Rubert *et al. BMC Bioinformatics* 2018, **19**(Suppl 6):152
https://doi.org/10.1186/s12859-018-2130-5

BMC Bioinformatics

RESEARCH

Open Access

Computing the family-free DCJ similarity

Diego P. Rubert[1], Edna A. Hoshino[1], Marília D. V. Braga[2], Jens Stoye[2] and Fábio V. Martinez[1*]

From RECOMB-CG - 2017 : The Fifteenth RECOMB Comparative Genomics Satellite Conference
Barcelona, Spain. 04-06 October 2017

Abstract

Background: The genomic similarity is a large-scale measure for comparing two given genomes. In this work we study the (NP-hard) problem of computing the genomic similarity under the DCJ model in a setting that does not assume that the genes of the compared genomes are grouped into gene families. This problem is called family-free DCJ similarity.

Results: We propose an exact ILP algorithm to solve the family-free DCJ similarity problem, then we show its APX-hardness and present four combinatorial heuristics with computational experiments comparing their results to the ILP.

Conclusions: We show that the family-free DCJ similarity can be computed in reasonable time, although for larger genomes it is necessary to resort to heuristics. This provides a basis for further studies on the applicability and model refinement of family-free whole genome similarity measures.

Keywords: Genome rearrangement, Double-cut-and-join, Family-free genomic similarity

Background

A central question in comparative genomics is the elucidation of similarities and differences between genomes. Local and global measures can be employed. A popular set of global measures is based on the number of genome rearrangements necessary to transform one genome into another one [1]. Genome rearrangements are large scale mutations, changing the number of chromosomes and/or the positions and orientations of DNA segments. Examples of such rearrangements are inversions, translocations, fusions, and fissions.

As a first step before such a comparison can be performed, some preprocessing is required. The most common method, adopted for about 20 years [1, 2], is to base the analysis on the order of conserved syntenic DNA segments across different genomes and group homologous segments into *families*. This setting is said to be *family-based*. Without duplicate segments, i.e., with the additional restriction that at most one representative of each family occurs in any genome, several polynomial time algorithms have been proposed to compute genomic distances and similarities [3–7]. However, when duplicates are allowed, problems become more intricate and many presented approaches are NP-hard [2, 8–13].

Although family information can be obtained by accessing public databases or by direct computing, data can be incorrect, and inaccurate families could be providing support to erroneous assumptions of homology between segments [14]. Thus, it is not always possible to classify each segment unambiguously into a single family, and an alternative to the family-based setting was proposed recently [15]. It consists of studying genome rearrangements without prior family assignment, by directly accessing the pairwise similarities between DNA segments of the compared genomes. This approach is said to be *family-free* (FF).

The *double cut and join* (DCJ) operation, that consists of cutting a genome in two distinct positions and joining the four resultant open ends in a different way, subsumes most large-scale rearrangements that modify genomes [5]. In this work we are interested in the problem of computing the overall similarity of two given genomes in a family-free setting under the DCJ model. This problem is called FFDCJ similarity, and in some contexts it may be more

*Correspondence: fhvm@facom.ufms.br
[1]Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brazil
Full list of author information is available at the end of the article

powerful than a distance measure, where it is known that the parsimony assumption holds only for closely related genomes [16], while a well-designed similarity measure may allow more flexibility. As shown in [17], the complexity of computing the FFDCJ similarity was proven to be NP-hard, while the FFDCJ distance was already proven to be APX-hard. In the remainder of this paper, after preliminaries and a formal definition of the FFDCJ similarity problem, we first present an exact ILP algorithm to solve it. We then show the APX-hardness of the FFDCJ similarity problem and present four combinatorial heuristics, with computational experiments comparing their results to the ILP for datasets simulated by a framework for genome evolution.

A preliminary version of this paper appeared in the Proceedings of the 15th RECOMB Satellite Workshop on Comparative Genomics (RECOMB-CG 2017) [18].

Methods

Each segment (often called *gene*) g of a genome is an oriented DNA fragment and its two distinct *extremities* are called *tail* and *head*, denoted by g^t and g^h, respectively. A genome is composed of a set of chromosomes, each of which can be circular or linear and is a sequence of genes. Each one of the two extremities of a linear chromosome is called a *telomere*, represented by the symbol $\circ$. An *adjacency* in a chromosome is then either the extremity of a gene that is adjacent to a telomere, or a pair of consecutive gene extremities. As an example, observe that the adjacencies 5^h, 5^t2^t, 2^h4^t, 4^h3^t, 3^h6^t, 6^h1^h and 1^t can define a linear chromosome. Another representation of the same linear chromosome, flanked by parentheses for the sake of clarity, would be $(\circ\ -5\ 2\ 4\ 3\ 6\ -1\ \circ)$, in which the genes preceded by the minus sign $(-)$ have reverse orientation.

A *double cut and join* or DCJ operation applied to a genome A is the operation that cuts two adjacencies of A and joins the separated extremities in a different way, creating two new adjacencies. For example, a DCJ acting on two adjacencies *pq* and *rs* would create either the adjacencies *pr* and *qs*, or the adjacencies *ps* and *qr* (this could correspond to an inversion, a reciprocal translocation between two linear chromosomes, a fusion of two circular chromosomes, or an excision of a circular chromosome). In the same way, a DCJ acting on two adjacencies *pq* and r would create either *pr* and q, or p and *qr* (in this case, the operation could correspond to an inversion, a translocation, or a fusion of a circular and a linear chromosome). For the cases described so far we can notice that for each pair of cuts there are two possibilities of joining. There are two special cases of a DCJ operation, in which there is only one possibility of joining. The first is a DCJ acting on two adjacencies p and q, that would create only one new adjacency *pq* (that could represent a circularization of one or a fusion of two linear chromosomes). Conversely, a DCJ can act on only one adjacency *pq* and create the two adjacencies p and q (representing a linearization of a circular or a fission of a linear chromosome).

In the remainder of this section we extend the notation introduced in [17]. In general we consider the comparison of two distinct genomes, that will be denoted by A and B. Respectively, we denote by $\mathcal{A}$ the set of genes in genome A, and by $\mathcal{B}$ the set of genes in genome B.

Adjacency graph and family-based DCJ similarity

In most versions of the family-based setting the two genomes A and B have the same content, that is, $\mathcal{A} = \mathcal{B}$. When in addition there are no duplicates, that is, when there is exactly one representative of each family in each genome, we can easily build the *adjacency graph* of genomes A and B, denoted by $AG(A, B)$ [6]. It is a bipartite multigraph such that each partition corresponds to the set of adjacencies of one of the two input genomes, and an edge connects the same extremities of genes in both genomes. In other words, there is a one-to-one correspondence between the set of edges in $AG(A, B)$ and the set of gene extremities. Since the graph is bipartite and vertices have degree one or two, the adjacency graph is a collection of paths and even cycles. An example of an adjacency graph is presented in Fig. 1.

It is well known that a DCJ operation that modifies $AG(A, B)$ by increasing either the number of even cycles by one or the number of odd paths by two decreases the DCJ distance between genomes A and B [6]. This type of DCJ operation is said to be *optimal*. Conversely, if we are interested in a DCJ similarity measure between A and B, rather than a distance measure, then it should be increased by such an optimal DCJ operation. This suggests that a formula for a DCJ similarity between two genomes should correlate to the number of connected components (in the following just *components*) of the corresponding adjacency graph.

When the genomes A and B are identical, their corresponding adjacency graph is a collection of c 2-cycles and b 1-paths [6], so that $c + \frac{b}{2} = |\mathcal{A}| = |\mathcal{B}|$. This should be the upper bound of our DCJ similarity measure, and the

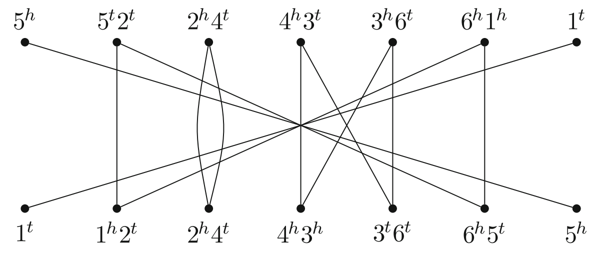

Fig. 1 The adjacency graph for the genomes $A = \{(\circ\ -5\ 2\ 4\ 3\ 6\ -1\ \circ)\}$ and $B = \{(\circ\ 1\ 2\ 4\ -3\ 6\ 5\ \circ)\}$

contribution of each component in the formula should be upper bounded by 1.

We know that an optimal operation can always be applied to adjacencies that belong to one of the two genomes and to one single component of $AG(A, B)$, until the graph becomes a collection of 2-cycles and 1-paths. In other words, each component of the graph can be *sorted*, that is, converted into a collection of 2-cycles and 1-paths independently of the other components. Furthermore, it is known that each of the following components – an even cycle with $2d + 2$ edges, or an odd path with $2d + 1$ edges, or an even path with $2d$ edges – can be sorted with exactly d optimal DCJ operations. Therefore, for the same d, components with more edges should actually have higher contributions in the DCJ similarity formula.

With all these considerations, the contribution of each component C in the formula is then defined to be its *normalized length* $\widehat{\ell}(C)$:

$$\widehat{\ell}(C) = \begin{cases} \frac{|C|}{|C|} = 1\,, & \text{if } C \text{ is a cycle}\,,\\ \frac{|C|}{|C|+1}\,, & \text{if } C \text{ is an odd path}\,,\\ \frac{|C|}{|C|+2}\,, & \text{if } C \text{ is an even path}\,. \end{cases}$$

Let $\mathcal{C}$ be the set of all components in $AG(A, B)$. The formula for the family-based DCJ similarity is the sum of their normalized lengths:

$$\mathrm{s}_{\mathrm{DCJ}}(A, B) = \sum_{C \in \mathcal{C}} \widehat{\ell}(C)\,. \tag{1}$$

Observe that $\mathrm{s}_{\mathrm{DCJ}}(A, B)$ is a positive value, indeed upper bounded by $|\mathcal{A}|$ (or, equivalently, by $|\mathcal{B}|$). In Fig. 1 the DCJ similarity is $\mathrm{s}_{\mathrm{DCJ}}(A, B) = 2 \cdot \frac{1}{2} + 3 \cdot 1 = 4$. The formula of Eq. 1 is the family-based version of the family-free DCJ similarity defined in [17], as we will see in the following subsections.

Gene similarity graph

In the family-free setting, each gene in each genome is represented by a unique (signed) symbol, thus $\mathcal{A} \cap \mathcal{B} = \emptyset$ and the cardinalities $|\mathcal{A}|$ and $|\mathcal{B}|$ may be distinct. Let a be a gene in A and b be a gene in B, then their *normalized gene similarity* is given by some value $\sigma(a, b)$ such that $0 \le \sigma(a, b) \le 1$.

We can represent the gene similarities between the genes of genome A and the genes of genome B with respect to σ in the so called *gene similarity graph* [15], denoted by $GS_\sigma(A, B)$. This is a weighted bipartite graph whose partitions $\mathcal{A}$ and $\mathcal{B}$ are the sets of (signed) genes in genomes A and B, respectively. Furthermore, for each pair of genes (a, b) such that $a \in \mathcal{A}$ and $b \in \mathcal{B}$, if $\sigma(a, b) > 0$ then there is an edge e connecting a and b in $GS_\sigma(A, B)$ whose weight is $\sigma(e) := \sigma(a, b)$. An example of a gene similarity graph is given in Fig. 2.

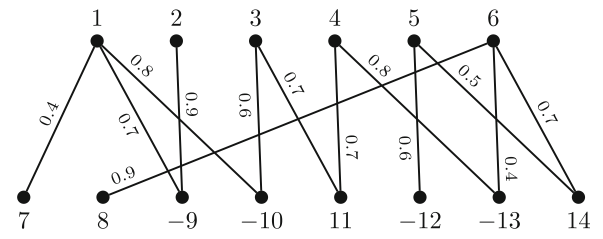

Fig. 2 Representation of a gene similarity graph $GS_\sigma(A, B)$ for two unichromosomal linear genomes $A = \{(\circ\ 1\ 2\ 3\ 4\ 5\ 6\ \circ)\}$ and $B = \{(\circ\ 7\ 8\ {-9}\ {-10}\ 11\ {-12}\ {-13}\ 14\ \circ)\}$

Weighted adjacency graph

The *weighted adjacency graph* $AG_\sigma(A, B)$ of two genomes A and B has a vertex for each adjacency in A and a vertex for each adjacency in B. For a gene a in A and a gene b in B with gene similarity $\sigma(a, b) > 0$ there is one edge e^h connecting the vertices containing the two heads a^h and b^h and one edge e^t connecting the vertices containing the two tails a^t and b^t. The weight of each of these edges is $\sigma\left(e^h\right) = \sigma\left(e^t\right) = \sigma(a, b)$. Differently from the simple adjacency graph, the weighted adjacency graph cannot be easily decomposed into cycles and paths, since its vertices can have degree greater than 2. As an example, the weighted adjacency graph corresponding to the gene similarity graph of Fig. 2 is given in Fig. 3.

We denote by $w(G)$ the weight of a graph or subgraph G, that is given by the sum of the weights of all its edges, that is, $w(G) = \sum_{e \in G} \sigma(e)$. Observe that, for each edge $e \in GS_\sigma(A, B)$, we have two edges of weight $\sigma(e)$ in $AG_\sigma(A, B)$, thus, the total weight of the weighted adjacency graph is $w\left(AG_\sigma(A, B)\right) = 2\, w\left(GS_\sigma(A, B)\right)$.

Reduced genomes

Let A and B be two genomes and let $GS_\sigma(A, B)$ be their gene similarity graph. Now let $M = \{e_1, e_2, \ldots, e_n\}$ be a matching in $GS_\sigma(A, B)$ and denote by $w(M) = \sum_{e_i \in M} \sigma(e_i)$ the weight of M, that is the sum of its edge weights. Since the endpoints of each edge $e_i = (a, b)$ in M are not saturated by any other edge of M, we can unambiguously define the function $\ell^M(a) = \ell^M(b) = i$ to relabel each vertex in A and B [17]. The *reduced genome* A^M is obtained by deleting from A all genes not saturated by M, and renaming each saturated gene a to $\ell^M(a)$, preserving its orientation (sign). Similarly, the reduced genome B^M is obtained by deleting from B all genes that are not saturated by M, and renaming each saturated gene b to $\ell^M(b)$, preserving its orientation. Observe that the set of genes in A^M and in B^M is $\mathcal{G}(M) = \left\{\ell^M(g) : g \text{ is saturated by the matching } M\right\} = \{1, 2, \ldots, n\}$.

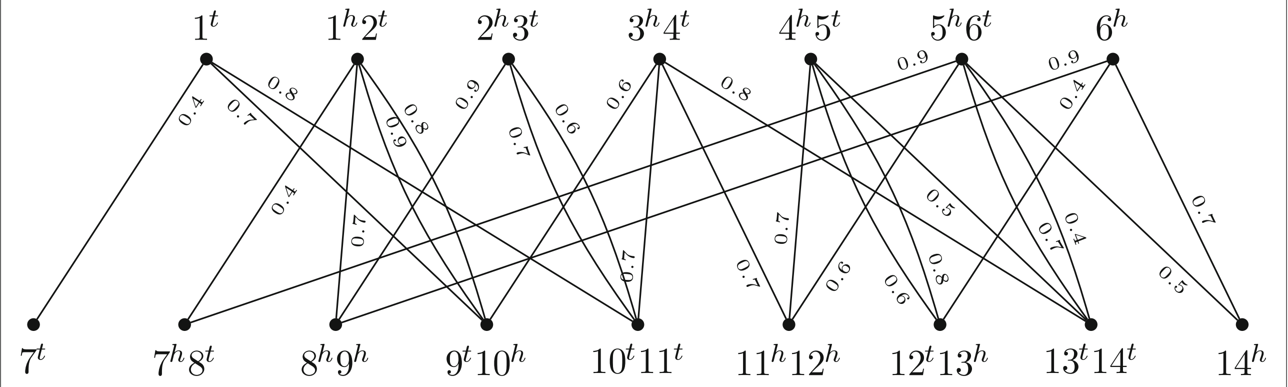

Fig. 3 The weighted adjacency graph $AG_\sigma(A,B)$ for two unichromosomal linear genomes $A = \{(\circ\ 1\ 2\ 3\ 4\ 5\ 6\ \circ)\}$ and $B = \{(\circ\ 7\ 8\ {-9}\ {-10}\ 11\ {-12}\ {-13}\ 14\ \circ)\}$

Weighted adjacency graph of reduced genomes

Let A^M and B^M be the reduced genomes for a given matching M of $GS_\sigma(A,B)$. The weighted adjacency graph $AG_\sigma\left(A^M,B^M\right)$ can be obtained from $AG_\sigma(A,B)$ by deleting all edges that are not elements of M and relabeling the adjacencies according to ℓ^M. Vertices that have no connections are then also deleted from the graph. Another way to obtain the same graph is building the adjacency graph of A^M and B^M and adding weights to the edges as follows. For each gene i in $\mathcal{G}(M)$, both edges $i^t i^t$ and $i^h i^h$ inherit the weight of edge e_i in M, that is, $\sigma\left(i^t i^t\right) = \sigma\left(i^h i^h\right) = \sigma(e_i)$. Consequently, the graph $AG_\sigma\left(A^M,B^M\right)$ is also a collection of paths and even cycles and differs from $AG\left(A^M,B^M\right)$ only by the edge weights.

For each edge $e \in M$, we have two edges of weight $\sigma(e)$ in $AG_\sigma\left(A^M,B^M\right)$, therefore $w\left(AG_\sigma\left(A^M,B^M\right)\right) = 2\,w(M)$. Examples of weighted adjacency graphs of reduced genomes are shown in Fig. 4.

The family-free DCJ similarity

For a given matching M in $GS_\sigma(A,B)$, a first formula for the weighted DCJ (wDCJ) similarity s_σ of the reduced genomes A^M and B^M was proposed in [15] only considering the cycles of $AG_\sigma\left(A^M,B^M\right)$. After that, this definition was modified and extended in [17], in order to consider all components of the weighted adjacency graph.

First, let the *normalized weight* $\widehat{w}(C)$ of a component C of $AG_\sigma\left(A^M,B^M\right)$ be:

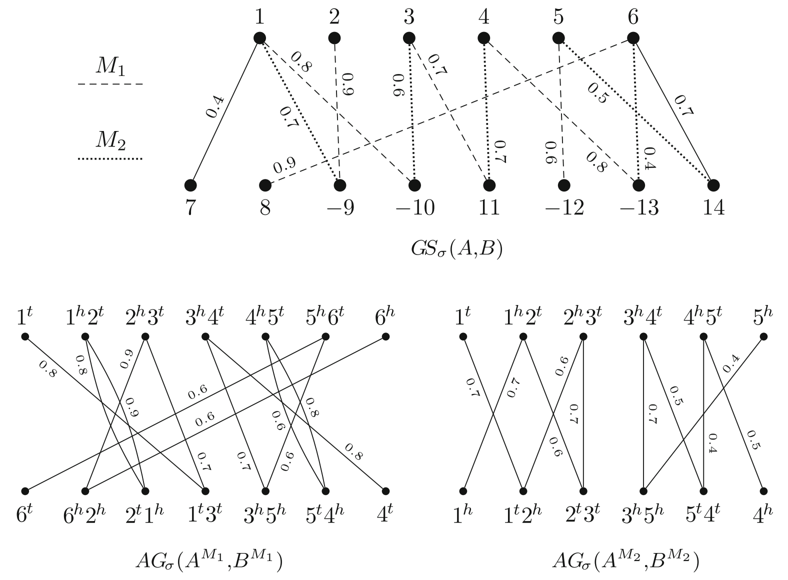

Fig. 4 Considering, as in Fig. 2, the genomes $A = \{(\circ\ 1\ 2\ 3\ 4\ 5\ 6\ \circ)\}$ and $B = \{(\circ\ 7\ 8\ {-9}\ {-10}\ 11\ {-12}\ {-13}\ 14\ \circ)\}$, let M_1 (dashed edges) and M_2 (dotted edges) be two distinct maximal matchings in $GS_\sigma(A,B)$, shown in the upper part. The two resulting weighted adjacency graphs $AG_\sigma\left(A^{M_1},B^{M_1}\right)$, that has two cycles and two even paths, and $AG_\sigma\left(A^{M_2},B^{M_2}\right)$, that has two odd paths, are shown in the lower part

$$\widehat{w}(C) = \begin{cases} \frac{w(C)}{|C|}\,, & \text{if } C \text{ is a cycle}\,, \\ \frac{w(C)}{|C|+1}\,, & \text{if } C \text{ is an odd path}\,, \\ \frac{w(C)}{|C|+2}\,, & \text{if } C \text{ is an even path}\,. \end{cases}$$

Let $\mathcal{C}$ be the set of all components in $AG_\sigma\left(A^M, B^M\right)$. Then the wDCJ similarity s_σ is given by the following formula [17]:

$$s_\sigma\left(A^M, B^M\right) = \sum_{C\in\mathcal{C}} \widehat{w}(C)\,. \tag{2}$$

Observe that, when the weights of all edges in M are equal to 1, this formula is equivalent to the one in Eq. 1.

The goal now is to compute the family-free DCJ similarity, i.e., to find a matching in $GS_\sigma(A,B)$ that maximizes s_σ. However, although $s_\sigma\left(A^M, B^M\right)$ is a positive value upper bounded by $|M|$, the behaviour of the wDCJ similarity does not correlate with the size of the matching, since smaller matchings, that possibly discard gene assignments, can lead to higher wDCJ similarities [17]. For this reason, the wDCJ similarity function is restricted to *maximal matchings* only, ensuring that no pair of genes with positive gene similarity score is simply discarded, even though it might decrease the overall wDCJ similarity. We then have the following optimization problem:

Problem FFDCJ-SIMILARITY(A,B): Given genomes A and B and their gene similarities σ, calculate their family-free DCJ similarity

$$s_{\text{FFDCJ}}(A,B) = \max_{M\in\mathbb{M}} \left\{s_\sigma\left(A^M, B^M\right)\right\}, \tag{3}$$

where $\mathbb{M}$ is the set of all maximal matchings in $GS_\sigma(A,B)$.

Problem FFDCJ-SIMILARITY is NP-hard [17]. Moreover, one can directly correlate the problem to the adjacency similarity problem, where the goal is to maximize the number of preserved adjacencies between two given genomes [11, 19]. However, since there the objective is to maximize the number of cycles of length 2, even an approximation for the adjacency similarity problem is not a good algorithm for the FFDCJ-SIMILARITY problem, where cycles of higher lengths are possible in the solution [20].

Capping telomeres

A very useful preprocessing to $AG_\sigma(A,B)$ is the *capping* of telomeres, a general technique for simplifying algorithms that handle genomes with linear chromosomes, commonly used in the context of family-based settings [4, 5, 21]. Given two genomes A and B with i and j linear chromosomes, respectively, for each vertex representing only one extremity we add a *null extremity* τ to it (e.g., 1^t of Fig. 4 becomes $\tau 1^t$). Furthermore, in order to add the same number of null extremities to both genomes, $|j-i|$ *null adjacencies* $\tau\tau$ (composed of two null extremities) are added to genome A, if $i < j$, or to genome B, if $j < i$. Finally, for each null extremity of a vertex in A we add to $AG_\sigma(A,B)$ a *null edge* with weight 0 to each null extremity of vertices in B. Consequently, after capping of telomeres the graph $AG_\sigma(A,B)$ has no vertex of degree one. Notice that, if before the capping p was a path of weight w connecting telomeres in $AG_\sigma(A,B)$, then after the capping p will be part of a cycle closed by null extremities with normalized weight $\frac{w}{|p|+1}$ if p is an odd path, or of normalized weight $\frac{w}{|p|+2}$ if p is an even path. In any of the two cases, the normalized weight is consistent with the wDCJ similarity formula in Eq. 2.

Results and discussion

An exact Algorithm

In order to exactly compute the family-free DCJ similarity between two given genomes, we propose an integer linear program (ILP) formulation that is similar to the one for the family-free DCJ distance given in [17]. It adopts the same notation and also uses an approach to solve the maximum cycle decomposition problem as in [13].

Let A and B be two genomes, let $G = GS_\sigma(A,B)$ be their gene similarity graph, and let X_A and X_B be the extremity sets (including null extremities) with respect to A and B for the capped adjacency graph $AG_\sigma(A,B)$, respectively. The weight $w(e)$ of an edge e in G is also denoted by w_e. For the ILP formulation, an extension $H = (V_H, E_H)$ of the capped weighted adjacency graph $AG_\sigma(A,B)$ is defined such that $V_H = X_A \cup X_B$, and $E_H = E_m \cup E_a \cup E_s$ has three types of edges: (*i*) *matching edges* that connect two extremities in different extremity sets, one in X_A and the other in X_B, if they are null extremities or there exists an edge connecting these genes in G; the set of matching edges is denoted by E_m; (*ii*) *adjacency edges* that connect two extremities in the same extremity set if they form an adjacency; the set of adjacency edges is denoted by E_a; and (*iii*) *self edges* that connect two extremities of the same gene in an extremity set; the set of self edges is denoted by E_s. Matching edges have weights defined by the normalized gene similarity σ, all adjacency and self edges have weight 0. Notice that any edge in G corresponds to two matching edges in H.

The description of the ILP follows. For each edge e in H, we create a binary variable x_e to indicate whether e will be in the final solution. We require first that each adjacency edge be chosen:

$$x_e = 1, \qquad \forall\, e \in E_a.$$

Now we rename each vertex in H such that $V_H = \{v_1, v_2, \ldots, v_k\}$ with $k = |V_H|$. We require that each of these vertices be adjacent to exactly one matching or self edge:

$$\sum_{e=v_r v_t \in E_m \cup E_s} x_e = 1, \forall\, v_r \in X_A, \quad \text{and}$$
$$\sum_{e=v_r v_t \in E_m \cup E_s} x_e = 1, \forall\, v_t \in X_B.$$

Then, we require that the final solution be valid, meaning that if one extremity of a gene in A is assigned to an extremity of a gene in B, then the other extremities of these two genes have to be assigned as well:

$$x_{a^h b^h} = x_{a^t b^t}, \qquad \forall\, ab \in E_G.$$

We also require that the matching be maximal. This can easily be ensured if we guarantee that at least one of the vertices connected by an edge in the gene similarity graph be chosen, which is equivalent to not allowing both of the corresponding self edges in the weighted adjacency graph be chosen:

$$x_{a^h a^t} + x_{b^h b^t} \leq 1, \qquad \forall\, ab \in E_G.$$

To count the number of cycles, we use the same strategy as described in [13]. For each vertex v_i we define a variable y_i that labels v_i such that

$$0 \leq y_i \leq i, \qquad 1 \leq i \leq k.$$

We also require that adjacent vertices have the same label, forcing all vertices in the same cycle to have the same label:

$$y_i \leq y_j + i \cdot (1 - x_e), \qquad \forall\, e = v_i v_j \in E_H,$$
$$y_j \leq y_i + j \cdot (1 - x_e), \qquad \forall\, e = v_i v_j \in E_H.$$

We create a binary variable z_i, for each vertex v_i, to verify whether y_i is equal to its upper bound i:

$$i \cdot z_i \leq y_i, \qquad 1 \leq i \leq k.$$

Since all variables y_i in the same cycle have the same label but a different upper bound, only one of the y_i can be equal to its upper bound i. This means that z_i is 1 if the cycle with vertex i as representative is used in a solution.

Now, let $L = \{2j : j = 1, \ldots, n\}$ be the set of possible cycle lengths in H, where $n := \min(|A|, |B|)$. We create the binary variable x_{ei} to indicate whether e is in i, for each $e \in E_H$ and each cycle i. We also create the binary variable x_{ei}^{ℓ} to indicate whether e belongs to i and the length of cycle i is ℓ, for each $e \in E_H$, each cycle i, and each $\ell \in L$.

We require that if an edge e belongs to a cycle i, then it can be true for only one length $\ell \in L$. Thus,

$$\sum_{\ell \in L} x_{ei}^{\ell} \leq x_{ei}, \qquad \forall\, e \in E_H \text{ and } 1 \leq i \leq k. \tag{4}$$

We create another binary variable z_i^{ℓ} to indicate whether cycle i has length ℓ. Then $\ell \cdot z_i^{\ell}$ is an upper bound for the total number of edges in cycle i of length ℓ:

$$\sum_{e \in E_M} x_{ei}^{\ell} \leq \ell \cdot z_i^{\ell}, \qquad \forall\, \ell \in L \text{ and } 1 \leq i \leq k.$$

The length of a cycle i is given by $\ell \cdot z_i^{\ell}$, for $i = 1, \ldots, k$ and $\ell \in L$. On the other hand, it is the total amount of matching edges e in cycle i. That is,

$$\sum_{\ell \in L} \ell \cdot z_i^{\ell} = \sum_{e \in E_m} x_{ei}, \qquad 1 \leq i \leq k.$$

We have to ensure that each cycle i must have just one length:

$$\sum_{\ell \in L} z_i^{\ell} = z_i, \qquad 1 \leq i \leq k.$$

Now we create the binary variable y_{ri} to indicate whether the vertex v_r is in cycle i. Thus, if $x_{ei} = 1$, i.e., if the edge $e = v_r v_t$ in H is chosen in cycle i, then $y_{ri} = 1 = y_{ti}$ (and $x_e = 1$ as well). Hence,

$$\left.\begin{aligned} x_{ei} &\leq x_e, \\ x_{ei} &\leq y_{ri}, \\ x_{ei} &\leq y_{ti}, \\ x_{ei} &\geq x_e + y_{ri} + y_{ti} - 2, \end{aligned}\right\} \quad \forall\, e = v_r v_t \in E_H \text{ and } 1 \leq i \leq k. \tag{5}$$

Since y_r is an integer variable, we associate y_r to the corresponding binary variable y_{ri}, for any vertex v_r belonging to cycle i:

$$y_r = \sum_{i=1}^{r} i \cdot y_{ri}, \qquad \forall\, v_r \in V_H.$$

Furthermore, we must ensure that each vertex v_r may belong to at most one cycle:

$$\sum_{i=1}^{r} y_{ri} \leq 1, \qquad \forall\, v_r \in V_H.$$

Finally, we set the objective function as follows:

$$\text{maximize} \quad \sum_{i=1}^{k} \sum_{\ell \in L} \sum_{e \in E_m} \frac{w_e x_{ei}^{\ell}}{\ell}.$$

Note that, with this formulation, we do not have any path as a component. Therefore, the objective function above is exactly the family-free DCJ similarity $\mathrm{S_{FFDCJ}}(A,B)$ as defined in Eqs. (2) and (3).

Notice that the ILP formulation has $O(N^4)$ variables and $O\left(N^3\right)$ constraints, where $N = |A| + |B|$. The number of variables is proportional to the number of variables x_{ei}^{ℓ}, and the number of constraints is upper bounded by (4) and (5).

APX-hardness and heuristics

In this section we first state that problem FFDCJ-SIMILARITY is APX-hard and provide a lower bound for the approximation ratio.

Theorem 1 FFDCJ-SIMILARITY *is APX-hard and cannot be approximated with approximation ratio better than* $22/21 = 1.0476\ldots$, *unless* $P = NP$.

Proof See Additional file 1. □

We now propose four heuristic algorithms to compute the family-free DCJ similarity of two given genomes: one that is directly derived from a maximum matching of the gene similarity graph GS_σ and three greedy-like heuristics that, according to different criteria, select cycles from the weighted adjacency graph AG_σ, such that the cycles selected by each heuristic induce a matching in GS_σ.

Maximum matching

In the first heuristic, shown in Algorithm 1 (MAXIMUM-MATCHING), we find a maximum weighted bipartite matching M in GS_σ by the Hungarian Method, also known as Kuhn-Munkres Algorithm [22–24]. Given the matching M, it is straightforward to obtain the reduced genomes A^M and B^M and return the similarity value $s_\sigma\left(A^M, B^M\right)$.

Algorithm 1 MAXIMUM-MATCHING(A, B, σ)

```
Input: genomes $A$ and $B$, gene similarity function $\sigma$
Output: a family-free DCJ similarity between $A$ and $B$
1: Build the gene similarity graph $GS_\sigma(A,B)$.
2: Obtain a maximum weighted matching $M$ in $GS_\sigma(A,B)$ defining reduced genomes $A^M$ and $B^M$.
3: Build the capped weighted adjacency graph $AG_\sigma\left(A^M,B^M\right)$ of the reduced genomes.
4: Let $\mathcal{C}$ be the set of all cycles in $AG_\sigma\left(A^M,B^M\right)$.
5: Return $s_\sigma\left(A^M,B^M\right) = \sum_{C\in\mathcal{C}} \widehat{w}(C)$.
```

For the implementantion of this heuristic we cast similarity values (floating point edge weights in $[0,1]$) in $GS_\sigma(A,B)$ to integers by multiplying them by some power of ten, depending on the precision of similarity values. Given real or general simulated instances, and for a power of ten large enough, this operation has little impact on the optimality of the weighted matching M for the original weights in $GS_\sigma(A,B)$ obtained from the Kuhn-Munkres algorithm, i.e., the weight of M for the original weights in $GS_\sigma(A,B)$ is optimal or near-optimal since only less significant digits are not considered.

Greedy heuristics

Before describing the greedy heuristics, we need to introduce the following concepts. We say that two edges in $AG_\sigma(A,B)$ are *consistent* if one connects the head and the other connects the tail of the same pair of genes, or if they connect extremities of distinct genes in both genomes. Otherwise they are *inconsistent*. A set of edges, in particular a cycle, is consistent if it has no pair of inconsistent edges. A set of cycles is consistent if the union of all of their edges is consistent. Observe that a consistent set of cycles in $AG_\sigma(A,B)$ induces a matching in $GS_\sigma(A,B)$.

Each one of the three greedy algorithms selects disjoint and consistent cycles in the capped $AG_\sigma(A,B)$. The consistent cycles are selected from the set of all cycles of $AG_\sigma(A,B)$, that is obtained in Step 4 of each heuristic (see Algorithms 2, 3 and 4 below), using a cycle enumeration algorithm by Hawick and James [25], which is based on Johnson's algorithm [26]. For this reason, the running time of our heuristics is potentially exponential in the number of vertices of $AG_\sigma(A,B)$.

In the three heuristics, after completing the cycle selection by iterating over the set of all cycles of $AG_\sigma(A,B)$, the induced matching M in $GS_\sigma(A,B)$ could still be non-maximal. Whenever this occurs, among the genes that are unsaturated by M, we can identify *disposable genes* by one of the two following conditions:

1. Any unsaturated gene in $GS_\sigma(A,B)$ that is connected only to saturated genes, is a disposable gene;
2. For a given set of vertices $S \subseteq \mathcal{A}$ (or $S \subseteq \mathcal{B}$) in $GS_\sigma(A,B)$ such that, for the set of connected genes $\mathrm{N}(S)$, we have $|S| > |\mathrm{N}(S)|$ (Hall's theorem), then any subset of size $|S| - |\mathrm{N}(S)|$ of unsaturated genes of S can be set as disposable genes. In our implementation we choose those $|S| - |\mathrm{N}(S)|$ unsaturated genes with the smallest labels. Such $S \subseteq \mathcal{A}$ can be found as follows. Let v be the set of vertices saturated by M, and let M' be a maximum cardinality matching in $GS_\sigma(A,B) \setminus v$. Consider the sets $\mathcal{A}' = \mathcal{A} \setminus v$ and $\mathcal{B}' = \mathcal{B} \setminus v$. Now let $GS'_\sigma(A,B)$ be a directed bipartite graph on the vertex set $\mathcal{A}' \cup \mathcal{B}'$, which includes the edges of M' oriented from $\mathcal{B}'$ to $\mathcal{A}'$ and the remaining edges of $GS_\sigma(A,B) \setminus v$ oriented from $\mathcal{A}'$ to $\mathcal{B}'$, and let $U \subseteq \mathcal{A}'$ be the set of vertices of $\mathcal{A}'$ unsaturated by M'. $S \subseteq \mathcal{A}$ is the corresponding set of vertices reachable from U in $GS'_\sigma(A,B)$, if any. $S \subseteq \mathcal{B}$ can be found analogously.

If there is no consistent cycle to be selected and the matching M is still non-maximal, new consistent cycles appear in $AG_\sigma(A,B)$ after the deletion of all identified disposable genes (see Fig. 5). In order to delete a disposable gene g, we need to remove from $AG_\sigma(A,B)$ the edges corresponding to extremities g^t or g^h and "merge"

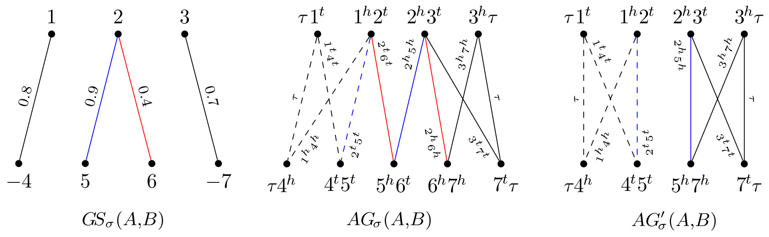

Fig. 5 Consider genomes $A = \{(\circ\ 1\ 2\ 3\ \circ)\}$ and $B = \{(\circ\ -4\ 5\ 6\ -7\ \circ)\}$ and their gene similarity graph $GS_\sigma(A,B)$. The selection of the dashed cycle in $AG_\sigma(A,B)$ adds to the matching M in $GS_\sigma(A,B)$ the edges connecting gene 1 to gene 4 and gene 2 to gene 5. After this selection, although the matching M is not yet maximal, there are no more consistent cycles in $AG_\sigma(A,B)$. Observe that in $GS_\sigma(A,B)$ gene 6 is unsaturated and its single neighbor - gene 2 - is already saturated. Since gene 6 can no longer be saturated by M, it is a disposable gene and is deleted from $AG_\sigma(A,B)$, resulting in $AG'_\sigma(A,B)$, where a new consistent cycle appears. The selection of this new cycle adds to the matching M the edge connecting gene 3 to gene 7. Both $AG_\sigma(A,B)$ and $AG'_\sigma(A,B)$ have a simplified representation, in which the edge weights, as well as two of the four null edges of the capping, are omitted. Furthermore, for the sake of clarity, in this simplified representation each edge has a label describing the extremities connected by it

the two vertices that represent these extremities. Every time disposable genes are deleted from $AG_\sigma(A,B)$, a new iteration of the algorithms starts from Step 4 (see again Algorithms 2, 3 and 4). This procedure assures that, in each one of the three algorithms, the final set of selected cycles defines a maximal matching M, such that $AG_\sigma\left(A^M, B^M\right)$ is exactly the union of those selected cycles.

Best density The best density heuristic is shown in Algorithm 2 (Greedy-Density). The *density* of a cycle C is given by $\frac{w(C)}{|C|^2}$ (its weight divided by the square of its length). The cycles of $AG_\sigma(A,B)$ are arranged in decreasing order of their densities, and consistent cycles are selected following this order.

Algorithm 2 Greedy-Density(A, B, σ)

```
Input: genomes $A$ and $B$, gene similarity function $\sigma$
Output: a family-free DCJ similarity between $A$ and $B$
1: $M := \emptyset$; $\mathcal{C} := \emptyset$.
2: Build the gene similarity graph $GS_\sigma(A,B)$.
3: Build the capped weighted adjacency graph $AG_\sigma(A,B)$.
4: for $\ell = 10, 20, \ldots$, maximum cycle length possible do
5:     List all cycles of $AG_\sigma(A,B)$ of length at most $\ell$ in decreasing order of their densities.
6:     While it is possible, select the best density consistent cycle $C$ that is also consistent with all cycles in $\mathcal{C}$ and add it to $\mathcal{C}$, let $AG_\sigma(A,B) := AG_\sigma(A,B) \setminus C$, update $M$ by adding the new gene connections induced by $C$.
7: If $M$ is not a maximal matching of $GS_\sigma(A,B)$, find and delete disposable genes from $AG_\sigma(A,B)$ and go back to Step 4.
8: Return $\sum_{C \in \mathcal{C}} \widehat{w}(C)$.
```

Since the number of cycles of any length may be exponential in the size of the input graph, in our implementation we add a heuristic in which initially the search is restricted to cycles of length up to ten. Then, as long as the obtained matching is not maximal, Steps 4 to 7 are repeated, while gradually increasing the allowed maximum cycle length in steps of ten.

Best length The best length heuristic is shown in Algorithm 3 (Greedy-Length). The cycles of $AG_\sigma(A,B)$ are found in increasing order of their lengths, and ties are broken by the decreasing order of their weights. Here we first find and select cycles of length 2, then of length 4, and so on, for each fixed length iterating over the set of all cycles in decreasing order of their weights. Consistent cycles are selected following this procedure.

Algorithm 3 Greedy-Length(A, B, σ)

```
Input: genomes $A$ and $B$, gene similarity function $\sigma$
Output: a family-free DCJ similarity between $A$ and $B$
1: $M := \emptyset$; $\mathcal{C} := \emptyset$.
2: Build the gene similarity graph $GS_\sigma(A,B)$.
3: Build the capped weighted adjacency graph $AG_\sigma(A,B)$.
4: for $\ell = 2, 4, \ldots$, maximum cycle length possible do
5:     List all cycles of $AG_\sigma(A,B)$ of length $\ell$ in decreasing order of their weights.
6:     While it is possible, select the best weight consistent cycle $C$ that is also consistent with all cycles in $\mathcal{C}$ and add it to $\mathcal{C}$, let $AG_\sigma(A,B) := AG_\sigma(A,B) \setminus C$, update $M$ by adding the new gene connections induced by $C$.
7: If $M$ is not a maximal matching of $GS_\sigma(A,B)$, find and delete disposable genes from $AG_\sigma(A,B)$ and go back to Step 4.
8: Return $\sum_{C \in \mathcal{C}} \widehat{w}(C)$.
```

Best length with weighted maximum independent set The best length heuristic with WMIS is shown in Algorithm 4 (GREEDY-WMIS) and is a variation of GREEDY-LENGTH. Instead of selecting cycles of greater weights for a fixed length, this algorithm selects the greatest amount of cycles for a fixed length by a WMIS algorithm. The heuristic builds a *cycle graph* where each vertex is a cycle of $AG_\sigma(A,B)$, the weight of a vertex is the weight of the cycle it represents and two vertices are adjacent if the cycles they represent are inconsistent. The heuristic tries to find next an independent set with the greatest weight in the cycle graph. Since this graph is not d-claw-free for any fixed d, the WMIS algorithm [27] does not guarantee any fixed ratio.

Algorithm 4 GREEDY-WMIS(A, B, σ)

```
Input: genomes $A$ and $B$, gene similarity function $\sigma$
Output: a family-free DCJ similarity between $A$ and $B$
1: $M := \emptyset$; $\mathcal{C} := \emptyset$.
2: Build the gene similarity graph $GS_\sigma(A,B)$.
3: Build the capped weighted adjacency graph $AG_\sigma(A,B)$.
4: for $\ell = 2, 4, \ldots,$ maximum cycle length possible do
5:     List all cycles of $AG_\sigma(A,B)$ of length $\ell$.
6:     Select a set $\mathcal{C}'$ of consistent cycles trying to maximize the sum of weights by a WMIS algorithm and add $\mathcal{C}'$ to $\mathcal{C}$, let $AG_\sigma(A,B) := AG_\sigma(A,B) \setminus \mathcal{C}'$, update $M$ by adding the new gene connections induced by $\mathcal{C}'$.
7: If $M$ is not a maximal matching of $GS_\sigma(A,B)$, find and delete disposable genes from $AG_\sigma(A,B)$ and go back to Step 4.
8: Return $\sum_{C \in \mathcal{C}} \widehat{w}(C)$.
```

Experimental results

Experiments for the ILP and our heuristics were conducted on an Intel i7-4770 3.40GHz machine with 16 GB of memory. In order to do so, we produced simulated datasets by the Artificial Life Simulator (ALF) [28] and obtained real genome data from NCBI, using the FFGC tool [29] to obtain similarity scores between genomes. Gurobi Optimizer 7.0 was set to solve ILP instances with default parameters, time limit of 1800 s and 4 threads, and the heuristics were implemented in C++.

Simulated data

We generated datasets with 10 genome samples each, running pairwise comparisons between all genomes in the same dataset. Each dataset has genomes of sizes around 25, 50 or 1000 (the latter used only for running the heuristics), generated based on a sample from the tree of life with 10 leaf species and PAM distance of 100 from the root to the deepest leaf. Gamma distribution with parameters $k = 3$ and $\theta = 133$ was used for gene length distribution. For amino acid evolution we used the WAG substitution model with default parameters and the preset of Zipfian indels with rate 0.00005. Regarding genome level events, we allowed gene duplications and gene losses with rate 0.002, and reversals and transpositions (which ALF refers to as translocations) with rate 0.0025, with at most 3 genes involved in each event. To test different proportions of genome level events, we also generated simulated datasets with 2- and 5-fold increase for reversal and transpositions rates.

Results are summarized in Table 1. Each dataset is composed of 10 genomes, totaling 45 comparisons of pairs per dataset. Rate $r = 1$ means the default parameter set for genome level events, while $r = 2$ and $r = 5$ mean the 2- and 5-fold increase of rates, respectively. For the ILP the table shows the average time for instances for which an optimal solution was found, the number of instances for which the optimizer did not find an optimal solution within the given time limit and, for the latter class of instances, the average relative gap between the best solution found and the upper bound found by the solver, calculated by $\left(\frac{\text{upper bound}}{\text{best solution}} - 1\right) \times 100$. For our heuristics, the running time for all instances of sizes 25 and 50 was negligible, therefore the table shows only the average relative gap between the solution found and the upper bound given by the ILP solver (if any).

Results clearly show the average relative gap of heuristics increases proportionally to the rate of reversals and

Table 1 Results of experiments for simulated genomes

	ILP			MAXIMUM-MATCHING	GREEDY-DENSITY	GREEDY-LENGTH	GREEDY-WMIS
	Time (s)	Not finished	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)
25 genes, $r = 1$	19.50	0	–	16.26	5.03	5.84	5.97
25 genes, $r = 2$	84.60	2	69.21	58.69	30.77	43.57	43.00
25 genes, $r = 5$	49.72	0	–	81.39	43.83	55.38	55.38
50 genes, $r = 1$	265.23	12	23.26	63.02	24.76	27.86	26.94
50 genes, $r = 2$	463.50	29	38.12	123.71	65.41	66.52	64.78
50 genes, $r = 5$	330.88	29	259.72	281.70	177.58	206.60	206.31

transpositions. This is expected, as higher mutation rates often result in higher normalized weights on longer cycles, thus the association of genes with greater gene similarity scores will be subject to the selection of longer cycles. Interestingly, for some larger instances the relative gap for heuristics is very close to the values obtained by the ILP solver, suggesting the use of heuristics may be a good alternative for some classes of instances or could help the solver finding lower bounds quickly. It is worth noting that the GREEDY-DENSITY heuristic found solutions with gap smaller than 1% for 38% of the instances with 25 genes.

In a single instance (25 genes, $r = 2$), the gap between the best solution found and the upper bound was much higher for the ILP solver and for the heuristics. This instance in particular is precisely the one with the largest number of edges in $GS_\sigma(A, B)$ in the dataset. This may indicate that a moderate increase in degree of vertices (1.3 on average to 1.8 in this case) may result in much harder instances for the solver and the heuristics, as after half of the time limit the solver attained no significant improvement on solutions found, and the heuristics returned solutions with a gap even higher.

We also simulated 10 genomes of sizes around 50, with PAM distance of 15 from the root to the deepest leaf, therefore evolutionarily "closer" to each other and for which higher similarity values are expected. For these genomes the default rates were multiplied by ten (10-fold) for Zipfian indels, gene duplications, gene losses, reversals and transpositions, otherwise there would be no significative difference between them. The exact ILP algorithm found an optimal solution for only 4 of the 45 instances, taking 840.59 s on average. For the remaining instances, where the ILP did not finish within the time limit, the average gap is 329.53%. Regarding the heuristics (Table 2), that all run in negligible time, GREEDY-DENSITY outperforms the others, with an average gap of 163% compared to the best upper bound found by the ILP solver. Surprisingly, values returned by greedy heuristics are better than values obtained by the ILP for these instances. Results again suggest that the ILP could benefit greatly from heuristics by using their results as initial lower bounds. Moreover, for some groups of instances even heuristics alone can obtain excellent results.

Although we have no upper bounds for comparing the results of our heuristics for genome sizes around 1000, they are still very fast. For these genomes we analyze the MAXIMUM-MATCHING algorithm separately afterwards, taking into account for now only the other three heuristics. The average running times are 0.30 s, 15.11 s and 12.16 s for GREEDY-DENSITY, GREEDY-LENGTH and GREEDY-WMIS, respectively, showing nevertheless little difference on results.

However, in 25% of the instances with $r = 5$, the difference from the best to the worst solutions provided by these heuristics varied between 10% and 24%, the best of which were given by GREEDY-DENSITY. That is probably because, instead of prioritizing shorter cycles, GREEDY-DENSITY attempts to balance both normalized weight and length of the selected cycles. The average running times for the instances with $r = 5$ are 1.84 s, 76.02 s and 80.67 s for GREEDY-DENSITY, GREEDY-LENGTH and GREEDY-WMIS, respectively.

Still for genomes of size around 1000 and $r = 5$, the MAXIMUM-MATCHING heuristic is the fastest, with an average running time of 1.70 s. Despite being the best heuristic for a few cases, the similarity value given by this heuristic is merely 27% of the value given by the best heuristic, on average. While the MAXIMUM-MATCHING heuristic is clearly not useful for calculating similarity values, these results show how significant it is choose cycles with the best normalized weights versus prioritizing edges with best weights in the gene similarity graph for the FFDCJ-SIMILARITY problem. Since this property of the MAXIMUM-MATCHING somehow reflects the strategy of family-based comparative genomics, this observation indicates an advantage of family-free analysis compared to family-based analysis.

To better understand how cycles scale, we generated 5-fold larger instances (up to 10000 genes), running the GREEDY-DENSITY heuristic. Results show that most of the cycles found are of short lengths compared to the genome sizes and in practice their number does not increase exponentially, providing some insight on why our heuristics are fast.

Finally, as expected, experiments for genomes simulated with different parameters indicate the FFDCJ similarity decreases as the PAM distance or the rates of genome level events increases (data not shown).

Real genome data

To show the applicability of our methods to real data, we obtained from NCBI protein-coding genes of X chromosomes of human (*Homo-sapiens*, assembly GRCh38.p7), house mouse (*Mus musculus*, assembly GRCm38.p4 C57BL/6J), and Norway rat (*Rattus norvegicus*, assembly Rnor_6.0). In mammals, the set of genes on the X

Table 2 Results of experiments for 10 simulated genomes (45 pairwise comparisons) with smaller PAM distance

	ILP			MAXIMUM-MATCHING	GREEDY-DENSITY	GREEDY-LENGTH	GREEDY-WMIS
	Time (s)	Not finished	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)
50 genes, $r = 10$	840.59	41	329.53	415.57	163.00	172.02	168.58

Table 3 Results for heuristics on real genomes

	Smaller genome	Matching size				Time (s)				Similarity			
		MM	GD	GL	GW	MM	GD	GL	GW	MM	GD	GL	GW
Human/mouse	696	643	643	643	643	0.07	19.6	0.1	8.6	404.56	420.64	421.48	420.72
Human/rat	672	613	611	611	612	0.05	11.6	0.04	3.3	358.36	374.17	374.27	373.82
Mouse/rat	746	690	689	689	689	0.17	0.18	0.13	0.18	481.53	500.59	500.57	500.36

Smaller genome column shows for each pair of genomes the number of genes in the smaller one, an upper bound for the matching size. Heuristics are represented by their initials (e.g. GREEDY-LENGTH = GL)

chromosome has been reasonably conserved throughout the last several million years [30], having however their order disrupted many times.

Since protein sequences are used to obtain the similarity scores (with the help of the BLASTp tool) instead of nucleotide sequences, 76 genes from the rat genome were excluded because no protein sequence was available. Besides, when a gene has multiple isoforms, the longest is kept. The number of genes in the resulting genomes were 822, 953 and 863 for human, mouse and rat, respectively, some of them removed from the pairwise genome comparison due to the pruning process of FFGC.

Table 3 shows, as expected, that the two rodent X chromosomes have a higher similarity than any of them to the human X chromosome. The values returned by the greedy heuristics are very similar, where GREEDY-LENGTH is the fastest. MAXIMUM-MATCHING results are less than 5% distant from the results of the greedy heuristics, which indicates the choice of cycles has some influence but does not dominate the similarity values obtained for these instances. Matching sizes are similar for all heuristics, showing that about 8% of the genes of the smaller genomes could not be matched to some gene of the other genome and had to be removed, that is, they are disposable genes.

Conclusions

In this paper we developed methods for computing the (NP-hard) family-free DCJ similarity, which is a large-scale rearrangement measure for comparing two given genomes. We presented an exact algorithm in form of an integer linear program and extended our previous hardness result by showing that the problem is APX-hard and has a lower bound of 22/21 for its approximation ratio. Therefore, we developed four heuristic algorithms and could show that they perform well while having reasonable running times also for realistic-size genomes.

Our initial experiment on real data can be considered a proof of concept. In general, the computational results of this paper can be used to more systematically study the applicability of the DCJ similarity measure in various contexts. One important point to be investigated is whether, differently from parsimonious distance measures that usually only hold for closely related genomes, a genomic similarity would allow to perform good comparisons of more distant genomes as well. Fine-tuning of both the data preparation and objective function may be necessary, though.

For example, one drawback of the function s_{FFDCJ} as defined in Eq. 3 is that distinct pairs of genomes might give family-free DCJ similarity values that cannot be compared easily, because the value of S_{FFDCJ} varies between 0 and $|M|$, where M is the matching giving rise to S_{FFDCJ}. Therefore some kind of normalization would be desirable. A simple approach could be to divide S_{FFDCJ} by the size of the smaller genome, because this is a trivial upper bound for $|M|$. Moreover, it can be applied as a simple postprocessing step, keeping all theoretical results of this paper valid. A better normalization, however, might be to divide by $|M|$ itself. An analytical treatment here seems more difficult, though. Therefore we leave this and the application to multiple genomes in a phylogenetic context as an open problem for future work.

Other questions that can be studied in the future are the relationships between family-based and family-free genomic similarity measures in general.

Additional file

Additional file 1: APX-hardness proof of the FFDCJ-SIMILARITY problem. (PDF 379 kb)

Acknowledgements
We would like to thank Pedro Feijão and Daniel Doerr for helping us with hints on how to get simulated and real data for our experiments.

Funding
The publication cost of this article was funded by the author's home institutions.

Availability of data and materials
Source code of the algorithms implementation is available from https://git.facom.ufms.br/diego/ffdcj-sim.

About this supplement
This article has been published as part of *BMC Bioinformatics* Volume 19 Supplement 6, 2018: Proceedings of the 15th Annual Research in Computational Molecular Biology (RECOMB) Comparative Genomics Satellite Workshop: bioinformatics. The full contents of the supplement are available online at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-19-supplement-6.

Authors' contributions
All authors developed the theoretical results and wrote the manuscript. EAH developed the ILP. DPR implemented the algorithms, devised and performed

the experimental evaluation. All authors read and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details
[1]Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brazil . [2]Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, Germany .

Published: 8 May 2018

References
1. Sankoff D. Edit distance for genome comparison based on non-local operations. In: Proc. of CPM 1992 LNCS, vol. 644. 1992. p. 121–35.
2. Sankoff D. Genome rearrangement with gene families. Bioinformatics. 1999;15(11):909–17.
3. Bafna V, Pevzner P. Genome rearrangements and sorting by reversals. In: Proc. of FOCS 1993. Palo Alto: IEEE; 1993. p. 148–57.
4. Hannenhalli S, Pevzner P. Transforming men into mice (polynomial algorithm for genomic distance problem). In: Proc. of FOCS 1995. Milwaukee: IEEE; 1995. p. 581–92.
5. Yancopoulos S, Attie O, Friedberg R. Efficient sorting of genomic permutations by translocation, inversion and block interchanges. Bioinformatics. 2005;21(16):3340–6.
6. Bergeron A, Mixtacki J, Stoye J. A unifying view of genome rearrangements. In: Bucher P, Moret BME, editors. Proc. of WABI 2006. LNBI, vol. 4175. Zurich: Springer; 2006. p. 163–73.
7. Braga MDV, Willing E, Stoye J. Double cut and join with insertions and deletions. J Comput Biol. 2011;18(9):1167–84.
8. Bryant D. The complexity of calculating exemplar distances. In: Sankoff D, Nadeau JH, editors. Comparative Genomics. Dortrecht: Kluwer Academic Publishers; 2000. p. 207–11.
9. Bulteau L, Jiang M. Inapproximability of (1,2)-exemplar distance. IEEE/ACM Trans. Comput. Biol. Bioinf. 2013;10(6):1384–90.
10. Angibaud S, Fertin G, Rusu I, Vialette S. A pseudo-boolean framework for computing rearrangement distances between genomes with duplicates. J Comput Biol. 2007;14(4):379–93.
11. Angibaud S, Fertin G, Rusu I, Thévenin A, Vialette S. Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. J Comput Biol. 2008;15(8): 1093–115.
12. Angibaud S, Fertin G, Rusu I, Thévenin A, Vialette S. On the approximability of comparing genomes with duplicates. Journal of Graph Algorithms and Applications. 2009;13(1):19–53.
13. Shao M, Lin Y, Moret B. An exact algorithm to compute the DCJ distance for genomes with duplicate genes. In: Proc. of RECOMB 2014. LNBI. Pittsburg: Springer; 2014. p. 280–292.
14. Doerr D, Thévenin A, Stoye J. Gene family assignment-free comparative genomics. BMC Bioinformatics. 2012;13(Suppl 19):3.
15. Braga MDV, Chauve C, Doerr D, Jahn K, Stoye J, Thévenin A, Wittler R. The potential of family-free genome comparison. In: Chauve C, El-Mabrouk N, Tannier E, editors. Models and Algorithms for Genome Evolution. London: Springer; 2013. p. 287–307. Chap. 13.
16. Durrett R, Nielsen R, York TL. Bayesian estimation of genomic distance. Genetics. 2004;166(1):621–9.
17. Martinez FV, Feijão P, Braga MDV, Stoye J. On the family-free DCJ distance and similarity. Algoritm Mol Biol. 2015;10:13.
18. Rubert DP, Medeiros GL, Hoshino EA, Braga MDV, Stoye J, Martinez FV. Algorithms for computing the family-free genomic similarity under DCJ. In: Proc. of RECOMB-CG 2017. LNBI. Barcelona: Springer International Publishing; 2017. p. 76–100.
19. Chen Z, Fu B, Xu J, Yang B, Zhao Z, Zhu B. Non-breaking similarity of genomes with gene repetitions. In: Proc. of Combinatorial Pattern Matching (CPM 2007). Heidelberg: Springer; 2007. p. 137–43.
20. Rubert DP, Feijão P, Braga MDV, Stoye J, Martinez FV. Approximating the DCJ distance of balanced genomes in linear time. Algoritm Mol Biol. 2017;12:3.
21. Shao M, Lin Y. Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. BMC Bioinformatics. 2012;13(Suppl 19):13.
22. Munkres J. Algorithms for the assignment and transportation problems. J SIAM. 1957;5(1):32–28.
23. Edmonds J, Karp RM. Theoretical improvements in algorithmic efficiency for network flow problems. J ACM. 1972;19(2):248–64.
24. Tomizawa N. On some techniques useful for solution of transportation network problems. Networks. 1971;1(2):173–94.
25. Hawick KA, James HA. Enumerating circuits and loops in graphs with self-arcs and multiple-arcs, Technical Report CSTN-013: Massey University; 2008.
26. Johnson D. Finding all the elementary circuits of a directed graph. SIAM J Comput. 1975;4(1):77–84.
27. Berman P. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. In: Halldórsson MM, editor. Proc. of SWAT 2000. Bergen: Springer-Verlag Berlin Heidelberg; 2000. p. 214–9.
28. Dalquen DA, Anisimova M, Gonnet GH, Dessimoz C. Alf – a simulation framework for genome evolution. Mol Biol Evol. 2012;29(4):1115.
29. Doerr D. Family Free Genome Comparison (FFGC). 2017. https://bibiserv2.cebitec.uni-bielefeld.de/ffgc. Accessed 31 Jan 2018.
30. Ohno S. Sex Chromosomes and Sex-linked Genes. Endocrinology, vol. 1. Berlin, Heidelberg: Springer; 2013.