NAR Genomics and Bioinformatics, 2023, **5**, 1–6
https://doi.org/10.1093/nargab/lqad096
Advance access publication date: 2 November 2023
Application Notes

SDRAP for annotating scrambled or rearranged genomes

Jasper Braun [1,4], Rafik Neme [2,3], Yi Feng [2], Laura F. Landweber [2] and Nataša Jonoska [1,*]

[1]Department of Mathematics and Statistics, University of South Florida, Tampa, FL 33620, USA
[2]Departments of Biochemistry and Molecular Biophysics, and Biological Sciences, Columbia University, New York, NY 10032, USA
[3]Department of Chemistry and Biology, Universidad del Norte, Barranquilla, Colombia
[4]Division of Clinical Pathology, Department of Pathology, Beth Israel Deaconess Medical Center, Boston, MA 02215, USA

[*]To whom correspondence should be addressed. Tel: +1 813 974 9566; Email: jonoska@mail.usf.edu

Abstract

Genomes sometimes undergo large-scale rearrangements. Programmed genome rearrangements in ciliates offer an extreme example, making them a compelling model system to study DNA rearrangements. Currently, available methods for genome annotation are not adequate for highly scrambled genomes. We present a theoretical framework and software implementation for the systematic extraction and analysis of DNA rearrangement annotations from pairs of genome assemblies corresponding to precursor and product versions. The software makes no assumptions about the structure of the rearrangements, and permits the user to select parameters to suit the data. Compared to previous approaches, this work achieves more complete precursor-product mappings, allows for full transparency and reproducibility, and can be adapted to genomic data from different sources.

Introduction

DNA rearrangements are important in many contexts, such as vertebrate immunity (1), and cancer genome instability (2). In some eukaryotes, such as the ciliate *Oxytricha trifallax*, programmed genome rearrangements permit massive structural modifications from a precursor genome (in the germline micronucleus) into a product genome (in the somatic macronucleus) (3,4). Consequently, these two genomes, cohabiting the same cell, have reduced similarity to each other, permitting comparative genomics within a single cell. In *Oxytricha*, genome rearrangement eliminates over 90% of the precursor DNA (3), and reorganizes the remaining DNA fragments into ∼18 000 different gene-sized nanochromosomes as the product (5). The order or orientation of DNA segments in the precursor often differs from the product (6). The presence of such complex DNA rearrangements during macronuclear development makes *O. trifallax* and other species of ciliates powerful model systems to study DNA rearrangement processes that appear in a wider range of organisms.

While our approach is tailored to the annotation of rearranged genomes, other models, such as (7–13) offer additional insights. Previous approaches to annotating scrambled genomes have had limitations with segmental duplications in scrambled genomes (14), or annotations of eliminated sequence (15) and were more limited, using custom scripts, with many complex cases skipped, and some annotation parameters chosen arbitrarily without a precise definition of the parameters associated with the output (3,16,17); these approaches limit annotation reproducibilty and cross-species comparisons, and lack transparency and comprehensiveness. Here, we present a protocol that readily describes and outputs relationships between a pair of precursor and product genomes. The algorithm also annotates chromosome ends/telomere-addition sites, specifies the types of rearrangements between precursor/product loci, and detects inversions or translocations (scrambled maps). We present an implementation of our algorithm, *Scrambled DNA Rearrangement Annotation Protocol (SDRAP)*, that allows a range of parameters to accommodate and process diverse genomes with similar properties to each other. Furthermore, we specify definitions related to retained and eliminated DNA sequences, scrambling, paralogy and completeness of loci, together improving existing models of genome rearrangements. Overall, this annotation tool allows consistent, automated, adaptable and reproducible analysis of scrambled genome pairs, including complex rearrangements that occur during somatic differentiation.

Materials and methods

Preliminaries

Annotation of DNA rearrangements

DNA rearrangements considered here involve reordering and/or inversion of segments from a precursor sequence to form a product sequence. In the context of ciliate biology, precursor and product sequences represent portions of the micronuclear and macronuclear chromosomes, respectively.

Matches and arrangements

Here, a pair of corresponding precursor and product segments is represented by a triple $M = ([a, b], [c, d], \sigma)$, called a **match**, where $[a, b]$ is an integer interval called **precursor interval** (denoted Prec(M)) that describes the location of the segment in the precursor, and $[c, d]$ is an integer interval called **product interval** (denoted Prod(M)) that describes the location of the segment in the product, while $\sigma \in \{0, 1\}$ is the **orientation** of the match, denoted $\sigma(M)$ (note that the term 'match'

Received: June 10, 2023. **Revised:** September 7, 2023. **Editorial Decision:** September 20, 2023. **Accepted:** October 3, 2023

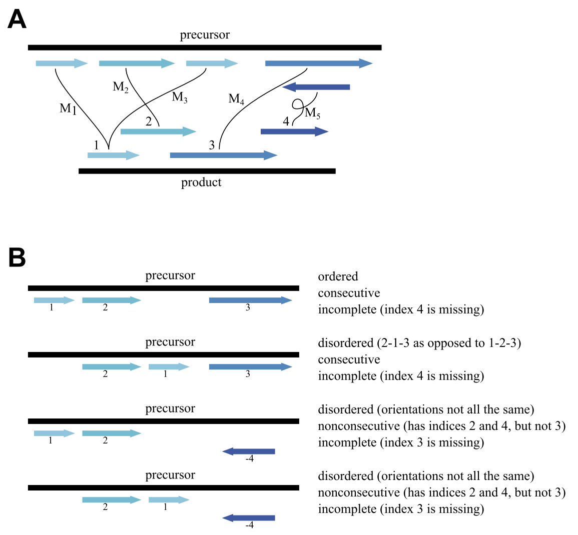
A
precursor
M1
M2
M3
M4
M5
1
2
3
4
product
B
precursor
ordered
consecutive
incomplete (index 4 is missing)
precursor
disordered (2-1-3 as opposed to 1-2-3)
consecutive
incomplete (index 4 is missing)
precursor
disordered (orientations not all the same)
nonconsecutive (has indices 2 and 4, but not 3)
incomplete (index 3 is missing)
precursor
disordered (orientations not all the same)
nonconsecutive (has indices 2 and 4, but not 3)
incomplete (index 3 is missing)
-4

Figure 1. (**A**) An arrangement of matches $\{M_1, \ldots, M_5\}$ with their indices labeling the product intervals. Matches M_1 and M_3 share a product interval and are given the same index 1. A precursor and a product interval connected by a black line form a match. An inverted match is indicated by the loop in the black line and reverse oriented arrows. The black lines are labelled by the indices of the corresponding matches. (**B**) Unambiguous subarrangements of the arrangment in (A). Only the precursor intervals are shown from the top figure. The index is negated for the reverse oriented match. For each of the unambiguous subarrangements, it is indicated whether or not the arrangement is ordered, consecutive, or complete. There is no complete unambiguous subarrangement because $\mathrm{Prec}(M_5) \subseteq \mathrm{Prec}(M_4)$.

here does not refer to a pair of matching residues in a sequence alignment as commonly expressed in the literature). We follow the convention that $\sigma = 1$ indicates that the two segments are found in the same orientation, and $\sigma = 0$ indicates that the two segments are found in opposite orientations in the respective precursor and product sequences. Local sequence alignments can be viewed as matches in the obvious way, and the language introduced here for matches is often also used for alignments.

To compare order and orientation of precursor intervals with corresponding product intervals, we define an ordering on a set of intervals by:

$$[c_1, d_1] \leq [c_2, d_2] \text{ if and only if } c_1 \leq c_2 \text{ and } d_1 \leq d_2. \quad (1)$$

However, this ordering is a total ordering only on sets in which no interval is contained in another. Thus, we are interested in sets of matches that satisfy:

$$\mathrm{Prod}(M_1) \subseteq \mathrm{Prod}(M_2) \Rightarrow \mathrm{Prod}(M_1) = \mathrm{Prod}(M_2), \quad (2)$$

for all matches M_1, and M_2 in a given set. In this way distinct product intervals of the matches in the set are never contained in one another. A set of matches that satisfies (2) is called an **arrangement**. We observe that a subset of an arrangement is also an arrangement.

For a match M in an arrangement $\mathcal{M}$, we define the **index of** M **in** $\mathcal{M}$, denoted $i_{\mathcal{M}}(M)$, to be the position of $\mathrm{Prod}(M)$ in the order of $\mathcal{M}$.

Properties of arrangements

The fact that precursor intervals of matches in an arrangement may be contained in those of other matches of the arrangement (for example M_4 and M_5 in Figure 1), and the potential lack of a 1–1 correspondence complicates a comparison of order and orientations between precursor and product intervals. The general strategy taken in the algorithm below is to consider all maximal subsets of an arrangement that do not have these issues.

Given a positive integer p, two matches M_1, M_2 are considered ***p*-overlapping** if:

$$\left|\mathrm{Prec}(M_1) \cap \mathrm{Prec}(M_2)\right| \geq \min\left\{p, |\mathrm{Prec}(M_1)|, |\mathrm{Prec}(M_2)|\right\}. \quad (3)$$

A subset $\mathcal{M}'$ of an arrangement $\mathcal{M}$ is **unambiguous** if:

(i) $i_{\mathcal{M}}(M_1) \neq i_{\mathcal{M}}(M_2)$, for all $M_1, M_2 \in \mathcal{M}'$, and
(ii) No two matches in $\mathcal{M}'$ are p-overlapping, and
(iii) $\mathcal{M}'$ is a maximal set of matches with the first two properties.

We consider an unambiguous set of matches **ordered** if either they all have orientation $\sigma = 1$ and appear in the same order on the precursor as on the product, or if they all have orientation $\sigma = 0$ and they appear on the precursor in reverse order compared to the product (see Figure 1(b)).

An unambiguous subset $\mathcal{M}'$ of an arrangement $\mathcal{M}$ is called **consecutive** if the set of indices $\mathcal{I}_{\mathcal{M}}(\mathcal{M}') = \{i_{\mathcal{M}}(M) : M \in \mathcal{M}'\}$ forms a consecutive set of integers, and it is called **complete** with respect to $\mathcal{M}$ if the set of indices $\mathcal{I}_{\mathcal{M}}(\mathcal{M}')$ is the same as the set of indices $\mathcal{I}_{\mathcal{M}}(\mathcal{M}) = \{i_{\mathcal{M}}(M) : M \in \mathcal{M}\}$ of matches in $\mathcal{M}$. An unambiguous arrangement $\mathcal{M}$ is not scrambled if it is ordered, consecutive and complete. Given a set of properties $\mathcal{S} \subseteq \{\text{ordered, consecutive, complete}\}$, an unambiguous subset of an arrangement is called $\mathcal{S}$-scrambled if one or more of the properties in $\mathcal{S}$ is violated.

To assess whether an arrangement $\mathcal{M}$ is scrambled, we consider all possible unambiguous subsets of $\mathcal{M}$. Whenever at least one of the unambiguous subsets of $\mathcal{M}$ is $\mathcal{S}$-scrambled, the parent set $\mathcal{M}$ is called **weakly $\mathcal{S}$-scrambled**. If all non-singleton unambiguous subsets of $\mathcal{M}$ are $\mathcal{S}$-scrambled, the parent set $\mathcal{M}$ is called **strongly $\mathcal{S}$-scrambled**. Weak and strong versions of the properties ordered, consecutive, and complete are defined analogously. Here, the terms weak and strong refer to the strength of the available evidence that designates an arrangement as scrambled, such that the user can decide the level of stringency whenever the feature 'scrambled' becomes biological or statistically relevant. Figure 1 shows an example of an arrangement together with its unambiguous subarrangements. With $\mathcal{S} = \{\text{ordered, consecutive}\}$, the depicted arrangement is weakly, but not strongly $\mathcal{S}$-scrambled. However this set is strongly scrambled because there is no complete unambiguous subarrangement.

Annotation protocol

Since efficient and well-established sequence alignment tools (like BLAST (18,19)) for the identification of the rearranging precursor and product segments already exist, our protocol starts with a set $\mathcal{H}$ of local sequence alignments. The algorithm then extracts arrangements $\mathcal{M}$ with information whether each $\mathcal{M}$ is weakly and strongly $\mathcal{S}$-scrambled. Informally, the procedure can be broken down into 3 steps:

(1) Extract 'preliminary matches' from $\mathcal{H}$ for which condition (2) is enforced (the set $\mathcal{M}_{\mathrm{pre}}$).
(2) Add additional, potentially paralogous matches from the remaining alignments that may violate condition (2).

(3) Determine whether or not the set of matches obtained in the first two steps is weakly and/or strongly $\mathcal{S}$-scrambled.

The three steps are summarized in the following paragraphs and more detail is provided in the supplementary section 'Methods' and Supplementary Figures S1–S4.

Merging alignments

While high sequence similarity between precursor and product segments can be expected in the context of DNA rearrangements in ciliates, some insertions, deletions and substitutions can be attributed to allelic variation, sequencing errors, and ambiguities during read assembly. Thus, an arrangement $\mathcal{M}$ consists of matches that represent members of $\mathcal{H}$ directly, or are obtained by merging members of $\mathcal{H}$. A custom method for merging alignments is used to allow finer control over which alignments are merged, as opposed to taking advantage of generic gapped alignment tools (supplementary section 'Merging alignments' and Supplementary Figure S1). This fine control is needed in the annotation of ciliate DNA-rearrangements since a run of non-scrambled rearranging segments can sometimes be confused with a single longer gapped alignment. One strategy taken in the past has been to instruct BLAST to search for ungapped alignments and then to further process these alignments (3,16,20). We adopt this general strategy in our algorithm and provide complete user control for choosing parameters that combine ungapped alignments into longer gapped alignments. Informally, the algorithm merges two ungapped alignments when they have the same orientation, the relative positions of their precursor intervals is 'similar enough' to the relative positions of their product intervals, and the gap that must be introduced to connect the two alignments is not 'too large'. A more precise specification of mergeability of two matches is given in the supplementary section 'Merging alignments'.

Step 1: preliminary matches

In order to ensure property 2 holds and to reduce the redundancies in the matches, we enforce non-containment of product intervals in a preliminary arrangement annotation step. This is done by iterating over the ungapped alignments in $\mathcal{H}$ and adding each to a growing set of preliminary matches $\mathcal{M}_{\text{pre}}$ whenever its product interval spans a 'sufficiently large' number of residues that are not already in one of the product intervals of members of $\mathcal{M}_{\text{pre}}$. This number of additional product residues that a member H of $\mathcal{H}$ must cover to be added to $\mathcal{M}_{\text{pre}}$ is user-defined and may be greater than 1 for lenience to account for imperfections in the data. When an alignment is added to $\mathcal{M}_{\text{pre}}$ during an iteration of the algorithm, it may be added in one of two ways: it is either merged with a member of $\mathcal{M}_{\text{pre}}$ if appropriate, or otherwise added as its own independent match. The ungapped alignments in $\mathcal{H}$ are iterated over in lexicographic order by bitscore in descending order to give preference to higher quality alignments. The final step of this part of the protocol is assignment of indices to each match in $\mathcal{M}_{\text{pre}}$. A more detailed description of the preliminary arrangement annotation step is given in supplementary section 'Preliminary matches' and Supplementary Figure S2 depicts a flowchart of the algorithm.

Step 2: additional matches

Similar to the algorithm described in the first step, a set of additional matches $\mathcal{M}_{\text{add}}$ is built while iterating over the alignments $\mathcal{H}$ which were not added to $\mathcal{M}_{\text{pre}}$. Each of the alignments H is merged with a member of $\mathcal{M}_{\text{add}}$ whenever possible. If not, the algorithm checks if its product interval overlaps 'sufficiently' (see supplement for details) with the product interval of any of the members of $\mathcal{M}_{\text{pre}}$ to add it to $\mathcal{M}_{\text{add}}$, however assigning to it the index of that preliminary match. Each additional match inherits the index of some preliminary match and can be viewed as a repeated or paralogous region in the precursor that matches the same region in the product. Below we somewhat abuse the notion of arrangement and we call the set $\mathcal{M} = \mathcal{M}_{\text{pre}} \cup \mathcal{M}_{\text{add}}$ an arrangement although the condition (2) may be violated with addition of $\mathcal{M}_{\text{add}}$. A more detailed description of the algorithm annotating additional matches is given in supplementary section 'Additional matches' and Supplementary Figure S3 depicts a flowchart of the algorithm.

Step 3: determine scrambling

In order to determine whether the arrangement $\mathcal{M} = \mathcal{M}_{\text{pre}} \cup \mathcal{M}_{\text{add}}$ is weakly or strongly $\mathcal{S}$-scrambled, the algorithm must test whether the unambiguous subsets of $\mathcal{M}$ are $\mathcal{S}$-scrambled. This is done by associating a graph to $\mathcal{M}$, by letting matches be the vertices and connecting two matches with an edge whenever they do not share the same index and are not p-overlapping. Then the maximal cliques in the graph correspond precisely to the maximal unambiguous subarrangements of $\mathcal{M}$. While a set of matches theoretically can have an exponential number of unambiguous subsets with respect to the number of matches, in practice this number is much lower in most cases and a user-specified cutoff is implemented in SDRAP (supplementary section 'Algorithm complexity' and Supplementary Figure S4). For that reason, we use the output-sensitive algorithm introduced by (21) to list maximal cliques. Once an unambiguous subarrangement is detected, the procedure determines whether that subarrangement is weakly or strongly $\mathcal{S}$-scrambled by checking the properties of the corresponding index sets.

Implementation

The algorithm described in Annotation Protocol was implemented as a web application called **Scrambled DNA Rearrangement Annotation Protocol**, or **SDRAP**, using PHP 5.3.3 and MySQL 5.6.31 with Apache 2.2.15 on a linux server running with the CentOS 6.7 operating system. The user interface was implemented using HTML, CSS and javascript and can be accessed at knot.math.usf.edu/SDRAP. The code and a complete documentation for SDRAP are available on zenodo at https://doi.org/10.5281/zenodo.8305849 or on GitHub at github.com/JasperBraun/SDRAP. SDRAP accepts additional input parameters b_{min} and q_{min} which set minimum threshold on bitscore and percent identity for high-scoring pairs to be included in the set $\mathcal{H}$ from which matches are extracted as described in Annotation Protocol. To limit extraction of potentially exponential numbers of unambiguous subsets for each of the arrangements, SDRAP accepts an input parameter k_{max} which determines the maximum number of unambiguous subsets extracted from each arrangement. A more detailed description of the implementation is given in supplementary section 'Implementation'.

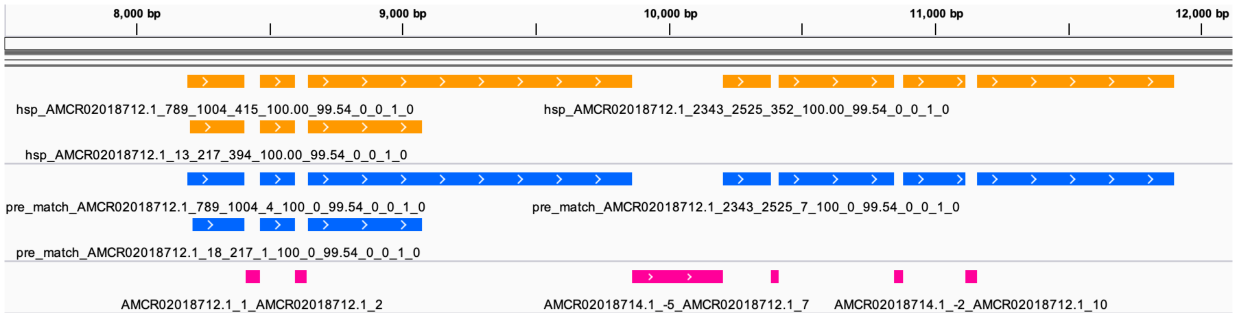
8,000 bp
9,000 bp
10,000 bp
11,000 bp
12,000 bp
hsp_AMCR02018712.1_789_1004_415_100.00_99.54_0_0_1_0
hsp_AMCR02018712.1_2343_2525_352_100.00_99.54_0_0_1_0
hsp_AMCR02018712.1_13_217_394_100.00_99.54_0_0_1_0
pre_match_AMCR02018712.1_789_1004_4_100_0_99.54_0_0_1_0
pre_match_AMCR02018712.1_2343_2525_7_100_0_99.54_0_0_1_0
pre_match_AMCR02018712.1_18_217_1_100_0_99.54_0_0_1_0
AMCR02018712.1_1_AMCR02018712.1_2
AMCR02018714.1_-5_AMCR02018712.1_7
AMCR02018714.1_-2_AMCR02018712.1_10

Figure 2. BLAST high-scoring pairs (yellow), matches (blue) (preliminary and additional) and eliminated sequences (pink) on micronuclear sequence ctg7180000068813 from (3). All alignments and precursor intervals match macronuclear sequence AMCR02018712.1 from (5). For an explanation of the labelling of the features, see the software documentation at https://doi.org/10.5281/zenodo.8305849 or https://github.com/JasperBraun/SDRAP. The picture was obtained using IGV version 2.4.10. Each label refers to a single segment and IGV hides some labels to avoid overcrowding of the figure.

Table 1. Total counts of arrangements of 2-telomeric product sequences covered at least 90% by the matches in their respective arrangements as well as counts of the subsets of those arrangements that contain ambiguities, along with counts of product sequences only found in these various sets of arrangements

	Mac genome from (17)		Mac genome from (5)	
	Strict	Lenient	Strict	Lenient
# arrangements*	10 071	15 683	23 718	33 705
# arrangements with p-overlapping matches*	90	399	653	1182
# arrangements with matches sharing the same index*	181	628	409	1062
# arrangements with ambiguities*, **	267	940	1051	2149
# product sequences*	9166	12 701	21 580	27 204
# product sequences only in arrangements* with p-overlapping matches	82	248	598	891
# product sequences only in arrangements with matches sharing the same index*	136	386	294	658
# product sequences only in arrangements with ambiguities*, **	215	607	884	1500

* Only arrangements and product sequences from arrangements with coverage 90% and where the product sequence has two telomeres are considered.
** Ambiguities are defined as p-overlapping matches and matches sharing the same index where $p = 5$ which is the default value.
Arrangements were obtained in the two SDRAP test runs applying parameter values described in supplementary Table S2 to the micronuclear genome assembly from (3) and the macronuclear genome assemblies from from (17) and (5).

Algorithm complexity

Since SDRAP is applied to potentially large datasets, the procedure's computational complexity must be considered. The preliminary arrangement annotation step has complexity $O(n_{\text{pre}}h)$ time, where $n_{\text{pre}} = |\mathcal{M}_{\text{pre}}|$ and $h = |\mathcal{H}|$. After the set $\mathcal{M}_{\text{pre}}$ is determined, assigning indexes $i_{\mathcal{M}_{\text{pre}}}(M)$ for each $M \in \mathcal{M}_{\text{pre}}$ requires sorting and subsequent iteration over $\mathcal{M}_{\text{pre}}$, totalling $O(n_{\text{pre}} \log n_{\text{pre}})$ operations. The additional arrangement annotation step has complexity $O(h(n_{\text{pre}} + n_{\text{add}}))$, where $n = n_{\text{pre}} + |\mathcal{M}_{\text{add}}|$. The algorithm for determining scrambling has complexity $O(n^4)$. Finally, the protocol is applied to all of N precursor-product sequence pairs which have high-scoring pairs between them. The complexity calculations are broken down in more detail in supplementary section 'Algorihtm complexity'.

Results and discussion

Test runs and default parameter values

To compare SDRAP to previous annotation procedures (3,16) and establish reasonable default parameter values, 52 test runs were conducted using precursor and product genomes of the organism *Oxytricha trifallax*, obtained from (3) and (17), respectively. The 2013 assembly of the macronuclear genome was chosen so that the results can be compared with previous annotations. A detailed description of the test runs is given in supplementary section 'Test runs and default parameter values', supplementary Tables S1 and S2. Supplementary Tables S3–S7 provide summary statistics for the test runs and supplementary Tables S8 and S9 provide a detailed comparison to previous annotations (3,16). SDRAP was run two more times with the parameter value sets described in supplementary Table S2 using the most recent macronuclear genome assembly published in (5). An example of an arrangement that has p-overlapping matches is given in Figure 2. The figure visualizes a region of the micronuclear ctg7180000068813 from (3) containing the BLAST high-scoring pairs and precursor intervals matching macronuclear contig AMCR02018712.1 from (5). Such complete annotation of this arrangement was not obtained with any other previous annotation. Previous annotations would have filtered out ambiguously annotated segments.

Increase in data coverage in the annotation of precursor and product genome of *O. trifallax*

One of the two parameter value sets used in the two test runs applied to the most recent macronuclear genome assembly (5) used a more lenient choice of values for b_{min} and q_{min}, whereas the other used stricter values. The arrangements of complete (2-telomere-containing) product sequences covered >90% (where coverage is defined as the coverage of the portion of the product sequence between the telomeres) were counted along with the subsets of those arrangements which had p-overlapping matches, or matches sharing the same index. These ambiguous rearrangements, or their matches, were filtered out in previous procedures. The counts are summarized in Table 1. For the more lenient or stricter parameter choices, $b_{\min}$ and $q_{\min}$ 4.4% and 6.4%, respectively, all arrangements of 2-telomere product sequences with at least 90% coverage contained ambiguities in the form of p-overlapping matches or matches sharing the same index. Out of all 2-telomeric product sequences that achieved 90% or more coverage in at least one arrangement, 4.1% and 5.5%, respectively, can only be found in arrangements that contain ambiguities. Thus, a more complete annotation of up 6.4% of the data was achieved.

Applying SDRAP to other datasets

SDRAP was successfully used to annotate the precursor and product genomes of the ciliates *Tetmemena* sp. and *Euplotes woodruffi* to investigate the origin and evolution of gene scrambling in ciliates (22).

SDRAP was also successfully run on an assembly of PacBio reads from the SK-BR-3 cancer cell line (23) with the PacBio assembly as precursor and the ERBB2 gene region on chromosome 17 in the human reference (GRCh38) as the product. The results confirm extensive duplications and translocations of this locus in the SK-BR-3 cell line (23), and illustrate the utility of SDRAP for a broad range of data. The output files can be found at https://doi.org/10.5281/zenodo.8299368 or knot.math.usf.edu/SDRAP/annotations/sdrap_skbr3_falcon/.

Data availability

All datasets used to showcase SDRAP were previously published and are cited in the text. The precursor and product genomes of all runs of SDRAP to annotate rearrangements of O. trifallax were obtained from the NCBI Assembly database under assembly names Oxytricha_MIC_v2.0, oxytricha_asm_v1.1, and oxytricha_jrb310_mac_pacbio. Oxytricha_MIC_v2.0 is the precursor assembly used for all runs of SDRAP on the genome of O. trifallax. Oxytricha_asm_v1.1 is the product assembly for the test and comparison runs. Oxytricha_jrb310_mac_pacbio is the product genome used to generate Figure 2, and both oxytricha_asm_v1.1 and oxytricha_jrb310_mac_pacbio were used to generate the data in Table 1. The annotations generated during the current study are available at https://doi.org/10.5281/zenodo.8299368 or knot.math.usf.edu/SDRAP/annotations/ (see SDRAP_TESTS_README.tsv). For the precursor of the SK-BR-3 cancer cell line annotation, the Falcon assembly of PacBio reads was downloaded from schatz-lab.org/publications/SKBR3/, and for the product, the ERBB2 locus on chromosome 17 in the human reference Ch38 was obtained from the NCBI Genome Data Viewer. The code and a complete documentation for SDRAP are available on Zenodo at https://doi.org/10.5281/zenodo.8305849 or on GitHub at github.com/JasperBraun/SDRAP.

Supplementary data

Supplementary Data are available at NARGAB Online.

Acknowledgements

Portions of this paper reflect results obtained in the PhD dissertation of author Jasper Braun (24).

Funding

NSF [DMS-1800443/1764366 to N.J., L.L.]; N.J. acknowledges partial support from the Simons Foundation and the Southeast Center for Mathematics and Biology.

Conflict of interest statement

None declared.

References

1. Roth,D.B. (2014) V(D)J recombination: mechanism, errors, and fidelity. *Microbiol. Spectr.*, **2**, https://doi.org/10.1128/microbiolspec.MDNA3-0041-2014.
2. Cortés-Ciriano,I., Lee,J.J.-K., Xi,R., Jain,D., Jung,Y.L., Yang,L., Gordenin,D., Klimczak,L.J., Zhang,C.-Z., Pellman,D.S., *et al.* (2020) Comprehensive analysis of chromothripsis in 2,658 human cancers using whole-genome sequencing. *Nat. Genet*, **52**, 331–341.
3. Chen,X., Bracht,J.R., Goldman,A.D., Dolzhenko,E., Clay,D.M., Swart,E.C., Perlman,D.H., Doak,T.G., Stuart,A., Amemiya,C.T., *et al.* (2014) The architecture of a scrambled genome reveals massive levels of genomic rearrangement during development. *Cell*, **158**, 1187–1198.
4. Yerlici,V.T. and Landweber,L.F. (2014) Programmed genome rearrangements in the ciliate *Oxytricha*. *Microbiol. Spectr.*, **2**, https://doi.org/10.1128/microbiolspec.MDNA3-0025-2014.
5. Lindblad,K.A., Pathmanathan,J.S., Moreira,S., Bracht,J.R., Sebra,R.P., Hutton,E.R. and Landweber,L.F. (2019) Capture of complete ciliate chromosomes in single sequencing reads reveals widespread chromosome isoforms. *BMC Genomics*, **20**, 1037.
6. Prescott,D.M. (2000) Genome gymnastics: unique modes of DNA evolution and processing in ciliates. *Nat. Rev. Genet.*, **1**, 191–198.
7. Altenhoff,A.M., Train,C.-M., Gilbert,K.J., Mediratta,I., Mendes de Farias,T., Moi,D., Nevers,Y., Radoykova,H.-S., Rossier,V., Warwick Vesztrocy,A., *et al.* (2020) OMA orthology in 2021: website overhaul, conserved isoforms, ancestral gene order and more. *Nucleic Acids Res.*, **49**, D373–D379.
8. Bafna,V. and Pevzner,P.A. (1996) Genome rearrangements and sorting by reversals. *SIAM J. Comput.*, **25**, 272–289.
9. Bhatia,S., Feijão,P. and Francis,A.R. (2018) Position and content paradigms in genome rearrangements: the wild and crazy world of permutations in genomics. *Bull. Math. Biol.*, **80**, 3227–3246.
10. Bohnenkämper,L., Braga,M. D.V., Doerr,D. and Stoye,J. (2021) Computing the rearrangement distance of natural genomes. *J. Comput. Biol.*, **28**, 410–431.
11. Khan,N.A. and McQuillan,I. (2017) Descrambling order analysis in ciliates. In: *International Conference on Unconventional Computation and Natural Computation*. Springer, pp. 206–219.
12. Kinsella,M., Patel,A. and Bafna,V. (2014) The elusive evidence for chromothripsis. *Nucleic Acids Res.*, **42**, 8231–8242.
13. Stevenson,J., Terauds,V. and Sumner,J. (2023) Rearrangement events on circular genomes. *Bull. Math. Biol.*, **85**, 107.
14. Darling,A.C., Mau,B., Blattner,F.R. and Perna,N.T. (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome research*, **14**, 1394–1403.
15. Zheng,W., Chen,J., Doak,T.G., Song,W. and Yan,Y. (2020) ADFinder: accurate detection of programmed DNA elimination using NGS high-throughput sequencing data. *Bioinformatics*, **36**, 3632–3636.
16. Burns,J., Kukushkin,D., Lindblad,K., Chen,X., Jonoska,N. and Landweber,L.F. (2016) <mds_ies_db>: a database of ciliate genome rearrangements. *Nucleic Acids Res.*, **44**, D703–D709.

17. Swart,E.C., Bracht,J.R., Magrini,V., Minx,P., Chen,X., Zhou,Y., Khurana,J.S., Goldman,A.D., Nowacki,M., Schotanus,K., *et al.* (2013) The *Oxytricha trifallax* macronuclear genome: a complex eukaryotic genome with 16,000 tiny chromosomes. *PLOS Biol.*, **11**, e1001473.
18. Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
19. Altschul,S.F., Madden,T.L., Schäffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
20. Burns,J., Kukushkin,D., Chen,X., Landweber,L.F., Saito,M. and Jonoska,N. (2016) Recurring patterns among scrambled genes in the encrypted genome of the ciliate *Oxytricha trifallax*. *J. Theor. Biol.*, **410**, 171–180.
21. Makino,K. and Uno,T. (2004) New algorithms for enumerating all maximal cliques. In: Hagerup,T. and Katajainen,J. (eds.) *Algorithm Theory - SWAT 2004*. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Vol. **3111**, pp. 260–272.
22. Feng,Y., Neme,R., Beh,L.Y., Chen,X., Braun,J., Lu,M.W. and Landweber,L.F. (2022) Comparative genomics reveals insight into the evolutionary origin of massively scrambled genomes. *eLife*, **11**, e82979.
23. Nattestad,M., Goodwin,S., Ng,K., Baslan,T., Sedlazeck,F.J., Rescheneder,P., Garvin,T., Fang,H., Gurtowski,J., Hutton,E., *et al.* (2018) Complex rearrangements and oncogene amplifications revealed by long-read DNA and RNA sequencing of a breast cancer cell line. *Genome research*, **28**, 1126–1135.
24. Braun,J. (2020) Discrete Models and Algorithms for Analyzing DNA Rearrangements. PhD thesis, University of South Florida.

Received: June 10, 2023. **Revised:** September 7, 2023. **Editorial Decision:** September 20, 2023. **Accepted:** October 3, 2023