Gärtner *et al. Algorithms Mol Biol (2018) 13:15*
https://doi.org/10.1186/s13015-018-0133-4

Algorithms for Molecular Biology

RESEARCH Open Access

Coordinate systems for supergenomes

Fabian Gärtner[1,2*], Christian Höner zu Siederdissen[2,3], Lydia Müller[1,3,4] and Peter F. Stadler[1,2,3,5,6,7,8]

Abstract

Background: Genome sequences and genome annotation data have become available at ever increasing rates in response to the rapid progress in sequencing technologies. As a consequence the demand for methods supporting comparative, evolutionary analysis is also growing. In particular, efficient tools to visualize-omics data simultaneously for multiple species are sorely lacking. A first and crucial step in this direction is the construction of a common coordinate system. Since genomes not only differ by rearrangements but also by large insertions, deletions, and duplications, the use of a single reference genome is insufficient, in particular when the number of species becomes large.

Results: The computational problem then becomes to determine an order and orientations of optimal local alignments that are as co-linear as possible with all the genome sequences. We first review the most prominent approaches to model the problem formally and then proceed to showing that it can be phrased as a particular variant of the Betweenness Problem. It is NP hard in general. As exact solutions are beyond reach for the problem sizes of practical interest, we introduce a collection of heuristic simplifiers to resolve ordering conflicts.

Conclusion: Benchmarks on real-life data ranging from bacterial to fly genomes demonstrate the feasibility of computing good common coordinate systems.

Keywords: Comparative genomics, Comparative transcriptomics, Big data, Graph theory, Betweenness ordering, Colored multigraph, Combinatorial optimization

Background

The past decade has seen rapid progress of sequencing technologies [1]. The dramatic decrease of sequencing costs has enabled an ever-accelerating flood of genomic and transcriptomic data [2] that in turn have lead to the development of a wide array of methods for data analysis. Despite recent efforts to study transcriptome evolution at large scales [3–7] the capability to analyze and integrate -omics data in large-scale phylogenetic comparisons lags far behind data generation. One key aspect of this shortcoming is the current lack of powerful tools for visualizing comparative -omics data. Available tools such as [8, 9] have been designed with closely related species or strains in mind. The visualizations become difficult to read for multiple species and larger evolutionary distances, where homologous genomic regions may differ substantially in their lengths, an issue that becomes more pressing the larger regions of interest become. A common coordinate system for multiple genomes is not only a convenience for graphical representations of -omics data, however. It would also greatly facilitate the systematic analysis of all those genomic features that are not sufficiently local to be completely contained within individual blocks of a genome-wide multiple sequence alignment (gMSA).

Still, gMSAs are the natural starting point. Several pipelines to construct such alignments have been deployed over the past two decades, most prominently the `tba/multiz` pipeline [10, 11] employed by the `UCSC` genome browser and the `Enredo/Pecan/Ortheus` (`EPO`) pipeline [12] featured in the `ensembl` system. For the ENCODE project data, in addition alignments generated with `MAVID` [13] and `M-LAGAN` [13] have become available, see [14] for a comparative assessment. A common feature of gMSAs is that they are composed of a large number of alignment blocks. At least in the case of MSAs of higher animals and plants the individual blocks are typically (much) smaller than individual genes. As a consequence, they are not ready-to-use for detailed comparative studies e.g. of transcriptome or epigenome [15]

*Correspondence: fabian@bioinf.uni-leipzig.de
[2] Bioinformatics Group, Department of Computer Science, Universität Leipzig, Härtelstraße 16–18, 04107 Leipzig, Germany
Full list of author information is available at the end of the article

structure. In the gMSA-based splice site maps of [16], for example, it is easy to follow the evolution of individual splice junctions as they are localized within a block. At the same time it is difficult to collate the global differences of extended transcripts, which may span hundreds of blocks and to relate changes in transcript structure with genomic rearrangements, insertions of repetitive elements or deletion of chunks of sequence.

To a certain extent this problem is alleviated by considering the blocks arranged w.r.t. a reference genome. For many applications, however, this does not appear to be sufficient. For sufficiently similar genomes with only few rearrangements gMSA blocks are large or can at least be arranged so that large syntenic regions can be represented as a single aligned block. Any ordering of these large syntenic blocks, termed a supergenome in [17], then yields an informative common coordinate system. So far, this approach has been applied only to closely related procaryotic genomes. Prime examples are a detailed comparative analysis of the transcriptome of multiple isolates of *Campylobacter jejuni* [18] or the reconstruction of the phylogeny of mosses from the "nucleotide pangenome" of mitogenomic sequences [19]. We remark that some approaches to "pangenomes" are concerned with gMSAs of (usually large numbers of) closely related isolates; most of this literature, however, treats pangenomes as sets of orthologous genes [20].

Here we are concerned with the coordinatization of supergenomes, i.e., the question how gMSA blocks can be ordered in a way that facilitates comparative studies of genome annotation data. In contrast to previous work on supergenomes we are in particular interested in large animal and plant genomes and in large phylogenetic ranges. We therefore assume that we have short alignment blocks and abundant genome rearrangement, leaving only short sequences of alignment blocks that are perfectly syntenic between all genomes involved. The problem of optimally sorting the MSA blocks can, as we shall see, be regarded as a quite particular variant of a vertex ordering problem, a class of combinatorial problems that recently has received increasing attention in computer science [21–24]. In the computational biology literature, furthermore, several graph-based methods have been proposed to solve the problem of sorting sequence blocks for supergenomes, see e.g. [12, 25–29].

This contribution is organized as follows: In the following section we first analyze the concept of the supergenome and its relationship to gMSAs in detail. We then review combinatorial optimization problems that are closely related to the "supergenome sorting problem", and argue that the most appropriate modeling leads to a special type of betweenness ordering problem. Next, we introduce a heuristic solution that is geared towards very large input alignments and proceeds by step-wise simplification of the supergenome multigraph. Finally, we outline a few computational results.

Theory

Genome-wide multiple sequence alignments

Our starting point is a set of genome assemblies. For our purposes an assembly is simply a set of sequences representing chromosomes, scaffolds, reftigs, contigs, etc. In the following, we will use *contig* to refer to any such sequence. On each of these constituent sequences we assume the usual coordinate system defining sequence positions. Since DNA is double stranded, a piece of genomic sequence is either contained directly ($\sigma = +1$) in the assembly or it is represented by its reverse complement ($\sigma = -1$). We write $(\mathcal{G}, c, i, j, \sigma)$ to identify the *sequence interval* from positions i to j on contig c of genome assembly $\mathcal{G}$ with reading direction σ. We assume, w.l.o.g., $i \leq j$.

Most comparative methods require multiple sequence alignments (MSAs) as input. An MSA $\mathfrak{A}$ is composed of *alignment blocks*, each of which consists of an alignment of sequence intervals. For the purposes of this paper it its sufficient to characterize an alignment block by the coordinates of its constituent sequence intervals. That is, a block $B \in \mathfrak{A}$ has the form $B = \{(\mathcal{G}_u, c_u, i_u, j_u, \sigma_u) | u \in \text{rows of } B\}$ where the index u runs over the rows of the alignment block. It will be convenient to allow alignment blocks also to consist of a single interval only, thus referring to a piece of sequence that has not been aligned. Note that at this stage we do not assume that an alignment block contains only one interval from each assembly.

The projection $\pi_{\mathcal{G}}(B)$ extracts from an alignment block the union of its constituent sequence intervals belonging to assembly $\mathcal{G}$. If the assembly $\mathcal{G}$ is not represented in the alignment block B we set $\pi_{\mathcal{G}}(B) = \emptyset$.

The projection operation collapses pairs of overlapping sequence intervals $(i \leq i' \leq j \leq j')$ in to a single interval: $(\mathcal{G}, c, i, j, \sigma) \cup (\mathcal{G}, c, i', j', \sigma') = (\mathcal{G}, c, i, j', +1)$ without regard for the orientation, which is set to $+1$ and will have *no bearing* on the algorithm we develop further down.

The projection $\pi_{\mathcal{G}}$ of $\mathfrak{A}$ onto one of its constituent assemblies $\mathcal{G}$ is the union of the sequence intervals from $\mathcal{G}$ that are contained in its alignment blocks, i.e., $\pi_{\mathcal{G}}(\mathfrak{A}) = \bigcup_{B \in \mathfrak{A}} \pi_{\mathcal{G}}(B)$.

Definition 1 Let $\mathfrak{A}$ be an MSA.

- $\mathfrak{A}$ is *complete* if $\pi_{\mathcal{G}}(\mathfrak{A}) = \mathcal{G}$, i.e., if each position in each assembly is represented in at least one alignment block.

- $\mathfrak{A}$ is *irredundant* $\pi_{\mathcal{G}}(B') \cap \pi_{\mathcal{G}}(B'') = \emptyset$ for any two distinct blocks B' and B'', i.e., if every sequence interval from assembly $\mathcal{G}$ is contained in at most one alignment block.
- $\mathfrak{A}$ is *injective* if no alignment block comprises more than one interval from each of its constituent assemblies.

Clearly, every given MSA can easily be completed by simply adding all unaligned sequence intervals as additional blocks.

Just like a contig c in a (genome) assembly $\mathcal{G}$, each block $B \in \mathfrak{A}$ has an internal coordinate system defined by its columns. We write (B, k) for column k in block B. We write $\ell(B)$ for the number of columns in block B. If $\mathfrak{A}$ is irredundant, then there are functions $f_{\mathcal{G},c}$ that map position i within $(\mathcal{G}, c)$ to a corresponding MSA coordinate (B, k). If $\mathfrak{A}$ is complete, the individual $f_{\mathcal{G},c}$ can be combined to a single function $f : (\mathcal{G}, c, i) \mapsto (B, k)$. Completeness implies that every position $(\mathcal{G}, c, i)$ is represented in the MSA, and irredundancy guarantees that the relation between assembly and alignment coordinates is a function by ensuring that $(\mathcal{G}, c, i)$ corresponds to at most one alignment column. The following definition is therefore equivalent to the notion of a supergenome introduced in [17].

Definition 2 An MSA $\mathfrak{A}$ is a *supergenome* if it is complete, irredundant, and injective.

The most commonly used genome-wide MSAs cannot be completed to supergenomes. The MSAs produced by the `multiz` pipeline are usually not irredundant: different intervals of the "reference sequence" may be aligned to the same interval of another assembly. While `multiz` [11] alignments are injective this is in general not the case with the `EPO` [12] alignments. In these, multiple paralogous sequences from the same genome may appear in one alignment block.

Now consider an MSA $\mathfrak{A}$ and an arbitrary order $<$ of the alignment blocks of $\mathfrak{A}$. Then there is a (unique) function ϕ that maps the pairs (B, k) injectively to the interval $[1, n]$, where $n = \sum_{B \in \mathfrak{A}} \ell(B)$ is the total number of columns in $\mathfrak{A}$ such that $\phi(B, i) < \phi(B', i')$ whenever $B < B'$ or $B = B'$ and $i < i'$. If $\mathfrak{A}$ is a supergenome, then $\phi(f)$ is clearly an injective function from a genome assembly $\mathcal{G}$ to $[1, n]$. We call $\phi(f(\mathcal{G}, c, i))$ the *coordinate* of position i of contig c of assembly $\mathcal{G}$ in the ordered supergenome $(\mathfrak{A}, <)$.

As pointed out in [17], the existence of a coordinate system for the supergenome $\mathfrak{A}$ is independent of the block order $<$. However, the order $<$ is crucial for the practical use of the coordinate system.

Adjacency and betweenness of MSA blocks

The natural starting point for considering adjacency and betweenness of alignment blocks are their constituent intervals $(\mathcal{G}, c, i, j, \sigma)$ on a fixed assembly $\mathcal{G}$ and contig c. Intervals have a natural partial order defined by $(\mathcal{G}, c, i, j, \sigma) \prec (\mathcal{G}, c, k, l, \sigma)$ whenever $i < k$ and $j < l$. Two intervals are incomparable in this *interval order* if and only if one is contained in the other. Note that the interval order allows comparable intervals to overlap. We also consider intervals incomparable that belong to different contigs and/or assemblies.

Given three intervals $\alpha = (\mathcal{G}, c, i', j', \sigma')$, $\beta = (\mathcal{G}, c, i'', j'', \sigma'')$, and $\gamma = (\mathcal{G}, c, i, j, \sigma)$ (on the same genome assembly and contig), we say that γ is *between* the two distinct intervals α and β if $\alpha \prec \gamma \prec \beta$ or $\beta \prec \gamma \prec \alpha$.

Given a collection $\mathcal{Q}$ of intervals on the same assembly $\mathcal{G}$ and contig c, we say that $\alpha = (\mathcal{G}, c, i', j', \sigma')$ and $\beta = (\mathcal{G}, c, i'', j'', \sigma'')$ are *adjacent* if there is no interval γ between α and β. We say that α is a *predecessor* of β if α and β are adjacent and $\alpha \prec \beta$. Analogously, α is a a *successor* of β if α and β are adjacent and $\beta \prec \alpha$.

Lemma 1 *Let $\mathfrak{A}$ be a supergenome and consider the the collection $\{\pi_{\mathcal{G}}(B) | B \in \mathfrak{A}\}$ of intervals on a given $\mathcal{G}$. Then (i) no two intervals overlap, (ii) the interval order $\prec$ is a total order on every contig c, (iii) every interval has at most one predecessor and one successor, and hence is adjacent to at most two intervals, and (iv) if γ is adjacent to both α and β, then γ is between α and β.*

Proof Property (i) follows directly from the condition. (ii) As a consequence, any two intervals on a fixed contig c are comparable, i.e., the restriction of $\prec$ to c is a total order. Since only intervals on the same contig can be adjacent, (iii) is an immediate consequence of (ii) and the fact that the number intervals is finite. Property (iv) is now a trivial consequence of the fact that by (iii) α and β must be the predecessor and successor of γ.

A key construction in this contribution is the notion of betweenness relations for alignment blocks.

Definition 3 Given three blocks $A, B, C \in \mathfrak{A}$, we say that C *is between A and B with respect to $\mathcal{G}$* if $\pi_{\mathcal{G}}(C)$ is between $\pi_{\mathcal{G}}(A)$ and $\pi_{\mathcal{G}}(B)$. The ternary relation $\mathscr{C}(\mathfrak{A})$ is the defined by $(A, C, B) \in \mathscr{C}(\mathfrak{A})$ whenever C is between A and B for some assembly $\mathcal{G}$.

We note that contradicting betweenness relations resulting from different genome assemblies $\mathcal{G}$ are to be expected, i.e., the relation $\mathscr{C}(\mathfrak{A})$ will in general not satisfy the properties of a well-formed betweenness relation. We will return to this issue below.

Definition 4 Two blocks $A, B \in \mathfrak{A}$ are *adjacent* if there is an assembly $\mathcal{G}$ such that $\pi_G(A)$ and $\pi_G(B)$ are adjacent w.r.t. $\{\pi_G(C)|C \in \mathcal{A}\}$.

It is useful to regard $\mathfrak{A}$ with its adjacency relation as a graph. In order to keep track of the individual contigs, we use an edge-colored multigraph, with $\mathcal{G}$ serving as edge color.

Definition 5 The *supergenome graph* $\Gamma(\mathfrak{A})$ of an MSA $\mathfrak{A}$ is the directed, edge-colored multigraph whose vertices are the alignment blocks of $\mathfrak{A}$ and whose directed edges (A, B) connect an alignment block A to an alignment block B with color $\mathcal{G}$ whenever there are sequence intervals $\alpha \in A$ is a predecessor of $\beta \in B$ in assembly $\mathcal{G}$.

An example how the prejection $\Gamma(\mathfrak{A})$ is done is shown in Fig. 1. The projection of $\Gamma(\mathfrak{A})$ to a constituent assembly $\mathcal{G}$ is a (not necessarily induced) subgraph. As an immediate consequence of Lemma 1, each projection is a disjoint union of directed paths, each of which represents a contig. Conversely, every colored directed multigraph whose restriction to a single color is a set of vertex-disjoint directed paths is a supergenome graph. It therefore makes sense to talk about a supergenome graph Γ without explicit reference to an underlying alignment $\mathfrak{A}$.

The structure of the *supergenome graph* strongly depends on the evolutionary history of the genomes that it represents. In the absence of genome rearrangements (i.e., when the only genetic changes are substitutions, insertions (including duplications), and deletions) then all genomes remain colinear with their common ancestor. In other words, a single, canonical [30] global alignment describes a common coordinate system that is unique up to the (arbitrary) order of contigs and each trace [31] of insertions and deletions. In terms of the block adjacency relation, each block has at most two adjacent neighbors in this scenario.

Genome rearrangements are by no means infrequent events [32–35], and thus cannot be neglected. Every breakpoint introduced by a genome rearrangement operation, be it a local reversal or a cut-and-join type dislocation, introduces an ambiguous adjacency, i.e., a block that has two or more predecessors or successors. The task of

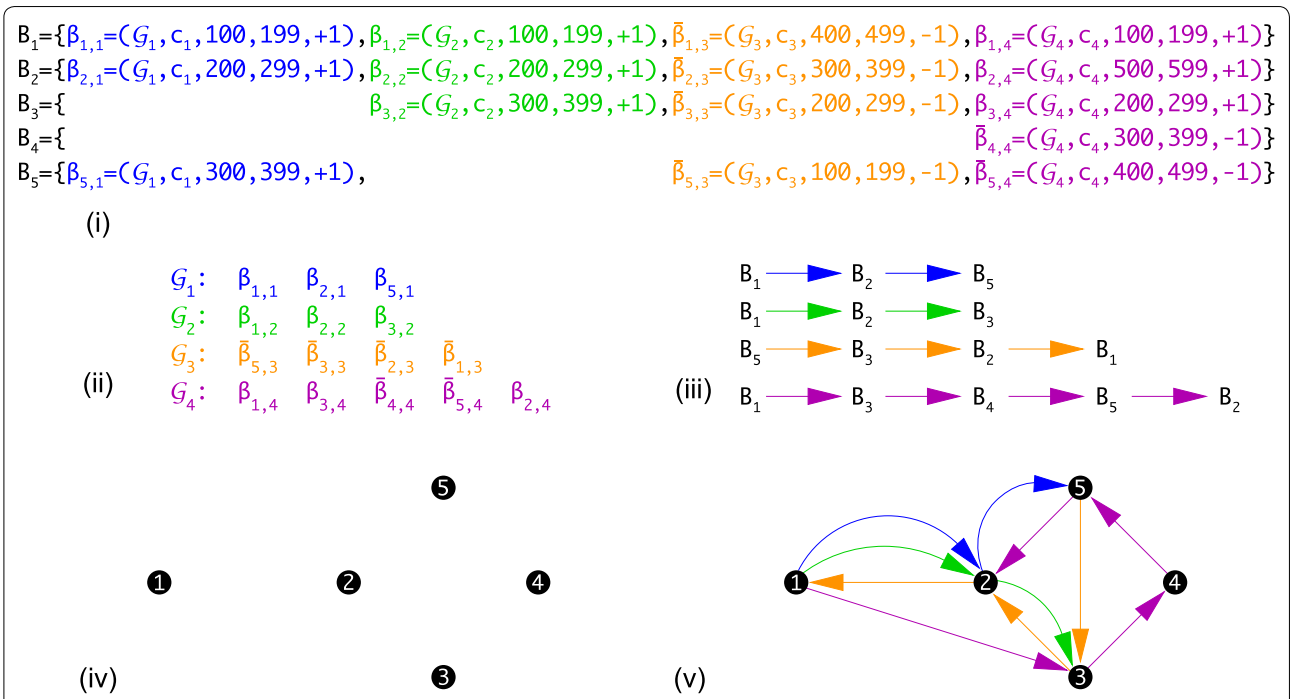

Fig. 1 Projections of a supergenome graph. An example how the projection of an artificial MSA $\mathfrak{A}$ to a supergenome graph $\Gamma(\mathfrak{A})$ can be done. Starting point is the MSA $\mathfrak{A}$ shown in **(i)**, which comprises five blocks (B_1, ..., B_5), each consisting of up to four intervals from the four genome assemblies (distinguished by colors: $\mathcal{G}_1$ blue, $\mathcal{G}_2$ green, $\mathcal{G}_3$ orange, $\mathcal{G}_4$ purple) with one contig each (c_1, ..., c_4). The intervals are designated as $\beta_{k,l} = (\mathcal{G}_l, c_l, i_{k,l}, j_{k,l}, +1) \in B_k$ and $\bar{\beta}_{k,l} = (\mathcal{G}_l, c_l, i_{k,l}, j_{k,l}, -1) \in B_k$. To construct the supergenome graph first the predecessor of each interval is determined by sorting the intervals separately for each assembly. By assumption, intervals do not overlap. The ordered set of each assembly is shown in panel **(ii)**. Only these orders are relevant in subsequent steps, hence we suppress the positional information from now on. The interval order implies a separate predecessor relation among blocks **(iii)**, with a colored arrow from B_1 to B_2 implying that B_1 is a predecessor of B_2 w.r.t. the assembly indicated by the color. The supergenome graph $\Gamma(\mathfrak{A})$ has all blocks of $\mathfrak{A}$ as its vertices **(iv)**. The predecessor relations among the blocks define the colored, directed edges **(v)**

identifying an appropriate ordering of the MSA blocks therefore is a non-trivial one for realistic data, even in the absence of alignment errors.

Modeling the "Supergenome Sorting Problem"

Informally, we may consider the "*supergenome sorting problem*" (SSP) as the task of finding an order $<$ (or, equivalently, a permutation ρ) of the alignment blocks of $\mathfrak{A}$ *such that the orders of the constituent assemblies are preserved as much as possible*. Somewhat more precisely, we are to find an order $<$ on the vertex set of the supergenome graph $\Gamma(\mathfrak{A})$ that as many of its directed edges as possible are "consistent" with the order $<$. It is not clear from the outset, however, how "consistency" should be defined for our application. A large number of related models have been proposed and analyzed in the literature that make this condition precise in different ways, leading to different combinatorial optimization problems. We therefore proceed with a brief review of some paradigmatic approaches. A reader primarily interested in our proposed approach to the problem might want to skip this section.

Hamiltonian paths

A plausible attempt is to view the SSP as a variant of the Hamiltonian path problem on the *supergenome graph* Γ. A Hamiltonian path defines a total order of the vertices and therefore a solution to the SSP. This is idea is similar to the use of Hamiltonian graphs for genome assembly from read overlap graphs [36]. There are several quite obvious difficulties, however. First, it is not sufficient to consider only paths that are entirely confined to pass through the adjacencies. The simplest counterexample consists of only 4 MSA blocks B_1, B_2, B_3 ,B_4 and three assemblies $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$. Given eight sequence intervals $\beta_{k,l} = (\mathcal{G}_l, c_{k,l}, i_{k,l}, j_{k,l}, +1) \in B_k$ we construct the following alignment:

$$\begin{array}{rcccc} & B_1 & B_2 & B_3 & B_4 \\ g_1 = & \beta_{1,1} & & & \beta_{4,1} \\ g_2 = & \beta_{1,2} & \beta_{2,2} & & \beta_{4,2} \\ g_3 = & \beta_{1,3} & & \beta_{3,3} & \beta_{4,3} \end{array}$$

This situation arises in practice e.g. when B_2 and B_3 are two independent, unrelated inserts between B_1 and B_4. The block adjacency graph is the graph

$$B_2 - B_1 - B_4 - B_3,$$

which violates the desired betweenness relation $(\beta_{1,3}, \beta_{3,3}, \beta_{4,3})$.

In this case there are only two biologically correct solutions: $B_1 < B_2 < B_3 < B_4$ (or the inverse order) and $B_1 < B_3 < B_2 < B_4$ (or its inverse). In either case, the solution contains two consecutive blocks (B_2 and B_3) that are not adjacent in the block graph. This example also serves to demonstrate that the block graph alone does not contain the complete information on the supergenome. It appears that in addition we will need to know the *betweenness* relation among the blocks i.e., that both B_2 and B_3 are between B_1 and B_4.

Feedback arc sets and topological sorting

An other possibility to determine a well-defined order of the vertices of the supergenome graph Γ is to first extract a maximum acyclic subgraph and then to compute a topological sorting of this subgraph. An equivalent formulation asks for the removal of a minimum set of edges that close cycles. This Maximum Acyclic Subgraph or Minimum Feedback Arc Set problem (MFAS) is well-known to be NP-hard [37]. Nevertheless fast, practicable heuristics have been devised, see e.g. [38, 39]. From the resulting directed acyclic graph (DAG) an admissible ordering of blocks can be obtained efficiently by topological sorting [40]. A closely related approach is the Linear Ordering Problem (LOP): given a complete weighted directed graph, find a tournament with maximum total edge weights [41]. It yields essentially the same model since LOP and MFAS can be transformed into each other quite easily [42]. Cost functions designed to define consensus orderings for sets of total and partial orders have been considered in different fields starting with the work of Spearman [43] and Kendall [44], see also e.g. [45–47]. The reconciliation of partial orders in investigated in detail e.g. in [48].

The key problem of modeling the SSP in terms of MFAS is highlighted in Fig. 2. It shows that even when undirected adjacencies would allow for a perfect solution, it may not be uncovered directly by the MFAS approach.

Simultaneous consecutive ones and matrix banding

Instead of adjacencies we may consider the incidence matrix $\mathbf{A}$ of the supergenome graph Γ and try to sort both the alignment blocks and their adjacencies in such a way that, to the extent that this is possible, (i) adjacent blocks are consecutive and (ii) adjacencies that have a block in common are consecutive. In more formal terms, we wish to sort both the rows (alignment blocks) and columns (adjacencies) of the incidence matrix in such a way that rows and columns show all non-zero entries consecutively. A rectangular matrix $\mathbf{A}$ that admits such a pair of row and column permutations is said to have the *simultaneous consecutive ones property* (C1S) [49]. This is possible if Γ is a union of paths. Note that instead of adjacencies we could also cover the graph with short paths $\wp_k$. In this case column k identifies the vertices incident with path k. Again, if Γ is a union of paths, the path-incidence matrix satisfies (C1S). It is not difficult to see [49]

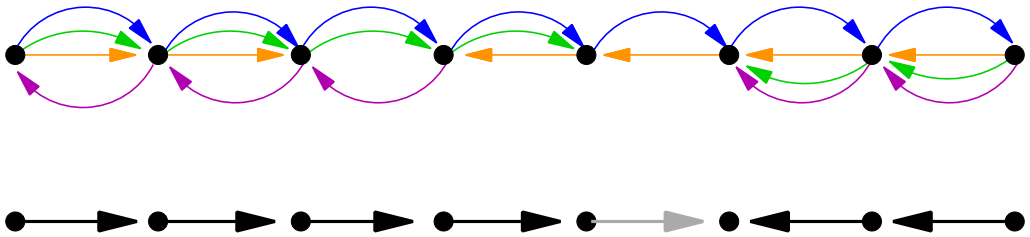

Fig. 2 Minimum feedback arc set (MFAS) does not necessarily yield an optimial solution of the SSP. Due to the arbitrariness of the orientation of the edges, the best solution of the SSP may contain cycles, which by definition is excluded in MFAS. Top: *supergenome graph* representation $\Gamma(\mathfrak{A})$ of an artificial alignment $\mathfrak{A}$. Bottom: simplified solution of the (uniformly weighted) MFAS. To turn $\Gamma(\mathfrak{A})$ into an acyclic graph, at least one edge has to be deleted. Two such solutions exist, differing only by the orientation of the gray arrow. The corresponding topological sorting breaks the genome into two distinct colinear pieces with opposite orientation. There is, however, a consistent order of the entire graph—the linear left-to-right or right-to-left order is consistent

that $\mathbf{A}$ satisfies (C1S) if and only if $\mathbf{A}$ has the well-studied consecutive ones property [50, 51] for both its rows and columns. Thus (C1S) can be checked in linear time [50]. Furthermore, Tucker's characterization of (C1S) in terms of forbidden sub-matrices [52] also carries over. Some direct connection between the consecutive ones property and the Betweenness Problem, which we will consider below, are discussed in [53].

In general, $\mathbf{A}$ will now have the consecutive ones property. The problem of identifying a minimal number of columns (adjacencies) whose removal leaves a (C1S) matrix is NP-complete [49]. In practise it may be desirable to quantify the extent of the violation of (C1S) in terms of intervals of consecutive zeros enclosed by the two 1s. For instance, one may want to use $\omega = \sum_i h(\ell_i)$, where the sum runs over all intervals i of consecutive zeros enclosed by the two 1s, and $h(\ell_i) \geq 0$ is some contribution that monotonically grows with the length ℓ_i of the 0-interval. For a given ordering of the rows and columns, the total violation is quantified as the sum of the ω values. It should be noted, however, that (C1S) does not imply Γ is a union of disjoint paths, i.e., that Γ is a valid supergenome graph.

A related set of optimization problems is concerned with reducing the bandwidth of matrices, i.e., the maximal distance of non-zero entries from the diagonal (in a symmetric case) or the parameter $\min(l, u) + l + u$ (for rectangular matrices); here $u = \max_{\mathbf{A}_{ij} \neq 0}(i - j)$ and $l = \max_{\mathbf{A}_{ij} \neq 0}(j - i)$ [54]. In the symmetric case, several good heuristics are known, starting with the Cuthill-McKee [55] and GPS [56] algorithms even though the problem is NP-hard [57], while the general case has received much less attention [54]. Bandwidth reduction methods do not eliminate "bad" adjacencies that eventually determine bandwidth. The resulting ordering of rows and columns thus may be very inaccurate locally.

Bidirected graphs

Several specialized graph structures have been introduced recently to tackle the problem of ordering sequence blocks, which is a problem very similar to SSP. Among these constructions are A-Bruijn graphs, Enredo graphs, and Cactus graphs, see [58] for a review. A key insight of [58] is that these graph representations are equivalent in the sense that they can be transformed into each other. They differ, however, in additional information extracted from the input alignment that is stored as vertex and edge labels, see Fig. 3. For instance, the bidirected graphs of [28] have the same underlying graph as our supergenome graph. While the supergenome graph used a standard directed graph structure, bidirected graphs encode directional information independently for the endpoint of each edge, distinguishing three cases: (i) adjacent alignment blocks have the same orientation, (ii) the connected blocks switch from minus to plus orientation, or (iii) vice versa. The latter two cases indicate a change of orientation between two changes. In [28], this basic structure is extended by additional transitive edges given by a legal path through the graph with an exponential weight function. The task is then to find a consistent (non-conflicting) set of edges with maximal weight. This form of the weight function takes into account that, due to genetic linkage, the more closely positioned on a genome two blocks are, the less likely it is that the edge between these blocks is broken. This gives higher weight to locally correct block positions and orientations.

While this sorting problem is (NP-) hard in general, the sets of genomes considered by the authors are particularly suitable for this kind of calculations. The genomes considered for pangenome construction are typically closely related, or even of the same species. Thus one can expect many paths with high weights of the conserved consensus [29]. This approach also fits well to the analysis of genomic regions that are under constraint to maintain

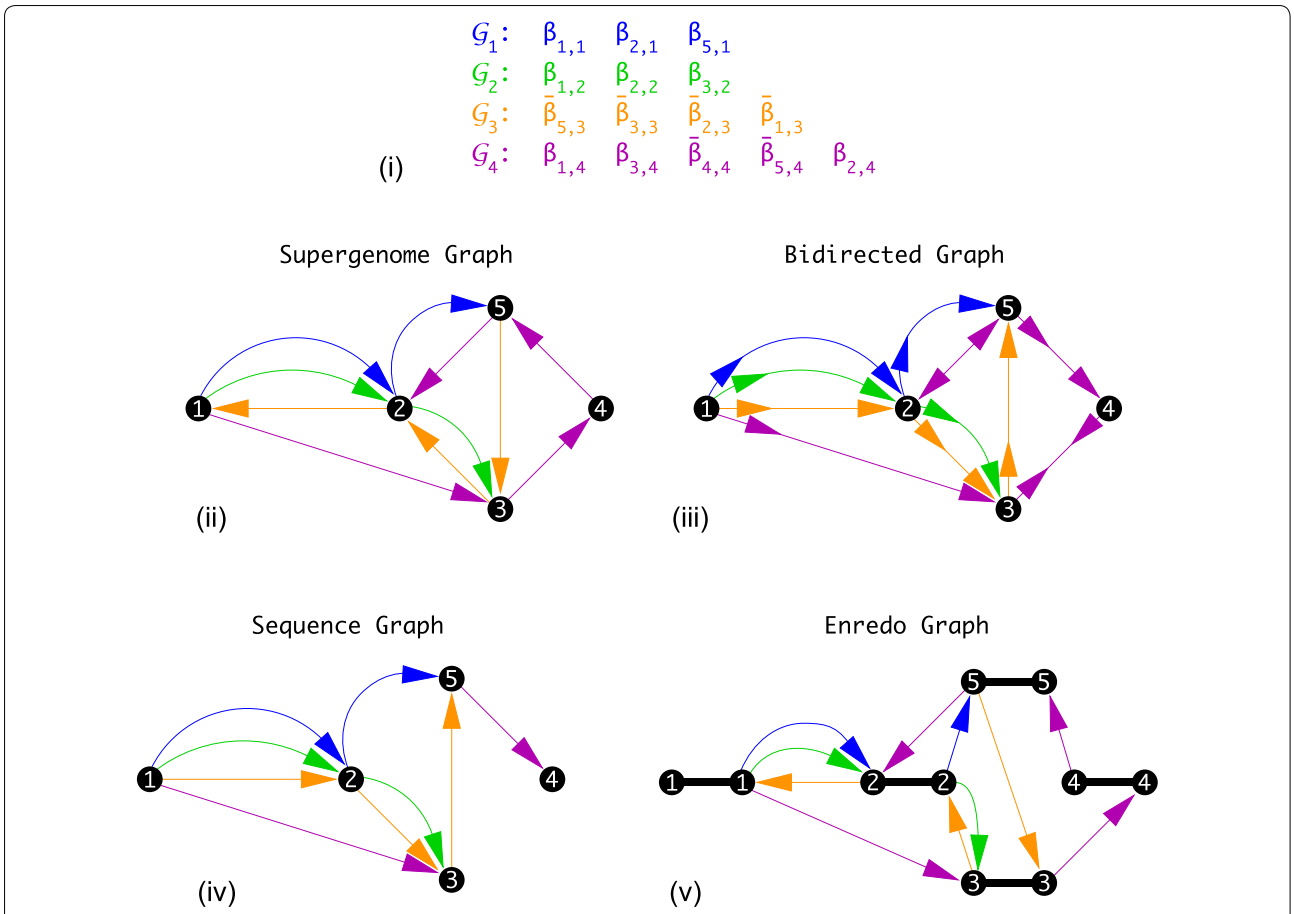

Fig. 3 Comparison of different graph definitions. Consider an alignment comprising of five MSA blocks (B_1, B_2, B_3, B_4, B_5) in four assemblies (indicated by different edge colors: $\mathcal{G}_1$ blue, $\mathcal{G}_2$ green, $\mathcal{G}_3$ orange, $\mathcal{G}_4$ purple), consisting of a total of sequence intervals denoted by $\beta_{k,l} = (\mathcal{G}_l, c_{k,l}, i_{k,l}, j_{k,l}, +1) \in B_k$ and $\bar{\beta}_{k,l} = (\mathcal{G}_l, c_{k,l}, i_{k,l}, j_{k,l}, -1) \in B_k$ for positive and negative orientation, respectively. In the drawings, block B_i is simply denoted by i. Most of the edges in the different graphs represent the adjacency of two intervals in one of the constituent assemblies. These edges are displayed in the corresponding color. **(i)** Genome order and orientation of the intervals. Every line shows one assembly and the order on this assembly. **(ii)** The supergenome graph. **(iii)** The bidirected graph. Note that the edges are the same as in the supergenome graph except for the shown direction. **(iv)** The sequence graph of Haussler et al. [29]. Note that the edges (3,4) and (5,2) are not part of the graph because of the change in orientation. **(v)** The `enredo` graph. An example of the other special graph structures discussed in Kehr *et al.* [58]

syntenic order for functional reasons, such as the MHC locus used as an example in [28]. In distantly related genomes, however, synteny tends not to be well preserved. In addition, we are interested in particular in data sets that contain genomes in preliminary draft forms, i.e., linkage information that is at least partially limited to short contigs or scaffolds. As a consequence we have less confidence in linkage and orientation information than one can expect in a typical pangenome scenario.

Sequence graphs

The sequence graphs of Haussler et al. [29] are also closely related to the *supergenome graph* representation $\Gamma(\mathfrak{A})$ of a multiple alignment $\mathfrak{A}$. The key difference is that the the orientation of the blocks is used to determine the direction of the edge. Two adjacent intervals with negative orientation thus imply an edge that is reversed compared to the supergenome graph. This situation is problematic in the case where the orientation of the intervals is switched. In [29] a preprocessing step is performed to minimize the number of such edges. The sequence graph approach was designed for the comparison of human genomes of different individuals. In such a scenario, the resulting information loss is small and does not present a practical problem. It is likely to become an issue, however, for large phylogenetic distances with frequent genome rearrangements.

The natural formulation of SSP on a sequence graph is to find a vertex ordering that minimizes weighted feedback edges and the average cut width. These optimization

criteria ensure that the successor relations are mostly kept intact and at the same time successors are placed close to each other in the solution. Both problems the Minimum Feedback Arc Set Problem [37] and the Average Cut-Width Minimization Problem [59–61] are known to be NP-hard. Cutwidth minimization problems ask for a linear ordering of the vertices of a graph such that the average or maximum number of edges spanning across the gap between a pair of consecutive vertices is minimized. Conceptually, cutwidth problems are quite similar to bandwidth problems [62]. In [29] a heuristic is presented that first extracts a totally ordered "backbone" and then inserts the remaining vertices into the backbone order that is kept intact in the process. While the presence of a global common backbone order is a well founded assumption for pangenomes of a single or very closely related species, it is violated substantially for phylogenetically diverse data.

Betweenness problems

In this work we interpret SSP as a betweenness (ordering) problem rather than a vertex ordering problem on a directed graph. Instead of (oriented) adjacencies, which are defined on pairs of blocks, one considers the relative order of three blocks:

Betweenness Problem [63, 64]: *Given a finite set X and a collection $\mathscr{C}$ of triples from X, is there a total order on X such that $\forall (i,j,k) \in \mathscr{C}$ either $i < j < k$ or $i > j > k$?*

This decision problem is NP-complete [63, 64]. A triple $(i,j,k) \in \mathscr{C}$ is called a betweenness triple.

The Betweenness Problem can be adapted to model the SSP by means of a suitable cost function b designed to penalize violations of the betweenness relation. Consider an order ρ used to coordinatize the supergenome. We may think of ρ as a bijective function from $[1,\ldots,|\mathfrak{A}|] \to \mathfrak{A}$. In other words, we number the blocks contained in $\mathfrak{A}$ and work on this set of block indices.

For $i < j < k$ we set $b_{\rho,\mathcal{G}}(i,j,k) = 1$ if the projections of the three alignment blocks $\rho(i)$, $\rho(j)$, and $\rho(k)$ exist and violate the betweenness relation for a given assembly $\mathcal{G}$, i.e., if $\pi_{\mathcal{G}}(\rho(i))$, $\pi_{\mathcal{G}}(\rho(j))$ and $\pi_{\mathcal{G}}(\rho(k))$ are located on the same contig and $\pi_{\mathcal{G}}(\rho(j))$ is not located between $\pi_{\mathcal{G}}(\rho(i))$ and $\pi_{\mathcal{G}}(\rho(k))$. Otherwise we set $b_{\rho,\mathcal{G}}(i,j,k) = 0$. A natural cost function is now the total number of betweenness violations

$$b(\rho) := \sum_{\mathcal{G}\in G(\mathfrak{A})} \sum_{i<j<k} b_{\rho,\mathcal{G}}(i,j,k)\,, \tag{1}$$

where $G(\mathfrak{A})$ is the set of genome assemblies that are contained in $\mathfrak{A}$. If genome evolution were to preserve gene order, i.e., only local duplications and deletions are allowed, the betweenness relation of the ancestral state would be preserved, guaranteeing a perfect solution ρ with $b(\rho) = 0$.

Since this decision problem is NP-complete [63, 64], so is the problem of optimizing $b(\rho)$ NP-hard. The proof is shown in Additional file 1: Section 1 The cost function $b(\rho)$ involves the sum over all triples of alignment blocks and thus is fairly expensive to evaluate. It is interesting in practice, therefore, to consider a modified cost function that restricts the sum in Eq. (1) to local information. This idea leads us to the rather natural extension of the Betweenness Problem to colored multigraphs.

Definition 6 Given an (undirected) *colored multigraph* $\hat{\Gamma} = (V,E)$, the triple (i,j,k) is part of the collection $\mathscr{C}(\hat{\Gamma})$, iff there are edges $\{i,j\} \in E$ and $\{j,k\} \in E$ with color $\mathcal{G}$.

Colored Multigraph Betweenness Decision Problem: given the colored multigraph $\hat{\Gamma} = (V,E)$, is there a total order on V such that $\forall (i,j,k) \in \mathscr{C}(\hat{\Gamma})$ either $i < j < k$ or $i > j > k$?

The reformulation as an optimization problem that maximizes the number of edges is straightforward: Colored Multigraph Betweenness Problem: Given a colored multigraph $\hat{\Gamma} = (V,E)$, find a total order on V such that $E^* \subseteq E$ is maximal under the condition that $\forall (i,j,k) \in \mathscr{C}(V,E^*)$ either $i < j < k$ or $i > j > k$.

This problem can be viewed as an analog of the Minimum Feedback Arc Set problem [38] for betweenness data. To our knowledge is has not been studied so far.

Lemma 2 *The (decision version of the) Colored Multigraph Betweenness Problem is NP-complete.*

Proof Every set $\mathscr{C}(\hat{\Gamma})$ of triples can be obtained from an edge-colored multigraph $\hat{\Gamma}$ (with vertices corresponding to alignment blocks and colored edges corresponding to adjacencies deriving from a genome identified by the color). Thus, the total order on the vertices of $\hat{\Gamma}$ is a solution of the Colored Multigraph Betweenness Problem if and only if the answer to the NP-complete Betweenness Problem is positive. In Additional file 1: Section 1 the reduction in both directions is shown.

In the example of Fig. 2 the optimal solution of the Colored Multigraph Betweenness Problem retains all *unordered* adjacencies and creates a unique coordinatization (up to orientation) that leaves all alignment blocks ordered as drawn.

Seriation

An alternative framework for solving the SSP by construction of a preferred ordering is seriation. The

Robinson seriation problem [65] starts from a dissimilarity measure $d : X \times X \to \mathbb{R}$, and seeks a linear order ρ on X that satisfies the inequality

$$\max\{d(\rho(i),\rho(j)),d(\rho(j),\rho(k))\} \leq d(\rho(i),\rho(k))\,. \quad (2)$$

A dissimilarity d for which an ordering ρ exists that satisfies Eq. (2) for all $\rho(i) < \rho(j) < \rho(k)$ is called *Robinsonian*. It is worth noting that Robinsonian dissimilarities are intimately related with pyramidal clustering problems [66, 67].

The seriation problem [65, 68] consists of finding a total order for which the given pairwise distances violates the Robinson conditions as little as possible. To link this seriation problem with the Colored Multigraph Betweenness Problem or Betweenness Problem we consider a collections $\mathscr{C}$ of triples such that

$$\begin{aligned}(\rho(i),\rho(j),\rho(k)) \in \mathscr{C} \text{ implies max} \\ \{d(\rho(i),\rho(j)),d(\rho(j),\rho(k))\} < d(\rho(i),\rho(k))\end{aligned} \quad (3)$$

Clearly, if the dissimilarity is Robinsonian, then ρ defines a total order on X that solves the Betweenness Problem for $(X,\mathscr{C})$.

The relevant optimization task in our context is therefore to minimize the number of ordered triples that violate Eq. (2). A variety of heuristics for this problem have been developed, see e.g. [69]. It is important to note, however, that in our setting the distance between alignment blocks is not defined directly. In order to obtain a seriation problem that approximates the Supergenome Sorting Problem we will have to resort to a heuristic that summarizes the distances between two blocks in all genomes and reflects the betweenness relationships. For pangenome-like models, the cost function advocated in [28] is a very plausible choice.

Graph simplification

Each of the plausible models for the "Supergenome Sorting Problem" discussed in the previous sections leads to NP-hard computational problems. The size of typical genome-wide alignments by far exceeds the range where exact solutions can be hoped for, except possibly for the smallest and most benign examples such as the ones used as examples in [17]. We therefore will have to resort to fast heuristics. In this section we focus on the conceptual ideas behind the simplification steps. More detailed implementation details are given in the "Methods" section.

Nevertheless it is possible to isolate certain sub-problems that can be solved exactly and independently of the remainder of the input graph. Since "linearized" portions of the vertex set can be contracted to a single vertex set, this leads to a reduction of problem size.

Lemma 3 *If the supergenome graph* Γ *is a directed acyclic graph then topological sorting of* Γ *solves the* Colored Multigraph Betweenness Problem.

Proof In this case betweenness is established exactly by the directed paths in the DAG. Hence any topological sorting preserves all betweenness triples of G and thus presents a perfect solution to the Colored Multigraph Betweenness Problem as well.

This simple observation suggests to identify subgraphs with DAG structure and to replace them with a representative for each replaced DAG. These can later be replaced by the solution that is created with topological sorting. We note that this does not necessarily preserve optimality. It is conceivable that a local DAG structure has to be broken up into two disjoint subsets that are integrated in larger surrounding structures in a way that requires reversal of the edge directions in one or even both parts. Nevertheless, if the local DAG structures are sufficiently isolated they are likely to be part of the optimal solution as a unit. The motif that describes such a local DAG structure has been introduced as "superbubble" in the context of graph structures arising in sequence assembly problems [70, 71].

Source and sink vertices s in the supergenome graph with only a single neighbor t are conceptually a special case of superbubbles. These can be sorted together with their unique neighbor t. Γ is thus simplified by contracting s and t, i.e., placing the source s immediately before t and sink s immediate after t. An example of such a source and a sink can be seen in Fig. 4.

In some cases it is helpful to reverse the direction of the coordinate system of a single species. This is in particular the case when a single genome is reversed compared to all others. The inversion of an entire path does not change the solution of the Colored Multigraph

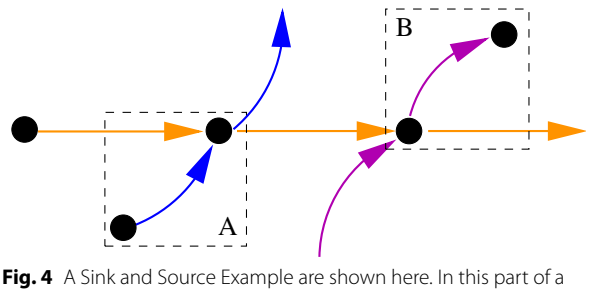

Fig. 4 A Sink and Source Example are shown here. In this part of a bigger supergenome graph a sink and a source exists. The source is in field A. It has only one successor and thus can be merged with this successor. The sink is in field B. It has only one predecessor and thus can be merged with this predecessor

Betweenness Problem but can make it easier to apply some of the reduction heuristics discussed above. In particular, if the relative orientation of the coordinatizations could be fixed in an optimal manner, the betweenness problem reduces to a much easier topological sorting problem. Finding this optimum, however, is equivalent to the Betweenness Problem, which is a NP-hard. Hence, we again have to resort to local heuristics.

Definition 7 Let $\Gamma = (V, E)$ be a supergenome graph. A pair of vertices $v, w \in V$ such that there are edges (v, w) and (w, v) in E is a *mini cycle*.

Mini-cycles are naturally removed by removing one of the two edge directions between v and w. More precisely, the less supported direction of an edge is dropped. The estimate for support is evaluated in a region around a mini-cycle since adjacent mini-cycles may yield contradictory majority votes.

Definition 8 Two mini-cycles are connected with each other if they share a vertex. A mini-cycle complex $\mathfrak{C}$ is a maximal connected set of mini-cycles.

Lemma 4 *The mini-cycle complexes of a supergenome graph* Γ *form a unique partition of the set of all minicycles. Any two classes of this partition are vertex and edge disjoint.*

Proof Consider the the graph H whose vertices are the mini-cycles, and there are edges between any two mini-cycles that share at least one vertex. Then every mini-cycle complex $\mathfrak{C}$ is a connected component of H. Since the connected components of a graph are uniquely defined, disjoint, and form a cover, they partition the vertex set of H. Every minicycle, furthermore, forms a connected subgraph of Γ by construction. Since any two minicycles that contain a common vertex belong to the same mini-cycle complex, two mini-cycle complexes cannot have a vertex in common. This implies that they are also edge disjoint.

The mini-cycle complexes therefore can be resolved independently of each other. The target is to remove edges that create cycles in order to obtain a DAG that can then be subjected to topological sorting. However, this topological sorting is a solution of the Colored Multigraph Betweenness Problem for a subgraph. This is still a hard problem, so that we again utilize a heuristic approach. In this step we only attempt to remove mini-cycles. Cycles that connect mini-cycle complexes with each other or with other vertices in the graph are therefore left untouched and have to be dealt with in a subsequent step.

The local sorting within a complex $\mathfrak{C}$ is achieved by considering adjacencies. To this end we annotate each adjacency with the number of edges and the ratio of the edges in the two directions. We identify the best supported edges as those with a high multiplicity and a strong bias for one direction over the other. This choice of a direction is then propagated. If a directed edge has more than one possible successor, we first propagate along the one with the largest support for the proposed direction. The issue now is when exactly to stop propagating this information. Clearly, it is forbidden to orient an edge that would close a directed cycle. Any such edge is instead seeded with the reverse directional information.

As part of this procedure it is possible that parts of a directed path from a given genome received contradictory orientations in different regions. If this is the case, the edge crossing the boundary between the differently oriented regions must be removed. Finally, the heuristic may terminate and still leave some edges unoriented. This indicates that the orientations are contradictory and need to be reversed. An example of the mini-cycle resolution process is shown in Fig. 5.

Methods

Curation of input data sets

We investigate here three genome-wide data sets. The smallest set, referred to as **B** (bacteria) below, is an alignment of four *Salmonella enterica* serovars. This alignment was produced with `Cactus` [27] using the *Salmonella enterica Newport* genome as reference and comprises 13, 416 blocks, 50, 932 sequence fragments, and 18, 047, 456 nucleotides. The medium-size set, termed **Y** (yeast), is an alignment of seven yeast species that uses the *Saccharomyces cerevisiae* genome as references. It comprises 49, 795 alignment blocks composed of 275, 484 sequences fragments that contains 71, 517, 259 nucleotides. The third, much larger set **F** (fly) is a alignment of 27 insect species that uses the *Drosophila melanogaster* genome as references. It comprises 2, 112, 962 blocks composed of 36, 139, 620 sequence fragments hat contains 2, 172, 959, 429 nucleotides. For more detailed information of the data sets refer to Additional file 1: Section 2.

The two large genome-wide multiple sequence alignments were produced by the `multiz` pipeline and were downloaded from the `UCSC` genome browser [72]. They are, as discussed above, injective but not irredundant. In order to remove spurious alignment blocks we filter the input blocks with respect to first length, then score, and finally mutual overlap. Very short alignment blocks are almost certainly either spurious matches or they were

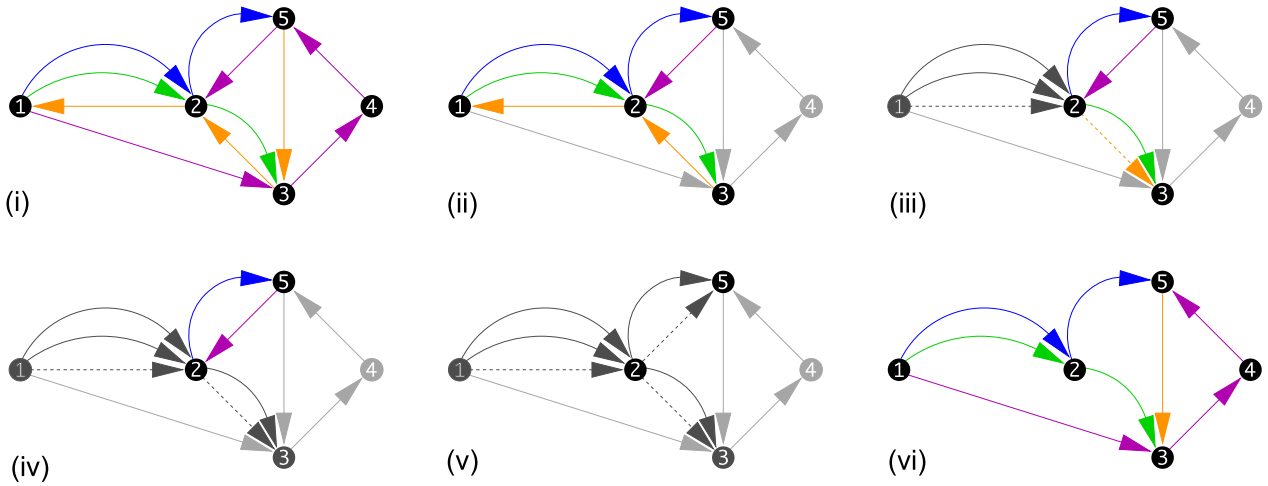

Fig. 5 Step-wise resolution of a complex of mini-cylces. **(i)** Starting point. **(ii)** The mini-cycle complex is highlighted. The complex is created from the mini-cycles {1, 2}, {2, 3}, and {2, 5}. Note that the edges (1, 3) and (5, 3) are not contained in the complex. The best supported directions are between (1, 2). **(iii)** This direction between (1, 2) is then set. The orange edges are therefore reversed (marked by dashed lines). The adjacency {1, 2} is decided and is no longer considered (marked with dark grey). **(iv)** In this step the best supported direction is (2, 3) and the graph is updated correspondingly. **(v)** Adjacency {2, 5} is left. No direction is superior. Since vertex 2 was solved for previously it is now used. This leads to direction (2, 5). **(vi)** Then the completed complex is decided and the edges that are contradictionary with the decisions are removed. Note that the circle 3, 4, 5, 3 that was not part of the complex is not removed

inserted to bridge gaps between larger blocks. Consequently, they convey little or no useful information for our purposes. We therefore remove all blocks with a length ≤ 10 nt.

Since genome-wide alignments tend to contain also very poorly aligned regions we require a minimum similarity, expressed here in the form of sum-of-pairs `blastz` scores [73]. Since these scale linearly with the number of columns $\ell(B)$ of the alignment block B and the number $\binom{r}{2}$ of pairwise alignments formed by the r rows in B, we normalize with $\binom{r}{2}\ell(B)$ to obtain a similarity measure that is independent of the size of the alignment block. Based on the parametrization of `blastz`, we set the threshold at a normalized score of -30, which corresponds to the gap extension penalty.

The coordinatization of the supergenome depends on the uniqueness of coordinate projections. There are three major reasons to observe overlaps, i.e., genomic regions that appear in more than one alignment: (i) the sequence is duplicated in some species. Then `multiz` tends to align the corresponding unduplicated sequence to both duplicates. (ii) Spurious similarities in particular in poorly conserved regions may lead to alignments containing a sequence element twice at the expense of the second copy. (iii) Short overlaps at the end of blocks may appear due to difficulties in determining the exact ends of alignable regions. The first two causes introduce undesirable noise and uncertainties. Therefore, we remove all such overlapping blocks in which sequences from the species overlap. Since there is no easy way to determine which one of two overlapping blocks is likely correct, we opt to remove both copies. The third case, in contrast, does not disturb the relative order of alignment blocks and thus can be ignored. The overlap filter is applied after low quality alignments already have been removed from the data set.

We tolerate an overlap of 20 nt at the borders of alignment blocks. This cutoff is designed to remove ambiguous alignments, while avoiding the removal of alignment blocks that overlap by a few nucleotides owing to overlapping extensions of local `blastz` seeds. In addition we remove sequences that completely overlap other sequences regardless of their size to further reduce the noise introduced by spurious alignments. We opt here for a stringent procedure and remove all alignment blocks that contain sequences tagged for removal. In practice, this step removes only a tiny fraction of the blocks and thus does not significantly influence the coverage of the retained data.

The initial data filtering steps removed almost 35% (40%, 30%) of the blocks from data set **F**, (**Y**, **B**). The majority were eliminated because of their minimal length. About 8.5% (27%, 0%) of the blocks were removed because they contained non-unique sequences. The sequences in the blocks that are removed with all filters contain less then 15% (26%, 0.4%) of the nucleotides in the alignment. Hence more than 85% (74%, 99%) of the

sequence information of the alignment is intact and the quality of the data is significant better. A more detailed summary of the filtering is compiled in Additional file 1: Section 2.

Graph simplification and DAG construction

The algorithmic ideas and their justifications for the graph reduction steps have already been discussed in "Theory" section. Here we briefly address implementation issues as well as particular choices of cost functions and parameters that were discussed in a more general setting above.

The filtered data is used to create an initial supergenome graph. Then we iterate the three different graph simplifiers until no further reduction steps can be applied: the mini-cycle remover, the source/sink simplifier, and the superbubble simplifier. The individual simplifiers are straightforward implementations of the basic ideas outlined above. The mini-cycle remover first identifies the mini-cycles, aggregates them into non-overlapping complexes, and then proceeds to remove contradictory edges in a greedy manner. The other two simplifiers first check for each vertex in the input graph whether it is a valid sink, source, or starting vertex of a superbubble. Pseudocode of the simplifiers is given in Additional file 1: Section 4.

The mini-cycle remover works more effectively on a single big complex than on many small ones separated by narrow gaps. The other two simplifiers therefore are applied until a fixed point is reached to close some of these gaps. The entire procedure is then iterated until the minicycle remover cannot change the graph any further.

Once a fixed point is reached we attempt to remove directed cycles. This amounts to solving the Minimum Feedback Arc Set Problem, which is known to be NP-hard [37]. Given the size of our input graphs we have to resort to linear-time heuristics. We use `Algorithm GR` [38] because it is known to work particularly well on sparse graphs. Cycle removal typically creates new possibilities to simplify the graph. For instance, a sink is created whenever the last outgoing edge of a vertex is removed. The new sink can then be simplified further. The graph simplifiers are therefore applied again after the cycle removal step.

The minicycle remover is not used in this second pass because it is not applicable to DAGs by construction. Instead, we use a generalized version of the source/sink simplifier in which a source s may have more than a single successor v, provided v is a predecessor of all other successors of s. The position of source s in the DAG is determined by v and thus s can be placed immediately before v. The corresponding arrangements for a sink and its predecessor is treated analogously.

The running time and the minimization of the data set while this process is applied is shown in Figs. 6, 7 and 8.

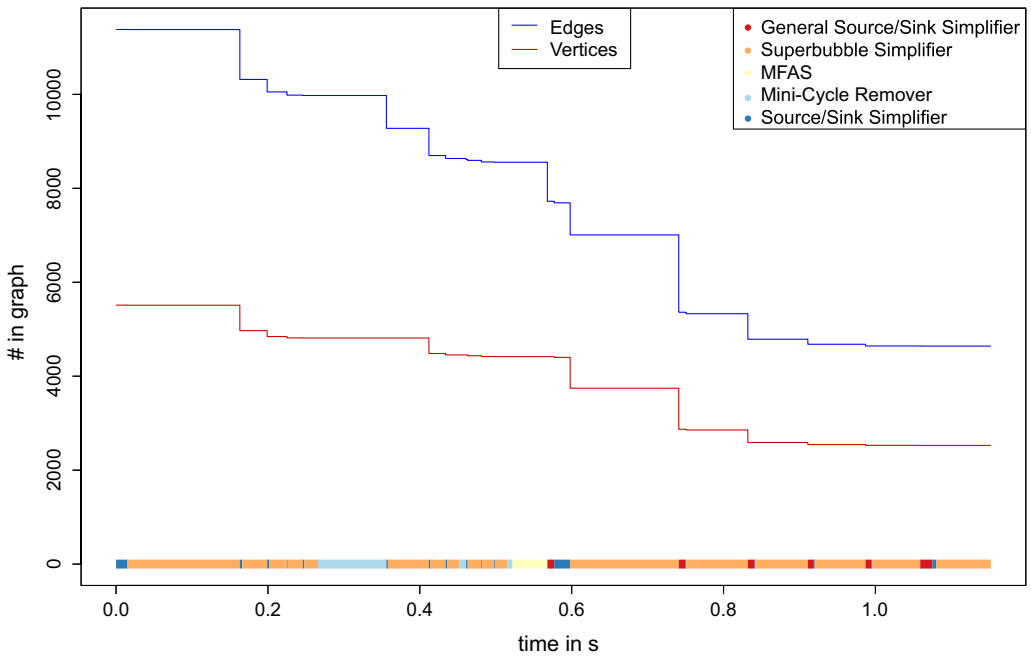

Fig. 6 Simplification process for data set **B**. The size of the graph (number of edges and vertices in the graph) is shown while the simplifier and the MFAS are applied. The running time is computed on a Intel(R) Xeon(R) CPU E7-8860 processor with 32 Gb RAM. At the bottom, the different processes are shown as color-coded bars

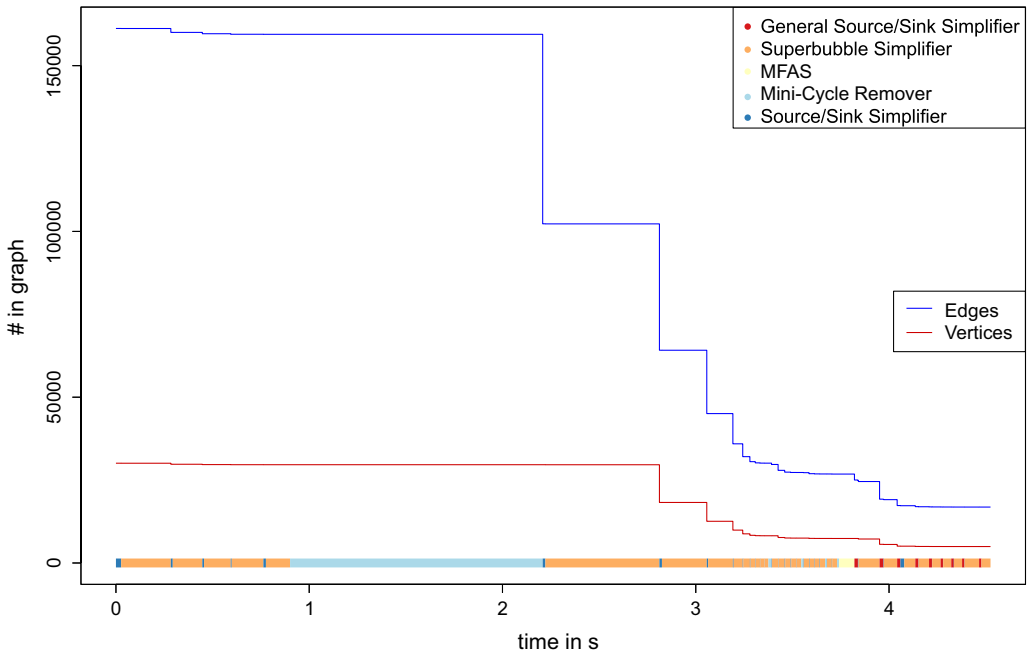

Fig. 7 Simplification process for data set **Y**. The size of the graph (number of edges and vertices in the graph) is shown while the simplifier is applied and the MFAS are solved. The running time is computed on a Intel(R) Xeon(R) CPU E7-8860 processor with 32 Gb RAM. At the bottom, the different processes are shown as color-coded bars.

Seriation

Finally, the common coordinate system is created by seriation of the DAG. The resulting supergenome, i.e. linear order of the vertices of the graphs corresponds to a linear order of all blocks. In particular, vertices resulting from a simplifier may contain more than one block. Those blocks, however, are already sorted and thus are inserted as a single block. Seriation is naturally divided into two steps. First, topological sorting is used to calculate an initial linear ordering from the DAG. Kahn's algorithm [40] is a classical solution to the topological sorting problem. For our purposes it is desirable that, if possible, two nodes v and w are placed consecutively whenever there is an edge (v, w) in the final DAG. To this end we modify Kahn's algorithm by sorting the successors of a node in the order of evidence for their adjacency in the original data.

The order obtained in this manner may not be optimal w.r.t. its agreement with the order of the blocks in the genomes. It provides a good starting point, however, for the final optimization step, which we phrase as minimizing the number τ of triplets (i, j, k) for which the Robinson condition, Eq. (2), page 12, is violated. We use the distance measure

$$d(i,k) = \begin{cases} \frac{1}{|(i,k)|} & \text{if an edge } (i,k) \text{ exists,} \\ \min_{i<j<k}\{d(i,j)+d(j,k)\} & \text{if a path from i to k through j exists,} \\ \infty & \text{if no path from i to k exists,} \end{cases} \tag{4}$$

where $|(i, k)|$ is the number of edges from i to k. Since d is a good measure of co-linearity only for short distances, we limit the path length in Eq. 4 to a small number of l edges. We set $l = 10$ in our implementation. In addition this reduces the effort of computing the distances from $O(|V|^2)$ to $O(|V|)$ as a consequence of the sparsity of the input graph.

We use a gradient descent-like optimization algorithm to minimize τ. We say that two nodes are siblings if they either share a predecessor in the DAG or if they are both sources. The move set for the gradient descent consists of swaps of siblings only. In addition, we allow to move a node directly in front of its sibling. The discrete gradient is computed exhaustively by generating and evaluating each potential move. Since non-overlapping swaps do not influence each other, we greedily execute a maximal set of non-overlapping swaps in a single optimization step.

Assessment of the quality of supergenome coordinate systems

Since no ground truth is available for this problem and the construction of simulated benchmarks for genome wide multiple sequence alignments would be a research

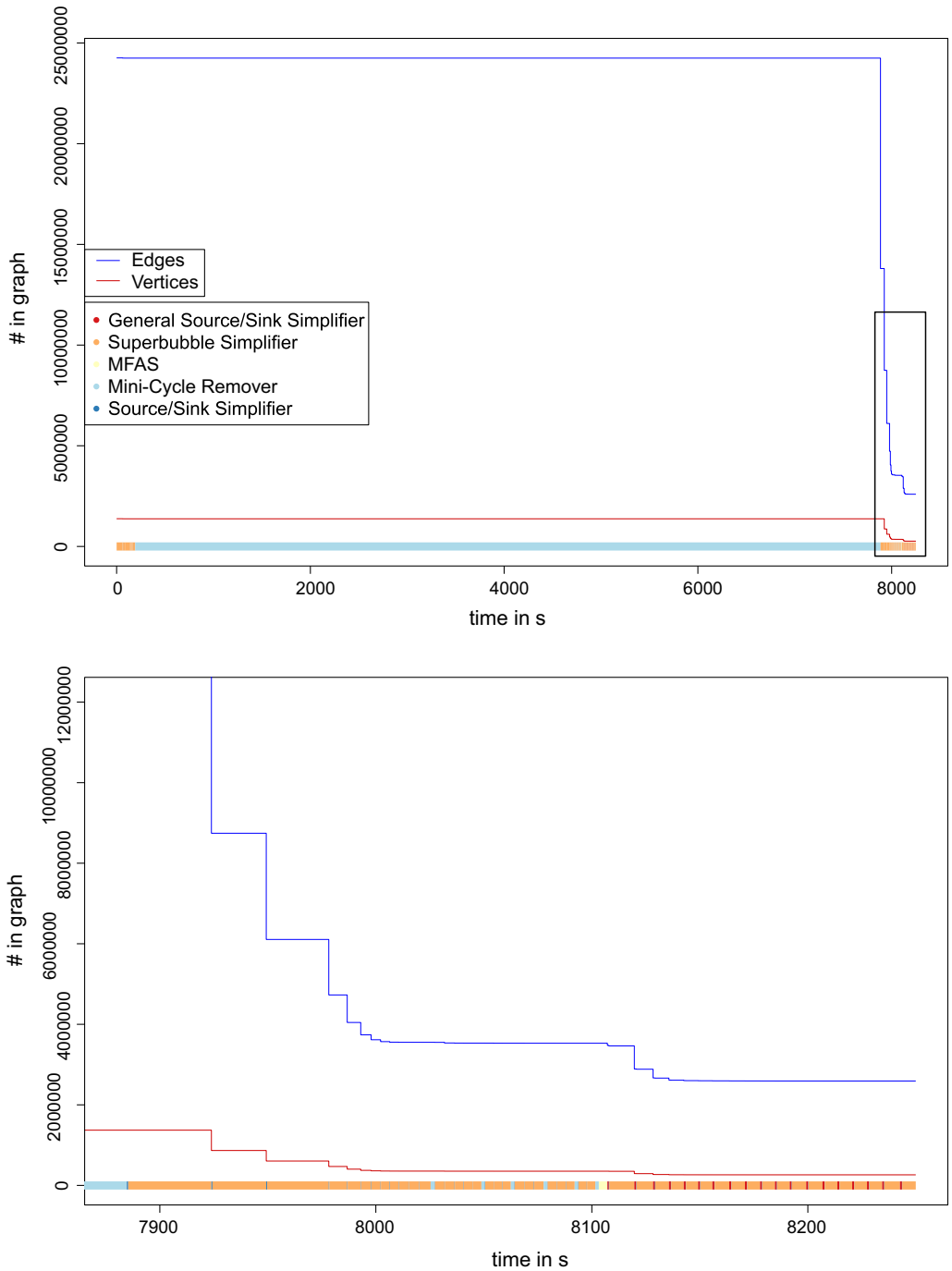

Fig. 8 Simplification process for data set **F**. The size of the graph (number of edges and vertices in the graph) is shown while the simplifier is applied and the MFAS is solved. The running time is computed on a Intel(R) Xeon(R) CPU E7-8860 processor with 32gb ram. At the bottom, the different processes are shown as color-coded bars

project in its own right, we have to resort to measuring quantities that are informative about the final choice of the coordinate system.

A straightforward measure is the distribution of distances in the output coordinate system of alignment blocks that are contiguous in at least one input genome. Since we are not interested in the length of alignment blocks, distance is measured here not in terms of sequence length but in terms of the number of alignment blocks, so that adjacent blocks have distance 0. It

is important here to keep track of the reading directions: contiguity with the same reading direction corresponds to preservation of the original genomic coordinates, while a change in reading direction indicates change of the order. Thus we distinguish preserved and inverted reading direction in our quantitative analysis.

Open reading frames (ORFs) are among the best-conserved features in the genome due to the strong selection pressures acting to preserve the corresponding proteins. As an immediate consequence we expect that ORFs are almost always preserved. This should be reflected also by the supergenome coordinates, i.e., blocks belonging to the same ORF should have only a small number (smaller then 5) of other blocks between them and retain their relative order. For higher eukaryotes, we cannot expect near perfect adjacency of coding blocks, however, because larger introns are subject to local rearrangements. To quantify the proximity of blocks of an ORF, the distances between all adjacent blocks are determined as described above and their absolute values are added up to yield a single characteristic value. In addition we count the number of exons that are "broken up" in the sense that consecutive pieces do not have consecutive coordinates or are placed in reverse order in the supergenome. Coding genes and exons are taken as annotated for the corresponding genomes. We note that in particular for large, intron-rich genomes such as the insect data set **F** this is an additional source of errors.

Results and discussion

We have devised a heuristic algorithm to extract a common coordinate system for a supergenome from a genome-wide multiple sequence alignment. The procedure has been tested on three alignments of very different size and difficulty: an easy instance comprising four closely related bacterial species, an intermediate size problem composed of seven yeast genomes, and the alignment of 27 insect genomes as the most difficult instance.

Performance of individual components

The heuristic algorithm outlined above is composed of several largely independent components. It is of interest, therefore to consider their relative impact on the final results. We find that most edges are removed by the mini-cycle remover, with a small contribution of `Algorithm GR`. On the other hand, the largest reduction of the vertex set is due to the merges identified by the closed DAG simplifier. More quantitative information is compiled in Figs. 6, 7 and 8 and in Additional file 1: Section 8. The simplifiers reduce the graph size by about an order of magnitude in both the number of vertices and edges, reducing it in size and complexity to a point where the seriation heuristic operates efficiently. The relative improvement is smallest for the bacterial data set.

Since the Colored Multigraph Betweenness Problem cannot be solved exactly in reasonable time for instances with sizes that are of interest for our application at hand, we cannot measure performance relative to the exact solution. The multigraphs obtained from real-life alignments contain a large number of conflicting edges. In the most difficult data set, **F**, for instance, the final order keeps more than 95% of the initial edges.

Quality of supergenome coordinate systems

The quality of the coordinate systems strongly depends on the quality of the input alignments. A detailed discussion of issues with the input alignments can be found in Additional file 1: Section 8. Here, we focus on an assessment of the coordinate systems themselves.

In order to check the overall quality of the solution we compute a betweenness graph from the supergenome coordinate systems. This is done by starting with a graph without edges. First, all edges that are supported by the total order of the supergenome are added. This is followed by edges that contradict the total order but do not create contradicting betweenness triples. Note that this graph is not necessary optimal but a good approximation that can easily be computed. The edge set of this graph is compared to the edge set of the initial graph. Good solutions are expected to retain most of the edges. For the three data sets we find that 95.3%, 97.5%, and 99.4% of the edges are retained in data sets **B**, **Y**, and **F**, respectively.

The distribution of block-wise distances in the supergenome of alignment blocks that are consecutive in the original genome serves as a simple measure of preserved synteny. The results are summarized in Fig. 9 and presented in full detail in Additional file 1: Section 8. Another measure is how many of the input orders are preserved. To measure this we consider every alignment block and all successors from the different genomes. For the bacterial data set **B** 89% of the successors preserve the order and 80% also preserve the adjacency. For the yeast data set **Y** we observe that 93% of the successors preserve the order and 84% also preserve adjacency. This is a very encouraging result, taking in mind that every true genome rearrangement necessarily introduces at least one non-adjacency. Even in the much larger and more difficult insect set **F** we still find 70% of the successors preserve the order and 66% also preserve adjacency. The overwhelming majority of non-contiguous successors are placed in the adjacent but order-reversed position, reflecting the level of local rearrangements in the insect data set. This is entirely reasonable given the much larger number of species and their larger phylogenetic depth compared to the yeast data. Taken together, these numbers already indicate that the supergenome coordinates

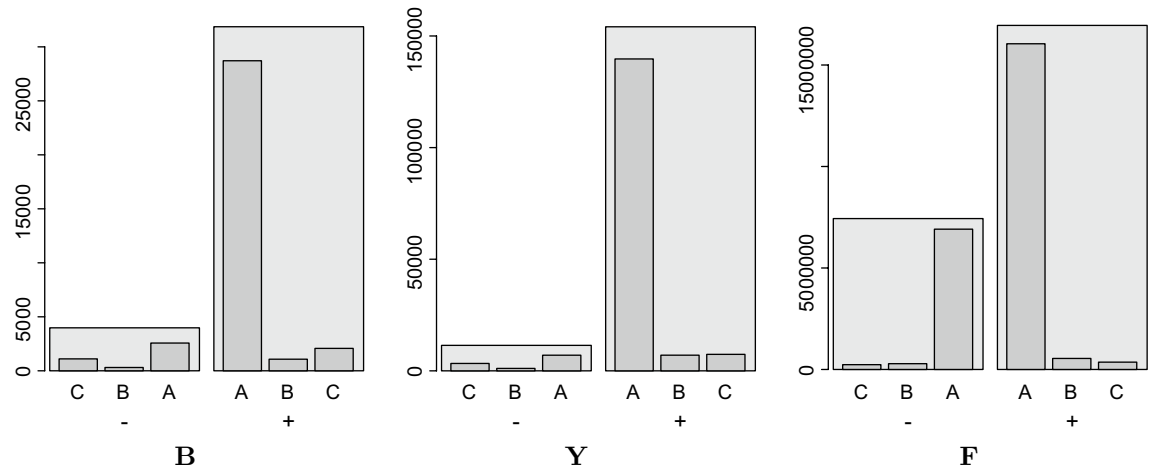

Fig. 9 Distribution of block-wise distances of consecutive alignment blocks in the original genomes. Data are shown separated for inverted (−) and preserved (+) orientation of consecutive blocks (light gray). As expected, the number of inverted blocks increases with the difficulty of the input alignment. In particular, there is a substantial number of local inversions in the insect data set **F**. Both the inverted (−) and preserved (+) bin are subdivided further into a bin of adjacent blocks (A), blocks with a distance of 1–5 blocks (B), and more distantly placed blocks (C), in the supergenome

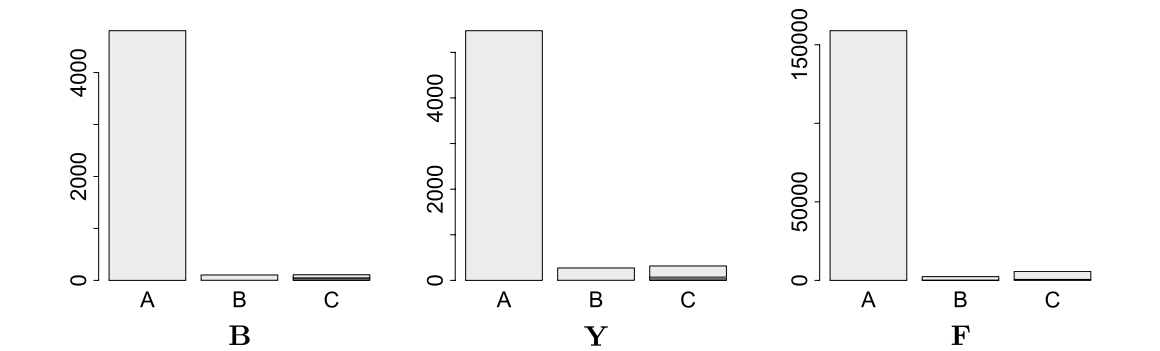

Fig. 10 Distribution of block-wise distances between alignment blocks that contain ORFs for the bacteria and yeast data set. For the insect data invidual exons are considered since rearrangements as well as alignment errors within introns are not infrequent. The black bar indicates the number of broken ORFs/exons. The data is binned in three distance ranges: a distance of 0 (A), a distance of 1–100 (B), and a distance larger than 100 (C)

are meaningful and indeed are likely a useful starting point for large-scale multi-genome comparisons.

Restricting our attention to coding sequences yields a more stringent quality measure, as shown in Fig. 10. As bacteria have essentially no introns, we expect that nearly all blocks belonging to the same ORF retain both adjacency and order. In the bacterial data set **B** 96% of the ORFs are in one stretch with no interruption and less then 1% of the ORFs are broken. Since yeasts have few and short introns [74] we expect that data set **Y** is also very well-behaved in this respect. It contains 6062 ORFs annotated for *Saccharomyces cerevisiae*. Of these, 5474, i.e., 90%, are consistently represented in the coordinate system. An additional 272 ORFs, about 5%, have a distance of less then 100 blocks between them. Only 73, i.e., a bit more than 1% of the ORFs are broken. For *Drosophila melanogaster* are 167, 051 exons annotated, and part of the alignment **F**. Due to large and abundant introns the analysis is based on individual exons rather than complete ORFs for set **F**. We observe that 95% of the ORFs/exons are consistently represented. Only 779, about 0.5%, are broken. Overall, thus, the supergenome coordinates behave very well for all three data sets.

As a specific example we consider the genes of the yeast TCA cycle [75] in more detail. It is one of the best-studied enzyme systems and known to be essential in *S. cerevisiae*. There, it comprises 20 genes [76–79], all of which are contained at least partially in the initial set of alignment blocks in the yeast data set **Y**. Only nine genes are included in their entirety, however. Seven of these nine

are represented colinearly in single blocks. The alignments for KGD2 and SDH2 cover multiple MSA blocks and there is intervening genomic sequence in the input alignment, leading to non-contiguous placement in the final coordinate system. The alignment blocks referring to the remaining 11 genes are difficult to analyze and may contain misassigned sequences. This example, similar to several other loci, suggests that the quality of the input alignment rather than the complexity of the betweenness problem is the limiting factor for the construction of supergenome coordinate systems.

Conclusion

In this contribution we have shown that the problem of computing a common coordinate system for supergenomes with the Colored Multigraph Betweenness Problem is NP-hard. It belongs to a class of relatively poorly studied betweenness problems for which few efficient heuristics have been developed so far. We introduced here several local simplification rules that can be applied iteratively to reduce the problem. It is important to note these reduction steps are only heuristics and do not guarantee optimal solutions. In conjunction with a simple serialization approach for the residual graph, they nevertheless yield practically useful results with acceptable computational efforts.

The most immediate application of the supergenome sorting problem is the direct comparison of genome annotations for multiple genomes. Hence it constitutes a prerequisite for comparative genome browsers. We have applied our approach to three real-life data sets of different sizes and difficulties. Our results indicate that practically useful coordinatizations can be computed. The computational requirement of the method scales favorably so that in principle even the largest genome-wide multiple sequence alignments could be processed.

The present study, however, also highlights the shortcomings of currently available genome-wide multiple sequence alignments [80, 81]. The issue is not only the relatively moderate coverage with alignment blocks that contain at least most of the species under consideration, but also the substantial fractions of alignment blocks that have been removed from our data set due to likely artefactual sequences. We have therefore not attempted to analyze the UCSC 100-way vertebrate alignments, since these data are even more complex than the insect data due to the very large number of paralogs introduced by genome duplications.

Synteny, i.e., the preservation of relative genome order, is in general a good predictor for homology. This fact suggests to use the common coordinate system to identify likely homologous regions that are not included in the initial alignment blocks. These could then be (re) aligned at sequence level and included in a revised multiple sequence alignment. This, in turn, could yield an improved common coordinate system. The systematic improvement of genome-wide alignments, albeit an interesting and extremely useful endeavor, is beyond the scope of this contribution.

Possible improvements of the approach taken here are conceivable in at least two directions. First, one may consider a hybrid algorithm that solves subgraphs with a dominant backbone use the method discussed in [29]. As discussed, we assume that large parts of the global graph structure are not amenable to such a solution, but it is also reasonable to assume that gene regions under strong conservation pressures can be solved fairly easily using a local backbone-based approach. A second venue of research is concerned with the determination of the final backbone order. Depending on the phylogenetic range under investigation, the ancestral gene order would provide a useful backbone based on the phylogeny of the species involved in the alignment.

Additional file

Additional file 1. Additional Text covering some proofs, details on the data set used for evaluation, pseudocode of the algorithms described in the main text, and additional benchmark data.

Authors' contributions

CHzS and PFS designed the study, FG and LM implemented the software and performed the computational analysis. All authors collaborated on the design of the algorithms and the overall work flow, the interpretation of results, and the writing of the manuscript. All authors read and approved the final manuscript.

Author details

[1] Competence Center for Scalable Data Services and Solutions Dresden/Leipzig, Universität Leipzig, Augustusplatz 12, 04107 Leipzig, Germany. [2] Bioinformatics Group, Department of Computer Science, Universität Leipzig, Härtelstraße 16–18, 04107 Leipzig, Germany. [3] Interdisciplinary Center for Bioinformatics, Universität Leipzig, Härtelstraße 16–18, 04107 Leipzig, Germany. [4] Automatic Language Processing Group, Department of Computer Science, Universität Leipzig, Augustusplatz 12, 04107 Leipzig, Germany. [5] Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, 04103 Leipzig, Germany. [6] Department of Theoretical Chemistry, University of Vienna, Währinger Straße 17, 1090 Vienna, Austria. [7] Center for non-coding RNA in Technology and Health, Grønegårdsvej 3, 1870 Frederiksberg C, Denmark. [8] Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501, USA.

Acknowledgements

This work was funded by the German Federal Ministry of Education and Research within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B). We acknowledge support for APCs from the German Research Foundation (DFG) and Universität Leipzig within the program of Open Access Publishing.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate
Not applicable.

Publisher's Note

Received: 14 June 2017 Accepted: 7 September 2018
Published online: 24 September 2018

References

1. Gawad C, Koh W, Quake SR. Single-cell genome sequencing: current state of the science. Nat Rev Genet. 2016;17(3):175–88.
2. Genomes Project Consortium. A global reference for human genetic variation. Nature. 2015;526(7571):68–74.
3. Hezroni H, Koppstein D, Schwartz MG, Avrutin A, Bartel DP, Ulitsky I. Principles of long noncoding RNA evolution derived from direct comparison of transcriptomes in 17 species. Cell Rep. 2015;11:1110–22. https://doi.org/10.1016/j.celrep.2015.04.023.
4. Lin S, Lin Y, Nery JR, Urich MA, Breschi A, Davis CA, Dobin A, Zaleski C, Beer MA, Chapman WC, Gingeras TR, Ecker JR, Snyder MP. Comparison of the transcriptional landscapes between human and mouse tissues. Proc Natl Acad Sci USA. 2014;111:17224–9. https://doi.org/10.1073/pnas.1413624111.
5. Necsulea A, Kaessmann H. Evolutionary dynamics of coding and non-coding transcriptomes. Nat Rev Genet. 2014;15:734–48. https://doi.org/10.1038/nrg3802.
6. Neme R, Tautz D. Fast turnover of genome transcription across evolutionary time exposes entire non-coding DNA to de novo gene emergence. Elife. 2016;5:e09977.
7. Washietl S, Kellis M, Garber M. Evolutionary dynamics and tissue specificity of human long noncodingRNAs in six mammals. Genome Res. 2014;24:616–28.
8. Nguyen N, Hickey G, Raney BJ, Armstrong J, Clawson H, Zweig A, Karolchik D, Kent WJ, Haussler D, Paten B. Comparative assembly hubs: web-accessible browsers for comparative genomics. Bioinformatics. 2014;30(23):3293–301.
9. Darling AE, Mau B, Perna NT. progressivemauve: multiple genome alignment with gene gain, loss and rearrangement. PloS ONE. 2010;5(6):11147.
10. Schwartz S, Kent WJ, Smit A, Zhang Z, Baertsch R, Hardison RC, Haussler D, Miller W. Human-mouse alignments with blastz. Genome Res. 2003;13(1):103–7.
11. Blanchette M, Kent WJ, Riemer C, Elnitski L, Smit AF, Roskin KM, Baertsch R, Rosenbloom K, Clawson H, Green ED. Aligning multiple genomic sequences with the threaded blockse aligner. Genome Res. 2004;14(4):708–15.
12. Paten B, Herrero J, Beal K, Fitzgerald S, Birney E. Enredo and pecan: genome-wide mammalian consistency-based multiple alignment with paralogs. Genome Res. 2008;18:1814–28.
13. Bray N, Pachter L. `MAVID`: constrained ancestral alignment of multiple sequences. Genome Res. 2004;14:693–9. https://doi.org/10.1101/gr.1960404.
14. Chen X, Tompa M. Comparative assessment of methods for aligning multiple genome sequences. Nat Biotech. 2010;28:567–72. https://doi.org/10.1038/nbt.1637.
15. Xiao S, Cao X, Zhong S. Comparative epigenomics: defining and utilizing epigenomic variations across species, time-course, and individuals. Wiley Interdiscip Rev Syst Biol Med. 2014;6:345–52. https://doi.org/10.1002/wsbm.1274.
16. Nitsche A, Rose D, Fasold M, Reiche K, Stadler PF. Comparison of splice sites reveals that long non-coding RNAs are evolutionarily well conserved. RNA. 2015;21:801–12. https://doi.org/10.1261/rna.046342.114.
17. Herbig A, Jäger G, Battke F, Nieselt K. GenomeRing: alignment visualization based on SuperGenome coordinates. Bioinformatics. 2012;28:7–15.
18. Dugar G, Herbig A, Förstner KU, Heidrich N, Reinhardt R, Nieselt K, Sharma CM. High-resolution transcriptome maps reveal strain-specific regulatory features of multiple *Campylobacter jejuni* isolates. PLoS Genet. 2013;9:1003495. https://doi.org/10.1371/journal.pgen.1003495.
19. Goryunov DV, Nagaev BE, Nikolaev MY, Alexeevski AV, Troitsky AV. Moss phylogeny reconstruction using nucleotide pangenome of complete mitogenome sequences. Biochemistry (Mosc). 2015;80:1522–7. https://doi.org/10.1134/S0006297915110152.
20. Medini D, Donati C, Tettelin H, Masignani V, Rappuoli R. The microbial pan-genome. Curr Op Genet Dev. 2005;15:589–94. https://doi.org/10.1016/j.gde.2005.09.006.
21. Bodlaender HL, Fomin FV, Koster AMCA, Kratsch D, Thilikos DM. A note on exact algorithms for vertex ordering problems on graphs. Theory Comput Syst. 2012;50:420–32.
22. Li K, Tang X, Veeravalli B, Li K. Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. IEEE Trans Comput. 2015;64(1):191–204.
23. Fellows MR, Hermelin D, Rosamond F, Shachnai H. Tractable parameterizations for the minimum linear arrangement problem. ACM Trans Comput Theory. 2016;8(2):6.
24. Pardo EG, Martí R, Duarte A. Linear layout problems. Berlin: Springer; 2016. p. 1–25.
25. Kececioglu J. The maximum weight trace problem in multiple sequence alignment. Combinatorial pattern matching. Berlin: Springer; 1993. p. 106–19.
26. Pevzner PA, Tang H, Tesler G. De novo repeat classification and fragment assembly. Genome Res. 2004;14(9):1786–96.
27. Paten B, Earl D, Nguyen N, Diekhans M, Zerbino D, Haussler D. Cactus: algorithms for genome multiple sequence alignment. Genome Res. 2011;21(9):1512–28.
28. Nguyen N, Hickey G, Zerbino DR, Raney B, Earl D, Armstrong J, Kent WJ, Haussler D, Paten B. Building a pan-genome reference for a population. J Comput Biol. 2015;22(5):387–401.
29. Haussler D, Smuga-Otto M, Paten B, Novak AM, Nikitin S, Zueva M, Miagkov D. A flow procedure for the linearization of genome sequence graphs. In: International conference on research in computational molecular biology. Berlin: Springer; 2017, p. 34–49.
30. Giegerich R. Explaining and controlling ambiguity in dynamic programming. Annual symposium on combinatorial pattern matching. Berlin: Springer; 2000. p. 46–59.
31. Sankoff D. Time warps, string edits, and macromolecules. The theory and practice of sequence comparison, reading. Boston: Addison-Wesley; 1983.
32. Belda E, Moya A, Silva FJ. Genome rearrangement distances and gene order phylogeny in γ-proteobacteria. Mol Biol Evol. 2005;22:1456–67. https://doi.org/10.1093/molbev/msi134.
33. Drillon G, Fischer G. Comparative study on synteny between yeasts and vertebrates. C R Biol. 2011;334:629–38. https://doi.org/10.1016/j.crvi.2011.05.011.
34. Fischer G, Rocha EPC, Brunet F, Vergassola M, Dujon B. Highly variable rates of genome rearrangements between hemiascomycetous yeast lineages. PLoS Genet. 2006;2:32. https://doi.org/10.1371/journal.pgen.0020032.
35. Friedberg R, Darling AE, Yancopoulos S. Genome rearrangement by the double cut and join operation. Methods Mol Biol. 2008;452:385–416.
36. El-Metwally S, Hamza T, Zakaria M, Helmy M. Next-generation sequence assembly: four stages of data processing and computational challenges. PLoS Comput Biol. 2013;9:1003345. https://doi.org/10.1371/journal.pcbi.1003345.
37. Karp RM. Reducibility among combinatorial problems. Complexity of computer computations. Berlin: Springer; 1972. p. 85–103.
38. Eades P, Lin X, Smyth WF. A fast and effective heuristic for the feedback arc set problem. Inf Process Lett. 1993;47:319–23.
39. Saab Y. A fast and effective algorithm for the feedback arc set problem. J Heuristics. 2001;7:235–50. https://doi.org/10.1023/A:1011315014322.
40. Kahn AB. Topological sorting of large networks. Commun ACM. 1962;5(11):558–62.
41. Martí R, Reinelt G. The linear ordering problem: exact and heuristic methods in combinatorial optimization, vol. 175. Berlin: Springer; 2011.
42. Grötschel M, Jünger M, Reinelt G. A cutting plane algorithm for the linear ordering problem. Oper Res. 1984;32:1195–220.
43. Spearman C. The proof and measurement of association between two things. Am J Psychol. 1904;15:72–101.

44. Kendall MG. A new measure of rank correlation. Biometrika. 1938;30:81–93.
45. Fagin R, Kumar R, Sivakumar D. Comparing top *k* lists. SIAM J Discrete Math. 2003;17:134–60. https://doi.org/10.1137/S0895480102412856.
46. Fried C, Hordijk W, Prohaska SJ, Stadler CR, Stadler PF. The footprint sorting problem. J Chem Inf Comput Sci. 2004;44:332–8.
47. Collier JH, Konagurthu AS. An information measure for comparing top *k* lists. In: 2014 IEEE 10th international conference on e-science, vol. 1. 2014, p. 127–34. https://doi.org/10.1109/eScience.2014.39.
48. Bertrand D, Blanchette M, El-Mabrouk N. Genetic map refinement using a comparative genomic approach. J Comput Biol. 2009;16:1475–86.
49. Oswald M, Reinelt G. The simultaneous consecutive ones problem. Theor Comput Sci. 2009;410:21–3.
50. Booth KS, Lueker GS. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J Comput Syst Sci. 1976;13:335–79.
51. Meidanis J, Porto O, Telles GP. On the consecutive ones property. Discrete Appl Math. 1998;88:325–54.
52. Tucker A. A structure theorem for the consecutive 1's property. J Comb Theory B. 1972;12:153–62.
53. Christof T, Oswald M, Reinelt G. Consecutive ones and a betweenness problem in computational biology. In: Bixby RE, Boyd EA, Ríos-Mercado RZ, eds. Integer programming and combinatorial optimization, vol. 1412. 1998, p. 213–28.
54. Reid JK, Scott JA. Reducing the total bandwidth of a sparse unsymmetric matrix. SIAM J Matrix Anal Appl. 2006;28:805–21.
55. Cuthill E, McKee J. Reducing the bandwidth of sparse symmetric matrices. In: Proceedings of 24th National Conference ACM. New York: ACM; 1969, p. 157–72. https://doi.org/10.1145/800195.805928.
56. Gibbs NE, Poole WG Jr, Stockmeyer PK. An algorithm for reducing bandwidth and profile reduction algorithms. SIAM J Numer Anal. 1976;13:236–50.
57. Feige U. Coping with the NP-hardness of the graph bandwidth problem. In: Algorithm Theory—SWAT 2000, vol. 1851. 2000, p. 129–45.
58. Kehr B, Trappe K, Holtgrewe M, Reinert K. Genome alignment with graph data structures: a comparison. BMC Bioinf. 2014;15(1):99.
59. Gavril F. Some NP-complete problems on graphs. In: Proceedings of the 11th Conference on Information Sciences and Systems. Baltimore: Johns Hopkins University; 1977, p. 91–5.
60. Makedon FS, Papadimitriou CH, Sudbourough IH. Topological bandwidth. SIAM J Algebraic Discrete Methods. 1985;6:418–44.
61. Martí R, Pantrigo JJ, Duarte A, Pardo EG. Branch and bound for the cutwidth minimization problem. Comput Oper Res. 2013;40:137–49.
62. Barth D, Pellegrini F, Raspaud A, Roman J. On bandwidth, cutwidth, and quotient graphs. Informatique théorique et applications. 1995;29:487–508.
63. Opatrny J. Total ordering problem. SIAM J Comput. 1979;8:111–4.
64. Chor B, Sudan M. A geometric approach to betweenness. SIAM J Discr Math. 1998;11:511–23.
65. Robinson WS. A method for chronologically ordering archaeological deposits. Amer Antiquity. 1951;16:293–301.
66. Bertrand P. Systems of sets such that each set properly intersects at most one other set—application to cluster analysis. Discrete Appl Math. 2008;156:1220–36.
67. Bertrand P, Diatta J. Multilevel clustering models and interval convexities. Discrete Appl Math. 2017;222:54–66. https://doi.org/10.1016/j.dam.2016.12.019.
68. Liiv I. Seriation and matrix reordering methods: an historical overview. Stat Anal Data Min. 2010;3:70–91.
69. Hahsler M, Hornik K, Buchta C. Getting things in order: an introduction to the R package `seriation`. J Stat Softw. 2008;25:3.
70. Onodera T, Sadakane K, Shibuya T. Detecting superbubbles in assembly graphs. In: International workshop on algorithms in bioinformatics. Berlin: Springer; 2013, p. 338–48.
71. Paten B, Novak AM, Garrison E, Hickey G. Superbubbles, ultrabubbles and cacti. In: International conference on research in computational molecular biology. Berlin: Springer; 2017, p. 173–89.
72. Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler D. The human genome browser at ucsc. Genome Res. 2002;12(6):996–1006.
73. Chiaromonte F, Yap V, Miller W. Scoring pairwise genomic sequence alignments. Pac Symp Biocomput. 2001;7:115.
74. Spingola M, Grate L, Haussler D, Ares M Jr. Genome-wide bioinformatic and molecular analysis of introns in *Saccharomyces cerevisiae*. RNA. 1999;5:221–34.
75. Krebs H, Gurin S, Eggleston L. The pathway of oxidation of acetate in Baker's yeast. Biochem J. 1952;51(5):614.
76. Saccharomyces Genome Database Community: SGD Yeast Pathway: *Saccharomyces cerevisiae* TCA cycle, aerobic respiration. http://pathway.yeastgenome.org/YEAST/NEW-IMAGE?object=TCA-EUK-PWY. Accessed 18 May 2017.
77. Haselbeck RJ, McAlister-Henn L. Function and expression of yeast mitochondrial nad-and nadp-specific isocitrate dehydrogenases. J Biol Chem. 1993;268(16):12116–22.
78. Oyedotun KS, Lemire BD. The carboxyl terminus of the *Saccharomyces cerevisiae* succinate dehydrogenase membrane subunit, sdh4p, is necessary for ubiquinone reduction and enzyme stability. J Biol Chem. 1997;272(50):31382–8.
79. Yasutake Y, Watanabe S, Yao M, Takada Y, Fukunaga N, Tanaka I. Crystal structure of the monomeric isocitrate dehydrogenase in the presence of nadp+ insight into the cofactor recognition, catalysis, and evolution. J Biol Chem. 2003;278(38):36897–904.
80. Earl D, Nguyen N, Hickey G, Harris RS, Fitzgerald S, Beal K, Seledtsov I, Molodtsov V, Raney BJ, Clawson H, Kim J, Kemena C, Chang JM, Erb I, Poliakov A, Hou M, Herrero J, Kent WJ, Solovyev V, Darling AE, Ma J, Notredame C, Brudno M, Dubchak I, Haussler D, Paten B. Alignathon: a competitive assessment of whole-genome alignment methods. Genome Res. 2014;24:2077–89. https://doi.org/10.1101/gr.174920.114.
81. Ezawa K. Characterization of multiple sequence alignment errors using complete-likelihood score and position-shift map. BMC Bioinf. 2016;17:133. https://doi.org/10.1186/s12859-016-0945-5.