - \frac{p}{2}, \tag{3}$$

where n is the number of *gene ends*, (double the number of genes , which, in circular genomes agrees with the number of adjacencies per genome, or N), and p is the number of permutation cycles in the $\sigma\pi$. We saw in the examples in previous sections that $p = 2C$, because each master graph cycle gives rise to two permutation cycles. As a result, the DCJ and algebraic distances agree.

More specifically, for the inversion of a circular (Fig. 4) and the circular fusion (Fig. 5), both cases consisted of one 2-cycle and one 1-cycle in their adjacency graphs. We recall that our cycle nomenclature labels a j-cycle by the number j of adjacencies in only one of the genomes.

Now, 1-cycles do not contribute to the distance. To see this, note that in the DCJ formulation, the distance contribution for $N = 1$ and $C = 1$ is 0. In the algebraic formulation, we saw (Fig. 8) that the 1-cycle *doubles into two cycles* in the permutation $\sigma\pi$, where the notation for the cycles is based on the number of *gene ends* in the adjacency graph, which is $n = 2N = 2$ for a 1-cycle. In the permutation $\sigma\pi$, the two cycles each become $n/2$-cycles, or $1-cycles$, for which the algebraic distance is $(n-p)/2 = (2-2)/2 = 0$. As for the 2-cycle, it contributes $N - C = 2 - 1 = 1$ in the DCJ formulation and $(n-p)/2 = (4-2)/2)$, which also equals 1, in the algebraic formulation. Hence the two distances agree.

In this section, we go on to prove that the agreement between the two distances (algebraic and DCJ) is a general result, as far as circular genomes are involved. This cannot be considered a new result, because it is implied by formulas for the distances given by Feijao and Meidanis [16], but since our approach here significantly differs from previous treatments, and one of our goals here is to view the algebraic theory in graphical terms, we developed the proof below.

Theorem 1. *If π and σ are circular genomes with the same genes, then*

$$d_{DCJ}(\pi,\sigma) = d_{alg}(\pi,\sigma).$$

Proof. In view of Formulas (2) and (3), it suffices to prove that the "cycle doubling" occurs in general. To see that this is actually the case, reason as follows. Every cycle in the master graph will have as number of edges a multiple of 4. This is because the edges alternate between green and adjacency edges, and, moreover, the adjacency edges alternate between red and black edges. In other words, starting from any gene end in π, for instance, and following the direction of the only black edge incident to it, we necessarily traverse four edges of colors black, green, red, and green again before we have the chance of closing the cycle. So the minimal cycle has four edges. If the cycle does not close after the first four edges, this means we reached a new, free gene end in π, and must traverse a new 4-edge walk (black, green, red, green) before being able to close.

Let's now see what happens with each such cycle as we compute the permutation cycles. Each permutation cycle is formed by blue edges, and each blue edge is the result of traversing a black-green-red walk in the master graph, that is, we are effectively walking over a master graph cycle. Moreover, in the end the green edges get contracted, and the net result of this can be seen as taking a fourth and last step, thus effectively continuing on the master cycle for another step.

It follows that each edge in a permutation cycle corresponds to four consecutive edges in the corresponding master cycle. In a master cycle with $4k$ edges, this will result in a permutation cycle with k edges. However, this permutation cycle involves only half of the π gene ends in the master cycle, because the adjacency partner of each gene end in the permutation cycle is missing from this cycle, since it is reachable in just one step, which is not a multiple of 4. The missing gene ends will be in another k cycle. It follows that each cycle in the master graph gives rise to two permutation cycles, that is,

$$p = 2C,$$

and therefore $d_{DCJ}(\pi,\sigma) = d_{alg}(\pi,\sigma)$. □

3 General transformations using "fictitious" elements

Having seen that for circular genome transformations the DCJ and algebraic distances turn out the same, we move on to the general case of multichromosomal genomes with no restrictions on chromosome types: circular, linear, or a mixture.

We review the original construction by Yancopoulos et al. which effectively transforms the general case into the circular case [20] by the addition of caps, and, where needed, null chromosomes to restore the balance in number of chromosomes between genomes. We call this procedure the **original closure rule** and the transformed graph the **augmented** master graph. Once this is done, the DCJ distance can be computed as in the circular case by using the augmented master graph.

Using the approach in the previous section, the augmented graph can also be used to derive permutation cycles, and thereby the algebraic distance. We find, perhaps not surprisingly, this coincides with the DCJ distance. We also investigate a new closure rule that keeps the caps but closes the paths into cycles without resorting to null chromosomes. Perhaps surprisingly, this **alternative closure rule** results in exactly the algebraic distance.

A final subsection summarizes these observations and contains proofs of results.

3.1 Examples of transformations with linear chromosomes

In this section we consider examples of transformations with linear chromosomes. We examining their master graphs we observe that in addition to cycles they also contain paths. Paths occur in the event of linear chromosomes, as there are vertices in the adjacency graph corresponding to "telomeric" gene ends, gene ends at the ends of chromosome without partners. We investigate how to close such paths to form cycles, so that we will be able to compute the distance as previously, by subtracting the number of cycles from the number of adjacencies.

3.1.1 Linearization

Consider the transformation from a genome with a circular chromosome, $\pi = (1)$, to that of a linear chromosome $\sigma = [1]$. This operation is the linearization of a circular chromosome. The master graph for this pair of genomes is shown in Fig. 9a. Contracting red and black edges we arrive at the adjacency graph in Fig. 9b.

As a result of the linear chromosome in the transformation, the master graph and adjacency graph have a path. To circularize this path, we add caps to the telomeres (unpaired extremities) in $\sigma = [1]$. This results in a disequilibrium of adjacencies in

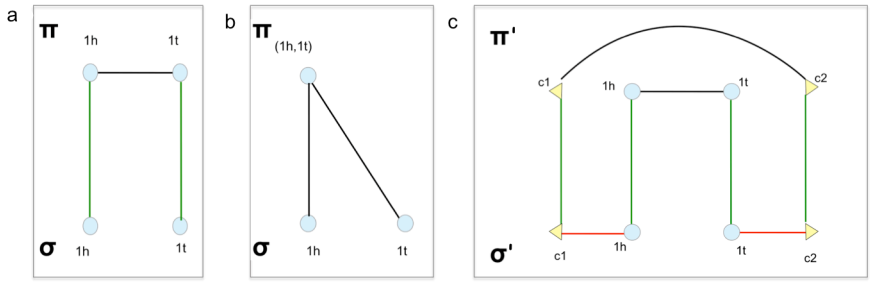

Fig. 9 Linearization. (a) Master Graph (b) Adjacency Graph (c) Augmented Master Graph with caps and null.

the two genomes π and σ. To restore the balance, we add null chromosome $(c1, c2)$, containing the same caps added in σ, to π, and close the path by adding connecting green lines to the null. The resulting augmented master graph can be seen in Fig. 9c.

3.1.2 Fission of a linear genome

Consider the transformation from a genome of a linear chromosome $\pi = [1,2]$ fissioning into two chromosomes $\sigma = \{[1],[2]\}$. The master graph for this pair of genomes is shown in Fig. 10a. Contracting red and black edges we arrive at the adjacency graph, shown in Fig. 10b.

Here again we have paths. To close them, we start by adding caps to each telomere. There are two in π and four in σ. After adding these caps there is a resulting imbalance of adjacencies in the two genomes. We compensate for that by adding a null chromosome, $(c2, c3)$, to π. Connecting corresponding gene ends with green edges we arrive at the augmented master graph in Fig. 10c.

3.2 The original closure rule, completing paths with caps and nulls

To understand the DCJ approach for general transformations including all kinds of chromosomes, we note that, with linear chromosomes present, the master graph contains paths in addition to cycles. The idea behind the method is to convert cases containing paths to the circular case by augmenting the master graph to circularize paths into cycles.

Linear chromosomes have unpaired gene ends at their ends called **telomeres**. We augment the master graph by attaching new, "dummy" gene ends called **caps** to the telomeres. Effectively we cap the ends of paths with these fictitious elements.

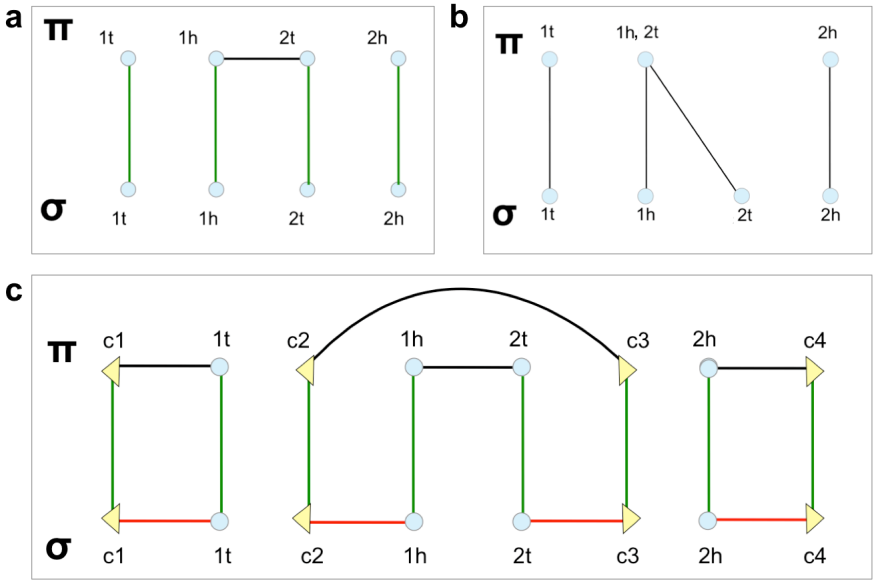

Fig. 10 Fission in a linear. (a) Master Graph (b) Adjacency Graph (c) Augmented Master Graph

Once we've capped the paths,we note there are two kinds of paths in their master (or equivalently their adjacency) graphs, those with both capped ends in the *same* genome, are called **even paths**, and those with their caps in *opposite* genomes, are called **odd paths**.

To close paths into cycles in the master (or adjacency graph), we add *green lines*, with two different approaches for odd and even paths. The **original closure rules** from Yancopoulos et al. [20] are:

1. For *odd paths* join the caps at the ends of the paths *directly* by a green edge.
2. For *even paths* create a **null chromosome** containing two caps in the genome opposite the genome with the two caps. Draw green edges linking each of the caps in the first genome to the caps of the null chromosome in the other genome.

We notice that, this method generates null chromosomes. The total number of adjacencies is augmented by counting nulls as a single adjacency and single caps as half. The total number of adjacencies is balanced since adjacencies containing actual gene ends are balanced in both genomes in cycles as well as in paths (where telomeres count as half an adjacency), *odd* paths have one cap in each of the genomes, and *even* paths, have two caps in one of the genomes balanced by a null in the other.

Denoting the augmented number of adjacencies in either genome by N', and the total number of cycles including paths closed by the closure rule by C', the DCJ distance can be computed as follows:

$$d_{DCJ} = N' - C',$$

3.3 Understanding the DCJ distance for general transformations

Having extended the formalism of closed cycles to the general case including linear chromosomes by our system of capping and the original closure rules, we now wish to make contact with the Bergeron *et al* [9] distance formula. Accordingly we will focus on the adjacency graph presentation, keeping in mind the adjacency graph can be derived from the master graph simply by compressing adjacencies. We cap and close paths using the original closure rule. Our goal is to reconsider the path formulation of Bergeron *et al* in light of caps, nulls and path closure. For illustrative purposes, consider the adjacency graph in Fig. 11 containing paths and cycles.

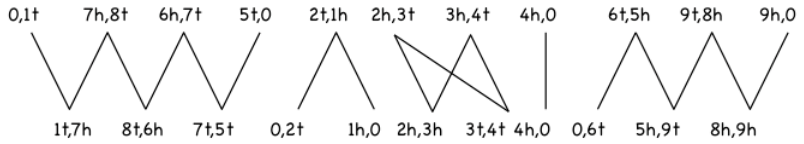

Fig. 11 A sample adjacency graph for a general transformation, that is, a transformation involving linear chromosomes. Here the top genome is $[1,2,3,4],[5,6,7,8,9]$, and the bottom genome is $[-1,-7,5,9,-8,-6],[2,-3,4]$.

Using the capping and closure formalism just developed, we know how to complete paths to transform them into cycles. We take note of some general principles:

- In general, the Adjacency Graph (AG) contains cycles and (even and odd) paths.
- A componentwise decomposition of the AG resolves it into these components.
- Each component makes an *independent* contribution to the DCJ distance.

We now consider how each individual component contributes to the total distance arrived at in the previous subsection.

3.3.1 Distance contribution from cycles

From Section 2.3 we know the distance contribution of an individual cycle containing k adjacencies is just $k-1$. Hence, if there are C cycles in the entire graph, involving a total of N_{cycle} genes, there are also N_{cycle} adjacencies, and the total contribution from cycles to the DCJ distance is

$$N_{cycle} - C.$$

3.3.2 Distance contribution from (closed) even paths

At the top of Fig. 12 we see a succession of even paths whose telomeric ends start and end in the top genome. Below these are listed the corresponding number of uncapped adjacencies belonging to each path, culminating in a general even path, represented by a "squiggly W", which does not have its adjacencies explicitly enumerated. For such a path involving k genes, there are k adjacencies, as there are $k-1$ "legitimate" adjacencies containing two gene ends, and 2 telomeric ends counting half an adjacency each.

Below the illustration of the uncapped even paths is a corresponding row with the paths closed by the original closure rule of Section 3.2 having caps attached in the top genome and a null chromosome in the bottom genome. The caps count as half an adjacency just as the telomeric ends. The two caps in the null chromosome count also as 1, so the amended adjacency total for either genome for each of these closed paths is one more than the previous uncapped version.

The path becomes a $(k+1)$-cycle. Using the DCJ formula for cycles, a closed general even path involving k genes contributes with

$$k+1-1=k$$

to the DCJ distance. The same holds if the even path starts and ends in the bottom genome.

If we consider all even paths, involving N_{even} genes altogether, their contribution to the DCJ distance is, therefore, N_{even}.

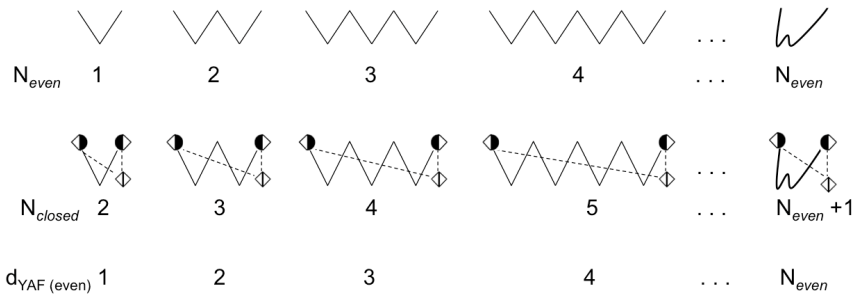

Fig. 12 Uncapped and capped even paths and their corresponding DCJ distances (called d_{YAF} in the picture).

3.3.3 Distance contributions from (closed) odd paths

Applying the capping and original closure rules for the odd paths we note that, once capped, an odd path involving $k+1/2$ adjacencies becomes a closed $(k+1)$-cycle, since the additional cap adds $1/2$ to the adjacency count. (The final count will still be an integer because odd paths occur in pairs.) Using the distance rule for closed cycles which is the number of adjacencies minus 1, this leads to a DCJ distance contribution from this odd path of

$$k-1/2.$$

Adding over all odd paths, a total contribution of $N_{odd}-I/2$ follows, where I is the number of odd paths.

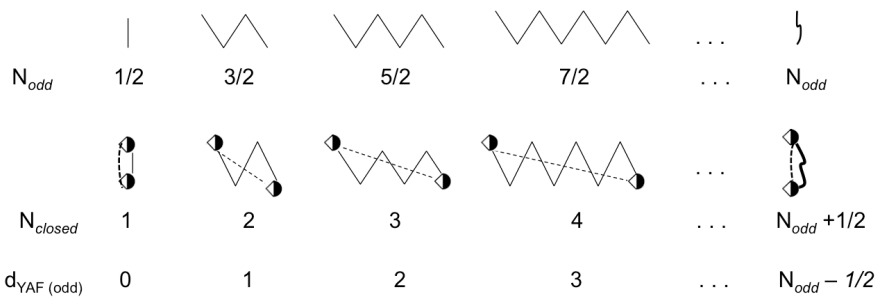

Fig. 13 Uncapped and capped odd paths and their corresponding DCJ distances (called d_{YAF} in the picture).

3.3.4 Adding all contributions from cycles, even and odd paths

To make contact with the Bergeron *et al* [9] formulation of the DCJ distance, we add all contributions from all components to the distance for a general transformation keeping in mind that each cycle, odd and even path contributes according to what was previously found. Summing on all components we arrive at:

$$\begin{aligned} d_{DCJ}(\pi,\sigma) &= \text{contribution from cycles} + \text{even paths} + \text{odd paths} \\ &= (N_{cycle}-C)+N_{even}+(N_{odd}-I/2) \\ &= N-C-I/2, \end{aligned}$$

because the total number of genes N is equal to $N_{cycle}+N_{even}+N_{odd}$, the number of cycles is C, and the total number of odd paths is I.

3.4 Algebraic Distance for permutation cycles using caps and nulls

In the previous section we observed that a general transformation contains cycles, and even and odd paths in the Adjacency Graph (AG) as first discussed by Bergeron, Mixtacki and Stoye [9] who introduced the AG. We went on to discuss how each of these components contributes to the DCJ distance, noting that the paths can be circularized by the addition of caps (for odd paths) and both caps and null chromosomes (for even paths). This circularization procedure using the original closure rules is achieved by the use of "fictitious" elements, but it is still possible to use the formalism for permutations as we did for pure circular genomes. Just as in the strictly circular case, since we've circularized all paths, all cycles will double when we go to the permutation cycles for the composition permutation. In addition, the vertices in the permutations will be twice the number of adjacencies. Hence since the algebraic distance is half of the norm for the permutation cycles , **we find that the algebraic distance using capped genomes with nulls is the *same* as the DCJ distance**.

3.5 Alternative closure scheme bypassing nulls

Instead of the "original closure rule" we can use an alternative closure scheme which includes caps and closes odd paths in the same way, but, by avoiding nulls, closes even paths by connecting the two cap-ends directly. Unlike the old closure rule, the number of chromosomes is no longer the same in both genomes.

With the new rule we use the number of adjacencies after capping *averaged* over both genomes, remembering that caps contribute $1/2$ an adjacency, as do telomeres. With the new rule, to average the number of adjacencies, we see that when two caps are added to genome π they get "averaged", so there is a resulting contribution of $1/2$ to the total number of adjacencies. The same happens when two caps are added to σ.

With the new closure rule, we introduce a new means of computing a distance, which we will call d_{new}. Since cycles and odd paths do not change their contributions to the total distance, the only difference could come from the even paths. In fact, the the new contribution from even paths differs from the DCJ contribution in $-1/2$ for each even path.

Summing over paths, we arrive at the new closure distance:

$$d_{new}(\pi,\sigma) = N - C - I/2 - E/2 = N - C - P/2,$$

where E is the number of even paths and $P = I + E$ is the total number of paths. But this is a know alternative formula for the algebraic distance [16]. We conclude that the new closure rule yields, in fact, the algebraic distance.

3.6 DCJ vs Algebraic distances for capped genomes

As we saw, there are ways of conceiving and computing the algebraic distance based on a graphical approach, using either the master graph or the adjacency graph. However, there is also a way of computing the DCJ distance using the algebraic approach: using capped genomes and the original closing rule. Indeed, let us recall for a moment the result of the original closure rule on the pair $\pi = [1,2]$, $\sigma = \{[1],[2]\}$. Fig. 14 is just the same as Fig. 10c, except that we named the caps with subscripts t and h so that they become more like gene ends.

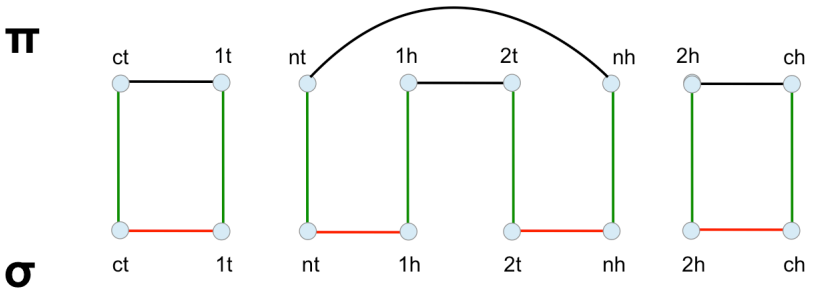

Fig. 14 The augmented master graph for genomes $\pi = [1,2]$ and $\sigma = \{[1],[2]\}$. Caps and nulls were named with subscripts t and h to resemble gene ends. With this, the figure looks exactly like the master graph of $\pi' = \{(1,2,-c),(n)\}$ and $\sigma' = (1,n,2,-c)$.

Doing that, we are able to write the chromosomal expression of the resulting genomes. If we add arrows going from x_t to the corresponding x_h for all entities x, be them genes or dummies, and follow the cycles formed by these arrow plus adjacencies, we end up with two circular genomes, namely, $\pi' = \{(1,2,-c),(n)\}$ on top and $\sigma' = (1,n,2,-c)$ in the bottom. We can then compute the algebraic distance between these two genomes, and the result, as we know, will also be the DCJ distance between these genomes. It turns out that, given a black box that computes algebraic distances, we are able to use it to compute the DCJ distance: just feed it with the capped versions of the genomes, augmented by the original closure rule. This is what the following result states.

Theorem 2. *Let π and σ be multichromosomal genomes over the same gene set, and let π' and σ' be their capped versions, augmented with the original closure rules. Then*

$$d_{DCJ}(\pi,\sigma) = d_{alg}(\pi',\sigma').$$

Proof. By definition, we know that

$$d_{DCJ}(\pi,\sigma) = d_{DCJ}(\pi',\sigma'),$$

since both pairs share the same master graph. Notice that the master graph of π' and σ' does not have to be augmented, because it already consists of cycles only. This means that π' and σ' are circular genomes. But for circular genomes, we know that equality holds between DCJ and algebraic:

$$d_{DCJ}(\pi', \sigma') = d_{alg}(\pi', \sigma'),$$

because of Theorem 1. The result then follows from these two equalities. □

The consequence of this result is that properties of the algebraic distance can be, in principle, applied to the DCJ distance via this correspondence. However, one possible difficulty here is the fact that π' does not depend solely on π, as the notation seems to imply. It also depends on σ.

4 Genome Permutations by the Adjacency Algebraic Theory

In this section we will apply algebraic rearrangement theory in its more recent, adjacency-based formulation to understand the algebraic distance formula and to see how differences with the DCJ distance can arise.

We start by revisiting two examples from the previous section, the linearization example from Fig. 9 and the linear fission example from Fig. 10. we note they both involve a single cut, which involves the breaking (or cutting) of exactly one adjacency. The master graphs for the two examples are very similar in terms of their both having a 2-cycle except that the master graph (Fig. 10c) for the linear fission is flanked by two 1-cycles. Since these are identity transformations, they don't contribute to the distance, and so we focus on the 2-cycle as in Fig. 9c. We use the procedure in Section 2.4 to draw the "blue lines" (Fig. 15a) and then derive the permutation cycles for the capped (Fig. 15b) and uncapped (Fig. 15c) master graphs.

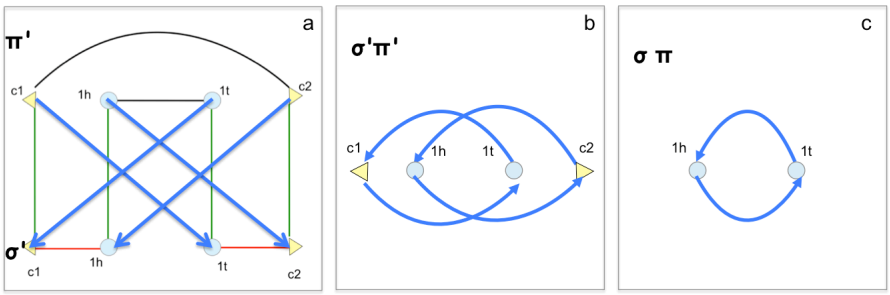

Fig. 15 Linearization permutation cycles (PC) a. blue lines in MG; b.capped PC c. uncapped PC.

To arrive at the uncapped permutation cycles, we can start from the capped master graph and "identify" the caps, as a device to reconnect the path/cycle without them. We imagine starting at a gene end such as $(1h)$ in the capped master graph and move along on an outgoing blue line; when we arrive at a cap (i.e. $(c1)$) we "jump" immediately to the "identified" cap, (i.e. $(c2)$) and continue along the outgoing blue lines onto gene end $(1h)$, completing the cycle.

Having found the permutation cycles, we note that the 2-cycle in the capped master graph resulted in two 2-cycles in the capped permutation cycles, and only a single 2-cycle in the uncapped case. To derive the permutation cycles for uncapped genomes in a general way, we move on to using the breakpoint graph, although we could do it with other graphs. The virtue of the breakpoint graph here is that its lines end on vertices representing gene ends just like the permutation cycles.

4.1 Component-wise decomposition of the Adjacency Graph

As mentioned in Section 1, permutations can be represented by directed graphs that are composed of one or more *cycles*. Every permutation can be written in an essentially unique way as a product of disjoint cycles, apart from the order in which the cycles are multiplied (recall that disjoint cycles commute). This is called the **cycle decomposition** of a permutation. The transformation of a genome π to another genome σ over the same genes is a permutation, and the master graph (MG), adjacency graph (AG), or the breakpoint graph (BP) of π and σ can graphically represent the components of this transformation. The permutation that transforms π into σ is $\sigma\pi$ in the adjacency algebraic theory — and $\sigma\pi^{-1}$ in the chromosomal algebraic theory, which we will not delve into here.

We discussed the separation of the adjacency graph for a general transformation into its connected components in Section 3.3, and saw they can be identified visually. As pointed out by Bergeron *et al* [9] these components are essentially *cycles* and *paths* in the AG. The same holds true for the breakpoint graph, which will be our main graphical tool in this section. Since separate components have no elements in common, their distance contributions can be resolved separately. We have seen in the section on transformation of circular chromosomes, Section 2, that every cycle in the MG (and hence in the breakpoint graph) generates two cycles in the permutation $\sigma\pi$. We will show that paths generate a single cycle in $\sigma\pi$ with the same vertices.

4.2 The Breakpoint Graph and Permutation Cycles

We are interested in looking at the connected components of the breakpoint graph between genomes π and σ to see how they contribute to the algebraic distance. In terms of components, the master, adjacency, and breakpoint graphs are very much

alike: cycles in one graph will correspond to cycles in the other two, perhaps with a different size, but always even. More specifically, a cycle with $4k$ edges in the MG will correspond to a cycle with $2k$ edges in the AG, and to a cycle with $2k$ edges in the BPG. And a path in one graph will always be a path in the other two, again with a difference in size, but not, in general, of the same parity. Specifically, a path with $2k+1$ edges in the MG will correspond to a path with $k+1$ edges in the AG, and to a path with k edges in the BPG.

4.3 The algebraic distance for paths

To understand more generally how permutation cycles result from paths in the breakpoint graph, we take a look at some generic paths. Instead of going from the construction just described for the previous example in Fig. 15, where we found the permutation cycle by starting from the capped master graph and then identified caps, we try to find a more direct root by starting with a path in the breakpoint graph and proceeding to add the blue lines to it, as depicted in Fig. 16.

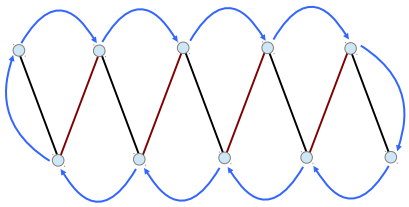

Fig. 16 A path in the breakpoint graph that begins and ends with a black edge. The breakpoint graph is drawn in zig-zag mode, much like the adjacency graph, to better illustrate the permutation cycle, (in blue), obtained in general by starting first in π (black edges) then σ (red edges). A possible pair of genomes that generates such a breakpoint graph is $\pi = \{(1),(2),(3),(4),(5)\}$ and $\sigma = [1,2,3,4,5]$.

To find the permutation cycles associated with a path we compute $\sigma\pi$, that is, apply π first and then σ. We indicate the permutation arrows in blue in Fig. 16. They were obtained, from each gene end, following a black edge and then a red edge. If there is no black edge incident to a certain gene end, just follow the red edge to get to the image under $\sigma\pi$. And if, after following a black edge, there is no red edge to continue, just take the result of the first step as the final result. Performing this

procedure for every gene end we get the blue arrows indicated in the figure. Notice that they involve all gene ends in the path, "enclosing" the path like a cloud.

Similar results are obtained with any path, not unlike what we have seen already. Paths of any size, starting with either black or red edges, and finishing with either black or red edges, all result in an enclosing blue cycle involving all the gene ends in the path.

We note that this procedure mimics that of Section 2.4, in tracing out the "blue lines" by walking 3 steps starting at every vertex, which is a gene end in π. In this method, the "3 steps" reduce to two, which are the black and red edges of the breakpoint graph since, in the breakpoint graph, the "green lines" have been contracted out of the picture and are represented by vertex-gene ends.

To see more explicitly how this procedure connects to specific gene ends, we return to the more elaborate approach using the master graph and follow this protocol for all three types of components in the MG, AG, or BPG, that is for *even* and *odd* paths as well as for *cycles*.

4.3.1 Even paths

Let us consider the telomeres and adjacencies of a general even path in the AG starting and ending in σ:

$$\underbrace{(e1)}_{\sigma},\underbrace{(e1,e2)}_{\pi},\underbrace{(e2,e3)}_{\sigma},\ldots,\underbrace{(e_{n-1},e_n)}_{\pi},\underbrace{(e_n)}_{\sigma},$$

where n is an even integer.

Drawing the master graph with caps (Fig. 17a), we use the procedure in Section 2.4 for constructing the "blue lines". Contracting the green lines we arrive at the "breakpoint graph" alternating between black and red edges, enveloped by the permutation lines (Fig. 17b). By identifying the caps at the ends (and eliminating them) we find that the permutation cycles of an even path involving n gene ends (n being therefore an even integer) is a single permutation n-cycle (Fig. 17c).

The entire construction works just as well if the even path starts and ends in π. Hence, a general "even path" in the MG, AG or BP, (starting and ending in either π or σ), with k adjacencies in one genome in the AG, and $n = 2k$ vertices in the BP, corresponds to an n-path in the AG and an n-cycle in the permutation $\sigma\pi$. The corresponding contribution to the algebraic distance from the even path is:

$$\frac{(n-1)}{2} = \frac{n}{2} - \frac{1}{2}.$$

4.3.2 Odds paths

Similarly, let's consider the telomeres and adjacencies of a general odd path starting in σ and ending in π.

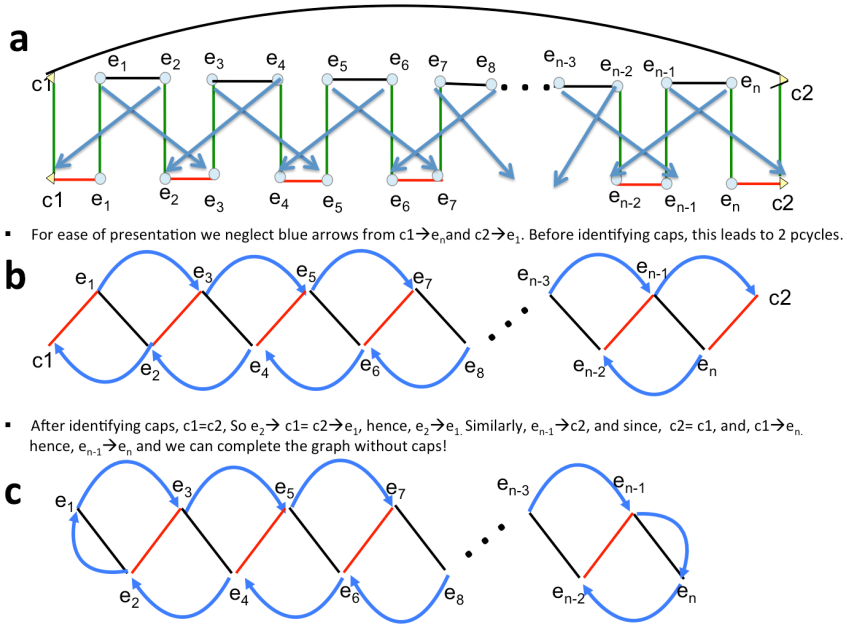

Fig. 17 An even path (n is an even integer). **a** Master Graph with caps and blue lines; π is on top. **b** Breakpoint Graph and permutation lines with caps. **c** Breakpoint Graph and a *single* permutation n-cycle without caps.

$$\underbrace{(e_1)}_{\sigma},\underbrace{(e_1,e_2)}_{\pi},\underbrace{(e_2,e_3)}_{\sigma},..,\underbrace{(e_{n-1},e_n)}_{\sigma},\underbrace{(e_n)}_{\pi},$$

where n is an odd integer.

Using a similar construction as for the general even path, we draw the capped master graph (Fig. 18a) and add blue lines as previously. In Fig. 18b we show the breakpoint graph containing alternating black and red lines and show the permutation lines with caps. In Fig. 18c we "absorb" the caps into their adjacency partner. What results is a permutation cycle with n edges, as each of the n vertices has an incoming and outgoing blue permutation line.

Even though this path starts in σ and ends in π, the odd path could "start" in π instead. Hence, for a general "odd path" in the MG, AG or BP, with n gene ends, and n vertices in the BP, the contribution to the algebraic distance is:

$$\frac{(n-1)}{2} = \frac{n}{2} - \frac{1}{2}.$$

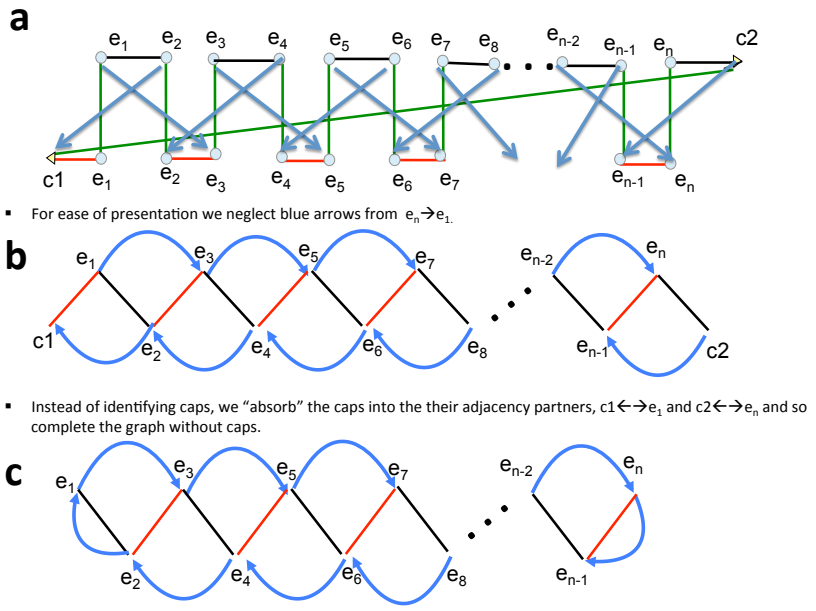

Fig. 18 An odd path (n is odd). **a** capped MG and blue lines (π on top). **b** capped BPG. **c** BPG and n-cycle, no caps.

4.4 The algebraic distance for cycles

We already discussed the algebraic distance for cycles in the section on circular transformations Section 2. To put our current treatment on par with that for paths, we will also treat cycles using the general framework of the previous sections. Consider a cycle containing n edges (corresponding to n "gene ends") in the adjacency graph of the transformation from π to σ corresponding to a master graph cycle component pictured in Fig. 19a.

$$\underbrace{(e_1,e_2)}_{\pi},\underbrace{(e_2,e_3)}_{\sigma},\ldots,\underbrace{(e_{n-1},e_n)}_{\pi},\underbrace{(e_n,e_1)}_{\sigma},$$

where n is necessarily even.

As we saw in Section 2.4, when following the procedure illustrated in Fig. 8 to find the composition permutation cycles from the master graph for circular genomes, there is a doubling of cycles which occurs in going from the master graph (MG) to the permutation cycles (PCs). At the same time, the number of vertices in the BPG

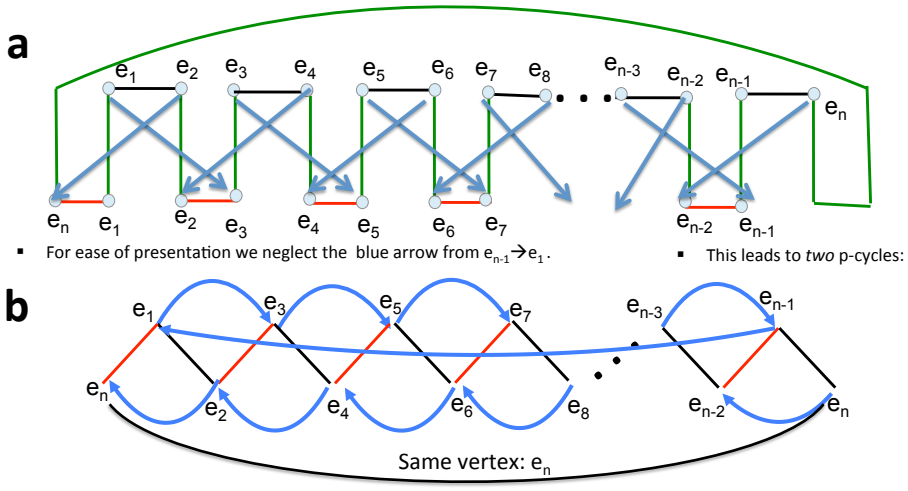

Fig. 19 A general cycle (n is even). **a** MG with blue lines (π on top). **b** BPG showing an MG cycle becomes *2* permutation cycles.

(equivalent to the number of red and black edges in the AG), is halved in a PC. This is easy to see in Fig. 19b, where it is clear that the splitting of the cycle segregates even and odd numbered adjacencies into two different permutation cycles. We note that $n = 2k$ is even, where k is the number of adjacencies in the cycle in either genome in the AG, and the permutation cycles each contain $n/2$ vertices. To find the contribution to the algebraic distance of a cycle in the AG (BPG, or MG) we find the norm of the corresponding permutation and divide by two. Hence, the contribution is:

$$2\frac{(\frac{n}{2}-1)}{2} = \frac{n}{2} - 1.$$

4.5 Total algebraic distance in the Adjacency graph

As we discussed, the adjacencies of the master graph can be segregated separately into *connected components* which contribute independently to the distance. We just saw how both even and odd paths in the adjacency graph (and also the master graph and breakpoint graph) turn into single cycles in the permutation $\sigma\pi$, whereas cycles in the adjacency graph double in number while halving their vertex counts.

Since we know how to compute the contribution of each individual component, it remains to tally all contributions from components in the adjacency graph. Collecting these, we sum distance contributions over all cycles and even and odd paths. The i^{th} cycle contributes $n_i^{cycle}/2 - 1$ to the distance, where n_i^{cycle} is the number of

gene ends in the i^{th} cycle. As for the paths, as we found, each path (even or odd) contributes $n_j^{path}/2 - 1/2$, where n_j^{path} is the number of gene ends in the j^{th} path. Summing over all components we get:

$$\begin{aligned} d_{alg}(\pi,\sigma) &= \text{ cycle contribution } + \text{ path contribution} \\ &= \frac{n_{cycles}}{2} - C + \frac{n_{paths}}{2} - \frac{P}{2} \\ &= N - C - \frac{P}{2} \end{aligned}$$

where $n_{cycles} = \sum n_i^{cycle}$ and $C = \sum_i 1$ are summed over cycles, and $n_{paths} = \sum n_j^{path}$, is summed over paths. Notice that $n_{cycles} + n_{path} = 2N$, because each gene corresponds to two gene ends. Since both even and odd cycles have the same formal contribution in this approach, we do not have to separate them in the sum over paths.

4.6 DCJ distance vs Algebraic distance

The difference between the DCJ distance and the (Adjacency) Algebraic distance boils down to the difference in the distance for even paths. For the DCJ distance the contribution from even and odd paths is asymmetric, whereas, as we just saw, for the algebraic method both contribute in the same way. Since both formulas can be stated simply in terms of the number of adjacencies and the number of odd and even paths, we can easily compute the difference. The distance $d_{dcj} = N - C - I/2$, where, as we recall, I is the number of *odd* paths, while $d_{alg} = N - C - P/2$ where $P = I + E$ is the sum over the number of odd *and* even paths. The difference, is therefore just $E/2$ which is just half the number of even paths!

Since we can say even paths occur whenever the number of chromosomes changes, or the type of chromosome changes, either from linear to circular or vice versa, these transformations may occur relatively rarely in genome transformations, in which case the two distances will appear to be nearly the same. We next re-examine some previous examples to compare similarities and differences between methods.

So, for example, operations considered in Section 2 which dealt with circular transformations were operations such as (internal) inversions, and the creation and absorption of circular chromosomes (from other circular chromosomes) have only cycles in their adjacency graph, and, as we have seen, cycles are treated in the same way by both methods.

We also know that differences can arise with the use of caps, and in the section involving such general transformations which include caps(Section 3), we discussed two transformations in particular where this may be the case, in particular, that of a linearization involving a circular transforming to a linear chromosome, and a linear fission of a single linear chromosome. Not only did these transformations involve caps, with the DCJ approach, but their “bare” uncapped master graphs contained

even paths, signaling there is likely to be a discrepancy. In fact, since both transformations contain exactly one even path, we know there is a difference: the DCJ distance is 1 for both the linearization and the linear fission, whereas the algebraic approach yields a distance of $1/2$.

Even though these are rather simple examples, there is more that can be gleaned from them. Not only can we compare the forward transformations, but the inverse transformations as well. And so we can also deduce that for the DCJ, a *circularization* of a linear chromosome, or a *fusion* of two linear chromosomes into one also costs a single DCJ, whereas the algebraic method yields again, a distance of $1/2$ for each. Finally, even if there are more "bystander" genes which are not actually involved in the transformation, these operations will still cost the same.

We conclude by summarizing the similarities and differences of the two methods for cycles, and odd and even paths in Fig. 20. This table shows the different contributions of each kind of component, with emphasis on cases where DCJ and algebraic distances differ, namely, the even paths. We examine these differences and the implications in further detail in the next section.

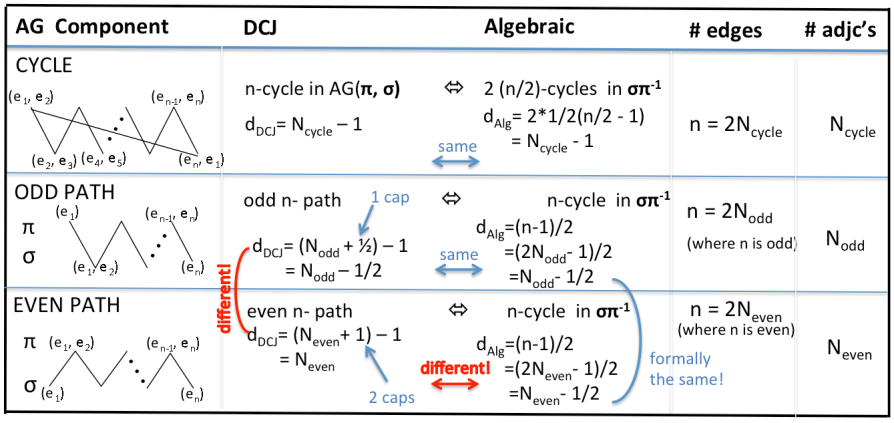

Fig. 20 Component-wise Summary of DCJ vs Algebraic distances.

5 Weights, Operations, and Biology

In this section we discuss the biological implications of the two approaches particularly by looking at the DCJ and algebraic formulations as methods which minimize the number of weighted operations needed to transform one genome into the other.

5.1 The relative weights of operations

Part of the challenge of modeling genome rearrangements is to find a successful strategy for dealing with the relative weights of operations. One of the original dilemmas for the DCJ was the relative weight of a transposition or a block interchange to an inversion. Our examination of the underpinnings of the DCJ and algebraic methods, has brought to light an interesting challenge: the relative weight of operations such as fissions, fusions, linearizations or circularizations vs inversion. In comparing these methods we saw that ultimately, the determination of this relative weight comes down to the underlying assumptions in the method. If caps are used, the ratio is 1 and when they aren't it depends on the method.

5.2 Comparing Weights for DCJ vs Algebraic Method

Yancopoulos and others [20] describe a complete list of operations characterized as DCJ. Table 2 shows the list adapted from their 2005 paper. We use "circular" in places where they use "circular intermediate" because here we study the general case, where the start and target genomes can have both types of chromosomes, linear and circular. Therefore, we do not necessarily think of circular chromosomes as intermediate. Each of these operations has a weight of 1 DCJ.

Table 2 A complete list of DCJ operations.

Operation	Example
Linear Translocation (incl fission/fusion)	$[1,2],[3,4] \mapsto [1,4],[3,2]$
Linear Reversal	$[1,2,3] \mapsto [1,-2,3]$
Creation of circular from linear (including circularization)	$[1,2,3] \mapsto [1,3],(2)$
Absorption of circular by linear (incl linearization)	inverse of previous
Circular Reversal	$(1,2,3) \mapsto (1,-2,3)$
Circular fission/fusion	$(1,2) \mapsto (1),(2)$

The same operations are available in the algebraic approach, but the weights are different. Table 3 shows the same operations with weights in both the DCJ scheme and the algebraic scheme. Notice that we have to split some operation classes because not all members have the same algebraic weight. In general, operations involving null chromosomes are weighted half a DCJ in the algebraic scheme.

Table 3 A comparison of weights in the DCJ and algebraic schemes. Operations described as 'proper' may not involve null chromosomes.

Operation	DCJ weight	Algebraic weight	Example
Linear Translocation, proper	1	1	$[1,2],[3] \mapsto [1],[3,2]$
Linear fission/fusion	1	$1/2$	$[1,2] \mapsto [1],[2]$
Linear Reversal	1	1	$[1,2,3] \mapsto [1,-2,3]$
Creation of circular from linear, proper	1	1	$[1,2,3] \mapsto [1,3],(2)$
Circularization of linear	1	$1/2$	$[1] \mapsto (1)$
Absorption of circular by linear, proper	1	1	$[1,2],(3) \mapsto [1,3,2]$
Linearization of circular	1	$1/2$	$(1) \mapsto [1]$
Circular Reversal	1	1	$(1,2,3) \mapsto (1,-2,3)$
Circular fission/fusion	1	1	$(1,2) \mapsto (1),(2)$

5.3 Fact or artifact: fictitious objects and dummy elements

We have discussed how it is possible to use caps and nulls and other such "fictitious" objects in order to close paths in the adjacency graph so as to use the formalism developed for cycles. We have shown how it is possible to arrive at the transformation distance, using capped versions of genomes. As caps and nulls are not "real" it may be thought that somehow these shouldn't "count" or "weigh in". Hence, some might think that the algebraic method, which doesn't use them, may seem more legitimate. Others may argue that though these dummy elements are strictly a device, they aid in simplifying the distance equations.

5.4 Fictitious operations and the "Basic" DCJ

We have made no bones about discussing the artificiality of caps; having given them the "power" of appearances, it remains to discuss the consequences in terms of the kinds of operations that result in the "basic" DCJ.

Ultimately, the double cut and join is a series of four more "elemental" individual **cut** and **join** operations, two of each. When the "single" DCJ operation is performed between two "real" adjacencies, then two "actual cuts" and two "actual joins" happen between "real" gene ends.

With the use of caps in the "basic" DCJ, concomitant **fictitious operations** can arise, which have no ultimate "reality". These fictitious operations involve any type of cut or join involving a cap. There are two possible adjacencies involving caps in the basic DCJ schema: either an adjacency between two caps which is a null chromosome, or an adjacency between a cap and single gene end, also known as a **telomere**. We define a fictitious operation as any operation involving either a cut or a join in an adjacency containing at least one cap.

To understand this in the context of an example, consider the case of the linearization example discussed in Section 3 using the capped master graph. Just as with any DCJ, there are essentially two cuts and two joins, but now, some of these involve caps. The null chromosome contains an adjacency which is severed as well as the adjacency between the two gene ends in the circular chromosome (1t and 1h). So one of the cuts performed in this DCJ is fictitious, and the other is real. After these adjacencies are broken, the gene ends are reconnected with caps, and both of these operations are fictitious. The total operation gets a DCJ "weight" of 1.

One might be tempted to infer that each individual *cut* and *join* is $1/4$, after all, there are four operations and the whole operation has a unit weight. If we do this, then in the linearization example the only "real" operation is the cut between 1t and 1h; the remaining 3 operations (the cut of the null chromosome and the joining of the 2 caps) are fictitious, so the "true cost" of this operation should only be 1/4!

In fact, Feijao and Meidanis [17] considered such a possible scheme with their "SCJ" operation such that any single cut or single join is weighted as little as $1/4$.

5.5 Fictitious Operations and the Algebraic Approach

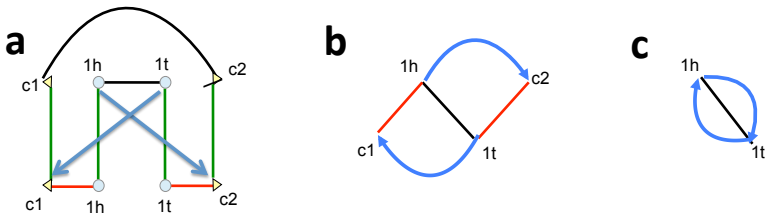

Fig. 21 Linearization in BPG representation. **a** MG with blue lines. **b** BPG with capped PC. **c** uncapped PC.

What about fictitious operations for examples like these? First let's start with the algebraic distance for these examples. As we know and saw in Fig. 15c, or here in Fig. 21, the permutation cycles for linearization consist of a single $2-cycle$, the norm of which is $n-1=2-1=1$; dividing by two, the distance is $1/2$. We have reached the *irreducible quantum* operation in the algebraic approach in a single fission/fusion/linearization. As we can see from Fig. 21c, the only "step down" from here which is a PC "cloud" surrounding a *single* breakpoint, or adjacency, is a single $n=1-cycle$ with a single gene end. But this last has *no* distance or cost. Hence, in the algebraic schema there is no "step down" below this operation which has a cost. The lowest we can go is to a cost of $1/2$ which straddles between the DCJ and the SCJ. This leaves us with more questions than answers. Had the cost of the

operation been $1/4$ of a full DCJ we might not be as perplexed, since it would seem the "fictitious" operations no longer "weigh in".

5.6 Does the BMS- DCJ do away with fictitious operations?

The beautiful formalism developed by Bergeron *et al* [9] in their approach to the DCJ, including the adjacency graph and the idea of odd and even paths, seems to liberate the DCJ from the clumsy use of caps and nulls. The beauty of the new approach was welcomed by many because of its elegant formulation, ease of calculation, and also because at least for some who may have been bothered by them, it seemed to do away with caps, and the issues that come with them; after all, caps are not only a bother, but they are an artificial construction.

Interestingly, even though the DCJ approach as formulated by Bergeron *et al* [9] does not *appear* to have caps and nulls explicitly, nevertheless, their version of the distance *agrees in value* with that of the "basic" DCJ distance computed with the use of caps and nulls. The weighting scheme of Bergeron *et al* is also identical to that of the DCJ with caps and nulls.

The fact that the algebraic distance *agrees* with the DCJ distance (BMS[9] or YAF[20]) when caps and nulls are used, but *differs* when they are not, begs the question: does the BMS approach somehow retain vestiges of fictitious operations?

5.7 Biological interpretation

The difference in weights between the DCJ and algebraic schemes poses interesting questions. For instance, is it biologically meaningful to give less weight to operations involving null chromosomes? On the one hand, this may make sense because apparently less "modification effort" is needed to effect, say, a linear fission than a linear translocation: in the first case, a single "cut" occurs, while in the second, there are actually two cuts and two rejoins. On the other hand, the creation of a new chromosome is no easy task biologically. A chromosome is not just genes. It needs, to begin with, a duplication apparatus, which includes a centromere or at least an origin of replication. Simplified genome modeling does not take these into account.

We must remember that a mutation has to be accepted by the environment to survive as such. Therefore, a lasting rearrangement must be "easy" with respect to modification effort, but also "stable" in terms of genome structure, and these two components should contribute to its weight.

To further illustrate this discussion, consider the following scenario: $\pi = [1,2]$ and $\sigma = [2,1]$. This is a linear transposition, and the DCJ distance between these two genomes is 2. However, the algebraic distance is just 1! How come? Because the algebraic method finds a shorter path of going from π to σ: linear fission of π, giving $[1],[2]$, which is the same as $[2],[1]$, and then linear fusion of these two chromosomes

yielding σ. Since a linear fission or fusion is worth $1/2$, the total comes to 1. Notice that this is only possible because the two blocks being interchanged comprise the entire chromosome.

One could argue that the algebraic path is way out of scope, since it involves both the creation and the destruction of a chromosome, which are supposedly expensive biological operations. However, one could also argue that this argument holds only if you assume that these two changes occurred separated by millions of years of evolution. What if they occurred in the *same* cell division?

It is hard to say what is correct or not in modeling biological processes. It may be the case that in some situations one approach is more suited, while in other situations the opposite is true. Experimental use of the distances will help us to shed more light into this issue.

5.8 Implications and concluding remarks

The fact that the algebraic method leads to a *different* weighting scheme than the *standard* DCJ not only lends a new approach to calculating the genomic distance, but offers the possibility of addressing issues having to do with the use of "dummy" elements such as caps and nulls. It is intriguing that we have found that by use of these elements in the algebraic approach, the distance agrees with the *standard* DCJ, but bypassing their use we arrive at a *different* distance and weighting scheme for the operations, primarily those involving even paths in the adjacency graph, and ultimately fissions and fusions in the transformation

By comparing the two formulations we have come to realize the consequences involving the assumptions behind the two methods. Ultimately we feel that neither method is "superior" in that either weighting scheme may be considered valuable under some circumstances and the possible use of either of these two methods increases the number of available options in analyzing genome transformations.

Acknowledgements We would like to thank Richard Friedberg for a long, inspiring conversation over dinner at an Indian restaurant in New York, on February 24th, 2013. We thank Pedro Feijao and Marilia Braga for shorter, but also inspiring, conversations. S.Y. thanks Nick Chiorazzi for his tolerance and support, David Sankoff for introducing Joao to me, and Judith Ficksman for her heartfelt encouragement. Finally, we thank our reviewer for meticulous and insightful comments.

References

1. Dobzhansky T, Sturtevant AH. Inversions in the chromosomes of *Drosophila pseudoobscura*. Genetics **23:** 28–64 (1984)
2. Sankoff D, Goldstein M, Probabilistic models for genome shuffling. Bulletin of Mathematical Biology **51:** 117–124 (1989)

3. Sankoff D, Cedergren R, Abel Y. Genome divergence through gene rearrangement. Methods in Enzymology **183:** 428–438 (1990)
4. Sankoff D, Leduc G, Antoine N, Paquin B, Lang BF, Cedergren R. Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. Proceedings of the National Academy of Sciences USA **89:** 6575–6579 (1992)
5. Sankoff D. Edit distance for genome comparison based on non-local operations. Combinatorial Pattern Matching. Third Annual Symposium, Lecture Notes in Computer Science. Springer Verlag. **644:** 121–135 (1992)
6. Hannenhalli,S. and Pevzner,P.A. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). Journal of the ACM (JACM), **46**1: 1–27 (1999). Previously appeared at *Proc. of the 27th Annual Symposium on the Theory of Computing (STOC 95), Las Vegas, Nevada* 178–189 (1995)
7. Hannenhalli,S. and Pevzner,P.A. Transforming mice into men (polynomial algorithm for genomic distance problem). *Proc. of the 36 Annual Symposium on Foundations of Computer Science (FOCS 95), Milwaukee, Wisconsin* 581–592 (1995)
8. Bafna V, Pevzner PA. Genome rearrangements and sorting by reversals. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press,**46:** 148–157(1993)
9. Bergeron A, Mixtacki J, Stoye J. A Unifying View of Genome Rearrangements. In *Algorithms in Bioinformatics Proceedings of WABI 2006.* Edited by Moret B; (2006)
10. Nadeau JH, Taylor BA.Lengths of chromosomal segments conserved since divergence of man and mouse. Proc Natl Acad Sci USA **81:** 814–818 (1984)
11. Tesler G. Efficient algorithms for multichromosomal genome rearrangements. J Comp Sys Sci **65:** 587–609 (2002)
12. Bafna V, Pevzner P. Sorting by transpositions. SIAM J. DISCRETE MATH. 224–240 (1998)
13. Christie D. Sorting permutations by block interchanges. Inform. Processing Lett. **60:** 165–169 (1996)
14. Blanchette M, Kunisawa T, Sankoff D. Parametric analysis of genome rearrangement Gene **172:** 11-17 (1996)
15. Bader M, Ohlebusch E. Sorting by weighted reversals, transpositions, and inverted transpositions. J. Comput. Biol. **14:** 615–636. (2007)
16. Feijao, P., Meidanis, J. Extending the algebraic formalism for genome rearrangements to include linear chromosomes. IEEE/ACM Transactions on Computational Biology and Bioinformatics **99**(PrePrints), 1 (2012). DOI http://doi.ieeecomputersociety.org/10.1109/TCBB.2012.161
17. Feijao, P., Meidanis, J. SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans Comput Biol Bioinform. **8**(5):1318–29. DOI 10.1109/TCBB.2011.34. (2011)
18. Friedberg, R., Darling, A.E., Yancopoulos, S.: Genome rearrangement by the double cut and join operation. In: J.M. Keith (ed.) Bioinformatics, *Methods in Molecular Biology*, vol. 452, pp. 385–416. Humana Press (2008)
19. Meidanis, J., Dias, Z.: An alternative algebraic formalism for genome rearrangements. In: D. Sankoff, J. Nadeau (eds.) Comparative Genomics, pp. 213–223. Kluwer Academic Publishers (2000)
20. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, in version and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)