Journal of Bioinformatics and Computational Biology
© Imperial College Press

Anchors for Homology-Based Scaffolding

Karl K. Käther*
karl@bioinf.uni-leipzig.de

Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Leipzig University, Härtelstrasse 16-18, D-04017, Germany

Thomas Gatter†

Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Leipzig University, Härtelstrasse 16-18, D-04017, Germany

Steffen Lemke‡

Zoology Department, University of Hohenheim, Garbenstraße 30, 70593, Germany

Peter F. Stadler§

Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Leipzig University, Härtelstrasse 16-18, D-04017, Germany

Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, D-04103, Germany

Department of Theoretical Chemistry, University of Vienna, Währingerstraße 17, A-1090, Austria

Facultad de Ciencias, Universidad National de Colombia, Colombia

Center for non-coding RNA in Technology and Health, University of Copenhagen, Ridebanevej 9, DK-1870, Denmark

Santa Fe Institute, 1399 Hyde Park Road, NM 87501, USA

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Homology-based scaffolding orders contigs based on conserved collinearity of homologous sequences across related species. Existing methods often rely on costly whole-genome alignments or show limited robustness when integrating multiple references. Here, we

*0009-0003-3439-5813, corresponding author
†0000-0001-5567-3016
‡0000-0001-5807-2865
§0000-0002-5016-5191

introduce an anchor-based scaffolding framework that adapts synteny anchors to efficiently infer contig order and orientation relative to one or more reference genomes. Our approach leverages precomputed, sufficiently unique anchors and their respective high-confidence homology matches in a greedy approach, combining single-reference to multi-reference scaffolds using a maximum matching. Across simulated and real datasets, anchor-based scaffolding achieves accuracy comparable to state-of-the-art methods. Notably, the approach shows particular strengths in multi-reference settings. These results demonstrate that synteny-anchor–based scaffolding provides an additional tool for homology-based scaffolding with robust accuracy and superior performance in multi-reference scenarios.

Keywords: Genome Assembly, Synteny, Maximum Matching

1. Introduction

High-throughput sequencing and assembly pipelines have been well established in the last decades, allowing researchers to create high quality contigs for genomes of varied species. Yet, scaffolding, the task of arranging them in correct chromosomal order and orientation, remains a challenging and contested area of research. Scaffolding typically requires additional data, because paired-end reads alone do not possess sufficient discriminatory power. While Hi-C approaches have become the de facto gold standard [1,2], homology-based scaffolding using closely related reference genomes offers an attractive alternative as high-quality reference assemblies become increasingly available. Several approaches have been developed for this task [3,4,5,6], including tools for multi-reference scaffolding [7,8].

Current homology-based scaffolding approaches face several challenges. Computationally, methods relying on whole-genome alignments can be resource-intensive; for example, Progressive Cactus alignments required for `Ragout2` [9] are extremely computationally demanding for large eukaryotic genomes [5]. Biologically, repetitive and duplicated genomic regions complicate scaffolding by introducing noisy signal, while larger-scale rearrangements between target and reference add further complexity (Fig. 1). Additionally, errors from earlier assembly steps can also introduce conflicting information. Employing multiple references amplifies these contradictions, requires phylogenetic information to weigh references appropriately, and demands additional computational resources.

Synteny anchors - sufficiently unique sequences forming unambiguous 1-1 orthologous matches [10,11] - offer solutions to these limitations. By construction, anchors exclude duplicated and many otherwise rearranged regions, reducing alignment ambiguity which promises to enable development of efficient algorithms for contig ordering.

In this contribution we adapt our recently developed `AncST` synteny anchor framework [11], originally designed for synteny detection, for the purpose of contig scaffolding. Formally, scaffolding assigns each contig i a position $\pi(i)$ and orientation $o(i) \in \{-1, +1\}$ in the final assembly, seeking to maximize a score $S(\pi, o)$ that rewards collinearity with the reference. This optimization problem is NP-hard [12], motivating heuristic approaches. Our method employs a greedy algorithm that or-

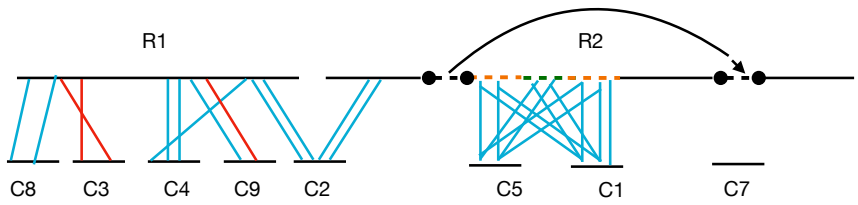

Fig. 1. Homology-based scaffolding. A list of contigs C1 to C8 is ordered along reference chromosomes R1 and R2 based on alignments indicated by colored lines between them. The contigs are initially unordered and placed into the order with respect to their alignments. Here, the `AncST`-based pipeline's strategy is depicted in which a contig gets placed into the order as soon as the majority of its alignments with its best-matching reference chromosome has been seen. Note that not all contigs have to be placed (C6 is missing). Different complications for homology-based scaffolding are shown: Blue lines indicate forward and red ones reverse matches; thus C3 is a contig which should either be turned or marked with one or more inversion events. C2 depicts the case when initial contigs get incorrectly fused in the assembly process which leads to one long scaffold consisting of all contigs aligning to R1 and R2. In R2 the arrow indicates a translocation event and the dashed colored lines denote local duplication events (colors do not matter). Hence, C5 and C1 are in a duplicated region which is difficult to reconcile and C7 is placed incorrectly.

ders contigs based on anchor positions along reference chromosomes, then applies a maximum weighted perfect matching approach [8] to reconcile single-reference scaffolds into multi-reference consensus scaffolds. This work is timely given massive genome sequencing efforts such as the Darwin Tree of Life,[a] the European Reference Genome Atlas,[b] and the Earth Biogenome Project.[c]

2. Methods

2.1. *Anchors*

In recent work, we investigated the notion of annotation-free anchors as a means to identify syntenic regions efficiently; see [10,11] for full detail. In brief, a subsequence w of a genome is called d_0-unique if $d(w, x) > d_0$ holds for every other subsequence x not overlapping w, and d denotes a metric pairwise sequence distance. Two d_0-unique subsequences w and w' in different genomes in G and G' are guaranteed to form a unique match if $d(w, w') < d_0/2$. In practice, it suffices to use a cutoff of d_0 minus some small tolerance. We call d_0-unique sequences anchor candidates. Pairs of anchor candidates w and w' satisfying $d(w, w') < d_0/2$ form *anchors* between the genomes G and G'.

Anchors were used in [11] to construct fine-grained synteny maps between not-too-distant genomes. In order to determine anchor candidates efficiently, we use infrequent k-mers with a length close to $\log_4 |G|$, where $|G|$ denotes the genome

[a] https://www.darwintreeoflife.org
[b] https://www.erga-biodiversity.eu
[c] https://www.earthbiogenome.org

size, as an indicator for the uniqueness of w. Details of the implementation are described in [11].

2.2. *Contig sorting*

The basic idea underlying our approach is rather straightforward: Since anchors between target contigs and reference chromosomes establish a strict 1-1 orthology relationship, they also anchor contigs at specific locations of the reference assembly. The same 1-1 relationship also justifies a simple greedy approach. Thus, we first determine, for each contig, the reference chromosome with the largest number of matched anchors. In the current implementation we do not attempt to identify genome rearrangements between reference and target explicitly. Instead, we directly construct homology-based scaffolds that (locally) match within chromosomes of the reference assembly. We will discuss this point in more detail in the Conclusions section.

After global construction of anchors, we consider each reference chromosome separately. Let $\mathcal{L}$ be an initially empty list. We traverse the anchors between the reference chromosome and its assigned contigs in the order of the reference. For each contig, we keep count of the number of its anchors that have been visited so far. A contig i is appended to $\mathcal{L}$ as soon as at least half of its anchors have been visited. The order of their insertion into $\mathcal{L}$ therefore determines the positions $\pi(i)$ of the contigs in the emerging scaffold. Their orientations $o(i)$ are assigned based on the predominant strand direction of the anchor matches with the assigned reference sequence. The use of `AncST` anchors, i.e., nearly perfect 1-1 orthologs, makes the greedy approach a natural choice since overlapping matches of contigs to the references are expected to be absent in theory and extremely rare in practice. This simple greedy strategy for determining a contig order relative to a reference assembly can easily be extended to multiple reference assemblies by simply running anchor detection and greedy ordering independently for each reference assembly. Different reference assemblies will yield slightly different contig orders; it then remains to reconcile the conflicting order information.

2.3. *Merging of alternative scaffolds*

Scaffolds obtained from different reference assemblies can be reconciled using a simple graph-theoretic approach [8]. To this end, we define a graph $\Gamma = (V, E, w)$ whose vertex set are the end points of the target contigs, i.e., each target contig is represented by *two* vertices. An edge $\{u, v\}$ is inserted if u and v are endpoints of two distinct contigs that are consecutive for one of the reference assemblies. Note that for a pair of consecutive contigs the choice of the adjacent endpoints determines the relative orientation of the two contigs. Each edge $\{u, v\}$ has a weight w that in the simplest case counts the number of reference assemblies in which u and v are consecutive. More sophisticated weights w can be introduced that account e.g. for the quality of anchors for the contigs of which u and v are endpoints. Weights are

aggregated if an edge appears for more than one reference assembly. For technical reasons, it is useful, moreover, to assign edges with weight 0 to all pairs of vertices that are neither consecutive w.r.t. to at least one reference assembly nor form the endpoints of the same contig. This amounts to building a single component with circular connections.

In order to extract a consensus scaffold from the weighted graph Γ, we observe that in a scaffold each contig has at most one predecessor and one successor and thus each end-point of a scaffold can be incident to at most one edge. Hence scaffolds that only contain observed adjacencies correspond to matchings in Γ. A maximum weighted matching in Γ thus serves as a best estimate for a consensus scaffolding. We note that a matching will be perfect, i.e. includes all vertices, only if all contigs are placed into circular chromosomes. Otherwise, there will be contig ends that correspond to ends of a scaffold and thus are not incident to any edge. The maximum weighted matching problem can be solved in polynomial time using the *blossom* algorithm [13].

Adding missing edges to Γ with weight 0 transforms the problem into a maximum weight (or minimum cost) *perfect matching* problem, which can be solved by the highly optimized `Blossom V` implementation [9]. This approach has been pioneered in [8]; we refer to Figure 2 in this publication for a detailed visual description of the algorithm.

In the resulting perfect matching, head and tail of the same contigs are connected, resulting in a regular graph Υ of vertex degree 2, i.e., a vertex disjoint union of cycles. After breaking each cycle by removing its minimum weight edge and removing completely unsupported edges with weight 0, we are left with a disjoint union of paths, each of which specifies a linear order of contigs as well as the reading direction of each contig, and thus a scaffold.

2.4. *Synthetic test data*

We produced three simulated draft assemblies of the human genome (GRCh38/hg38) in which the genome is chopped at repetitive sequences to generate contigs, based on typical fail scenarios of common assemblers where sequences cannot be uniquely elongated. In addition to this merely fragmented version we produced two further scenarios with additional noise using a combination of tandem duplications, translocations and inversions; see Suppl. Sect. A.1 which can be found at Zenodo. The genome assemblies of chimpanzee (Clint_PTRv2/panTro6) and bonobo (Mhudiblu_PPA_v0/panPan) were used as references for scaffolding. All sequence data were downloaded from the UCSC genome browser[d].

[d] https://hgdownload.soe.ucsc.edu/goldenPath/

2.5. *Real-life test data*

In addition to synthetic data we complement benchmarks with real life data sets. We employed our scaffolding pipeline to a set of lower quality 20 `Drosophila` genomes published on NCBI, for 11 of which chromosome level high quality genomes are also available as a reference. (Suppl. Tbl. B.1).

Here we focus on the application scenario in which genomes of multiple related species of limited assembly quality are available for homology-based scaffolding. For each of the 11 species with high quality genomes, we choose a lower quality genome assembly (labeled as `scaffold` or `contig` assembly level on NCBI) and scaffolded them with regards to the other lower quality 19 species.

2.6. *Scaffolding*

We compared our approach with some widely used homology-based scaffolding approaches: `CSAR` [3] and `Multi-CSAR` [8], `RagTag` [7] with their `merge` command for multi-ref scaffolding, `ntJoin` [5] which allows multi-ref scaffolding as a default and `Ragout2` [6] which uses `cactus` multiple sequence alignments as input. The details about versions and parameters can be found in Suppl. Sect. C.2.
`AncST` was used to produce pairwise alignments between the respective genomes (parameters noted in Suppl. Sect. C.2). The resulting pairwise scaffolds were reconciled using a reimplementation of the ideas behind the merging algorithm in `Multi-CSAR` [8] as described in the respective *Merging of Alternative Scaffolds* section and further details can also be found in Suppl. Sect. C.2. While `Multi-CSAR` calculates weights for each reference; `RagTag`, `ntJoin` and `Ragout2` do not produce weights. Since weights may have a significant impact on the results, however, we ran the multi-ref experiment with `ntJoin` and `RagTag` of scaffolding the chopped but not further rearranged version of the synthetic human genome with the weights used by our pipeline and noted that the results were exactly the same as for unit weights. Furthermore, for experiments with real data we show in Suppl. Sect. B.3 that using weights from our pipeline in `RagTag` and `ntJoin` does not significantly alter the results. `Ragout2` does not use weights but a multiple sequence alignment produced by `cactus` for which a weighted initial phylogeny can be provided.

2.7. *Evaluation*

As previously described, contiguity metrics alone are too simplistic to properly capture assembly quality [14,15,16,17]. Hence, we employ several custom metrics for a more comprehensive study of the resulting scaffolds' qualities. Firstly, we use a Needleman-Wunsch global alignment algorithm [18] to quantify the edit distance between reconstructed scaffolds and the correct order of the contigs in the reference. A score of 1 is used for matching contigs, while insertions, deletions, and mismatches are scored with 0. The total score of the optimal alignment therefore equals the number of matched contigs for an optimal solution and, after determining the correct orientations of the aligned contigs, can directly be used as a

performance measure. More precisely, we compute the *precision* as the percentage of the Needleman-Wunsch alignment score divided by the maximally achievable score, which is given by the total number of contigs minus 1/2 times the number of fused contigs. Alternatively, we may normalize the Needleman-Wunsch alignment score by the total number of contigs that are placed into any scaffold, resulting in a measure that is akin to *specificity*. Throughout, we refer to these metrics as *NW prec.* and *NW spec.*, respectively.

We also compute the total length of DNA covered by the correctly aligned and oriented contigs, i.e. exactly those with positive contribution in the NW alignment, normalized by the total length of all contigs from the original genomes (abbreviated as *NW DNA cov.*).

In addition to evaluating the alignments, we count the number of correctly connected contigs among all possible correct connections (*precision*) and all made connections (*specificity*). Similar as for NW, we also compute the total length of DNA covered by correctly connected contigs relative to genomic length. Here, exactly all contigs are included that are connected correctly on at least one side, even if they are also involved in a false connection. We further refer to this metric as *adj. DNA cov.* Finally, we record the number of *breaks* which are introduced to the original chromosomes when they are covered by more than one newly assembled scaffold.

Finally, we employ the widely used assembly evaluation tool `Quast` (`v5.3.0`) [19] and compare the contiguity statistic *auN* (`a`rea `u`nder `N`x curve, see [20]) for which a higher value indicates better performance. Additionally, we consider the number of *misassemblies* (for which a low number is preferable) reported against the respective reference genomes for both the synthetic and real data sets. We employ `auNA` as a measure of contiguity which takes into consideration only correct alignment blocks with the reference genome.

3. Results

3.1. *Synthetic Data - Custom Metrics*

Table 3.1 summarizes the performance results for scaffolding the contigs of the human genome produced by breaking it up at regions of considerable repeat content and length as well as reversing and fusing some of the resulting contigs. Tables 1 and 2 in Suppl. Sect. A.4 show the results for the rearranged assemblies. Across the benchmark suite, no single tool dominates all metrics, with different approaches showing advantages for different measures depending on the scenario. `AncST` consistently produces relatively contiguous single- and multi-ref scaffolds across the rearrangement levels which shows in the number of chromosome breaks and NW score in brackets for which only one contig is allowed to align to the original sequence. Considering the fragmentation of the multi-level scaffolds of all other tools, it can be argued that `AncST` performs multi-ref scaffolding best overall. `ntJoin` and `RagTag` perform the best most consistently in the NW scoring scheme while `CSAR`

and `Ragout2` generally perform best for the adjacency statistics.

Table 1. Comparison of scaffolding performance. Data refer to the synthetic contig sets obtained by fragmenting the human genome. The genomes of chimpanzee (tro) and bonobo (pan) are used as references. We refer to Section 2 for details on these metrics. The second columns for the alignment-based statistics in brackets considers the case in which only one new scaffold is allowed to be aligned to each reference chromosome as opposed to the figure before the brackets for which multiple scaffolds can align to each reference chromosome. All values (except for time measurements and the number of chromosome breaks) are percentages. Some of the best and worst performances are highlighted for each column (values in brackets are considered as separate columns except for the second time one). Accordingly, red means worst performing, magenta is within 3 percentage/time/count points of the worst, blue means best performing and cyan is within 3 percentage/time/count points of the best. Execution times are approximate wallclock times on a Linux server with 256 GB RAM and 64 available cores on two Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz processors. The second time column reports the running time of (1) `AncST` without the pre-computation of anchors for each genome, since the latter can be performed as a one-time preprocessing for each genome and (2) `cactus` for `Ragout2`.

Pipeline	Time[min]	NW			adj.			chr. breaks
		prec.	spec.	DNA cov.	prec.	spec.	DNA cov.	
CSAR pan	3200	65 (33)	63	64 (35)	94	90	91	83
CSAR tro	3200	63 (31)	62	59 (28)	95	91	91	75
Multi-CSAR	6500	87 (3)	82	86 (5)	43	68	70	1086
RagTag pan	25	79 (77)	82	80 (76)	93	90	87	9
RagTag tro	25	82 (80)	84	83 (79)	92	89	86	5
Multi-RagTag	52	60 (9)	92	80 (11)	48	86	75	327
ntJoin pan	20	79 (77)	81	80 (74)	92	88	82	10
ntJoin tro	20	82 (80)	83	82 (79)	92	87	82	4
Multi-ntJoin	28	89 (28)	96	77 (19)	85	93	74	178
Ragout2 (multi)	4100 (15)	74 (39)	80	79 (38)	93	97	91	58
AncST pan	600 (14)	79 (76)	84	80 (76)	91	91	88	7
AncST tro	600 (14)	82 (79)	87	82 (79)	92	92	88	4
Multi-AncST	1200 (30)	81 (79)	86	82 (79)	92	91	88	2

In terms of speed, the minimizer-based `RagTag` and `ntJoin` are orders of magnitude faster than the other tools if the extensive alignments needed by the other tools are not precomputed. A more detailed breakdown of the resource consumption of the `AncST` pipeline shows that about a quarter of the compute time is used for the collection of the k-mer statistics with `GenMap` as a basis for determining the anchor candidates. Further two thirds of the effort is expended for the computation of d_0-uniqueness of these candidates through a `blast` comparisons of the anchor candidates against their own genomes. All other tasks in the pipeline account for less than 10% of the wallclock time. Hence, the computation time for pairwise anchor candidate comparison and scaffolding alone are on par with those of the fastest methods.

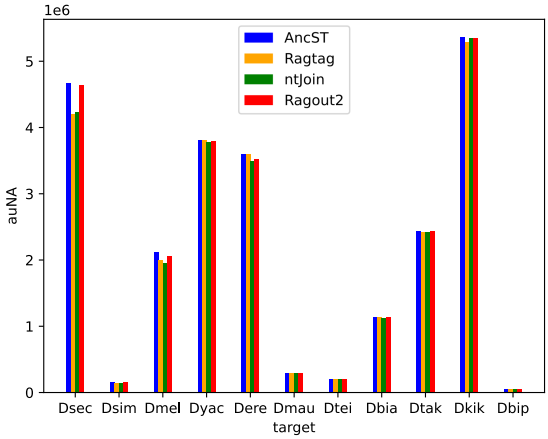

Fig. 2. `auNA` as computed by `Quast` for the 11 `Drosophila` newly scaffolded species with a chromosome- scale reference genome on NCBI. Further details about the species in Suppl. Tbl. B.1.

3.2. *Synthetic Data - Evaluation with `Quast`*

`Quast` evaluation further supports the results obtained with the metrics described above. The full results and corresponding figures are compiled in Suppl. Sect. A. In summary, `CSAR` performs worst overall, with one exception: It shows the best `auNA` metric for single-ref scaffolding of the chopped but not rearranged version of the human genome. `AncST`, `ntJoin` and `Ragout2` perform well overall. For the multi-ref scenario `AncST` shows robustness while the other tools can usually not only *not* benefit from using multiple references but performance drops significantly except for `ntJoin` with regards to `auN`.

In Suppl. Sect. A.3 we plot the relative coverage of the 24 human chromosomes (22+`X`+`Y`) estimated by the total length of all `Quast` alignments with the new scaffolds covering this chromosome. It can be seen that while for single reference homology-based scaffolding all tools except `CSAR` produce sensibly large scaffolds, only our pipeline achieves this for multi-ref runs.

3.3. *Real-Life Data - Evaluation with `Quast`*

As for the synthetic data above, we show `auN`, `auNA` and the number of `misassemblies` as computed by `Quast` to compare the scaffolding results `AncST`, `ntJoin`, `Ragout2` and `RagTag` shown in suppl. Sect. B.2. We excluded `Multi-CSAR` from the experiments as it implies resource-intensive whole-genome alignments of all pairs and this tool shows largely disappointing results for the synthetic data set. In Figure 2 we show that the tools are relatively comparable in their results but `AncST` performs best most consistently.

In the following, the detailed analyses are restricted to the two most promising tools based on the contiguity statistics `auN` and `auNA` which are our own and `RagTag`. Firstly, in order to assess whether the multi-ref approach improves over single-ref scenarios, as one would expect from the low quality of the genomes, we compare the fractions of single-ref scaffolds with higher contiguity as measured by `auN`. Comparing all 380 single-ref scaffolds with their respective multi-ref version, in 76 % of cases from `RagTag` the multi-ref version is more contiguous while our pipeline achieves higher contiguity in 90 % of cases. Hence, while both pipelines can utilize the multi-ref approach for a scenario in which a set of lower-quality genomes is available for scaffolding, our pipeline more consistently produces multi-ref scaffolds of higher contiguity. To estimate how many unrealistically large scaffolds are created in either approach, we count the number of cases in which the largest scaffold exceeds 50 million nts, a number significantly higher than any `Drosophila` reference chromosome. While no single-ref scaffold exceeds 50 million nts for either approach, `RagTag` produces 8 and `AncST` 4 multi-ref scaffolds longer than this threshold. Thus, both approaches produce unrealistically large scaffolds in a significant amount of cases of the multi-ref approach but the issue is more severe for `RagTag`. To further investigate the scaffolds' contiguity and severity of potential misassemblies we compare the coverage of large chromosomes of the reference genomes (if available) with potential new scaffolds and single original contigs in some cases. The complete results for each species are displayed in Suppl. Sect. B.3. As an example, consider the extreme case of `Drosophila biarmipes` whose `RagTag` scaffolds display an `auN` value of around 176 million and a longest contig of length 180 million while the total assembly length is 185 million. Related to the proportion of unrealistically large scaffolds, `RagTag` scaffolds show more cases in which multiple reference chromosomes are covered by the same scaffold. In order to assess whether `RagTag`'s performance can be improved by using weights for each reference we repeated the scaffolding and analysis using the same weights as in in our pipeline. In essence, using our weights for `RagTag` does not significantly improve its performance (Suppl. Sect. B.4) for this multi-ref scaffolding experiment.

4. Conclusions

Homology-based scaffolding is likely to gain increasing importance in non-model-organism genomics in response to the rapidly growing number of high quality genomes that are suitable as references. The phylogenetic distance of available reference genomes to a newly sequenced genome thus shrinks, increasing the accuracy of the reference-based approach and making multi-ref scaffolding more attractive. Hence, the methodology has the potential to provide biologists with large numbers of accessible (close to) chromosome-level genome assemblies for functional analyses, comparative genomics, and evolutionary studies at moderate experimental and computational costs. This is of particular relevance with respect to the large-scale efforts mentioned in the introduction.

We have shown that anchor candidates can be used effectively for homology-based contig scaffolding. Our anchor-based approach occupies a middle ground in the scaffolding landscape: computational cost and scalability fall between expensive whole-genome alignment methods and highly efficient minimizer-based approaches, while achieving comparable accuracy to established tools.

In terms of scores employed in the Results section, we observe that `AncST` usually scores within three points of the best performance for DNA covered by the NW alignments and chromosome breaks. Moreover, it shows only low or moderate distance to the best performing tools for all metrics and does not show any particular weakness. Its key strength lies in robust performance in multi-reference scaffolding for both synthetic and real data. A detailed algorithmic analysis of how each tool processes individual datasets is beyond the scope of this work given the extensive benchmark suite encompassing multiple tools, scenarios, and metrics. We can, however, identify plausible causes for the differences in results. At least in part, the choice of the underlying algorithms determines performance characteristics. With exception of `ntJoin`, all tools use alignments to generate single-ref contig orders. For multi-ref scaffolding all approaches rely on graph-based methods to reconcile evidence from the different references.

Theoretically, whole-genome alignments from `MUMmer` [21] in `CSAR` and `cactus` for `Ragout2` should achieve the highest resolution. The performance benchmarks show, however, that `CSAR` and `Multi-CSAR` arguably perform the worst overall and `Ragout2` only excels in contig adjacencies while producing relatively fragmented scaffolds. We suspect that these tools do not resolve the conflicting information from duplicated and rearranged sequences with sufficient accuracy and that these problems then propagate to the scaffolding algorithms. The algebraic rearrangement models implemented in `CSAR` do not seem to capture the important signals for single-reference scaffolding very well. Moreover, `Multi-CSAR's` implementation of the maximum perfect matching algorithm may be suboptimal because we observe similar or better results for the multi-reference scaffolds than for its single-ref inputs with our implementation.

`Ragout2` uses several sophisticated steps to first obtain synteny blocks of different granularity from the multiple sequence alignment and then refine the breakpoint graph. In contrast, our implementation only makes use of atomic synteny anchors without producing larger blocks first. Despite its sophisticated algorithms `Ragout2` performs worse than `AncST` except for correct contig adjacencies. Most likely, our anchors already provide a robust signal for scaffolding without further refinement of synteny blocks. In fact, we confirmed that using `MCScanX` collinearity blocks instead of single anchors did not improve the results of `AncST`. We are lead to believe that larger synteny blocks actually prevent the creation of some contig connections because the larger the size of a block, the more potentially conflicting sequences are included. Such ambiguities are filtered out from `AncST` by design and we only use the inherently less conflicting pairwise alignments as input for the multi-reference scaffolding. Accordingly, we observe that the number of cycles broken to obtain

scaffolds from the maximum perfect matching is extremely small in our pipeline, which eventually lead to the more contiguous scaffolds. In summary, these architectural differences likely explain the good performance of `AncST` in comparison to `Ragout2` despite using atomic anchors instead of larger synteny blocks, only pairwise alignments, and less sophisticated algorithms to produce multi-reference scaffolds.

`RagTag` and `ntJoin` also involve cycle-breaking steps in their graph-refinement strategies, which may be the cause of the relatively high fragmentation of their multi-reference scaffolds despite the high quality of the single-reference inputs. Furthermore, our pipeline readily produces sensitive weights for multi-reference scaffolding. Assuming that relative anchor coverage is similar across genomes, we should obtain a robust estimation of phylogenetic distance of the species by considering the total aligned length of pairwise anchor alignments normalized by the size of the two genomes. For single-reference scaffolding `RagTag` and `ntJoin` are clearly the best performing tools overall, particularly considering their extremely low run time. Hence, minimizer sketches and alignments based on them seem to provide most of the relevant signal for homology-based scaffolding and even avoid some of the issues leading to inferior performance of `CSAR` and `AncST`. Arguably, `AncST` still achieves comparable performance in single-reference scenarios since its NW alignments cover nearly as much DNA as the best performing tools, the adjacency statistics are within a few points of competing tools, and `AncST` still produces the smallest number of chromosome breaks. Moreover, `Quast` evaluation confirms the comparability in single-ref and superiority in multi-ref runs for `AncST`.

We conclude that `AncST` provides synteny anchor alignments highly suitable for homology-based scaffolding which can be leveraged by a greedy algorithm for single-ref and maximum perfect matching approach for multi-ref scaffolding. The issues of homology-based scaffolding are inherently addressed in the nature of the `AncST` anchors: By desgin, repetitive sequences are excluded and ambiguous alignments of anchors are marked as unreliable. This design may also facilitate the overall superior performance of `AncST` with respect to the real data experiments where we scaffolded explicitly poor assemblies with higher content of repeats and misassemblies. Furthermore, anchor candidates are computed for highly fragmented assemblies in just the same way and can readily indicate problematic genomic regions when their alignments to other genomes' candidates are ambiguous. As these alignments are marked as unreliable and not further used, performance can stay relatively high even for poor assemblies. A distinctive advantage of the anchor-based approach is that candidates can be precomputed and stored as simple annotation files that represent anchors as coordinate pairs associated with their level of the interval's d_0-uniqueness. Consequently, anchors of the target genome are the same in each pairwise comparison with the references which may promote the observed robustness for reconciliation of multiple single-ref scaffolds.

`AncST` anchor computation is more resource intensive than the minimizer-based approaches for homology-based scaffolding. Compared to whole-genome and multiple sequence alignment-based scaffolders, however, `AncST` is more scalable as it

avoid a full whole-genome alignment and the resource consumption for pairwise alignments is comparably low because the number of anchor candidates is limited and repetitive sequences are avoided. Other potential limitations of `AncST` in general include the reliance on nucleotide alignments and potential difficulties with polyploid genomes. Since homology-based scaffolding is usually restricted to closely related species, `AncST's` phylogenetic performance boundaries [11] are of no practical concern. Whether polyploid genomes can effectively be aligned by `AncST` depends on the relative timing of chromosome duplications: if they are sufficiently old, enough duplicated sequences may have diverged as to serve as anchor candidates.

A natural next step is leveraging non-collinear anchor alignments to identify breakpoints arising from genome rearrangements. The anchor matches themselves, moreover, can be fed directly into the analysis of macrosynteny, which is usually performed as part of genome assembly projects. For instance, we can readily track possible inversion scenarios by enumerating regions for which `AncST` anchors align on the opposing strand in considerable numbers. Translocations are indicated by significant collinear stretches of alignments with a breakpoint in one of the species. The current implementation of the `AncST`-based pipeline identifies contigs with alignments to more than one reference chromosome and those with alignments in opposite reading directions on the same reference chromosome. These are strong indicators for rearrangements relative to the reference genome. In principle, these data could be used to extend scaffolds beyond such breakpoints. Whether this can be done with acceptable reliability, however, is currently subject to investigation.

Acknowledgements

This work was funded by the German Research Foundation as part of SPP 2349 *"Genomic Basis of Evolutionary Innovations (GEvol)"* (grant no. 502862570). PFS acknowledges the financial support by the Federal Ministry of Education and Research of Germany (BMBF) through DAAD project 57616814 (SECAI, School of Embedded Composite AI), and jointly with the Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research *Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig*, project identification number: SCADS24B.

Competing Interests

The authors have no competing interests.

Availability and Supplementary Material

The appendix and AncST anchors for the human dataset are available at Zenodo (https://zenodo.org/records/18257364). Further supplementary material is available at https://zenodo.org/records/15061775. The prototypical pipeline is included in the supplementary material and `AncST` Github repository (https://github.com/

Norsbus/AncST) as well as part of results from the webserver (https://anchored.bioinf.uni-leipzig.de/).

Appendix A. Benchmark of Synthetic Human Data

A.1. *Simulation of Synthetic Fragmentation*

Locations of rearrangements were restricted to respecting annotation items, i.e. breakpoints were only introduced at the start or end of annotated items in the annotation file of the human genome. We distinguish between short-range and long-range rearrangements. The former only affect a single annotation item, while the latter pertain to an interval containing two or more annotation items separated by not more than 1 million bases. In addition to a local inversion of the interval, rearrangements were defined as follows: For translocation we randomly selected a different chromosome, an insertion position, and the orientation of the inserted fragment. Up to four duplication events are allowed, either local or remote. For a local duplication the copy of a selected interval is inserted, possibly in inverted orientation, at a location between 1000 and 100000 bases downstream of the interval in question. Alternatively, duplicates were inserted in a randomly chosen chromosome. If a translocated sequence was chosen to only be translocated but not copied, then the original source interval was deleted in order to avoid bloating the genome size. We created one version in which at least 5% of the human genome are rearranged and another one in which at least 10% are rearranged. The number of duplication events as well as sequence intervals are chosen arbitrarily. The rationale behind the moderate number of rearrangement events in the test data is that contigs with a large number of rearrangements relative to the reference cannot be safely placed into a scaffold since homology-based scaffolding only can be employed when co-linearity is the norm. The rearranged genomes are documented in the supplementary material. Contigs were extracted in the same manner from both the rearranged and conserved human genome. In order to mimic the fact that contig assembly usually stops at repetitive sequences, we defined contig ends at stretches of at least 10000 N in the hard-masked version of the human genome. In addition, we introduced random breakpoints within the initial fragments. At this stage we only accepted contigs consisting of at least 20000 bases and 30% unmasked sequence. Each initial fragment was subjected to 30 repetitions of the random fragmentation attempts. In order to simulate assembly errors [22], we fused 5% of the resulting fragments at random. Moreover, reading direction was determined at random.

A.2. *auN(A) and Misassemblies as Computed by Quast*

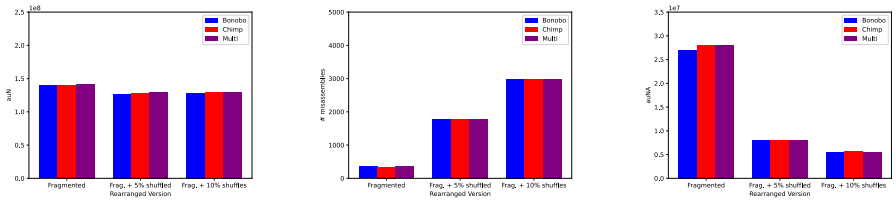

Fig. 3. (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `AncST`.

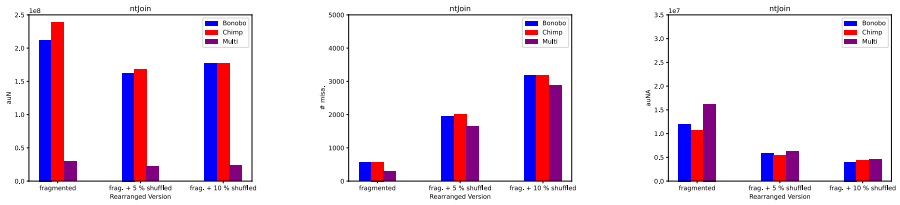

Fig. 4. (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `ntJoin`.

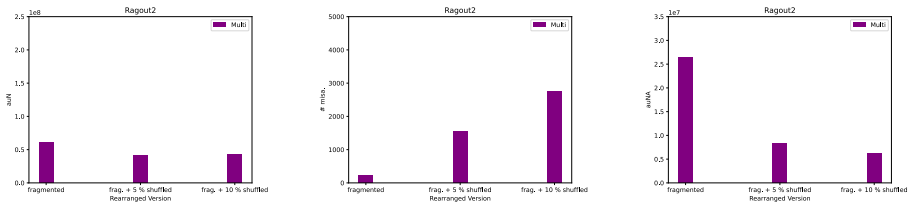

Fig. 5. (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `Ragout2`.

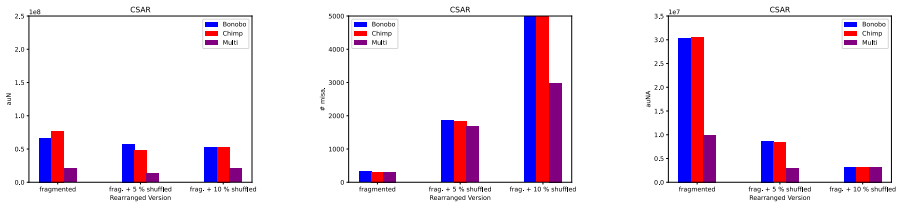

Fig. 6. (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `CSAR`.

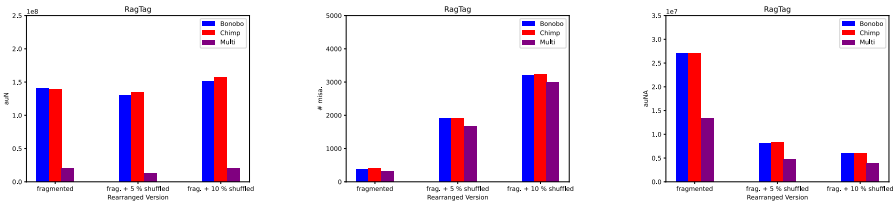

Fig. 7. (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `RagTag`.

A.3. *Coverage of Main Human Chromosomes with Quast Alignments of New Scaffolds*

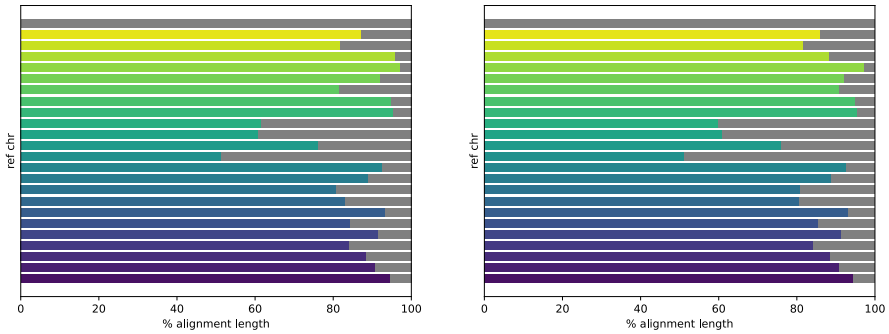

Fig. 8. Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `AncST`. Relative coverage (X-axis) of 24 human chromosomes (22+`X`+`Y`) (Y-axis) by all alignments produced by `Quast`. First, the total length of all alignments of a reference chromosome with any new scaffolds is noted. Then each new scaffold is assigned its relative coverage as the proportion of the length of its alignments with the reference. Scaffolds covering at least 50 % of a reference are colored and the rest is gray. Relative coverage by one contig is bordered by white vertical lines.

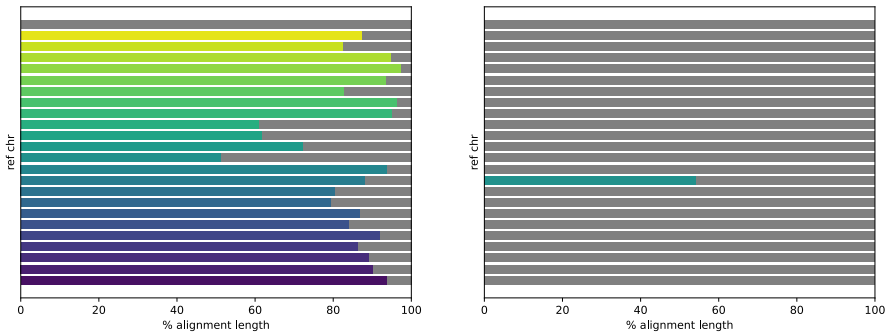

Fig. 9. Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `ntJoin`. Details can be found in caption of 8.

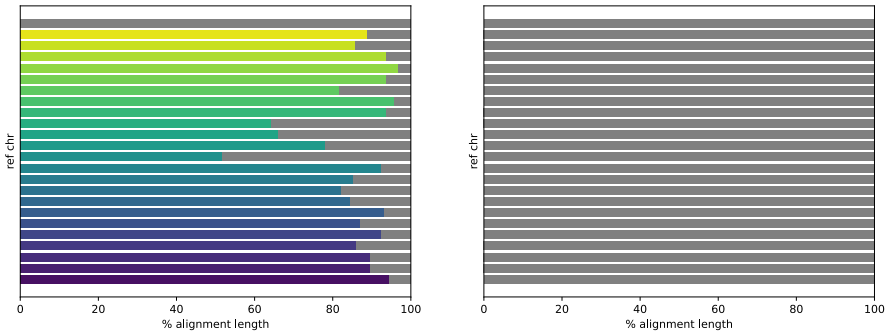

Fig. 10. Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `RagTag`. Details can be found in caption of 8.

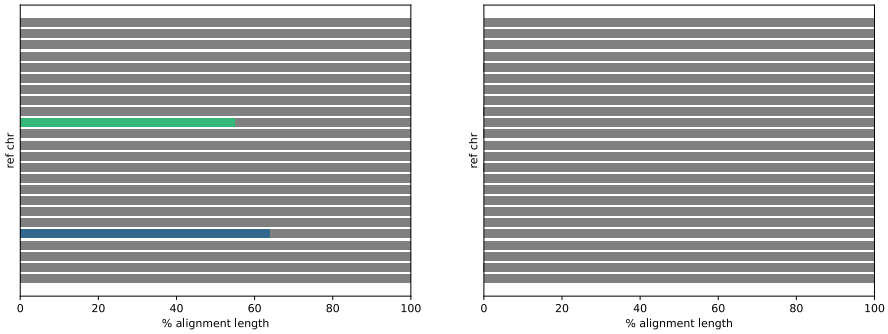

Fig. 11. Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `CSAR`. Details can be found in caption of 8.

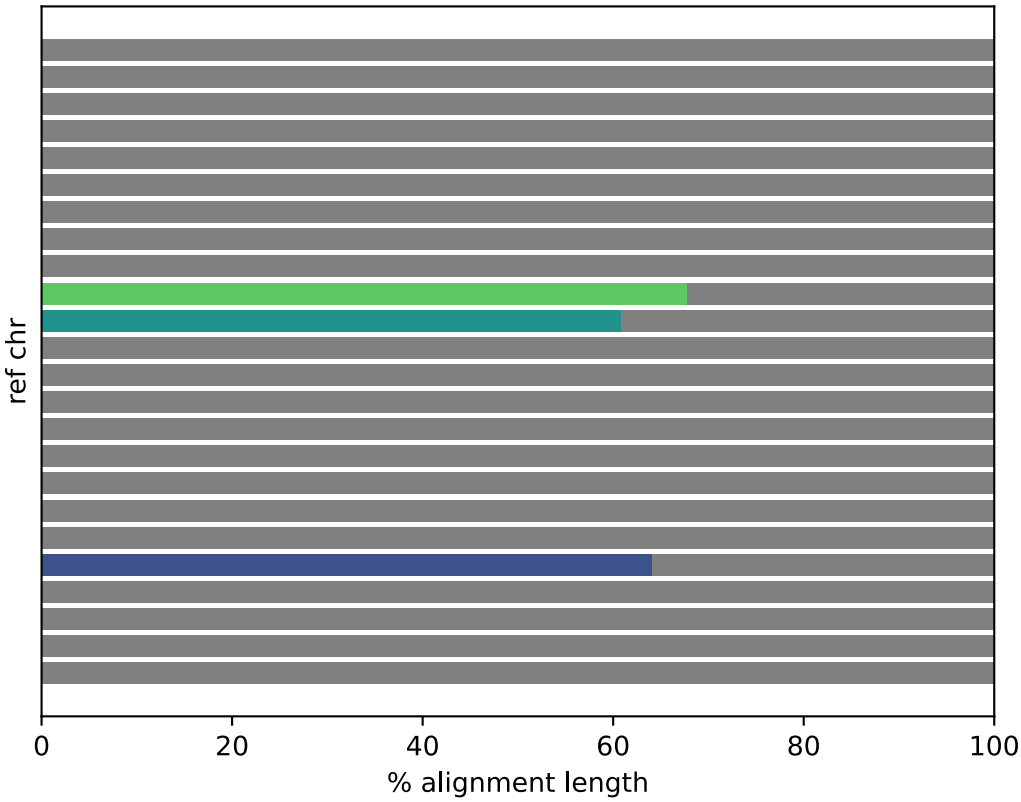

Fig. 12. Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against chimp and bonobo from `Ragout2`. Details can be found in caption of 8.

A.4. *Summary of Benchmarks*

Table 2. Scaffolding results for the chopped human genome with at least 5% rearranged sequence. The genomes of chimpanzee (tro) and bonobo (pan) are used as references. We refer to Section 2 for details on these metrics. The second columns for the alignment-based statistics in brackets considers the case in which only one new scaffold is allowed to be aligned to each reference chromosome as opposed to the figure before the brackets for which multiple scaffolds can align to each reference chromosome. All values (except for time measurements and the number of chromosome breaks) are percentages. Some of the best and worst performances are highlighted for each column (values in brackets are considered as separate columns except for the second time one). Accordingly, red means worst performing, magenta is within 3 percentage/time/count points of the worst, blue means best performing and cyan is within 3 percentage/time/count points of the best. Execution times are approximate wallclock times on a Linux server with 256 GB RAM and 64 available cores on two Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz processors. The second time column reports the running time of (1) `AncST` without the pre-computation of anchors for each genome, since the latter can be performed as a one-time preprocessing for each genome and (2) `cactus` for `Ragout2`.

Pipeline	Time[min]	NW			adj.			chr. breaks
		prec.	spec.	DNA cov.	prec.	spec.	DNA cov.	
CSAR pan	3200	50 (17)	53	50 (19)	82	85	83	156
CSAR tro	3200	53 (19)	56	56 (20)	83	83	85	161
Multi-CSAR	6500	79 (3)	80	75 (3)	37	63	60	1692
RagTag pan	28	67 (64)	79	73 (69)	78	86	79	24
RagTag tro	26	71 (67)	83	76 (72)	78	86	79	18
Multi-RagTag	56	43 (5)	89	71 (7)	31	78	65	460
ntJoin pan	20	68 (65)	81	72 (69)	79	89	79	17
ntJoin tro	20	72 (68)	85	76 (72)	79	88	78	13
Multi-ntJoin	28	77 (20)	96	73 (16)	74	93	69	246
Ragout2 (multi)	4700 (18)	66 (26)	83	70 (29)	79	95	83	85
AncST pan	650 (14)	65 (63)	82	73 (70)	76	90	79	8
AncST tro	650 (14)	69 (66)	87	76 (72)	76	89	80	8
Multi-AncST	1300 (30)	66 (64)	82	73 (70)	77	89	80	5

Table 3. Scaffolding results for the chopped human genome with at least 10% rearranged sequence. For details we refer to the caption of Table 2 in A.4

Pipeline	Time[min]	NW			adj.			chr. breaks
		prec.	spec.	DNA cov.	prec.	spec.	DNA cov.	
CSAR pan	3300	48 (51)	51	40 (14)	72	75	68	159
CSAR tro	3400	52 (18)	55	43 (16)	73	76	69	172
Multi-CSAR	6800	77 (3)	78	71 (4)	33	56	49	1687
RagTag pan	29	66 (63)	77	67 (63)	76	83	71	23
RagTag tro	27	70 (66)	80	69 (66)	77	83	72	16
Multi-RagTag	58	42 (5)	85	67 (8)	29	73	57	433
ntJoin pan	20	67 (65)	81	69 (66)	78	88	74	9
ntJoin tro	20	70 (67)	84	72 (68)	79	88	74	10
Multi-ntJoin	28	77 (22)	96	69 (16)	73	91	62	243
Ragout2 (multi)	4800 (18)	64 (29)	81	65 (25)	78	95	77	82
AncST pan	650 (15)	66 (64)	82	70 (67)	73	89	73	5
AncST tro	650 (15)	69 (66)	86	72 (69)	77	89	74	5
Multi-AncST	1300 (32)	66 (64)	82	69 (66)	77	89	73	2

Appendix B. Benchmark on Drosophila Assemblies

B.1. *Genomic References*

Table 4. Genomes of `Drosophila` species used. `same` in `NCBI Ref.`(erence) means that the Contig or Scaffold level assembly used as a scaffolding target is also marked as the reference chromosome for this species on NCBI.

Accession	Species	Abbreviation	Assembly Level	NCBI Ref.	Identifier
GCA_018904445.1	`D. sechellia`	Dsec	Scaffold	GCF_004382195.2	A
GCA_039725655.1	`D. simulans`	Dsim	Contig	GCF_016746395.2	B
GCA_000778455.1	`D. melanogaster`	Dmel	Contig	GCF_000001215.4	C
GCA_018904385.1	`D. yakuba`	Dyac	Contig	GCF_016746365.2	D
GCA_018904525.1	`D. erecta`	Dere	Scaffold	GCF_003286155.1	E
GCA_018904475.1	`D. mauritiana`	Dmau	Scaffold	GCF_004382145.1	F
GCA_018903625.1	`D. teissieri`	Dtei	Contig	GCF_016746235.2	G
GCA_005876975.1	`D. orena`	Dore	Contig	same	H
GCF_018153835.1	`D. eugracilis`	Deug	Contig	same	I
GCA_018148935.1	`D. biarmipes`	Dbia	Contig	GCF_025231255.1	J
GCA_018152695.1	`D. takahashii`	Dtak	Contig	GCF_030179915.1	K
GCF_018152265.1	`D. ficusphila`	Dfic	Contig	same	L
GCF_018152505.1	`D. elegans`	Dele	Contig	same	M
GCF_018152115.1	`D. rhopaloa`	Drho	Contig	same	N
GCA_008042655.1	`D. burlai`	Dbur	Scaffold	same	O
GCA_018152535.1	`D. kikkawai`	Dkik	Contig	GCF_030179895.1	P
GCA_008042735.1	`D. leontia`	Dleo	Scaffold	same	Q
GCA_021223765.1	`D. bipectinata`	Dbip	Scaffold	GCF_030179905.1	R
GCA_018153235.1	`D. malerkotliana`	Dmal	Contig	same	S
GCA_018148915.1	`D. ananassae`	Dana	Contig	same	T

B.2. *Scaffolding Evaluation of Drosophilas*

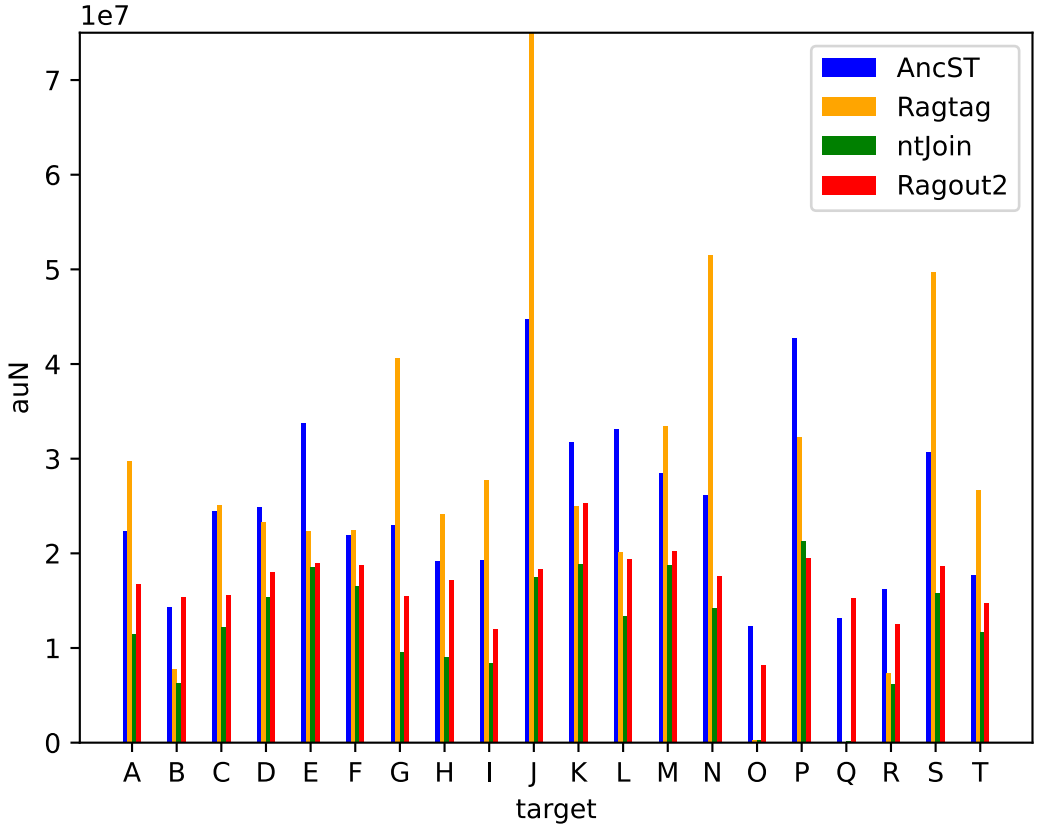

Fig. 13. *auN* as computed by `Quast` for 20 `Drosophila` newly scaffolded species. The new scaffolds for computed by `RagTag` for `Drosophila biarmipes` (J) show an *auN* of around 180 million while we set the upper limit of the y-axis to 75 million for clearer display. Identifier correspondence and further details in Table 4.

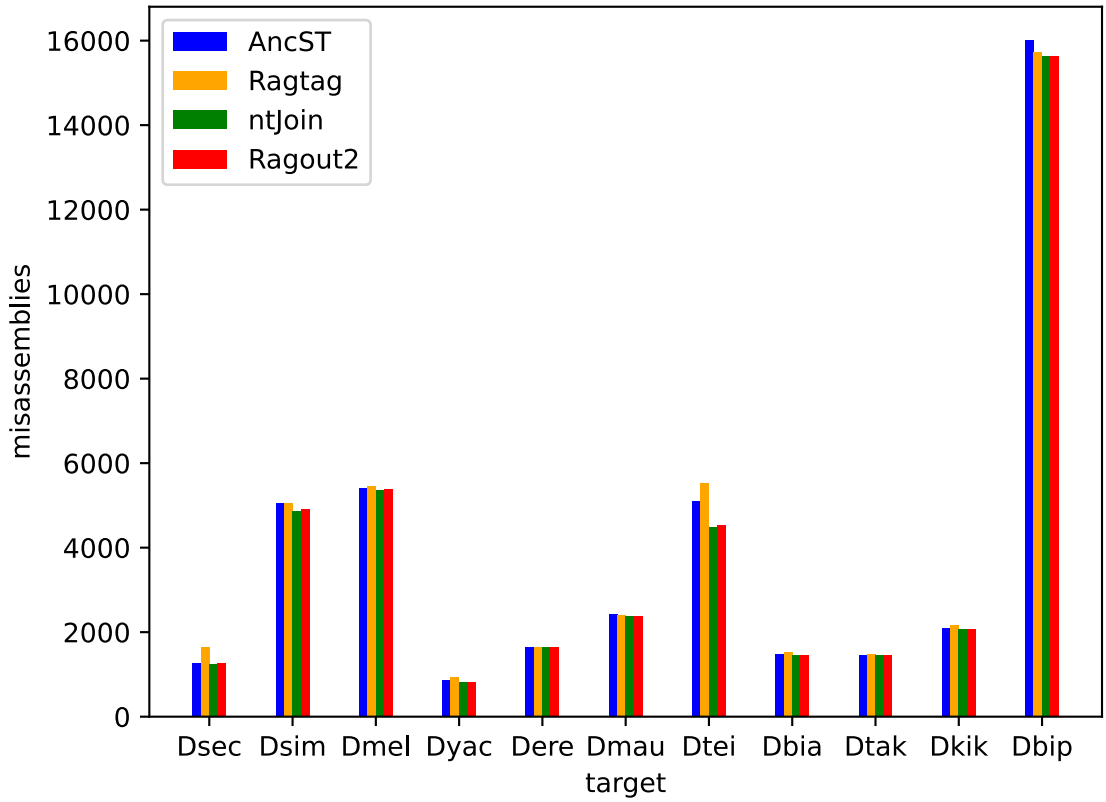

Fig. 14. Number of *misassemblies* as computed by `Quast` for the 11 `Drosophila` newly scaffolded species with a chromosome- scale reference genome on NCBI. Further details in Table 4.

B.3. *Assessment of Reference Chromosome Coverage with New Scaffolds*

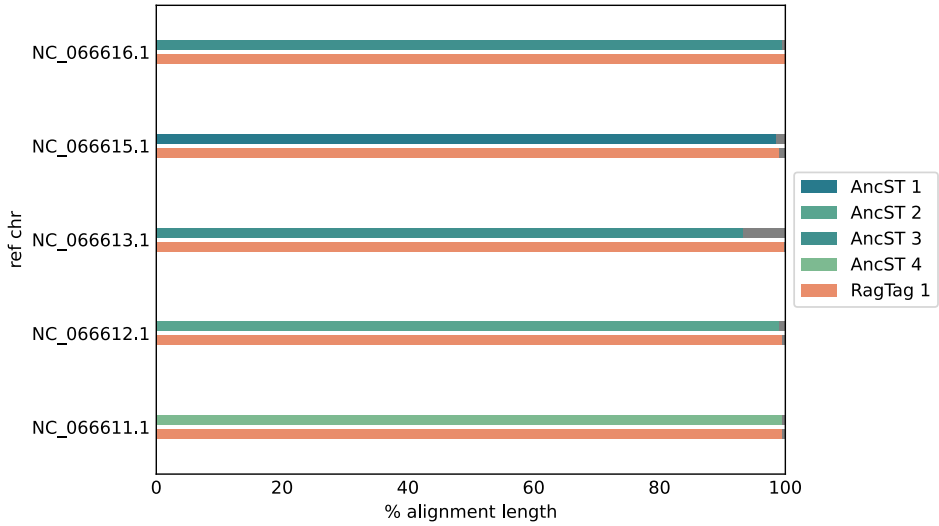

Fig. 15. Shown are all reference chromosomes for the `Drosophila biarmipes` official reference assembly on NCBI on the y-axis. For each reference chromosome, the upper bar displays results computed with the `AncST`-based pipeline and the lower bar the ones from `RagTag`. The bars are stacked according to the coverage of each reference chromosome by new scaffolds from the respective tool. The coverage is estimated by the total alignment length of all `minimap` alignments recorded in the output of `Quast`. Only contigs/scaffolds covering at least a third of the total alignment length are drawn colored while the rest is kept gray. Each color represents a different new scaffold which are indicated in the legend.

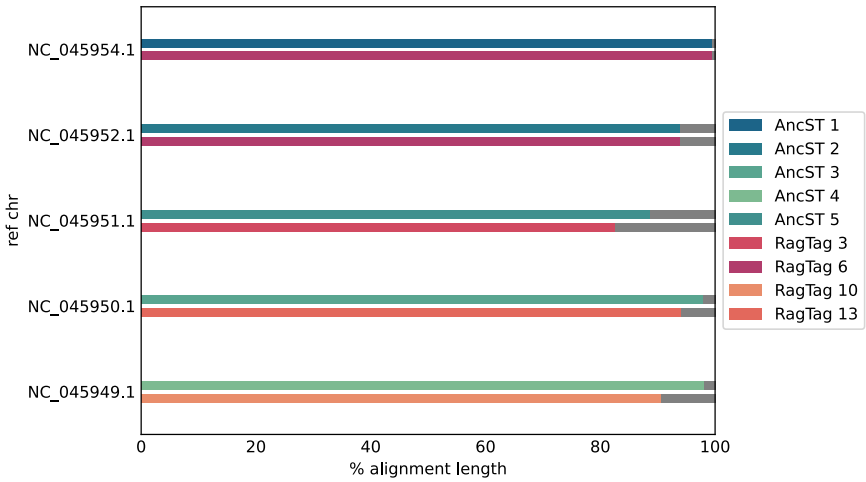

Fig. 16. Assembly: GCA_018904445.1. Shown are all reference chromosomes for a species' official reference assembly on NCBI on the y-axis. For each reference chromosome, the upper bar displays results computed with the `AncST`-based pipeline and the lower bar the ones from `RagTag`. The bars are stacked according to the coverage of each reference chromosome by original contigs or new scaffolds from the respective tool. Each color represents a different original contig or new scaffold which are indicated in the legend. Only contigs/scaffolds covering at least a third of the total alignment length are drawn colored while the rest is kept gray. The coverage is estimated by the total alignment length of all `minimap` alignments recorded in the output of `Quast`. `From A.` means as computed by `AncST` and `From R.` means as computed by `RagTag`.

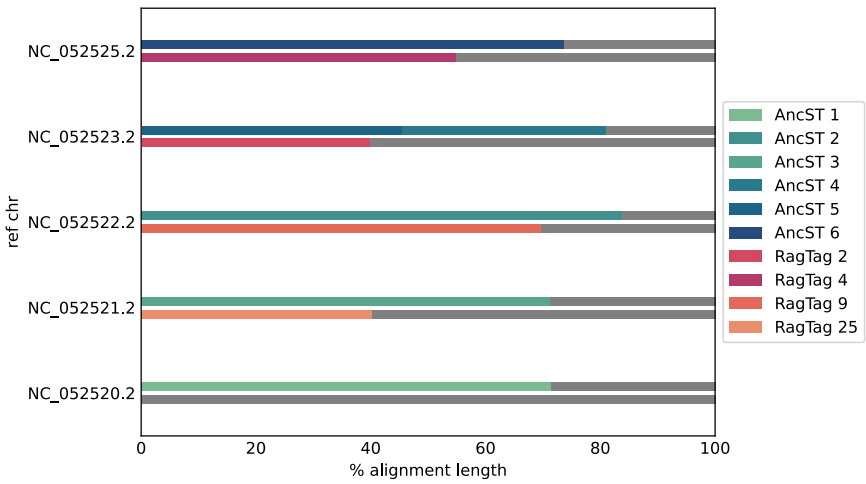

Fig. 17. Assembly: GCA_039725655.1. See caption of Figure 16.

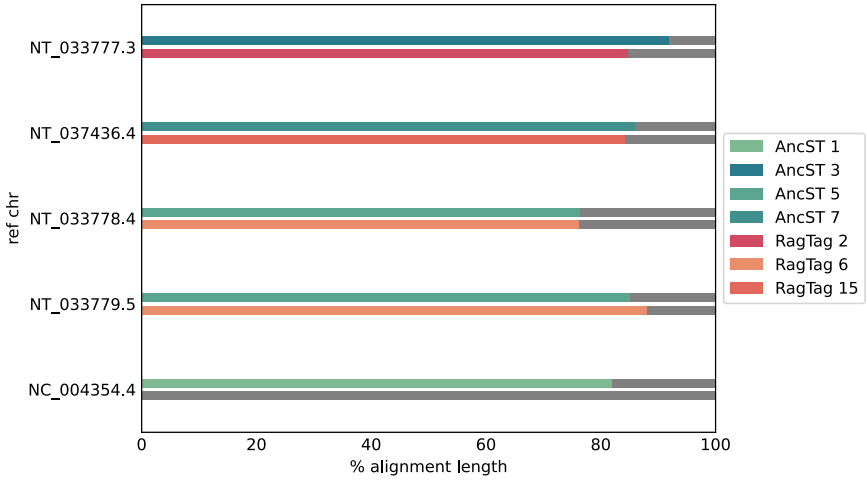

Fig. 18. Assembly: GCA_000778455.1. See caption of Figure 16.

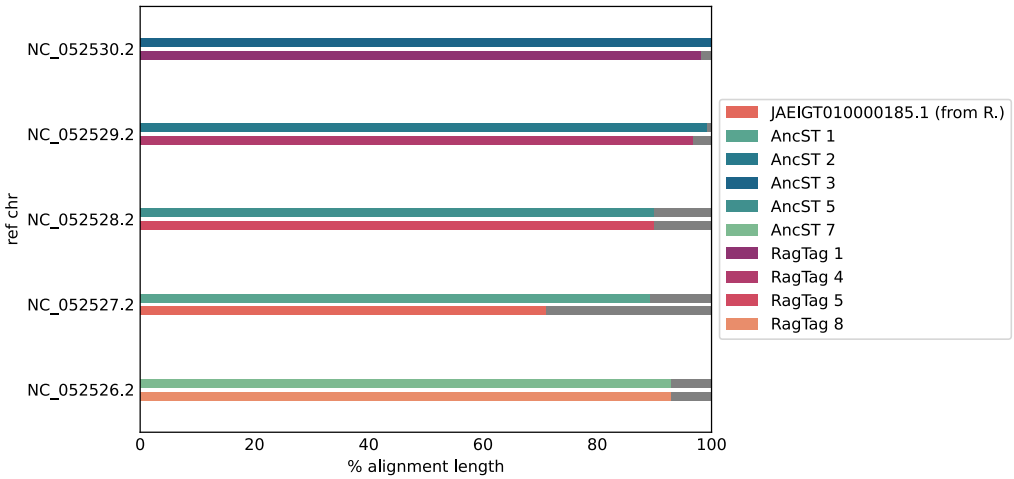

Fig. 19. Assembly: GCA_018904385.1. See caption of Figure 16.

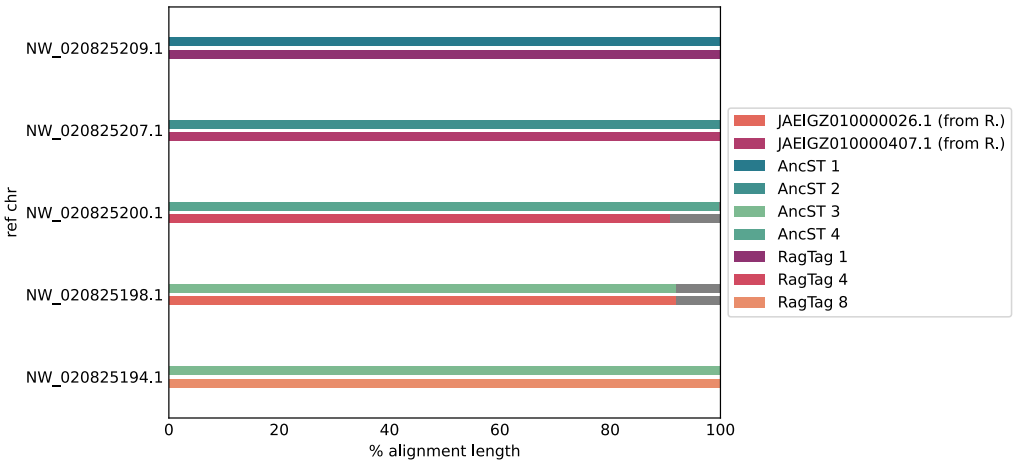

Fig. 20. Assembly: GCA_018904525.1. See caption of Figure 16.

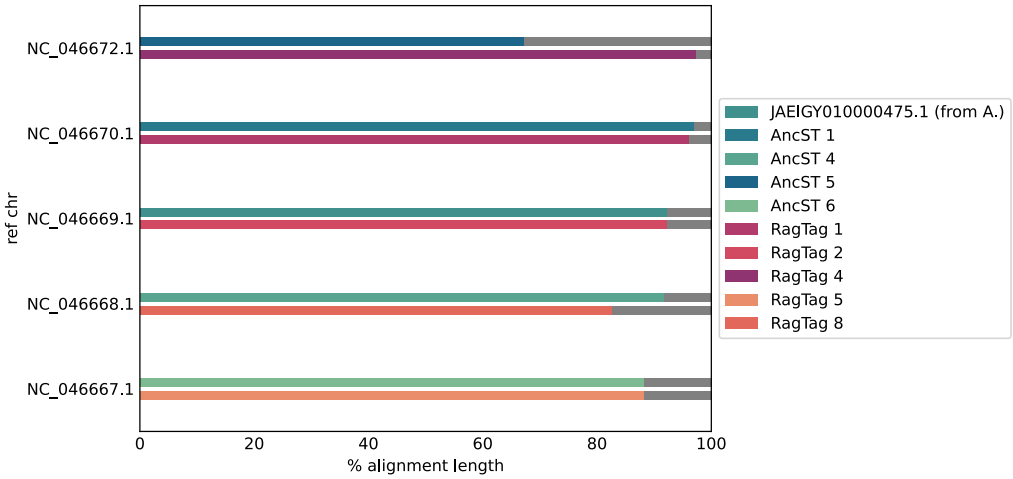

Fig. 21. Assembly: GCA_018904475.1. See caption of Figure 16.

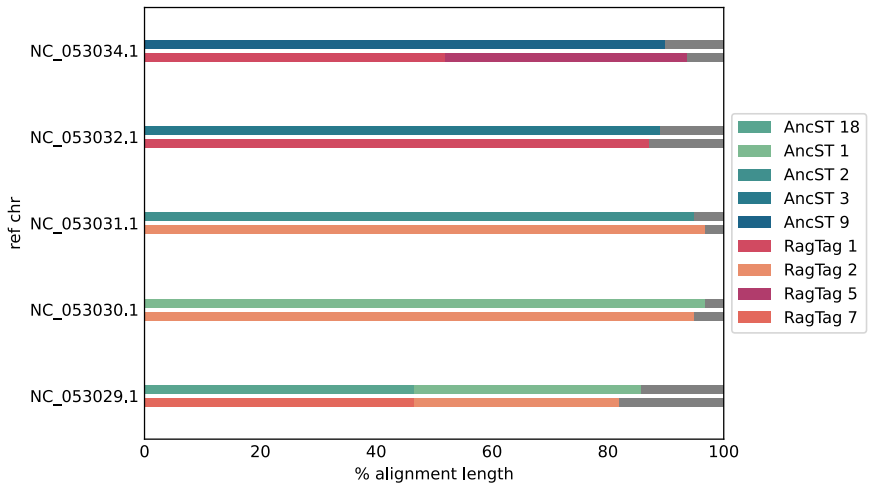

Fig. 22. Assembly: GCA_018903625.1. See caption of Figure 16.

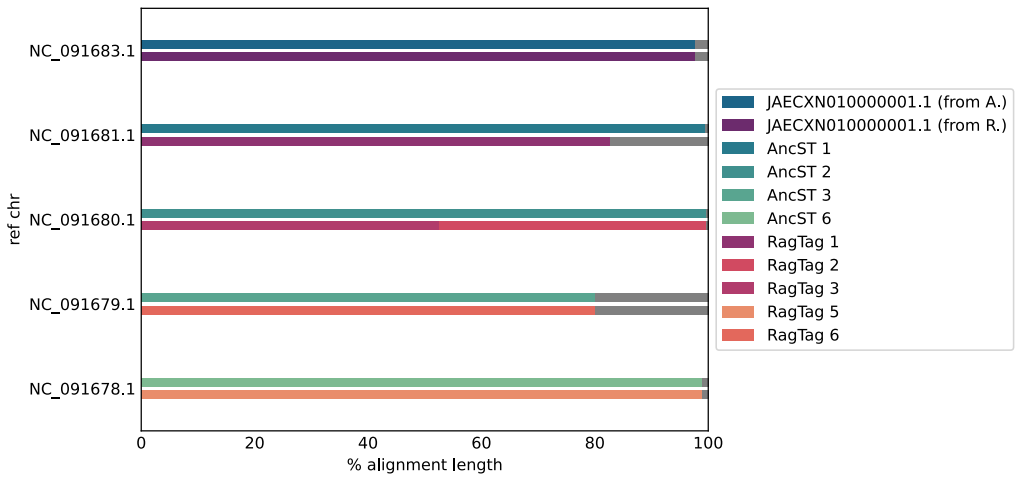

Fig. 23. Assembly: GCA_018152695.1. See caption of Figure 16.

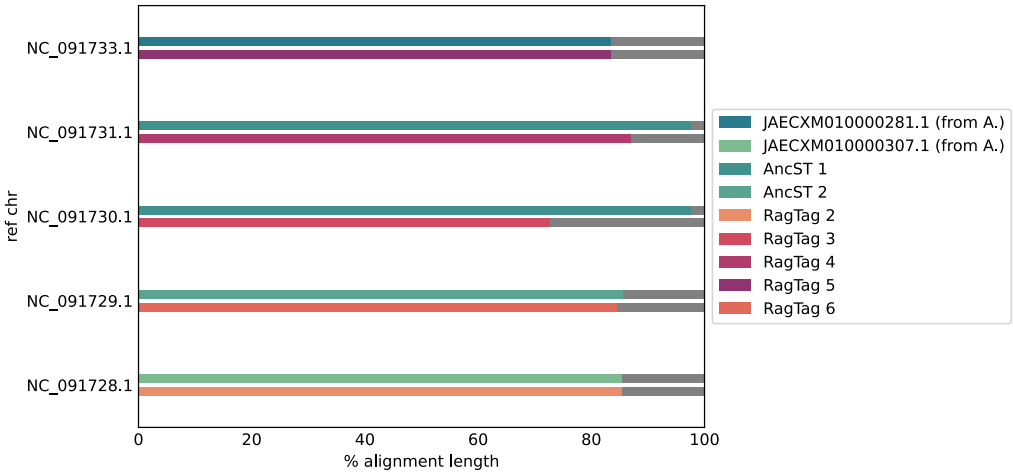

Fig. 24. Assembly: GCA_018152535.1. See caption of Figure 16.

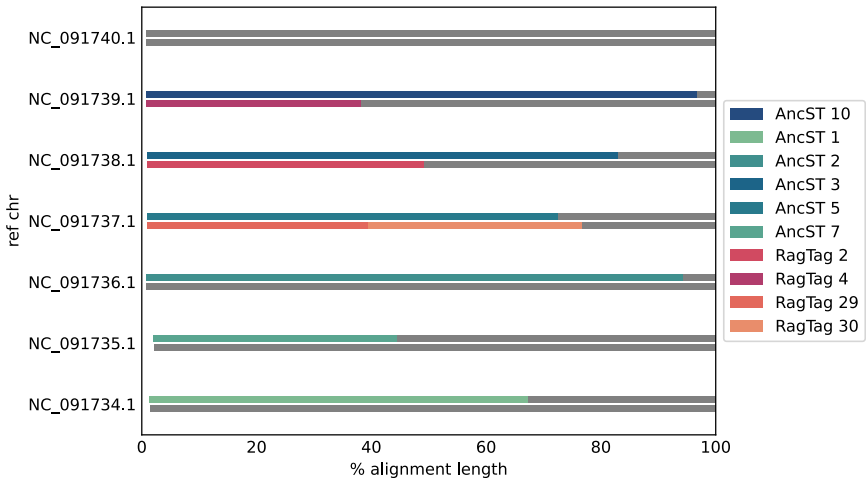

Fig. 25. Assembly: GCA_021223765.1. See caption of Figure 16.

B.4. *Influence of Using Weights for RagTag and ntJoin on auN and Number of Misassemblies for Drosophila Scaffolds*

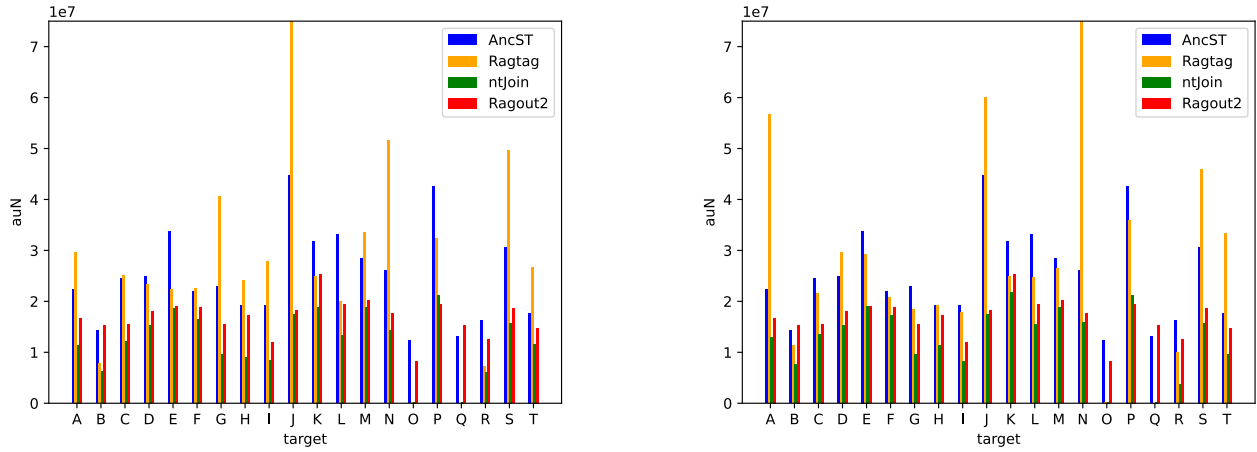

Fig. 26. Left: Same plot as Fig. 13. Right: Same analysis but using `AncST` weights for `RagTag` and `ntJoin`.

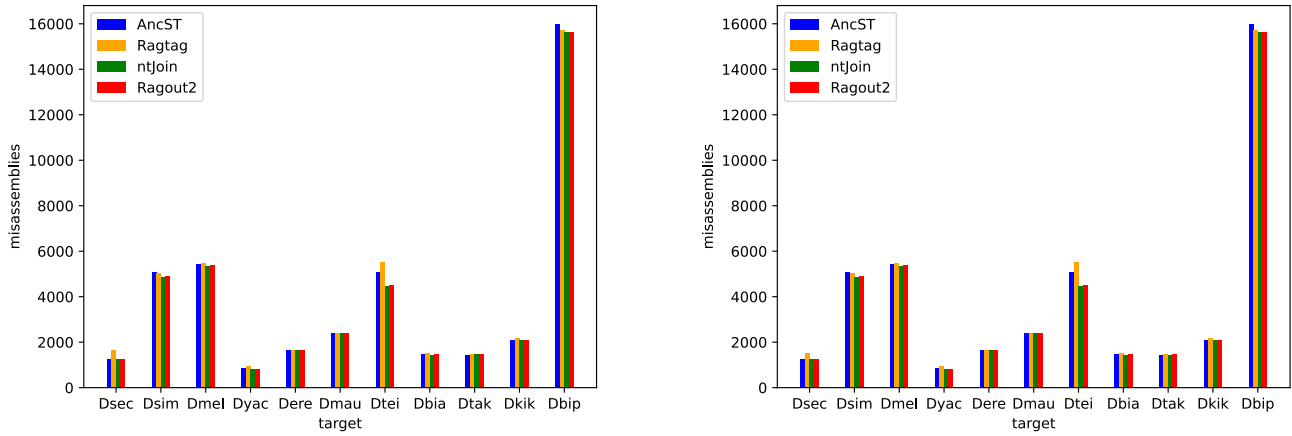

Fig. 27. Left: Same plot as Fig. 14. Right: Same analysis but using `AncST` weights for `RagTag` and `ntJoin`.
(yes, the two plots are different)

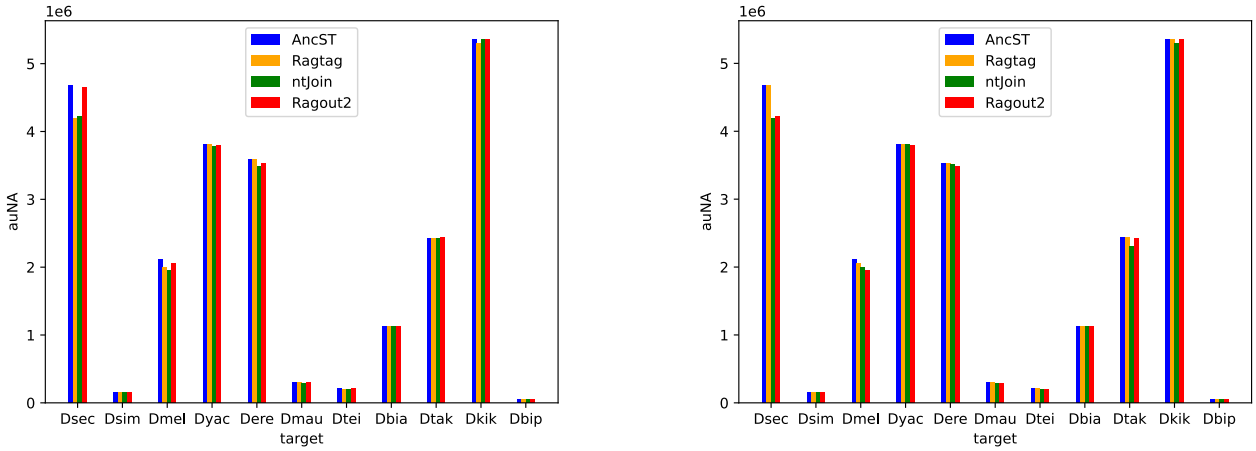

Fig. 28. Left: auNA from `Quast` for each `Drosophila` species with a chromosome-level reference as in main text. Right: Same analysis as in main text but using `AncST` weights for `RagTag` and `ntJoin`.

Appendix C. Details of Tool Versions and Parameters

C.1. *Installations of Scaffolders*

`CSAR` and `Multi-CSAR` were installed from their sources[e]. Default parameters were used with the option `--nuc` to run on the nucleotide level on the two pairs of genomes as well as both reference genomes for the multi-reference version.
`RagTag` was installed from its source[f] and run on the two pairs of genomes. To obtain a multi-reference version we used its `merge` command on the result of the pairwise results. Default parameters were used. For the real test data `RagTag` was run with the `minimap` setting *asm 20* to increase sensitivity as we compare genomes of considerable divergence. `ntJoin 1.15` was installed using the respective `bioconda` recipe and was run with the following options to ensure comparability of the results: `agp=True no_cut=True time=True overlap=False`. `Ragout 2.3` was installed using the respective `bioconda` recipe and was run with the following options: `--refine --solid_scaffolds`. Multiple sequences alignments were generated with `cactus 2.9.9` and used in `maf` format as input for `Ragout2`. We used 58 cores for `AncST` anchor computation while the other tools were run as single-threaded applications. `CSAR` cannot be run multi-threaded although the consumed `user` time indicates use of multiple cores. We decided not to benchmark `RagTag`, `Ragout2` and `ntJoin` with multiple threads as they are so fast that time consumption is negligible already without multithreading. Multiple alignment computation with `cactus`, however, was done using 64 available cores.

[e]https://github.com/ablab-nthu/CSAR, https://github.com/ablab-nthu/Multi-CSAR
[f]https://github.com/malonge/RagTag

C.2. *Details AncST run*

Both the primate and `Drosophila` genomes were processed at an instance of the `AncST` webserver available at https://anchored.bioinf.uni-leipzig.de with default parameters. That means that for k-mer statistics we calculate the k-0 mappability with $k = \frac{\log(genome_size)}{\log(4)}$ with `GenMap`, we use initial candidate window sizes of 300 with a pitch of 50 and take the candidates up to the 42th percentile of lowest aggregate k-mer counts. `macle` is used to calculate the genomes' match complexity and the highest scoring 42th percentile of windows of size 1000 with pitch 50 is used. We did not use the feature to resolve duplicates of a copy low number as additional anchor candidates for this study (for details see webserver documentation and help).

C.3. *Details weights of multi-ref AncST scaffolding*

As a weight for each single-reference scaffolding result we use the total `blast bit score` of all `AncST` alignments times 1000 divided by the length of the respective reference genome (to produce weights in a range of few dozens to few hundreds for human-readability). Accordingly, for each contig pair connected in one of the single-reference scaffolds, the produced weight is added to the corresponding edge in the superimposed graph for multi-reference scaffolding.

References

1. Burton JN, Adey A, Patwardhan RP, Qiu R, Kitzman JO, Shendure J, Chromosome-scale scaffolding of *de novo* genome assemblies based on chromatin interactions, *Nat Biotechnol* **31**(12):1119–1125, 2013. doi:10.1038/nbt.2727.
2. Yamaguchi K, Kadota M, Nishimura O, Ohishi Y, Naito Y, Kuraku S, Technical considerations in Hi-C scaffolding and evaluation of chromosome-scale genome assemblies, *Mol Ecol* **30**(23):5923–5934, 2021. doi:10.1111/mec.16146.
3. Chen KT, Liu CL, Huang SH, Shen HT, Shieh YK, Chiu HT, Lu CL, CSAR: a contig scaffolding tool using algebraic rearrangements, *Bioinformatics* **34**:109–111, 2018. doi:10.1093/bioinformatics/btx543.
4. Alonge M, Soyk S, Ramakrishnan S, Wang X, Goodwin S, Sedlazeck FJ, Lippman ZB, Schatz MC, RaGOO: fast and accurate reference-guided scaffolding of draft genomes, *Genome Biology* **20**:224, 2019. doi:10.1186/s13059-019-1829-6.
5. Coombe L, Nikolić V, Chu J, Birol I, Warren RL, ntJoin: Fast and lightweight assembly-guided scaffolding using minimizer graphs, *Bioinformatics* **36**(12):3885–3887, 2020. doi:10.1093/bioinformatics/btaa253.
6. Kolmogorov M, Armstrong J, Raney BJ, Streeter I, Dunn M, Yang F, Odom D, Flicek P, Keane TM, Thybert D, Paten B, Pham S, Chromosome assembly of large and complex genomes using multiple references, *Genome Res* **28**(11):1720–1732, 2018. doi:10.1101/gr.236273.118.
7. Alonge M, Lebeigle L, Kirsche M, Jenike K, Ou S, Aganezov S, Wang X, Lippman ZB, Schatz MC, Soyk S, Automated assembly scaffolding using RagTag elevates a new tomato system for high-throughput genome editing, *Genome biology* **23**:258, 2022. doi:10.1186/s13059-022-02823-7.
8. Chen KT, Shen HT, Lu CL, Multi-CSAR: a multiple reference-based contig scaffolder using algebraic rearrangements, *BMC Systems Biology* **12**:69–75, 2018. doi:10.1186/s12918-018-0654-y.

9. Kolmogorov V, Blossom V: a new implementation of a minimum cost perfect matching algorithm, *Mathematical Programming Computation* **1**:43–67, 2009. doi: 10.1007/s12532-009-0002-8.
10. Käther K, Lemke S, Stadler PF, Annotation-free identification of potential synteny anchors, Rojas I, Valenzuela O, Rojas Ruiz F, Herrera LJ, Ortuño F (eds.), *International Work-Conference on Bioinformatics and Biomedical Engineering (IWWBIO'23)*, Lect. Notes Comp. Sci., Vol. 13919, Springer, Cham, pp. 217–230, 2023. doi:10.1007/978-3-031-34953-9_17.
11. Käther KK, Remmel A, Lemke S, Stadler PF, Unbiased anchors for reliable genome-wide synteny detection, *Alg Mol Biol* **20**:5, 2025. doi:10.1186/s13015-025-00275-9.
12. Schrinner S, Goel M, Wulfert M, Spohr P, Schneeberger K, Klau GW, Using the longest run subsequence problem within homology-based scaffolding, *Algorithms for Molecular Biology* **16**:11, 2021. doi:10.1186/s13015-021-00191-8.
13. Edmonds J, Path, trees, and flowers, *Canadian J Math* **17**:449–467, 1965. doi:10.4153/CJM-1965-045-4.
14. Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, Treangen TJ, Schatz MC, Delcher AL, Roberts M, Marçais G, Pop M, Yorke J, GAGE: A critical evaluation of genome assemblies and assembly algorithms, *Genome research* **22**(3):557–567, 2012. doi:10.1101/gr.131383.111.
15. Thrash A, Hoffmann F, Perkins A, Toward a more holistic method of genome assembly assessment, *BMC bioinformatics* **21**(Suppl 4):249, 2020. doi:10.1186/s12859-020-3382-4.
16. Wang P, Wang F, A proposed metric set for evaluation of genome assembly quality, *Trends in Genetics* **39**(3):175–186, 2023. doi:10.1016/j.tig.2022.10.005.
17. Narzisi G, Mishra B, Comparing de novo genome assembly: the long and short of it, *PloS one* **6**(4):e19175, 2011. doi:10.1371/journal.pone.0019175.
18. Needleman SB, Wunsch CD, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J Mol Biol* **48**(3):443–453, 1970. doi: 10.1016/0022-2836(70)90057-4.
19. Gurevich A, Saveliev V, Vyahhi N, Tesler G, QUAST: quality assessment tool for genome assemblies, *Bioinformatics* **29**(8):1072–1075, 2013. doi:10.1093/bioinformatics/btt086.
20. Li H, *auN: a New Metric to Measure Assembly Contiguity*, *Tech Rep*, Broad Institute, 2020, URL https://lh3.github.io/2020/04/08/a-new-metric-on-assembly-contiguity.
21. Marçais G, Delcher AL, Phillippy AM, Coston R, Salzberg SL, Zimin A, MUMmer4: A fast and versatile genome alignment system, *PLoS Comp Biol* **14**(1):e1005944, 2018. doi:10.1371/journal.pcbi.1005944.
22. Li K, Xu P, Wang J, Yi X, Jiao Y, Identification of errors in draft genome assemblies at single-nucleotide resolution for quality assessment and improvement, *Nature Communications* **14**(1):6556, 2023. doi:10.1038/s41467-023-42336-w.