Supplemental Note

S1 Video

A video showcasing the main views and interactions of MULTIPLA can be accessed here: https://vimeo.com/1062599634.

S2 Spring Simulation Forces

The forces applied on each node by the immediate neighbors are described in Fig. 7. Gravity condenses the view and a repelling force reduces excessive clustering. The sequence resists being stretched and compressed through the genome force. To avoid overlapping, a minimum distance value determines how close two nodes can be. When at minimum distance, a normal force propagates forces between neighboring nodes (Fig. 7). Nodes can only push, not pull, each other – the force transfer from one node only occurs if its total force is directed toward the other node.

The forces, defined in Equations 1-4, were derived through iterative refinement of previous work. Spring simulation force equations:

$$F_{type} = \alpha_{type} sign(\Delta x) log_2(|\Delta x|) \tag{1}$$

$$F_{gravity} = \alpha_{gravity} sign(\Delta x) \sqrt{|\Delta x|} \tag{2}$$

$$F_{repel} = \alpha_{repel} sign(\Delta x) |\Delta x|^{\frac{1}{\beta_{repel}}} \tag{3}$$

$$F_{genome} = \begin{cases} |\Delta x| - |\Delta x_0| < 0, & \alpha_{g1} \frac{-\Delta x}{|\Delta x_0|} \frac{1}{log_2(1+|\Delta x|/\beta_g)} \\ 0 \leq |\Delta x| - |\Delta x_0| < 10 & \alpha_{g2}(\Delta x - \Delta x_0) \\ otherwise & \alpha_{g3} sign(\Delta x) \frac{|\Delta x - \Delta x_0|}{|\Delta x_0|} \end{cases} \tag{4}$$

Δx represents the distance between the two nodes, and Δx_0 represents the original distance between the nodes. The different α and β are constants, where α constants are user-controlled parameters.

S2.1 Performance and Termination

The simulation terminates after a specified amount of time has passed unless the other termination criteria are met first. This ensures that the runtime is never longer than the specified time limit (in our case, 3 seconds). However, termination by time means freezing the simulation even if it is moving. Thus it is not guaranteed that it will be close to a local minima, and results can look unexpected.

Since the selection of nodes to update each iteration is random, the simulation can come to different results with the same data, both due to local minima and to the runtime restriction. This limits the possibility of reproducing a layout. On the other hand, since running is relatively fast, the user can rerun the simulation until they get a layout that is useful for them.

S2.2 Scalability

The simulation selects 20 nodes each iteration and calculates the forces on them. How heavy these calculations are depends on how many nodes are in the same homology group and how many nodes are in (direct or indirect) contact with the selected node. The simulation might be optimized by changing the number of nodes considered each interaction, depending on the cost per node.

The quality of the result also depends on how many nodes there are, since only one node is updated per iteration. As the calculation time per node increases, fewer iterations fit within the time limit. Additionally, the more nodes there are the fewer average updates per node fit within the time limit.

Scalability is a limitation of the spring simulation, limiting how large datasets can be considered. However, our collaborators describe typically reviewing a set of around 10-20 homology groups in a locus, as in the case studies described in the paper. For this, the spring simulation is useful and functions with acceptable performance. It would, of course, be interesting to extend it further, but considering the cognitive load on the user the benefit could be limited.

S2.3 Possible Improvements

Since the spring simulation was implemented as a proof of concept, future fine-tuning to improve the performance remains to be done. Nodes close together could be grouped to get a faster but coarser result. Increasing the cooling rate might make the system converge faster, but could lead to reduced robustness. Similarly, tuning the number of nodes that are selected for evaluation based on the dataset could improve performance. It might also be possible to select the nodes for tuning more intelligently, for example by excluding nodes that were recently found to be under low stress.

S3 Implementation Details

MULTIPLA is implemented using a combination of web technologies to ensure responsiveness and interactivity. The frontend uses Vue and Pinia for state management, and there is a Python backend for interactive calculations. The Overviews make use of Pixi and Pixi Viewport for scalable graphics and interactions. D3 [BOH11] is employed for data loading, scale transformations, lasso selections, and interactive features in the Focus View. A demo instance of MULTIPLA on the Capsicum data is available here: http://vapp1.win.tue.nl/multipla.

S4 Zoom Implementation

The creation and management of the sequence scaling, compression, and viewing is described in Figure S1. The spring simulation changes the positions of the elements, giving them new positions in the same coordinate system. One gene-to-simulation transform is created per sequence, and one simulation-to-window transform is created for the whole view, linearly scaling the simulation coordinates to window coordinates. The range that is displayed in the viewport is defined in the simulation position space. Chaining of the gene-to-simulation transform and the simulation-to-window transform results in a gene-to-window transform, which is used to calculate where each element should be rendered. Zooming and navigation in the viewport result in updates of the window range (see orange box, Fig. S1), causing an update of the simulation-to-window transform and the gene-to-window transform.

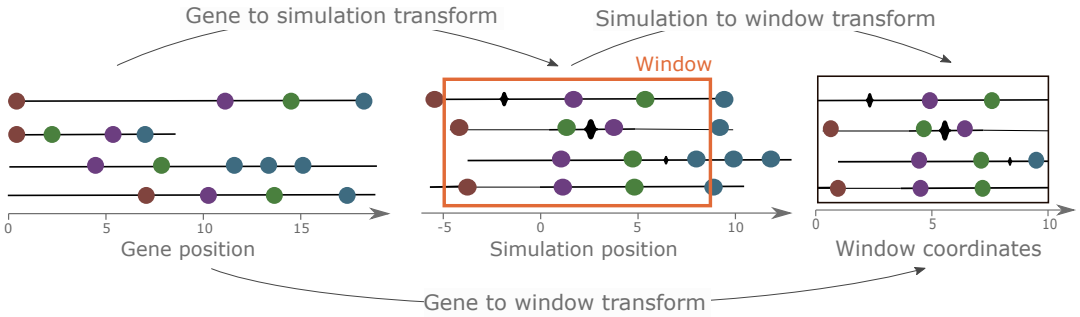

Figure S1: *Architecture of the zooming mechanism, from gene position (left) to window coordinates (right). For each sequence, we define a transform from the gene space to the simulated coordinate space (middle). In the simulated position space, we define a viewport range (orange) and then define a transform from the simulation positions to window coordinates. Based on these transitions, we define a gene-to-window transition from the original gene position to the window coordinates for each sequence.*

S5 Distance Metrics

S5.1 Sources

In this section, we report which references inspired the use of each distance metric and address whether they resulted from collaborator input during discussions.

Protein distance and Order Most existing tools for exploring loci, including those used for biosynthetic gene clusters (BGCs), use protein similarity and gene order, as they are familiar and serve as primary determinants of locus organization and similarity. (References: [ARJ*15, CF18, ZES19, GC21, BSK*21])

Orientation Synteny tools consider gene orientation as a factor in conservation because a change in orientation could point to an inversion of a segment of DNA. Logically, our collaborators also expressed interest in being able to explore the influence of this feature within the context of gene organization. However, the *change* in orientation within one sequence needs to be observed, otherwise, opposite gene orientation (and order) could be due to opposite sequence orientation (which is arbitrary). If sequence orientation is inconsistent across genomes, it complicates comparison. Therefore, flipping an entire sequence and its features interactively might be helpful. (References: [MMP09, ARJ*15, CF18])

Size Collaborators mentioned that truncated or inflated gene sizes are relevant, as they may indicate gene fusion, partial gene movement, or potential annotation errors. (References: [MMP09, ARJ*15, ZES19])

Location While relative gene location is of interest, the absolute gene location on the sequence is considered secondary because it is often highly variable due to differences in genome quality rather than biological factors. (References: [MMP09, ARJ*15, ZES19])

PAV and CNV Although presence-absence variation (PAV) and copy number variation (CNV) are not explicitly emphasized in most tools, they are crucial for comparing loci and analyzing large structural variations. (References: [ARJ*15, ZES19])

S5.2 Specifications

The distance metrics not specified in the main text are presented in this section.

PAV The PAV-based distance D_v is obtained by using the Jaccard dissimilarity J of the sets of genes $g(A)$ and $g(B)$ present in in A and B:

$$d_v(A,B) = J(g(A),g(B)) = 1 - \frac{|g(A) \cap g(B)|}{|g(A) \cup g(B)|}.$$

CNV The genes' CNV based distance D_c is obtained by taking the Jaccard dissimilarity J of the multisets (sets allowing duplicates) of genes $g(A)$ and $g(B)$ as they appear in A and B, respectively.

$$d_c(A,B) = J(g(A),g(B)) = 1 - \frac{|g(a) \cap g(b)|}{|g(a) \cup g(b)|}.$$

The distance $D_*(A,B)$ with $*$ one of $\{p,s,l\}$ is given as the sum of distances d_* of the mRNA sets corresponding to the gene sets $g(A)$ and $g(B)$ restricted to the different homology groups h:

$$D_*(A,B) = \frac{1}{|H|} \sum_{h \in H} d_*(m(g(A)) \cap m(h), m(g(B)) \cap m(h)),$$

where $m(x)$ is the set of mRNAs corresponding to the genes in set x and H is the set of all homology groups. We define d_* for non-empty sets a and b of mRNAs. For other subsets holds: $d_*(a,b) = 0$ if $a = b = \emptyset$ and $d_*(a,b) = 1$ if one of a and b is $\emptyset$.

Protein The protein distance d_p is based on the protein sequence similarity scores s between the mRNAs $m_a \in a$ and $m_b \in b$ as calculated in PanTools [SAdRSS18]:

$$d_p(a,b) = \sum_{m_a \in a} \sum_{m_b \in b} \frac{1 - s(m_a,m_b)}{|a| \cdot |b|}.$$

Size Distance d_s uses the average size $avg(x)$ of mRNAs in a set x and the ratio of the average sizes of mRNAs :

$$d_s(a,b) = 1 - \frac{min(avg(a), avg(b))}{max(avg(a), avg(b))}.$$

Location We define the location distance d_l as the absolute difference of the relative location of mRNAs x with respect to the total length of the sequence.

$$\mathrm{rl}(X) = \frac{\mathrm{loc}(X)}{|X|}$$

$$d_l(a,b) = |\mathrm{rl}(a) - \mathrm{rl}(b)|$$

References

[ARJ*15] AURISANO J., REDA K., JOHNSON A., MARAI E. G., LEIGH J.: BactoGeNIE: A large-scale comparative genome visualization for big displays. *BMC Bioinformatics 16*, 11 (2015), S6. doi:10.1186/1471-2105-16-S11-S6. 2

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2301–2309. doi:10.1109/TVCG.2011.185. 1

[BSK*21] BLIN K., SHAW S., KLOOSTERMAN A. M., CHARLOP-POWERS Z., VAN WEZEL G. P., MEDEMA M. H., WEBER T.: AntiSMASH 6.0: Improving cluster detection and comparison capabilities. *Nucleic Acids Research 49*, W1 (2021), W29–W35. doi:10.1093/nar/gkab335. 2

[CF18] CLEARY A., FARMER A.: Genome context viewer: visual exploration of multiple annotated genomes using microsynteny. *Bioinformatics 34*, 9 (2018), 1562–1564. doi:10.1093/bioinformatics/btx757. 2

[GC21] GILCHRIST C. L., CHOOI Y. H.: Clinker & clustermap.js: Automatic generation of gene cluster comparison figures. *Bioinformatics 37*, 16 (2021), 2473–2475. doi:10.1093/bioinformatics/btab007. 2

[MMP09] MEYER M., MUNZNER T., PFISTER H.: MizBee: A multiscale synteny browsers. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 897–904. doi:10.1109/TVCG.2009.167. 2

[SAdRSS18] SHEIKHIZADEH ANARI S., DE RIDDER D., SCHRANZ M. E., SMIT S.: Efficient inference of homologs in large eukaryotic pan-proteomes. *BMC bioinformatics 19*, 1 (2018), 340. doi:10.1186/s12859-018-2362-4. 2

[ZES19] ZHAO T., ERIC SCHRANZ M.: Network-based microsynteny analysis identifies major differences and genomic outliers in mammalian and angiosperm genomes. *Proceedings of the National Academy of Sciences of the United States of America 116*, 6 (2019), 2165–2174. doi:10.1073/pnas.1801757116. 2