Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Subject Section

Integrating Hi-C links with assembly graphs for chromosome-scale assembly

Jay Ghurye [1,2], Arang Rhie [2], Brian P. Walenz [2], Anthony Schmitt [3], Siddarth Selvaraj [3], Mihai Pop [1], Adam M. Phillippy [2,*], and Sergey Koren [2,*]

[1] Department of Computer Science, University of Maryland, College Park, USA
[2] Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, USA and
[3] Arima Genomics, Inc, San Diego, USA

[*] To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Long-read sequencing and novel long-range assays have revolutionized de novo genome assembly by automating the reconstruction of reference-quality genomes. In particular, Hi-C sequencing is becoming an economical method for generating chromosome-scale scaffolds. Despite its increasing popularity, there are limited open-source tools available. Errors, particularly inversions and fusions across chromosomes, remain higher than alternate scaffolding technologies.
Results: We present a novel open-source Hi-C scaffolder that does not require an *a priori* estimate of chromosome number and minimizes errors by scaffolding with the assistance of an assembly graph. We demonstrate higher accuracy than the state-of-the-art methods across a variety of Hi-C library preparations and input assembly sizes.
Availability and Implementation: The Python and C++ code for our method is openly available at https://github.com/machinegun/SALSA
Contact: sergey.koren@nih.gov, adam.phillippy@nih.gov
Supplementary information: Not available online.

1 Introduction

Genome assembly is the process of reconstructing a complete genome sequence from significantly shorter sequencing reads. Most genome projects rely on whole genome shotgun sequencing which yields an oversampling of each genomic locus. Reads originating from the same locus are identified using assembly software, which can use these overlaps to reconstruct the genome sequence (Nagarajan and Pop, 2013; Miller *et al.*, 2010). Most approaches are based on either a de Bruijn (Pevzner *et al.*, 2001) or a string graph (Myers, 2005) formulation. Repetitive sequences exceeding the sequencing read length (Nagarajan and Pop, 2009) introduce ambiguity and prevent complete reconstruction. Unambiguous reconstructions of the sequence are output as "unitigs" (or often "contigs"). Ambiguous reconstructions are output as edges linking unitigs. Scaffolding utilizes long-range linking information such as BAC or fosmid clones (Venter *et al.*, 1996; Gnerre *et al.*, 2011), optical maps (Schwartz *et al.*, 1993; Dong *et al.*, 2013; Shelton *et al.*, 2015), linked reads (Zheng *et al.*, 2016; Weisenfeld *et al.*, 2017; Yeo *et al.*, 2017), or chromosomal conformation capture (Simonis *et al.*, 2006) to order and orient unitigs. If the linking information spans large distances on the chromosome, the resulting scaffolds can span entire chromosomes or chromosome arms.

Hi-C is a sequencing-based assay originally designed to interrogate the 3D structure of the genome inside a cell nucleus by measuring the contact frequency between all pairs of loci in the genome (Lieberman-Aiden *et al.*, 2009). The contact frequency between a pair of loci strongly correlates with the one-dimensional distance between them. Hi-C data can provide linkage information across a variety of length scales, spanning tens of megabases. As a result, Hi-C data can be used for genome scaffolding. Shortly after its introduction, Hi-C was used to generate chromosome-scale scaffolds (Burton *et al.*, 2013; Kaplan and Dekker, 2013; Marie-Nelly *et al.*, 2014; Bickhart *et al.*, 2017; Dudchenko *et al.*, 2017).

LACHESIS (Burton *et al.*, 2013) is an early method for Hi-C scaffolding which first clusters unitigs into a user-specified number of chromosome groups and then orients and orders the unitigs in each group independently to generate scaffolds. Thus, the scaffolds inherit any assembly errors present in the unitigs. The original SALSA (Ghurye *et al.*, 2017) method first corrects the input assembly, using a lack of Hi-C coverage as evidence of error. It then orients and orders the corrected unitigs to generate scaffolds. However, SALSA requires manual parameter tuning for each dataset which affects the contiguity and correctness of the final scaffolds. Recently, the 3D-DNA (Dudchenko *et al.*, 2017) method was introduced and demonstrated on a draft assembly of the *Aedes aegypti* genome. 3D-DNA also corrects the errors in the input assembly and then iteratively orients and orders unitigs into a single megascaffold. This megascaffold is then broken into a user-specified number of chromosomes, identifying chromosomal ends based the on Hi-C contact map.

There are several shortcomings common across currently available tools. They require the user to specify the number of chromosomes *a priori*. This can be challenging in novel genomes where no karyotype is available. An incorrect guess often leads to mis-joins that fuse chromosomes. They are also sensitive to input assembly contiguity and Hi-C library variations and require tuning of parameters for each dataset. Inversions are common when the input unitigs are short, as orientation is determined by maximizing the interaction frequency between unitig ends across all possible orientations (Burton *et al.*, 2013). When unitigs are long, there are few interactions spanning the full length of the unitig, making the true orientation apparent from the higher weight of links. However, in the case of short unitigs, there are interactions spanning the full length of the unitig, making the true orientation have a similar weight to incorrect orientations. Biological factors, such as topologically associated domains (TADs) also confound this analysis (Dixon *et al.*, 2012).

In this work, we introduce SALSA2 – an open source software that combines Hi-C linkage information with the ambiguous-edge information from a genome assembly graph to better resolve unitig orientations. We also propose a novel stopping condition, which does not require an *a priori* estimate of chromosome count, as it naturally stops when the Hi-C information is exhausted. We show that SALSA2 has fewer orientation, ordering, and chimeric errors across a wide range of assembly contiguities. We also demonstrate robustness to different Hi-C libraries with varying intra-chromosomal contact frequencies. When compared to 3D-DNA, SALSA2 generates more accurate scaffolds across all conditions tested. To our knowledge, this is the first method to leverage assembly graph information for scaffolding Hi-C data.

2 Methods

Figure 1(A) shows the overview of the SALSA2 pipeline. A draft assembly is generated from long reads such as Pacific Biosciences (Eid *et al.*, 2009) or Oxford Nanopore (Jain *et al.*, 2016). SALSA2 requires the unitig sequences and, optionally, a GFA-format graph (Li, 2016) representing the ambiguous reconstructions. Hi-C reads are aligned to the unitig sequences, and unitigs are optionally split in regions lacking Hi-C coverage. A hybrid scaffold graph is constructed using both ambiguous edges from the GFA and edges from the Hi-C reads, scoring edges according to a "best buddy" scheme. Scaffolds are iteratively constructed from this graph using a greedy weighted maximum matching. A mis-join detection step is performed after each iteration to check if any of the joins made during this round are incorrect. Incorrect joins are broken and the edges blacklisted during subsequent iterations. This process continues until the majority of joins made in the prior iteration are incorrect. This provides a natural stopping condition, when accurate Hi-C links have been exhausted. Below, we describe each of the steps in detail.

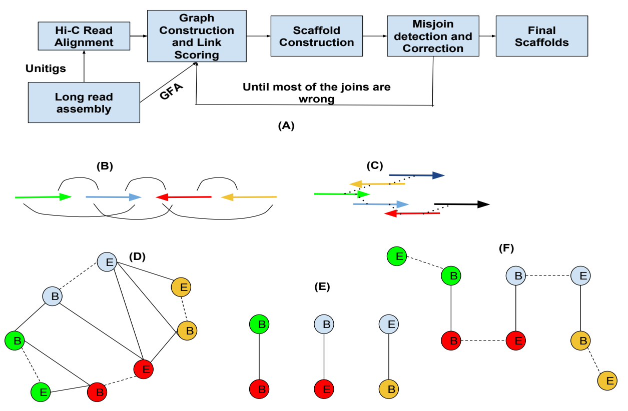

Fig. 1. (A) Overview of the SALSA2 scaffolding algorithm. (B) Linkage information obtained from the alignment of Hi-C reads to the assembly. (C) The assembly graph obtained from the assembler. (D) A hybrid scaffold graph constructed from the links obtained from the Hi-C read alignments and the overlap graph. Solid edges indicate the linkages between different unitigs and dotted edges indicate the links between the ends of the same unitig. (E) Maximal matching obtained from the graph using a greedy weighted maximum matching algorithm. (F) Edges between the ends of same unitigs are added back to the matching.

2.1 Read alignment

Hi-C paired end reads are aligned to unitigs using the BWA aligner (Li and Durbin, 2009)(parameters: -t 12 -B 8) as single end reads. Reads which align across ligation junctions are chimeric and are trimmed to retain only the start of the read which aligns prior to the ligation junction. After filtering the chimeric reads, the pairing information is restored. Any PCR duplicates in the paired-end alignments are removed using Picard tools (Wysoker *et al.*, 2013). Read pairs aligned to different unitigs are used to construct the initial scaffold graph. The suggested mapping pipeline is available at http://github.com/ArimaGenomics/mapping_pipeline.

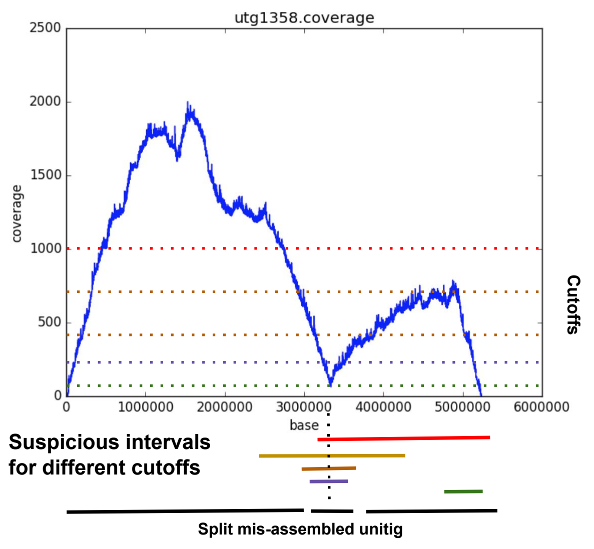

Fig. 2. Example of the mis-assembly detection algorithm in SALSA2. The plot shows the position on x-axis and the physical coverage on the y-axis. The dotted horizontal lines show the different thresholds tested to find low physical coverage intervals. The lines at the bottom show the suspicious intervals identified by the algorithm. The dotted line through the intervals shows the maximal clique. The smallest interval (purple) in the clique is identified as mis-assembly and the unitig is broken in three parts at its boundaries.

2.2 Unitig correction

As any assembly is likely to contain mis-assembled sequences, SALSA2 uses the physical coverage of Hi-C pairs to identify suspicious regions and break the sequence at the likely point of mis-assembly. We define the physical coverage of a Hi-C read pair as the region on the unitig spanned by the start of the leftmost fragment and the end of the rightmost fragment. A drop in physical coverage indicates a likely assembly error. We extend the mis-assembly detection algorithm from SALSA which split a unitig when a fixed minimum coverage threshold was not met. A drawback of this approach is that coverage can vary, both due to sequencing depth and variation in Hi-C link density.

Figure 2 sketches the new unitig correction algorithm implemented in SALSA2. Instead of a single coverage threshold, a set of suspicious intervals is found with a sweep of thresholds. Using the collection of intervals as an interval graph, we find the maximal clique. This can be done in $O(NlogN)$ time, where N is the number of intervals. For any clique of a minimum size, the region between the start and end of the smallest interval in the clique is flagged as a mis-assembly and the unitig is split into three pieces — the sequence to the left of the region, the junction region itself, and the sequence to the right of the region.

2.3 Assembly graph construction

For our experiments, we use the unitig assembly graph produced by Canu (Koren *et al.*, 2017) (Figure 1(C)), as this is the more conservative graph output. SALSA2 requires only a GFA format (Li, 2016) representation of the assembly. Since most long read genome assemblers such as FALCON (Chin *et al.*, 2016), miniasm (Li, 2016), Canu (Koren *et al.*, 2017), and Flye (Kolmogorov *et al.*, 2018) provide assembly graphs in GFA format, their output is compatible with SALSA2 for scaffolding.

2.4 Scaffold graph construction

The scaffold graph is defined as $G(V, E)$, where nodes V are the ends of unitigs and edges E are derived from the Hi-C read mapping (Figure 1B). The idea of using unitig ends as nodes is similar to that used by the string graph formulation (Myers, 2005).

Modeling each unitig as two nodes allows a pair of unitigs to have multiple edges in any of the four possible orientations (forward-forward, forward-reverse, reverse-forward, and reverse-reverse). The graph then contains two edge types - one explicitly connects two different unitigs based on Hi-C data, while the other implicitly connects the two ends of the same unitig.

We normalize the Hi-C read counts by the frequency of restriction enzyme cut sites in each unitig. This normalization reduces the bias in the number of shared read pairs due to the unitig length as the number of Hi-C reads sequenced from a particular region are proportional to the number of restriction enzyme cut sites in that region. For each unitig, we denote the number of times a cut site appears as $C(V)$. We define edges weights of G as:

$$W(u, v) = \frac{N(u, v)}{C(u) + C(v)}$$

where $N(u, v)$ is the number of Hi-C read pairs mapped to the ends of the unitigs u and v.

We observed that the globally highest edge weight does not always capture the correct orientation and ordering information due to variations in Hi-C interaction frequencies within a genome. To address this, we defined a modified edge ratio, similar to the one described in (Dudchenko *et al.*, 2017), which captures the relative weights of all the neighboring edges for a particular node.

The best buddy weight $BB(u, v)$ is the weight $W(u, v)$ divided by the maximal weight of any edge incident upon nodes u or v, excluding the (u, v) edge itself. Computing best buddy weight naively would take $O(|E|^2)$ time. This is computationally prohibitive since the graph, G, is usually dense. If the maximum weighted edge incident on each node is stored with the node, the running time for the computation becomes $O(|E|)$. We retain only edges where $BB(u, v) > 1$. This keeps only the edges which are the best incident edge on both u and v. Once used, the edges are removed from subsequent iterations. Thus, the most confident edges are used first but initially low scoring edges can become best in subsequent iterations.

For the assembly graph, we define a similar ratio. Since the edge weights are optional in the GFA specification and do not directly relate to the proximity of two unitigs on the chromosome, we use the graph topology to establish this relationship. Let $\bar{u}$ denote the reverse complement of the unitig u. Let $\sigma(u, v)$ denote the length of shortest path between u and v. For each edge (u, v) in the scaffold graph, we find the shortest path between unitigs u and v in every possible orientation, that is, $\sigma(u, v)$, $\sigma(u, \bar{v})$, $\sigma(\bar{u}, v)$ and $\sigma(\bar{u}, \bar{v})$. With this, the score for a pair of unitigs is defined as follows:

$$Score(u, v) = \frac{\min\limits_{x' \in \{u,\bar{u}\}-\{x\}, y' \in \{v,\bar{v}\}-\{y\}} \sigma(x', y')}{\min\limits_{x \in \{u,\bar{u}\}, y \in \{v,\bar{v}\}} \sigma(x, y)}$$

where x and y are the orientations in which u and v are connected by a shortest path in the assembly graph. Essentially, $Score(u, v)$ is the ratio of the length of the second shortest path to the length of the shortest path in all possible orientations. Once again, we retain edges where $Score(u, v) > 1$. If the orientation implied by the assembly graph differs from the orientation implied by the Hi-C data, we remove the Hi-C edge and retain the assembly graph edge (Figure 1D). Computing the score graph requires $|E|$ shortest path queries, yielding total runtime of $O(|E| * (|V| + |E|))$ since we do not use the edge weights.

2.5 Unitig layout

Once we have the hybrid graph, we lay out the unitigs to generate scaffolds. Since there are implicit edges in the graph G between the beginning and end of each unitig, the problem of computing a scaffold layout can be modeled as finding a weighted maximum matching in a general graph, with edge weights being our ratio weights. If we find the weighted maximum matching of the non-implicit edges (that is, edges between different unitigs) in the graph, adding the implicit edges to this matching would yield a complete traversal. However, adding implicit edges to the matching can introduce a cycle. Such cycles are removed by removing the lowest weight non-implicit edge. Computing a maximal matching takes $O(|E||V|^2)$ time (Edmonds, 1965). We iteratively find a maximum matching in the graph by removing nodes found in the previous iteration. Using the optimal maximum matching algorithm this would take $O(|E||V|^3)$ time, which would be extremely slow for large graphs. Instead, we use a greedy maximal matching algorithm which is guaranteed to find a matching within 1/2-approximation of the optimum (Poloczek and Szegedy, 2012). The greedy matching algorithm takes $O(|E|)$ time, thereby making the total runtime $O(|V||E|)$. The algorithm for unitig layout is sketched in Algorithm 1. Figure 1(D - F) show the layout on an example graph.

Junctions in the graph can prevent some nodes from being included in larger scaffolds. At a junction, only one of the possible unitigs can be included in the matching, demoting the other unitigs at the junction to alternate matchings. To account for this, we try to insert unitigs from small scaffolds (less than five unitigs) into all possible positions in the large scaffolds in all possible orientations. A unitig is inserted into the scaffold at the position and orientation which maximizes the sum of edge weights between it and all adjacent unitigs at that location. If the gain in the sum of edge weights is not sufficient, the unitig is not inserted into any of the existing scaffolds but can be scaffolded in subsequent iterations.

Algorithm 1 Unitig Layout Algorithm

```
$E$ : Edges sorted by the best buddy weight
$M$ : Set to store maximal matchings
$G$ : The scaffold graph
while all nodes in $G$ are not matched do
    $M^* = \{\}$
    for $e \in E$ sorted by best buddy weights do
        if $e$ can be added to $M^*$ then
            $M^* = M^* \cup e$
        end if
    end for
    $M = M \cup M^*$
    Remove nodes and edges which are part of $M^*$ from G
end while
```

Algorithm 2 Misjoin detection and correction algorithm

```
$Cov$ : Physical coverage array for a window size $w$ around a scaffold join at position $p$ on a scaffold
$A$ : Auxiliary array
$I$ : Maximum sum subarray intervals
for $\delta \in \{$min_coverage, max_coverage$\}$ do
    if $Cov[i] \leq \delta$ then
        $A[i] = 1$
    else
        $A[i] = -1$
    end if
    $s_\delta, e_\delta =$ maximum_sum_subarray$(A)$
    $I = I \cup \{s_\delta, e_\delta\}$
end for
$s, e =$maximal_clique_interval$(I)$
if $p \in \{s, e\}$ then
    Break the scaffold at position $p$
end if
```

2.6 Iterative mis-join correction

Since the unitig layout is greedy, it can introduce errors by selecting a false Hi-C link which was not eliminated by our ratio scoring. These errors propagate downstream, causing large chimeric scaffolds and chromosomal fusions. We examine each join made within all the scaffolds in the last iteration for correctness. Any join with low spanning Hi-C support relative to the rest of the scaffold is broken and the links are blacklisted for further iterations.

We compute the physical coverage spanned by all read pairs aligned in a window of size w around each join. For each window, w, we create an auxiliary array, which stores -1 at position i if the physical coverage is greater than some cutoff δ and 1, otherwise. We then find the maximum sum subarray in this auxiliary array, since it captures the longest stretch of low physical coverage. If the position being tested for a mis-join lies within the region spanned by the maximal clique generated with the maximum sum subarray intervals for different cutoffs (Figure 2), the join is marked as incorrect. The physical coverage can be computed in $O(w + N)$ time, where N is the number of read pairs aligned in window w. The maximum sum subarray computation takes $O(w)$ time. If K is the number of cutoffs(δ) tested for the suspicious join finding, then the total runtime of mis-assembly detection becomes $O(K(N + 2 * w))$. The parameter K controls the specificity of the mis-assembly detection, thereby avoiding false positives. The algorithm for mis-join detection is sketched in Algorithm 2. When the majority of joins made in a particular iteration are flagged as incorrect by the algorithm, SASLA2 stops scaffolding and reports the scaffolds generated in the penultimate iteration as the final result.

3 Results

3.1 Dataset description

We created artificial assemblies, each containing unitigs of same size, by splitting the GRCh38 (Schneider *et al.*, 2017) reference into fixed sized unitigs of 200 to 900 kbp. This gave us eight assemblies. The assembly graph for each input is built by adding edges for any adjacent unitigs in the genome.

For real data, we use the recently published NA12878 human dataset sequenced with Oxford Nanopore (Jain *et al.*, 2017) and assembled with Canu (Koren *et al.*, 2017). We use a Hi-C library from Arima Genomics (Arima Genomics, San Diego, CA) sequenced to 40x coverage (SRX3651893). We compare results with the original SALSA, SALSA2 without the assembly graph input, and 3D-DNA. We did not compare our results with LACHESIS because it is no longer supported and is outperformed by 3D-DNA (Dudchenko *et al.*, 2017). SALSA2 was run using default parameters, with the exception of graph incorporation, as listed. For 3D-DNA, alignments were generated using the Juicer alignment pipeline (Durand *et al.*, 2016b) with defaults (-m haploid -t 15000 -s 2), except for mis-assembly detection, as listed. The chromosome number was set to 23 for all experiments. A genome size of 3.2 Gbp was used for contiguity statistics for all assemblies.

For evaluation, we also used the GRCh38 reference to define a set of true and false links from the Hi-C graph. We mapped the assembly to the reference with MUMmer3.23 (nucmer -c 500 -l 20) (Kurtz *et al.*, 2004) and generated a tiling using MUMmer's show-tiling utility. For this "true link" dataset, any link joining unitigs in the same chromosome in the correct orientation was marked as true. This also gives the true unitig position, orientation, and chromosome assignment. We masked sequences in GRCh38 which matched known structural variants from a previous assembly of NA12878 (Pendleton *et al.*, 2015) to avoid counting true variations as scaffolding errors.

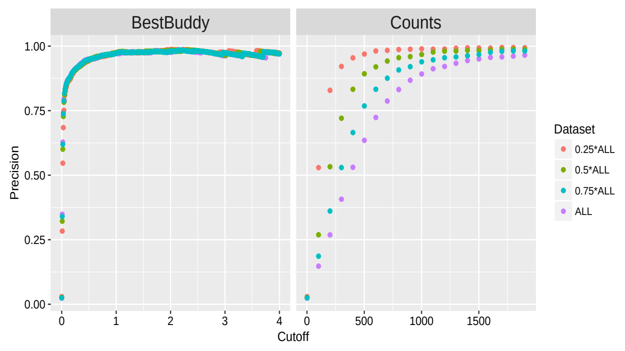

Fig. 3. Precision at different cutoffs for Hi-C links. The plot on the left shows the curve for the SALSA2 best buddy weight cutoffs and the plot on the right shows the curve for a fixed Hi-C pair count cutoff, used in SALSA1, across changing coverage.

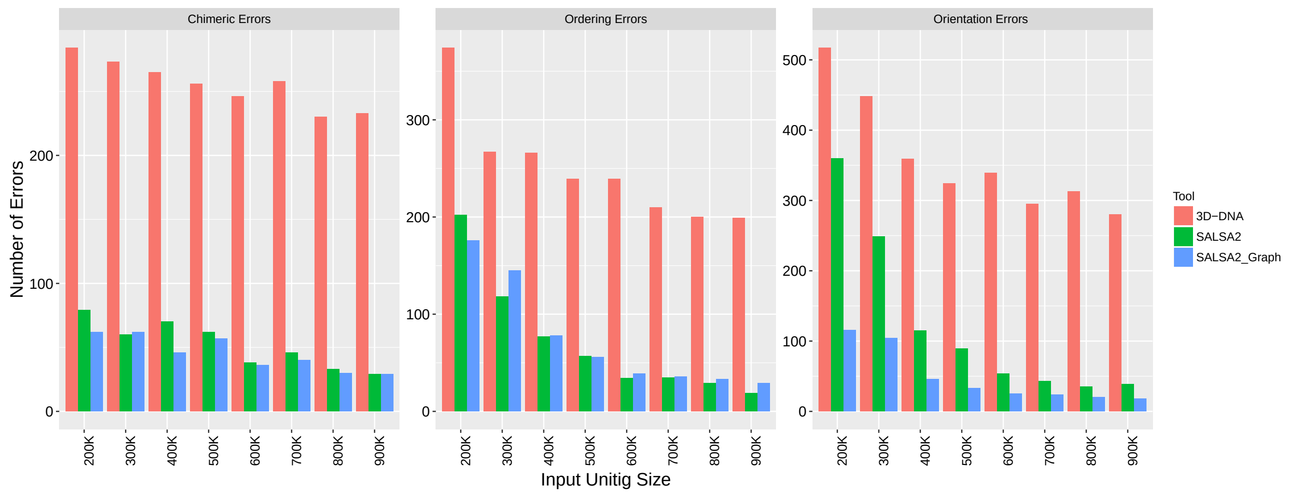

Fig. 4. Comparison of orientation, ordering, and chimeric errors in the scaffolds produced by SALSA2 and 3D-DNA on the simulated data. As expected, the number of errors for all error types decrease with increasing input unitig size. Incorporating the assembly graph reduces error across all categories and most assembly sizes, with the largest decrease seen in orientation errors. SALSA2 utilizing the graph has 2-4 fold fewer errors than 3D-DNA.

3.2 Scoring effectiveness

For correct scaffolding, we want to filter false edges and retain only the correct linkage information between pairs of unitigs. Our previous algorithm used a fixed, user-defined minimum for edges connecting a pair of unitigs. The drawback of a fixed cutoff is that it cannot handle variations in coverage within the assembly and varies between any pair of sequencing datasets. To compare the scoring methods, we down-sample the alignments into three different sets with 0.25, 0.5 and 0.75 of the original coverage and computed the precision of filtering based on the ratio score and a fixed threshold. The precision remained almost constant for the ratio cutoff on all datasets, whereas the precision changes rapidly for different coverages and a fixed threshold (Figure 3).

3.3 Evaluation on simulated unitigs

3.3.1 Assembly correction

We simulated assembly error by randomly joining 200 pairs of unitigs from each simulated assembly. All erroneous joins were made between unitigs that are more than 10 Mbp apart or were assigned to different chromosomes in the reference. The remaining unitigs were unaltered. We then aligned the Arima-HiC data and ran our assembly correction algorithm. When the algorithm marked a mis-join within 20 kbp of a true error we called it a true positive, otherwise we called it a false positive. Any unmarked error was called a false negative. The average sensitivity over all simulated assemblies was 77.62% and the specificity was 86.13%. The sensitivity was highest for larger unitigs (50% for 200 kbp versus >90% for untigs greater than 500 kbp) implying that our algorithm is able to accurately identify errors in large unitigs, which can have a negative impact on the final scaffolds if not corrected.

3.3.2 Scaffold mis-join validation

As before, we simulated erroneous scaffolds by joining unitigs which were not within 10 Mbp in the reference or were assigned to different chromosomes. Rather than pairs of unitigs, each erroneous scaffold joined 10 unitigs and we generated 200 such erroneous scaffolds. The remaining unitigs were correctly scaffolded (ten unitigs per scaffold) based on their location in the reference. The average sensitivity was 68.89% and specificity was 100% (no correct scaffolds were broken). Most of the un-flagged joins occurred near the ends of scaffolds and could be

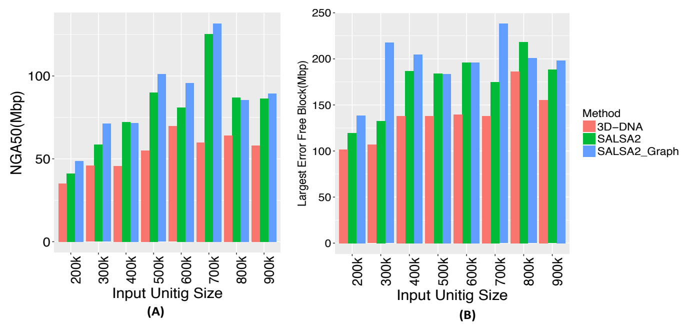

Fig. 5. (A) NGA50 statistic for different input unitig sizes and (B) The length of longest error-free block for different input unitig sizes. Once again, the assembly graph typically increases both the NGA50 and the largest correct block.

captured by decreasing the window size. Similar to assembly correction, we observed that sensitivity was highest with larger input unitigs. This evaluation highlights the accuracy of the mis-join detection algorithm to avoid over-scaffolding and provide a suitable stopping condition.

3.3.3 Scaffold accuracy

We evaluated scaffolds across three categories of error: orientation, order, and chimera. An orientation error occurs whenever the orientation of a unitig in a scaffold differs from that of the scaffold in the reference. An ordering error occurs when a set of three unitigs adjacent in a scaffold have non-monotonic coordinates in the reference. A chimera error occurs when any pair of unitigs adjacent in a scaffold align to different chromosomes in the reference. We broke the assembly at these errors and computed corrected scaffold lengths and NGA50 (analogous to the NGA50 defined by Salzberg et al. (Salzberg *et al.*, 2012)). This statistic corrects for large but incorrect scaffolds which have a high NG50 but are not useful for downstream analysis because of errors.

Hi-C scaffolding errors, particularly orientation errors, increased with decreasing assembly contiguity. We evaluated scaffolding methods across a variety of simulated unitig sizes. Figure 4 shows the comparison of these errors for 3D-DNA, SALSA2 without the assembly graph, and SALSA2 with the graph. SALSA2 produced fewer errors than 3D-DNA across

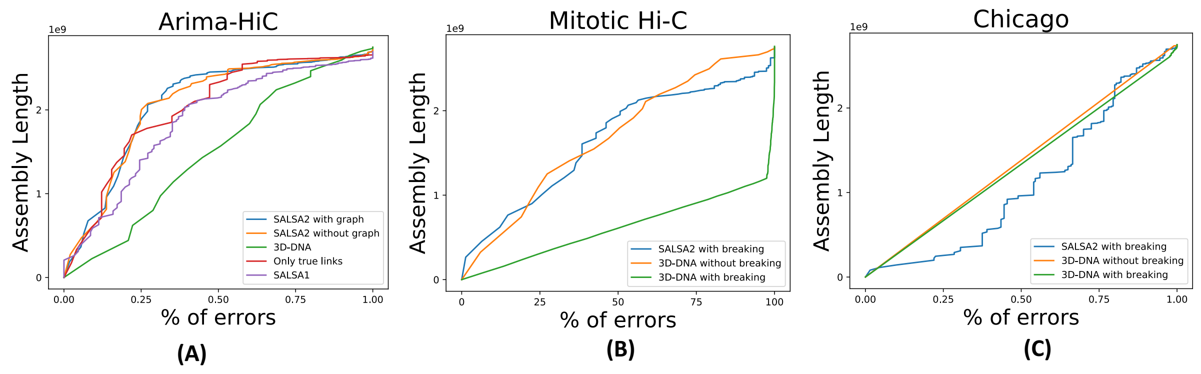

Fig. 6. Feature Response Curve for (A) assemblies obtained from unitigs as input (B) assemblies obtained from mitotic Hi-C data and (C) assemblies obtained using Dovetail Chicago data. The best assemblies lie near the top left of the plot, with the largest area under the curve. The FRC for 3D-DNA scaffolds with Chicago input is a straight line because 3D-DNA generated a single 2.7 Gbp super-scaffold which contained the majority of the genome sequence.

Dataset	Method	NG50(Mbp)	NGA50(Mbp)	Longest Chunk (Mbp)	Orientation Errors	Ordering Errors	Chimeric Errors
Arima-HiC	SALSA2 true links	83.31	79.48	172.19	78	101	0
	SALSA2 w graph	125.34	57.20	165.11	156	289	142
	SALSA2 wo graph	101.96	56.84	155.68	168	302	152
	3D-DNA	137.88	28.61	130.88	233	405	178
	SALSA1	19.09	14.81	73.14	99	176	96
Mitotic Hi-C	SALSA2 w graph	69.23	26.46	145.53	117	98	58
	3D-DNA w correction	16.34	0.064	0.96	12017	11687	7217
	3D-DNA wo correction	141.18	21.47	84.00	345	320	163
Chicago	SALSA2 w graph	6.15	4.63	34.60	59	72	128
	3D-DNA w correction	2,641.31	2.62	12.76	244	186	1550
	3D-DNA wo correction	1,648.92	4.52	34.60	119	100	711

Table 1. Assembly scaffold and correctness statistics for NA12878 assemblies scaffolded with different Hi-C libraries. The NG50 of human reference GRCh38 is 145 Mbp. The ratio between NG50 and NGA50 represents how many erroneous joins affect large scaffolds in the assembly. A high ratio between NGA50 and NG50 indicates a more accurate assembly. We observe that 3D-DNA mis-assembly detections shears the input with both the mitotic Hi-C and Chicago data so we include results both with and without this assembly correction. In case of Chicago data, 3D-DNA generates a large super-scaffold containing more than 50% of the genome, giving a very high NG50 but a poor NGA50 and ratio.

all error types and input sizes. The number of correctly oriented unitigs increased significantly when assembly graph information was integrated with the scaffolding, particularly for lower input unitig sizes (Figure 4). For example, at 400 kbp, the orientation errors with the graph were comparable to the orientation errors of the graph-less approach at 900 kbp. The NGA50 for SALSA2 also increased when assembly graph information was included (Figure 5). This highlights the power of the assembly graph to improve scaffolding and correct errors, especially on lower contiguity assemblies. This also indicates that generating a conservative assembly, rather than maximizing contiguity, can be preferable for input to Hi-C scaffolding.

3.4 Evaluation on NA12878

Table 1 lists the metrics for NA12878 scaffolds. We include an idealized scenario, using only reference-filtered Hi-C edges for comparison. As expected, the scaffolds generated using only true links had the highest NGA50 value and longest error-free scaffold block. SALSA2 scaffolds were more accurate and contiguous than the scaffolds generated by SALSA1 and 3D-DNA, even without use of the assembly graph. The addition of the graph further improved the NGA50 and longest error-free scaffold length.

We also evaluated the assemblies using Feature Response Curves (FRC) based on scaffolding errors (Vezzi *et al.*, 2012). An assembly can have a high raw error count but still be of high quality if the errors

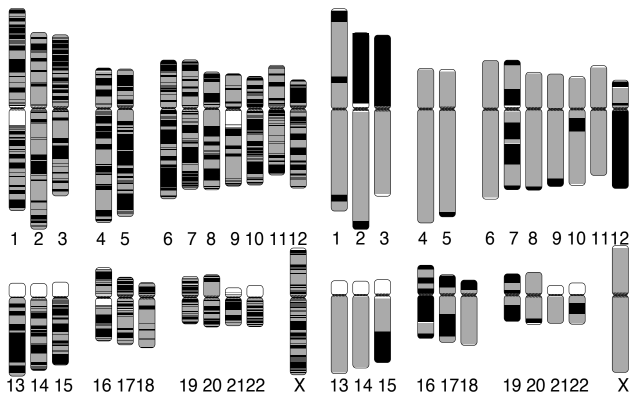
1 2 3 4 5 6 7 8 9 10 11 12
13 14 15 16 17 18 19 20 21 22 X
1 2 3 4 5 6 7 8 9 10 11 12
13 14 15 16 17 18 19 20 21 22 X

Fig. 7. Chromosome ideogram generated using the coloredChromosomes (Böhringer et al., 2002) package. Each color switch denotes a change in the aligned sequence, either due to large structural error or the end of a unitig/scaffold. Left: input unitigs aligned to the GRCh38 reference genome. Right: SALSA2 scaffolds aligned to the GRCh38 reference genome. More than ten chromosomes are in a single scaffold. Chromosomes 1 and 7 are more fragmented due to scaffolding errors which break the alignment.

are restricted to only short scaffolds. FRC captures this by showing how quickly error is accumulated, starting from the largest scaffolds. Figure 6(A) shows the FRC for different assemblies, where the X-axis

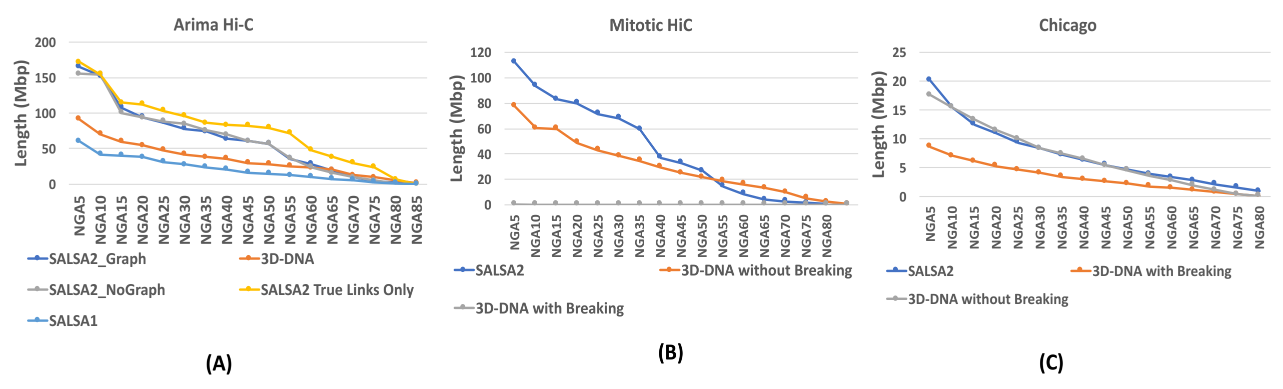

Fig. 8. Contiguity plot for scaffolds generated with (A) standard Arima-HiC data (B) mitotic Hi-C data and (C) Chicago data. The X-axis denotes the NGAX statistic and the Y-axis denotes the corrected block length to reach the NGAX value. SALSA2 results were generated using the assembly graph, unless otherwise noted.

denotes the cumulative % of assembly errors and the Y-axis denotes the cumulative assembly size. The assemblies with more area under the curve accumulate fewer errors in larger scaffolds and hence are more accurate. SALSA2 scaffolds with and without the graph have similar areas under the curve and closely match the curve of the assembly using only true links. The 3D-DNA scaffolds have the lowest area under the curve, implying that most errors in the assembly occur in the long scaffolds. This is confirmed by the lower NGA50 value for the 3D-DNA assembly (Table 1).

Apart from the correctness, SALSA2 scaffolds were highly contiguous and reached an NG50 of 125 Mbp (cf. GRCh38 NG50 of 145 Mbp). Figure 7 shows the alignment ideogram for the input unitigs as well as the SALSA2 assembly. Every color change indicates an alignment break, either due to error or due to the end of a sequence. The input unitigs are fragmented with multiple unitigs aligning to the same chromosome, while the SALSA2 scaffolds are highly contiguous and span entire chromosomes in many cases. Figure 8(A) shows the contiguity plot with corrected NG stats. As expected, the assembly generated with only true links has the highest values for all NGA stats. The curve for SALSA2 assemblies with and without the assembly graph closely matches this curve, implying that the scaffolds generated with SALSA2 are approaching the optimal assembly of this Arima-HiC data.

3.5 Robustness to input library

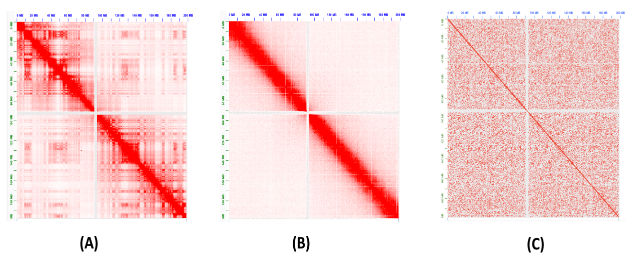

Fig. 9. Contact map of Hi-C interactions on Chromosome 3 generated by the Juicebox software (Durand et al., 2016a). The cells sequenced in (A) normal conditions, (B) during mitosis, and (C) Dovetail Chicago

We next tested scaffolding using two libraries with different Hi-C contact patterns. The first, from (Naumova *et al.*, 2013), is sequenced during mitosis. This removes the topological domains and generates fewer off-diagonal interactions. The second, the L1 library from (Putnam *et al.*, 2016), is an *in vitro* chromatin sequencing library (Chicago) generated by Dovetail Genomics. It also removes off-diagonal matches but has shorter-range interactions, limited by the size of the input molecules. As seen from the contact map in Figure 9, both the mitotic Hi-C and Chicago libraries follow different interaction distributions than the standard Hi-C (Arima-HiC in this case). We ran SALSA2 with defaults and 3D-DNA with both the assembly correction turned on and off.

For mitotic Hi-C data, we observed that the 3D-DNA mis-assembly correction algorithm sheared the input assembly into small pieces, which resulted in more than 12,000 errors and more than half of the unitigs incorrectly oriented or ordered. Without mis-assembly correction, the 3D-DNA assembly has a higher number of orientation (345 vs. 117) and ordering (320 vs. 98) errors compared to SALSA2. The feature response curve for the 3D-DNA assembly with breaking is almost a diagonal (Figure 6(B)) because the sheared unitigs appeared to be randomly joined. SALSA2 scaffolds contain longer stretches of correct scaffolds compared to 3D-DNA with and without mis-assembly correction (Figure 8(B)).

For the Chicago libraries, 3D-DNA mis-assembly detection once again sheared the input unitigs. It generated a single 2.7 Gbp scaffold and was unable to split it into the requested number of chromosomes. 3D-DNA uses signatures of chromosome ends (Dudchenko *et al.*, 2017) to identify break positions which are not present in Chicago data. As a result, it generated more chimeric joins compared to SALSA2 (1,550 vs. 128 errors). However, the number of order and orientation errors was similar across the methods. Even in the large single scaffold generated by 3D-DNA, the sizes of the correctly oriented and ordered blocks were smaller than SALSA2 (Figure 8(C)). Since Chicago libraries do not provide chromosome-spanning contact information for scaffolding, the NG50 value for SALSA is 6.15 Mbp, comparable to the equivalent coverage assembly (50% L1+L2) in (Putnam *et al.*, 2016) but much smaller than Hi-C libraries. SALSA2 is robust to changing contact distributions. In the case of Chicago data it produced a less contiguous assembly due to the shorter interaction distance. However, it avoids introducing false joins, unlike 3D-DNA, which appears tuned for a specific contact model.

4 Conclusion

In this work, we present the first Hi-C scaffolding method that integrates an assembly graph to produce high-accuracy, chromosome-scale assemblies. Our experiments on both simulated and real sequencing data for the human genome demonstrate the benefits of using an assembly graph to guide scaffolding. We also show that SALSA2 outperforms alternative Hi-C scaffolding tools on assemblies of varied contiguity, using multiple Hi-C library preparations.

Hi-C scaffolding has been historically prone to inversion errors when the input assembly is highly fragmented. The integration of the assembly graph with the scaffolding process can overcome this limitation. Existing Hi-C scaffolding methods also require an estimate for the number of chromosomes in the genome. Since SALSA2's mis-join correction algorithm stops scaffolding after the useful linking information in a dataset is exhausted, no chromosome count is needed as input. As the Genome10K consortium (Koepfli *et al.*, 2015) and independent scientists begin to sequence novel lineages in the tree of life, it may be impractical to generate physical or genetics maps for every organism. Thus, Hi-C sequencing combined with SALSA2 presents an economical alternative for the reconstruction of chromosome-scale assemblies.

Acknowledgements

AS and SS were funded by generous support from NHGRI (grant# 1R44HG009584). JG and MP were supported by NIH grant R01-AI-100947 to MP. SK, AR, BPW, and AMP were supported by the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health. AR was also supported by a grant from the Korean Visiting Scientist Training Award (KVSTA) through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number: HI17C2098). This work utilized the computational resources of the NIH HPC Biowulf cluster (http://hpc.nih.gov).

References

Bickhart, D. M. *et al.* (2017). Single-molecule sequencing and chromatin conformation capture enable de novo reference assembly of the domestic goat genome. *Nature Genetics*, **49**(4), 643–650.

Böhringer, S. *et al.* (2002). A software package for drawing ideograms automatically. *Online J Bioinformatics*, **1**, 51–61.

Burton, J. N. *et al.* (2013). Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. *Nature biotechnology*, **31**(12), 1119–1125.

Chin, C.-S. *et al.* (2016). Phased diploid genome assembly with single molecule real-time sequencing. *bioRxiv*.

Dixon, J. R. *et al.* (2012). Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, **485**(7398), 376–380.

Dong, Y. *et al.* (2013). Sequencing and automated whole-genome optical mapping of the genome of a domestic goat (capra hircus). *Nature biotechnology*, **31**(2), 135–141.

Dudchenko, O. *et al.* (2017). De novo assembly of the aedes aegypti genome using hi-c yields chromosome-length scaffolds. *Science*, **356**(6333), 92–95.

Durand, N. C. *et al.* (2016a). Juicebox provides a visualization system for hi-c contact maps with unlimited zoom. *Cell systems*, **3**(1), 99–101.

Durand, N. C. *et al.* (2016b). Juicer provides a one-click system for analyzing loop-resolution hi-c experiments. *Cell systems*, **3**(1), 95–98.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of mathematics*, **17**(3), 449–467.

Eid, J. *et al.* (2009). Real-time dna sequencing from single polymerase molecules. *Science*, **323**(5910), 133–138.

Ghurye, J. *et al.* (2017). Scaffolding of long read assemblies using long range contact information. *BMC genomics*, **18**(1), 527.

Gnerre, S. *et al.* (2011). High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, **108**(4), 1513–1518.

Jain, M. *et al.* (2016). The oxford nanopore minion: delivery of nanopore sequencing to the genomics community. *Genome biology*, **17**(1), 239.

Jain, M. *et al.* (2017). Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv*, page 128835.

Kaplan, N. and Dekker, J. (2013). High-throughput genome scaffolding from in vivo dna interaction frequency. *Nature biotechnology*, **31**(12), 1143–1147.

Koepfli, K.-P. *et al.* (2015). The genome 10k project: a way forward. *Annu. Rev. Anim. Biosci.*, **3**(1), 57–111.

Kolmogorov, M. *et al.* (2018). Assembly of long error-prone reads using repeat graphs. *bioRxiv*.

Koren, S. *et al.* (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, **27**(5), 722–736.

Kurtz, S. *et al.* (2004). Versatile and open software for comparing large genomes. *Genome biology*, **5**(2), R12.

Li, H. (2016). Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, **32**(14), 2103–2110.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, **25**(14), 1754–1760.

Lieberman-Aiden, E. *et al.* (2009). Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *science*, **326**(5950), 289–293.

Marie-Nelly, H. *et al.* (2014). High-quality genome (re) assembly using chromosomal contact data. *Nature communications*, **5**.

Miller, J. R. *et al.* (2010). Assembly algorithms for next-generation sequencing data. *Genomics*, **95**(6), 315–327.

Myers, E. W. (2005). The fragment assembly string graph. *Bioinformatics*, **21**(suppl 2), ii79–ii85.

Nagarajan, N. and Pop, M. (2009). Parametric complexity of sequence assembly: theory and applications to next generation sequencing. *Journal of computational biology*, **16**(7), 897–908.

Nagarajan, N. and Pop, M. (2013). Sequence assembly demystified. *Nature Reviews Genetics*, **14**(3), 157–167.

Naumova, N. *et al.* (2013). Organization of the mitotic chromosome. *Science*, **342**(6161), 948–953.

Pendleton, M. *et al.* (2015). Assembly and diploid architecture of an individual human genome via single-molecule technologies. *Nature methods*.

Pevzner, P. A. *et al.* (2001). An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, **98**(17), 9748–9753.

Poloczek, M. and Szegedy, M. (2012). Randomized greedy algorithms for the maximum matching problem with new analysis. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 708–717. IEEE.

Putnam, N. H. *et al.* (2016). Chromosome-scale shotgun assembly using an in vitro method for long-range linkage. *Genome research*, **26**(3), 342–350.

Salzberg, S. L. *et al.* (2012). Gage: A critical evaluation of genome assemblies and assembly algorithms. *Genome research*, **22**(3), 557–567.

Schneider, V. A. *et al.* (2017). Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, **27**(5), 849–864.

Schwartz, D. C. *et al.* (1993). Ordered restriction maps of saccharomyces cerevisiae chromosomes constructed by optical mapping. *Science*, **262**(5130), 110–114.

Shelton, J. M. *et al.* (2015). Tools and pipelines for bionano data: molecule assembly pipeline and fasta super scaffolding tool. *BMC genomics*, **16**(1), 734.

Simonis, M. *et al.* (2006). Nuclear organization of active and inactive chromatin domains uncovered by chromosome conformation capture–on-chip (4c). *Nature genetics*, **38**(11), 1348–1354.

Venter, J. C. *et al.* (1996). A new strategy for genome sequencing. *Nature*, **381**(6581), 364.

Vezzi, F. *et al.* (2012). Feature-by-feature–evaluating de novo sequence assembly. *PloS one*, **7**(2), e31002.

Weisenfeld, N. I. *et al.* (2017). Direct determination of diploid genome sequences. *Genome research*, **27**(5), 757–767.

Wysoker, A. *et al.* (2013). Picard tools version 1.90.

Yeo, S. *et al.* (2017). Arcs: Scaffolding genome drafts with linked reads. *Bioinformatics*.

Zheng, G. X. *et al.* (2016). Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. *Nature biotechnology*.