RESEARCH

Coordinate Systems for Supergenomes

Fabian Gärtner[1,2*], Christian Höner zu Siederdissen[2,3], Lydia Müller[4,3,1] and Peter F Stadler[2,3,1,5,6,7,8,9]

*Correspondence: fabian@bioinf.uni-leipzig.de
[1]Competence Center for Scalable Data Services and Solutions Dresden/Leipzig, Universität Leipzig, Augustusplatz 12, D-04107 Leipzig, Germany
[2]Bioinformatics Group, Department of Computer Science, Universität Leipzig, Härtelstraße 16–18, D-04107 Leipzig, Germany
Full list of author information is available at the end of the article

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 show that this problem can be phrased as a particular variant of the Betweenness Problem that 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.

Genome-wide multiple sequence alignments (gMSAs) provide a natural starting point. Several pipelines to construct such alignments have been deployed over the past two decades, most prominently the `tba/multiz` pipeline [8, 9] employed by the `UCSC` genome browser and the `Enredo/Pecan/Ortheus` (`EPO`) pipeline [10] featured in the `ensembl` system. For the ENCODE project data, in addition alignments generated with `MAVID` [11] and `M-LAGAN` [11] have become available, see [12] 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 [13] structure. 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 [14] then yields an informative common coordinate system. So far, this approach has been been applied only to closely related procaryotic genomes. Prime examples are a detailed comparative analysis of the transcriptome of multiple isolates of *Campylobacter jejuni* [15] or the reconstruction of the phylogeny of mosses from the "nucleotide pangenome" of mitogenomic sequences [16]. 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 [17].

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 [18].

This contribution is organized as follows: In the following section we first analyse the concept of the supergenome and its relationship to gMSAs in detail. We then argue that the most appropriate modelling of the "supergenome sorting problem" leads to a special type of betweenness ordering problem rather than intuitively more appealing graph problems such as Hamiltonian path or consecutive ones problems. We then introduce a heuristic solution that is geared towards very large input alignments and proceeds by stepwise simplification of the supergenome multigraph. We then outline a few computational results.

Theory

Genome-wide multiple sequence alignments

Our starting point is a set $\mathcal{G}$ of genome assemblies. For our purposes an assembly $g \in \mathcal{G}$ is simply a set of sequences representing chromosomes, scaffolds, reftigs, contigs, etc. 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 (g, c, i, j, σ) to identify the sequence interval from positions i to j on assembly element c of genome assembly 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 $B = \{(g_u, c_u, i_u, j_u, \sigma_u) | u = 1, ..., r\}$ 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 $\pi_g(B)$ extracts from an alignment block the union of its constituent sequence intervals belonging to assembly g. If the assembly g is not represented in the alignment block B we set $\pi_g(B) = \emptyset$. We define, furthermore, that the union of overlapping intervals is the union of the intervals $(g, c, i', j', \sigma') \cup (g, c, i'', j'', \sigma'') = (g, c, i, j, \sigma)$ with $i = \min(i', i'')$, $j = \max(j', j'')$ without regard for the orientation $\sigma = \pm 1$. The projection π_g of $\mathfrak{A}$ onto one of its constituent assemblies g is the union of the sequence intervals from g that is contained in its alignment blocks, i.e., $\bigcup_{B \in \mathcal{A}} \pi_g(B)$.

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

- $\mathfrak{A}$ *is* complete *if $\pi_g(\mathfrak{A}) = g$, i.e., if each position in each assembly is represented in at least one alignment block.*
- $\mathfrak{A}$ *is* irredundant *$\pi_g(B') \cap \pi_g(B'') = \emptyset$ for any two distinct blocks B' and B'', i.e., if every sequence interval from assembly 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 an item (g, c) in a (genome) assembly, each block $B \in \mathcal{A}$ has an internal coordinate system defined by its columns. We write (B, k) for column k in block B. If $\mathfrak{A}$ is irredundant, then there are functions $f_{g,c}$ that map position i within assembly item (g, c) to a corresponding MSA coordinate (B, k). If $\mathfrak{A}$ is complete, the individual $f_{g,c}$ can be combined to a single function $f : (g, c, i) \mapsto (B, k)$: completeness implies that every position (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 (g, c, i) corresponds to at most one alignment column. The following definition is therefore equivalent to the notion of a supergenome introduced in [14].

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` [9] alignments are injective this is in general not the case with the `EPO` [10] alignments. In these, multiple paralogous sequences from the same genome may appear in one alignment block.

Now consider a 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 $N=\sum_{B\in\mathfrak{A}}|B|$ is the total number of columns in $\mathfrak{A}$ such that $f(B,i)<f(B',i')$ 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 g to $[1,N]$. We call $\phi(f(g,c,i))$ the *coordinate* of position i of item c of assembly g in the ordered supergenome $(\mathcal{A},<)$.

As pointed out in [14], 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 two intervals $\alpha=(g',c',i',j',\sigma')$ and $\beta=(g'',c'',i'',j'',\sigma'')$ *overlap* if $g'=g''$, $c'=c''$, and $\max(i',i'')\leq\min(j',j'')$. The interval is *between* if $\gamma=(g,c,k,l,\sigma)$ is between the two distinct intervals: if $g=g'=g''$, $c=c'=c''$ and either $j'<k\leq l<i''$ or $j''<k\leq l<i'$. In particular, if γ is between α and β, than γ overlaps neither α nor β.

Definition 3 *Given a collection $\mathcal{Q}$ of intervals we say that $\alpha=(g',c',i',j',\sigma')$ and $\beta=(g'',c'',i'',j'',\sigma'')$ are* adjacent (in $\mathcal{Q}$) *if*

(i) $g'=g''$ and $c'=c''$;
(ii) α and β do not overlap;
(iii) There is no interval $\gamma\in\mathcal{Q}$ between α and β

We regard a genome assembly g as the collection of its intervals. By construction, no two distinct intervals $\alpha,\beta\in g$ overlap or are adjacent. Given a MSA $\mathfrak{A}$ consider the collections $\mathcal{Q}_g(\mathfrak{A})=\{\pi_g(B)|B\in\mathfrak{A}\}$ of projections of alignment blocks to fixed assembly g. Then no two alignment blocks overlap if and only if $\mathfrak{A}$ is irredundant. Since the projections of blocks that map to the same assembly item c is linearly ordered, any interval $\alpha\in\mathcal{Q}_g(\mathfrak{A})$ has at most two adjacent intervals, namely its successor and it predecessor along c.

Definition 4 *Two blocks $A,B\in\mathfrak{A}$ are adjacent if there are adjacent intervals $\alpha\in A$ and $\beta\in B$. We denote the set of adjacent pairs by $\mathfrak{E}$.*

In other words the alignment blocks $A,B\in\mathfrak{A}$ are adjacent if there is an assembly g so that $\pi_g(A)$ and $\pi_g(B)$ are adjacent in $\mathcal{Q}_g(\mathfrak{A})$.

It is useful to regard $\mathfrak{A}$ with its adjacency relation as a graph Γ. In a supergenome, the projection to each constituent assembly g can be regarded as a (not necessarily induced) subgraph Γ_g of Γ. Alternatively, we may view $\mathfrak{A}$ as an edge-colored multigraph Γ obtained as superposition of the simple graph Γ_g with color g. In the following it will sometimes be convenient to also record the reading direction on each assembly element.

Definition 5 *A* supergenome graph *is a directed, arc-colored multigraph such that (i) any two vertices are incident to a most one arc of each color and (ii) the subgraphs induced by the arcs of a given color are disjoint directed paths.*

In the absence of genome rearrangements (i.e., when the only genetic changes are substitutions, insertions (including duplications), and deletions) then all genomes remain collinear with their common ancestor. In other words, a single global alignment describes a common coordinate system that is unique up to the (arbitrary) order of chromosomes. 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, however [19–22], and thus cannot be neglected. Every break point introduced by a genome rearrangment 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 identifying an appropriate ordering of the MSA blocks therefore is a non-trivial one for realistic data, even in the absence of alignment errors.

Modelling 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 *such that the orders of the constituent assemblies are preserved as much as possible*. The latter condition is not quite well-defined, however.

Hamiltonian Paths

A plausible attempt is to view the SSP as variant of the Hamiltonian Path problem. There are several quite obvious difficulties. 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 α, β, γ, δ and three assemblies:

$$\begin{array}{cccc} \alpha & & & \delta \\ \alpha & \beta & & \delta \\ \alpha & & \gamma & \delta \end{array}$$

This situation arises e.g. when β and γ are two independent inserts between α and δ. The block adjacency graph is the graph β-α-δ-γ, which violates betweenness implied by the second and third line. In this case there are only two biologically correct solutions: $\alpha < \beta < \gamma < \delta$ (or the inverse order) and $\alpha < \gamma < \beta < \delta$ (or its inverse). In either case, the solution contains two consecutive blocks (β and γ) 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.

Feedback Arc Sets and Topological Sorting

The colored edges of $\tilde{\Gamma}$ can be directed so that they point in the direction of increasing genomic coordinates. We note, however, that this direction orientation is purely conventional and has no biological meaning. Nevertheless, a well-defined order of the blocks can be obtained by extracting a maximum acyclic subgraph. An equivalent formulation asks for removing a minimum set of arcs that close cycles. This Maximum Acyclic Subgraph or Minimum Feedback Arc Set problem

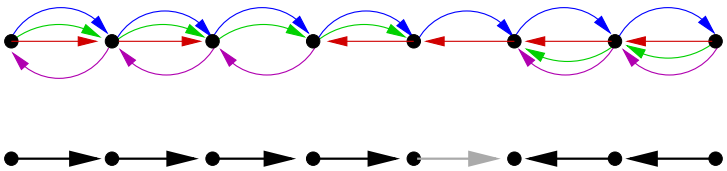

Figure 1 Minimum Feedback Arc Set (MFAS) does not necessarily yield the best betweenness order. Any acyclic solution requires the removal of at least one arc between adjacent vertices. Two such solutions exist, differing only by the orientation of the gray arrow. Both solve MFAS. The corresponding coordinatization breaks the genome into two distinct collinear pieces with opposite orientation. There is, however, a consistent coordinatization of the entire graph – the linear left-to-right or right-to-left order is consistent.

(MFAS) is well-known to be NP-hard [23]. It is fixed parameter tractable (FPT) [24] but APX hard [25]. Nevertheless fast, practicable heuristics have been devised, see e.g. [26, 27]. From the resulting directed acyclic graph an admissible ordering of blocks can be obtained efficiently by topological sorting [28]. A closely related approach is the Linear Ordering Problem (LOP): Given a complete weighted directed graph, find a tournament with maximum total arc weights [29]. It yields essentially the same model since LOP and MFAS can be transformed into each other quite easily [30].

The key problem of modelling the coordinatization problem in terms of DAGs is highlighted in Fig. 1. 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 of 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 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) [31]. 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 [31] that $\mathbf{A}$ satisfies (C1S) if and only if $\mathbb{A}$ has the well-studied consecutive ones property [32, 33] for both its rows and columns. Thus (C1S) can be checked in linear time [32]. Furthermore, Tucker's characterization of (C1S) in terms of forbidden submatrices [34] also carries over. Some direct connection between the consecutive ones property and the `Betweenness Problem` are discussed in [35].

In general, the consecutive ones property will be violated. The problem of identifying a minimal number of columns (adjacencies) whose removal leaves a (C1S) matrix is NP-complete [31]. 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 1's. For instance, one may want to use $\phi = \sum_i f(\ell_i)$, where the sum runs over all

intervals i of consecutive zeros enclosed by the two 1's, and $f(\ell_i) \geq 0$ is some contribution that monotonically grows with the length ℓ_i of the 0-interval. For a given ordering (π_1, π_2) of the rows and columns, the total violation is assessed summing the ϕ values. It should be noted, however, that (C1S) does not imply Γ is a union of disjoint paths.

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 $u = \max_{a_{ij}\neq 0}(i - j)$ and $l = \max_{a_{ij}\neq 0}(j - i)$ [36]. In the symmetric case, several good heuristics are known, starting with the Cuthill–McKee [37] and GPS [38] algorithms even though the problem is NP-hard [39], while the general case has received much less attention [36]. 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.

Betweenness Problems

It would appear that the most natural cost function for the SSG is to minimize the number of wrong betweenness triples. Consider an order ρ used to coordinatize the supergenome. We may think of ρ as a bijective function from $[1, \ldots, n] \to \mathfrak{A}$. For $i < j < k$ we set $b_g(i, j, k) = 1$ if the projections of the three alignment blocks $\rho(i)$, $\rho(j)$, and $\rho(k)$ violate the betweenness relation, i.e., if $\pi_g(\rho(i))$ and $\pi_g(\rho(k))$ are located on the same assembly item and $\pi_g(\rho(j))$ is not located between them. Otherwise we set $b_g(i, j, k) = 0$. A natural cost function is now the total number of betweenness violations

$$b(\rho) := \sum_g \sum_{i<j<k} b(i, j, k) \tag{1}$$

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 $b(\rho) = 0$.

This optimization problem is associated with the Betweenness Problem: *Given a finite set X and a collection $\mathscr{C}$ of triples from X, is there a total order on X such that $(i, j, k) \in \mathfrak{C}$ either $i < j < k$ or $i > j > k$?* Since this decision problem is NP-complete [40, 41], so is the problem of optimizing $b(\rho)$. Since the cost function $b(\rho)$ is fairly expensive to evaluate, one might want to consider variations on this theme. To address this issue, the sum in equ.(1) could be restricted to local information.

This idea leads us to the following, rather natural formulation for our problem at hand:

Colored Multigraph Betweenness Problem: Find a maximal subset of colored edges E^* of the multigraph $\tilde{\Gamma}$ such that the set of triples $\mathscr{C}(E^*)$ has a total order, where $(i, j, k) \in \mathscr{C}(E^*)$ if and only if there is color g such that $\{i, j\}$ and $\{j, k\}$ are edges with color g.

This problem can be viewed as an analog of the feedback arc set problem [26] for betweenness data. To our knowledge is has not been studied so far.

Lemma 1 *The (decision version of the)* Colored Multigraph Betweenness Problem *is NP-complete.*

Proof Every set $\mathscr{C}$ of triples can be obtained from an edge-colored multigraph (with vertices corresponding to alignment blocks and colored edges corresponding to adjacencies deriving from a genome identified by the color). Thus, $\mathscr{C}$ is a solution of the Colored Multigraph Betweenness Problem if and only if the answer to the NP-complete Betweenness Problem is "true". □

In the example of Fig. 1 the optimal solution of the Colored Multigraph Betweenness retains all *unordered* adjacencies and results a unique coordinatization (up to orientation) that leaves all alignment blocks ordered as drawn.

Seriation

If a distance measure $d : X \times X \to \mathbb{R}$ is given, a betweenness relation can be obtained by virtue of $(i, j, k) \in \mathscr{C}$ if $\max\{d(i, j), d(j, k)\} \leq d(i, j)$. If there is a linear order π on X such that the condition is satisfied for points in this order, i.e.,

$$\max\{d(\pi(i), \pi(j)), d(\pi(j), \pi(k))\} \leq d(\pi(i), \pi(k)) \tag{2}$$

is satisfied for all $i < j < k$, then the distance d is said be *Robinsonian*. Clearly, if the distance is Robinsonian, π defines a total order on X that solves the Betweenness Problem for $(X, \mathscr{C})$.

The seriation problem [42, 43] consists of finding a total order for which the given pairwise distance satisfies the Robinson conditions as well as possible. To this end we minimize the number of ordered triples that violate equ.(2). A variety of heuristics for this problem have been developed, see e.g. [44]. 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.

Graph Simplification

Each of the plausible models for the "Supergenome Sorting Problem" discussed in the previous section 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 [14]. 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 2 *If the supergenome graph G is a directed acyclic graph then the* Colored Multigraph Betweenness Problem *is solved by a topological sorting.*

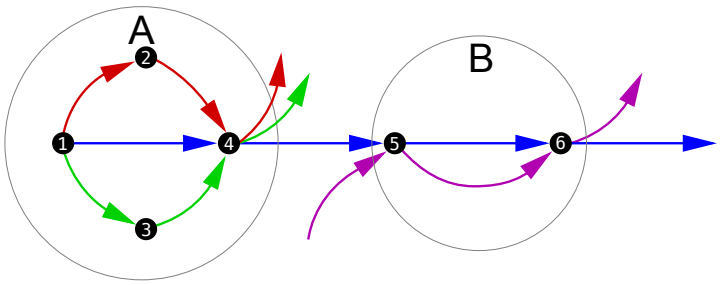

Figure 2 Closed DAGs. The fragment of a larger supergenome graph shown here contains two closed DAGs, A with source 1 and sink 4, and B with source 5 and sink 6 comprising only these two vertices.

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 hence presents a perfect solution to the Betweenness Problem as well. □

This simple observation suggests to identify subgraphs with DAG structure and to replace them by a 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 arc 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. We propose here a fairly general type of motif to be simplified:

Definition 6 *A subgraph $\hat{G}$ of the supergenome graph G is a* closed DAG *if*

(i) $\hat{G}$ *is a directed acyclic graph;*

(ii) $\hat{G}$ *is connected to* $G \setminus \hat{G}$ *by a single source vertex* v *and a single sink vertex* w*, where* $v \neq w$*;*

(iii) all direct successors of v *and all direct predecessors of* w *are contained in* $\hat{G}$*;*

(iv) all vertices in $\hat{G} \setminus \{v\}$ *are successors of* v *and all vertices in* $\hat{G} \setminus \{w\}$ *are predecessors of* w*.*

Two examples of closed DAGs are shown in Figure 2. We say that a closed DAG is trivial if it consists only of the source and sink vertices v and w and edges between them.

Source and sink vertices s in the supergenome graph with only a single neighbor t are conceptually a special case of closed DAGs. These can be sorted together with their unique neighbour t. G 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 a source and a sink can be seen in Figure 3.

In some cases it is helpful to reverse the direction of 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 affect the Colored Multigraph Betweenness Problem but can make it easier to apply some of the reduction heuristics discussed above. In particular, if the relative orientation of

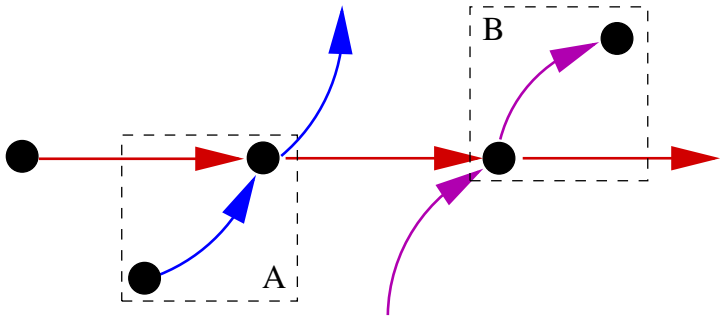

Figure 3 A Sink and Source example are shown here. In this part of a bigger supergenome graph a sink and a source exists. The source is in field A. It has only one successor and can be merged in this successor. The sink is in field B. It has only one predecessor and can be merged in this predecessor.

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 in itself a hard problem, 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 arcs (v, w) and (w, v) in Γ is a* mini cycle.

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

Definition 8 *Let M be the set of all mini-cycles in the supergenome graph Γ and let P be a partition of M*

(i) If C_1 and C_2 are two distinct complexes, then for all mini-cycles $c_1 \in C_1$ and all $c_2 \in C_2$ we have $c_1 \cap c_2 = \emptyset$, i.e., mini-cycles from two distinct complexes are vertex disjoint.

(ii) The classes P are C are minimal in the sense that they cannot be subdivided further. More precisely, $P' = (P \setminus C) \cup \{C \setminus S, S\}$ does not satisfy condition (i) for any C and any non-empty subset $S \subseteq C$.

We refer to the classes $C \in P$ as mini-cycle complex.

An example of a mini-cycle complex is shown in Figure 1.

Lemma 3 *For every given set of mini-cycles only one valid set of complexes exists.*

Proof Since P is a partition, it contains the all elements of the set of mini-cycles. Condition (i) makes these vertex and arc disjoint. Hence they form a lattice, of which condition (ii) picks the uniquely defined connected components. □

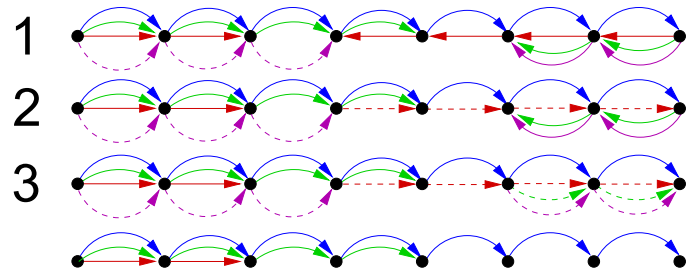

Figure 4 Stepweise resolution of a sequence of mini-cylces. Starting point is the graph of Figure 1. The blue, green, and red edges are the best supported directions at the beginning. The magenta edges are therefore reversed in the left part (marked by dashed lines). In the second step, a subset of red edges is marked as reversed since the conflict with majority vote (green and blue edges) in the part of the chain that has been resolved in the first step. In the third step edges that conflict with the consensus direction of the previous step are resolved. The last row then shows the edges that are retained; all dashed edges are removed.

The mini-cycle complexes can be resolved independent of each other. The target is to remove edges that create cycles in order to obtain a partial order, that can then be sorted topologically. Cycle removal in this context amounts to solving the minimum arc-set problem. This is still a hard problem, so that we again utilize a heuristic approach. In this step we only attempt to remove only 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 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 large number of arcs 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 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 arcs crossing the boundary between the differently oriented regions must be removed. Finally, the heuristic may terminate and still leave some arcs unoriented. This indicates that the orientations are contradictory and need to be reversed. An example of the mini-cycle resolution process is shown in Figure 4.

Methods

Curation of input data sets

We investigate here three genome-wide data sets. The smallest set, referred to as B below, is an aligment of four *Salmonella enterica* serovars. This alignment was produced with `Cactus` [45] using the *Salmonella enterica Newport* genome as reference and comprises 13 416 blocks, 50 932 sequences fragments, and 18 047 456 nucleotides. The medium-size set, termed Y, 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 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 the Additional File 1 Section 1.

The two large genome-wide multiple sequence alignments were produced by the `multiz` pipeline and were downloaded from the `UCSC` genome browser [46]. 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 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 [47]. Since these scale linearly with the length ℓ of the alignment block and the number $\binom{N}{2}\ell$ of pairwise alignments formed by the N sequences, we normalize with $\binom{N}{2}\ell$ to obtain a similarity measure that is independent of the size of the alignment block. Based on the parametrization of `blastz`, we use 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 my 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 avoid spurious random alignments of short sequence fragments, while on the other avoiding the removal of alignment blocks that overlap by a few nucleotides owing to overlapping extensions of local `lastz` 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 complete remove all alignment blocks that contain sequences tagged for removal. In practise, 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% of the blocks from data set F. The majority were eliminated because of their minimal length. About 8.5% 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% of the nucleotides in the alignment. Hence more than 85% 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 the 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 closed DAG simplifier. The individual simplifiers are the 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 closed DAG. Pseudocode of the simplifiers is given in Additional File 1 Section 3.

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 [23]. Given the size of our input graphs we have to resort to linear-time heuristics. We use `Algorithm GR` [26] 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. This is, 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.

Seriation

Finally, the common coordinate system is created by seriation of the DAG. The resulting 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 [28] 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 arc $v \to w$ 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, is violated. We use the distance measure

$$d_{ik} = \begin{cases} \frac{1}{|e_{ik}|} & \text{if an edge } e_{ik} \text{ exists,} \\ \min_{i<j<k}\{d_{ij} + d_{jk}\} & \text{if a path from } i \text{ to } k \text{ through } j \text{ exists,} \\ \infty & \text{if no path from } i \text{ to } k \text{ exists,} \end{cases} \tag{3}$$

where $|e_{ik}|$ is the number of edges from i to k. Since d_{ik} is a good measure of co-linearity only for short distances, we limit the path length in Eq. 3 to a small number of l edges. 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 one of a pair of siblings to be moved in front of its brother independently of its original location. 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 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 block 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 remain close-by and retain their relative order. For higher eukaryotes, of course, we cannot expect near perfect adjacency of coding blocks, however, because larger introns are of course 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 of course 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 the MFAS step. 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 Additional File 1 Section 7. 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. Not surprisingly, the relative improvement is smallest for the bacterial dataset.

Since the Colored Multigraph Betweenness Problem cannot be solved exactly 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 alignment contain a large number of conflicting arcs. In the most difficult data set, F, for instance, the final order keeps more than 95% of this initial edges.

Quality of supergenome coordinate systems

Not surprisingly, the quality of the coordinate systems depends strongly on the quality of the input alignments. A detailed discussion of issues with the input alignments can be found in Additional File 1 Section 7. Here, we focus on an assessment of the coordinate systems themselves.

In order to check the overall quality of the solution we compute the betweenness graph from the final order of the supergenome and compare its edge set to the initial

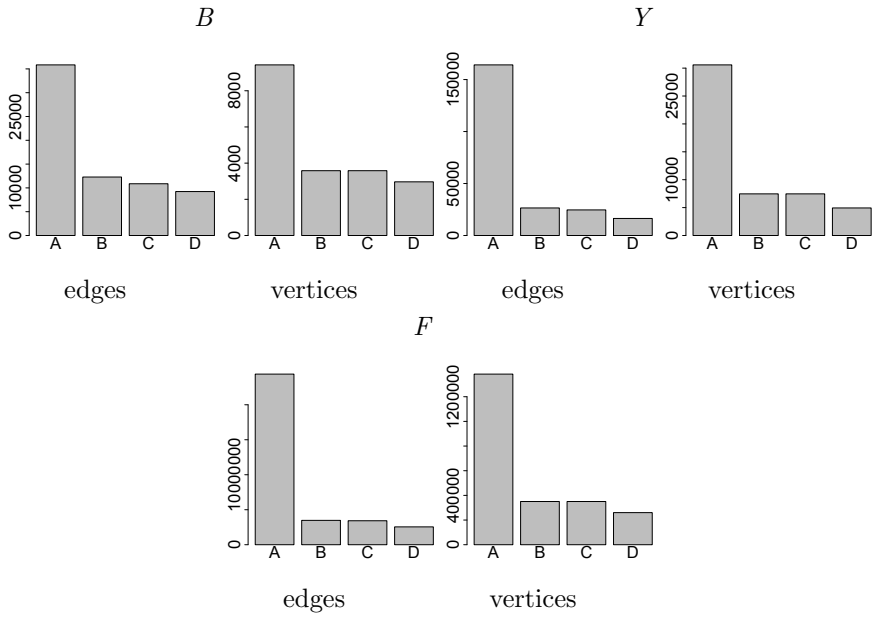

Figure 5 Relative effect of the different stages of the graph simplification heuristics. For each of the three data sets we show the number of edges and vertices in the original graph (A), after the initial simplification step (B), after the application of the FAS-algorithm (C), and at the end of the simplification procedure (D).

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 simple measure of preserved synteny. The results are summarized in Fig. 6 and presented in full detail in Additional File 1 Section 7. 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. Noting that every true genome rearrangement necessarily introduces at least one non-adjacency this is very encouraging result. Even in the much larger and more difficult insect set F we still find 70% conserved successors and 66% order preservation. 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 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 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 [48] we expect that data set Y is also very well-behaved in this respect. It contains 6062 ORFs annotated

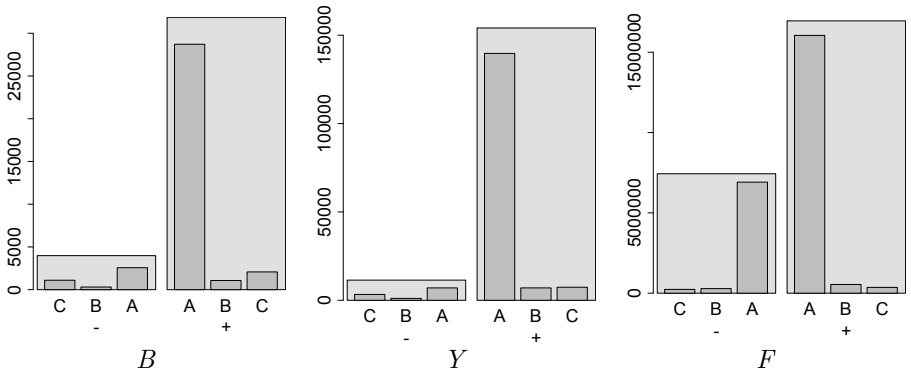

Figure 6 Distribution of block-wise distances of consecutive alignment blocks in the original genomes. Data are shown separated for inverted (-) and preserved (+) orientation of consecutive blocks (light gray). As expected, the number of inverted blocks increases with the difficulty of the input alignment. 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 1-5 intervening blocks (B), and more distantly placed blocks (C).

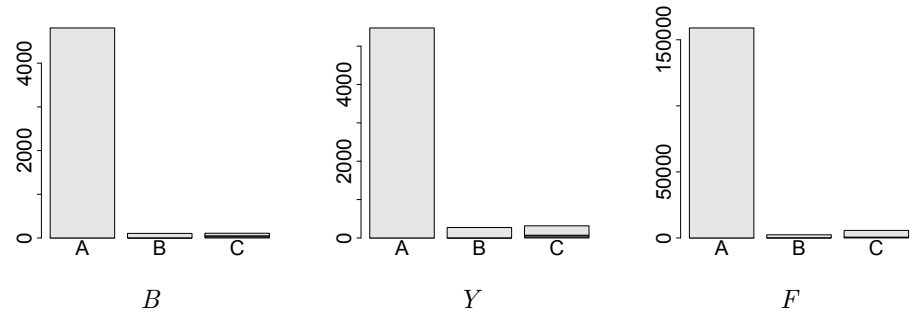

Figure 7 Distribution of block-wise distances between alignment blocks that contain ORFs for the bacteria and yeast data set. For the insect data invidual exons are considered since rearrangements as well as alignment errors within introns are not infrequent. The black bar indicates the number of broken 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 then 100 (C).

for *Saccharomyces cerevisiae*. Of these, 5 474, i.e., 90%, have no intervening blocks and are thus consistently represented in the coordinate system. An additional 272 ORFs, about 5%, have a distance of less then 100 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 appear without intervening blocks or rearrangements. Only 779, about 0.5%, are broken. Overall, thus, the supergenome coordinates are thus every well behaved for all three data sets.

As a specific example we consider the genes of the yeast TCA cycle [49] 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 [50–53], 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 material 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 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 coordinatization 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 the 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 [54, 55]. 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 difficult 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 endeavour, is beyond the scope of this contribution.

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

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Author's 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 workflow, the interpretation of results, and the writing of the manuscript.

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 from the German Research Foundation (DFG) and Universität Leipzig within the program of Open Access Publishing.

Author emails

FG fabian@bioinf.uni-leipzig.de; CHzS choener@bioinf.uni-leipzig.de; LM lydia@bioinf.uni-leipzig.de; PFS studla@bioinf.uni-leipzig.de

Author details

[1]Competence Center for Scalable Data Services and Solutions Dresden/Leipzig, Universität Leipzig, Augustusplatz 12, D-04107 Leipzig, Germany. [2]Bioinformatics Group, Department of Computer Science, Universität Leipzig, Härtelstraße 16–18, D-04107 Leipzig, Germany. [3]Interdisciplinary Center for Bioinformatics, Universität Leipzig, Härtelstraße 16–18, D-04107 Leipzig, Germany. [4]Automatic Language Processing Group, Department of Computer Science, Universität Leipzig, Augustusplatz 12, D-04107 Leipzig, Germany. [5] Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, D-04103 Leipzig, Germany. [6] Fraunhofer Institut for Cell Therapy and Immunology, Perlickstraße 1, D-04103 Leipzig, Germany. [7]Department of Theoretical Chemistry, University of Vienna Währinger Straße 17, A-1090 Vienna, Austria. [8] Center for non-coding RNA in Technology and Health, Grønegårdsvej 3, DK-1870 Frederiksberg C, Denmark. [9] Santa Fe Institute, 1399 Hyde Park Rd., NM87501 Santa Fe, USA.

References

1. Gawad, C., Koh, W., Quake, S.R.: Single-cell genome sequencing: current state of the science. Nature Reviews Genetics **17**(3), 175–188 (2016)
2. 1000 Genomes Project Consortium: A global reference for human genetic variation. Nature **526**(7571), 68–74 (2015)
3. Hezroni, H., Koppstein, D., Schwartz, M.G., Avrutin, A., Bartel, D.P., Ulitsky, I.: Principles of long noncoding RNA evolution derived from direct comparison of transcriptomes in 17 species. Cell Rep **11**, 1110–1122 (2015). doi:10.1016/j.celrep.2015.04.023
4. Lin, S., Lin, Y., Nery, J.R., Urich, M.A., Breschi, A., Davis, C.A., Dobin, A., Zaleski, C., Beer, M.A., Chapman, W.C., Gingeras, T.R., Ecker, J.R., Snyder, M.P.: Comparison of the transcriptional landscapes between human and mouse tissues. Proc Natl Acad Sci USA **111**, 17224–17229 (2014). doi:10.1073/pnas.1413624111
5. Necsulea, A., Kaessmann, H.: Evolutionary dynamics of coding and non-coding transcriptomes. Nat Rev Genet **15**, 734–748 (2014). doi: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 **5**, 09977 (2016). doi:10.7554/eLife.09977
7. Washietl, S., Kellis, M., Garber, M.: Evolutionary dynamics and tissue specificity of human long noncoding RNAs in six mammals. Genome Res. **24**, 616–628 (2014)
8. Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D., Miller, W.: Human–mouse alignments with blastz. Genome research **13**(1), 103–107 (2003)
9. Blanchette, M., Kent, W.J., Riemer, C., Elnitski, L., Smit, A.F., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E.D., *et al.*: Aligning multiple genomic sequences with the threaded blockset aligner. Genome research **14**(4), 708–715 (2004)
10. Paten, B., Herrero, J., Beal, K., Fitzgerald, S., Birney, E.: Enredo and pecan: genome-wide mammalian consistency-based multiple alignment with paralogs. Genome Res **18**, 1814–1828 (2008)
11. Bray, N., Pachter, L.: `MAVID`: Constrained ancestral alignment of multiple sequences. Genome Res **14**, 693–699 (2004). doi:10.1101/gr.1960404
12. Chen, X., Tompa, M.: Comparative assessment of methods for aligning multiple genome sequences. Nature Biotech **28**, 567–572 (2010). doi:10.1038/nbt.1637
13. 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 **6**, 345–352 (2014). doi:10.1002/wsbm.1274
14. Herbig, A., Jäger, G., Battke, F., Nieselt, K.: GenomeRing: alignment visualization based on SuperGenome coordinates. Bioinformatics **28**, 7–15 (2012)
15. Dugar, G., Herbig, A., Förstner, K.U., Heidrich, N., Reinhardt, R., Nieselt, K., Sharma, C.M.: High-resolution transcriptome maps reveal strain-specific regulatory features of multiple *Campylobacter jejuni* isolates. PLoS Genet **9**, 1003495 (2013). doi:10.1371/journal.pgen.1003495
16. Goryunov, D.V., Nagaev, B.E., Nikolaev, M.Y., Alexeevski, A.V., Troitsky, A.V.: Moss phylogeny reconstruction using nucleotide pangenome of complete mitogenome sequences. Biochemistry (Mosc) **80**, 1522–1527 (2015). doi:10.1134/S0006297915110152
17. Medini, D., Donati, C., Tettelin, H., Masignani, V., Rappuoli, R.: The microbial pan-genome. Curr Op Genet Devel **15**, 589–594 (2005). doi:10.1016/j.gde.2005.09.006
18. Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M.: A note on exact algorithms for vertex ordering problems on graphs. Theory Computing Syst **50**, 420–432 (2012)
19. Belda, E., Moya, A., Silva, F.J.: Genome rearrangement distances and gene order phylogeny in γ-proteobacteria. Mol Biol Evol **22**, 1456–1467 (2005). doi:10.1093/molbev/msi134
20. Drillon, G., Fischer, G.: Comparative study on synteny between yeasts and vertebrates. C R Biol **334**, 629–638 (2011). doi:10.1016/j.crvi.2011.05.011
21. Fischer, G., Rocha, E.P.C., Brunet, F., Vergassola, M., Dujon, B.: Highly variable rates of genome rearrangements between hemiascomycetous yeast lineages. PLoS Genet **2**, 32 (2006). doi:10.1371/journal.pgen.0020032

22. Friedberg, R., Darling, A.E., Yancopoulos, S.: Genome rearrangement by the double cut and join operation. Methods Mol Biol **452**, 385–416 (2008)
23. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103. Springer, ??? (1972)
24. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM **55**, 1–19 (2008)
25. Kann, V.: On the approximability of NP-complete optimization problems. PhD thesis, Royal Institute of Technology, Stockholm (1992)
26. Eades, P., Lin, X., Smyth, W.F.: A fast and effective heuristic for the feedback arc set problem. Inf Processing Let **47**, 319–323 (1993)
27. Saab, Y.: A fast and effective algorithm for the feedback arc set problem. J Heuristics **7**, 235–250 (2001). doi:10.1023/A:1011315014322
28. Kahn, A.B.: Topological sorting of large networks. Communications of the ACM **5**(11), 558–562 (1962)
29. Martí, R., Reinelt, G.: The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization vol. 175. Springer, Berlin, Heidelberg (2011)
30. Grötschel, M., Jünger, M., Reinelt, G.: A cutting plane algorithm for the linear ordering problem. Operations Res **32**, 1195–1220 (1984)
31. Oswald, M., Reinelt, G.: The simultaneous consecutive ones problem. Theor. Comp. Sci. **410**, 21–23 (2009)
32. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Systems Sci. **13**, 335–379 (1976)
33. Meidanis, J., Porto, O., Telles, G.P.: On the consecutive ones property. Discr Appl Math **88**, 325–354 (1998)
34. Tucker, A.: A structure theorem for the consecutive 1's property. J Comb Theory B **12**, 153–162 (1972)
35. Christof, T., Oswald, M., Reinelt, G.: Consecutive ones and a betweenness problem in computational biology. In: Bixby, R.E., Boyd, E.A., Ríos-Mercado, R.Z. (eds.) Integer Programming and Combinatorial Optimization, vol. 1412, pp. 213–228 (1998)
36. Reid, J.K., Scott, J.A.: Reducing the total bandwidth of a sparse unsymmetric matrix. SIAM J Matrix Anal Appl **28**, 805–821 (2006)
37. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proc. 24th Nat. Conf. ACM, pp. 157–172. ACM, New York (1969). doi:10.1145/800195.805928
38. Gibbs, N.E., Poole Jr., W.G., Stockmeyer, P.K.: An algorithm for reducing bandwidth and profile reduction algorithms. SIAM J. Numer. Anal. **13**, 236–250 (1976)
39. Feige, U.: Coping with the NP-hardness of the graph bandwidth problem. In: Algorithm Theory – SWAT 2000, vol. 1851, pp. 129–145 (2000)
40. Opatrny, J.: Total ordering problem. SIAM J Computing **8**, 111–114 (1979)
41. Chor, B., Sudan, M.: A geometric approach to betweenness. SIAM J Discr Math **11**, 511–523 (1998)
42. Robinson, W.S.: A method for chronologically ordering archaeological deposits. Amer. Antiquity **16**, 293–301 (1951)
43. Liiv, I.: Seriation and matrix reordering methods: An historical overview. Statistical Analysis & Data Mining **3**, 70–91 (2010)
44. Hahsler, M., Hornik, K., Buchta, C.: Getting things in order: An introduction to the `R` package `seriation`. J Statistical Software **25**, 3 (2008)
45. Paten, B., Earl, D., Nguyen, N., Diekhans, M., Zerbino, D., Haussler, D.: Cactus: Algorithms for genome multiple sequence alignment. Genome research **21**(9), 1512–1528 (2011)
46. Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M., Haussler, D.: The human genome browser at ucsc. Genome research **12**(6), 996–1006 (2002)
47. Chiaromonte, F., Yap, V., Miller, W.: Scoring pairwise genomic sequence alignments. In: Pacific Symposium on Biocomputing, vol. 7, p. 115 (2001)
48. Spingola, M., Grate, L., Haussler, D., Ares Jr., M.: Genome-wide bioinformatic and molecular analysis of introns in *Saccharomyces cerevisiae*. RNA **5**, 221–234 (1999)
49. Krebs, H., Gurin, S., Eggleston, L.: The pathway of oxidation of acetate in baker's yeast. Biochemical Journal **51**(5), 614 (1952)
50. 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: 2017-05-18
51. Haselbeck, R.J., McAlister-Henn, L.: Function and expression of yeast mitochondrial nad-and nadp-specific isocitrate dehydrogenases. Journal of Biological Chemistry **268**(16), 12116–12122 (1993)
52. Oyedotun, K.S., Lemire, B.D.: The carboxyl terminus of the saccharomyces cerevisiaesuccinate dehydrogenase membrane subunit, sdh4p, is necessary for ubiquinone reduction and enzyme stability. Journal of Biological Chemistry **272**(50), 31382–31388 (1997)
53. 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. Journal of Biological Chemistry **278**(38), 36897–36904 (2003)
54. Earl, D., Nguyen, N., Hickey, G., Harris, R.S., Fitzgerald, S., Beal, K., Seledtsov, I., Molodtsov, V., Raney, B.J., Clawson, H., Kim, J., Kemena, C., Chang, J.M., Erb, I., Poliakov, A., Hou, M., Herrero, J., Kent, W.J., Solovyev, V., Darling, A.E., Ma, J., Notredame, C., Brudno, M., Dubchak, I., Haussler, D., Paten, B.: Alignathon: a competitive assessment of whole-genome alignment methods. Genome Res. **24**, 2077–2089 (2014). doi:10.1101/gr.174920.114
55. Ezawa, K.: Characterization of multiple sequence alignment errors using complete-likelihood score and position-shift map. BMC Bioinformatics **17**, 133 (2016). doi:10.1186/s12859-016-0945-5

Additional Files

Additional File 1 — Supplemental text