OMBlast: Alignment Tool for Optical Mapping Using a Seed-and-extend Approach – Supplementary Document

Alden King-Yung Leung[1], Tsz-Piu Kwok[2], Raymond Wan[*,1], Ming Xiao[5], Pui-Yan Kwok[6,7], Kevin Y. Yip[2,3,†], and Ting-Fung Chan[1,2,3,4,†]

[1]School of Life Sciences
[2]Department of Computer Science and Engineering
[3]Hong Kong Bioinformatics Centre
[4]Centre for Soybean Research, State Key Laboratory of Agrobiotechnology, The Chinese University of Hong Kong, Hong Kong, China
[5]School of Biomedical Engineering, Science and Health System, Drexel University, Philadelphia, Pennsylvania, USA
[6]Institute for Human Genetics
[7]Cardiovascular Research Institute, University of California San Francisco, San Francisco, California, USA
[†]To whom correspondance should be addressed.

June 2016

Contents

[*]Currently with the Division of Life Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China

List of Figures

List of Tables

Abstract

The purpose of this Supplementary Document is to support the manuscript entitled "OMBlast: Alignment Tool for Optical Mapping Using a Seed-and-extend Approach". Sections, figures and tables in this document are all preceded by an **S** to distinguish them from those in the main manuscript.

S1 Introduction

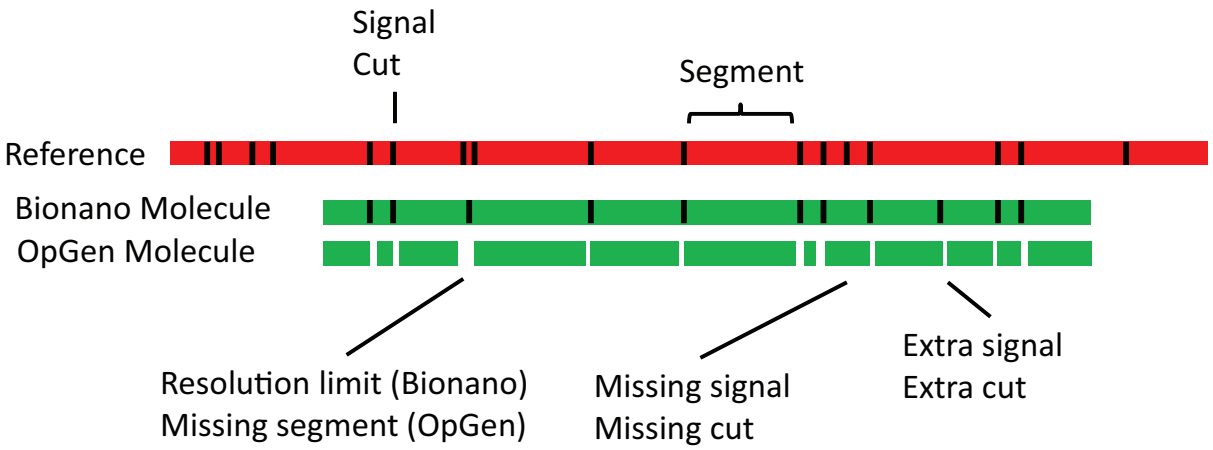


Figure S1: A comparision of the terminology used in optical mapping for instruments from Bionano Genomics Inc. and OpGen Inc.

Bionano Genomics Inc. and OpGen Inc. are two major providers of instruments that produce high-throughput optical mapping data. Supplementary Figure S1 shows how the terminology used between them differ slightly. We represent molecules as rectangles (i.e., red for the reference optical map; green for the query optical map) with black vertical bars to indicate the location of **signals**. Each section of an optical map bounded by two signals is a **segment**.

Supplementary Figure S1 also demonstrates that the terminology used by the two companies to describe the types of errors found in optical maps also differ slightly. This is further summarized in Supplementary Table S1.

Table S1: Sources of error in optical maps generated by Bionano Genomics Inc. and OpGen Inc.

Error	Bionano Genomics Inc.	OpGen Inc.
Scaling Error	Constant in all segments	Varies across all segments
Measuring Error	Imaging hardware	
Missing Cuts	Incomplete enzyme digestion	
Extra Cuts	Non-specific enzyme cut Random DNA break	
Smallest Segment Limit	Resolution limit	Missing Segment

S2 Algorithm Details

OMBlast consists of three main modules:

1. Alignment and extension;
2. Overlapping alignments merging; and
3. Alignment joining.

We describe each one in this section.

S2.1 Seeding and extension

Seeds are first obtained from the query optical map via a sliding window. Similar to BLAST (Altschul *et al.*, 1990), these seeds are sought in the reference genome and then extended. One notable difference between OMBlast and BLAST is that OMBlast operates on the set of signals instead of the set of nucleotides.

S2.1.1 Seeding

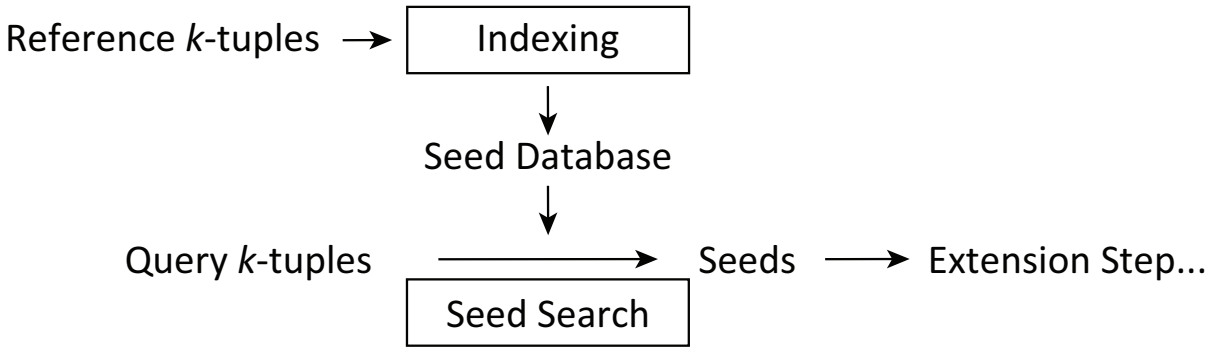


Figure S2: Overview of seed generation.

As shown in Supplementary Figure S2, the seeding module of OMBlast consists of two stages:

1. Indexing reference k-tuples into a seed database and
2. Searching matching seeds for the query k-tuple using the generated seed database.

There are two methods for identifying seeds of length k, which we have dubbed Seeding Method A and Seeding Method B.

With Seeding Method A, the reference k-tuples are indexed by generating k sorted lists. Each of these lists are sorted independently by a different key. So, one list is sorted by the first position, a second list is sorted by the second position, etc. For a given query k-tuple, binary search is applied on each of these lists, the results of which are intersected for further consideration.

In Seeding Method B, the reference k-tuples are indexed by placing them into bins of 5 kbp each. Each bin is enumerated so that they can be identified by a character, and then placed in a look-up table. Seeds from the query k-tuple are transformed in a similar way so that strings can be used to locate the bins of interest. In order to avoid missing any bins, a range of bins are obtained. After extracting all the reference k-tuples from the look-up table, they are subjected to further validation. A 3-tuple is shown in Supplementary Figure S3(B) as an example to demonstrate the iteration of combination of bins.

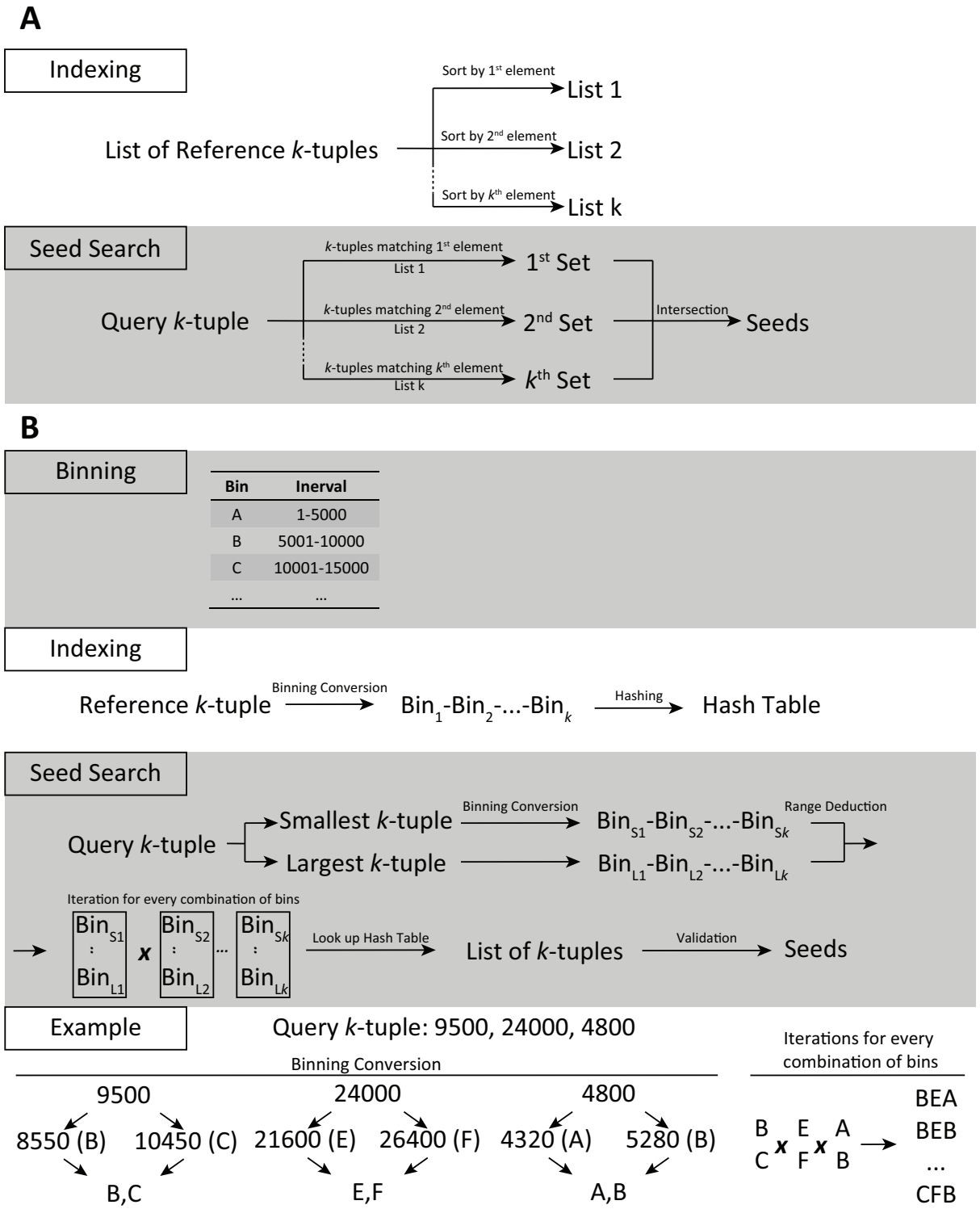


Figure S3: Details of seed generation.

S2.1.2 Extension

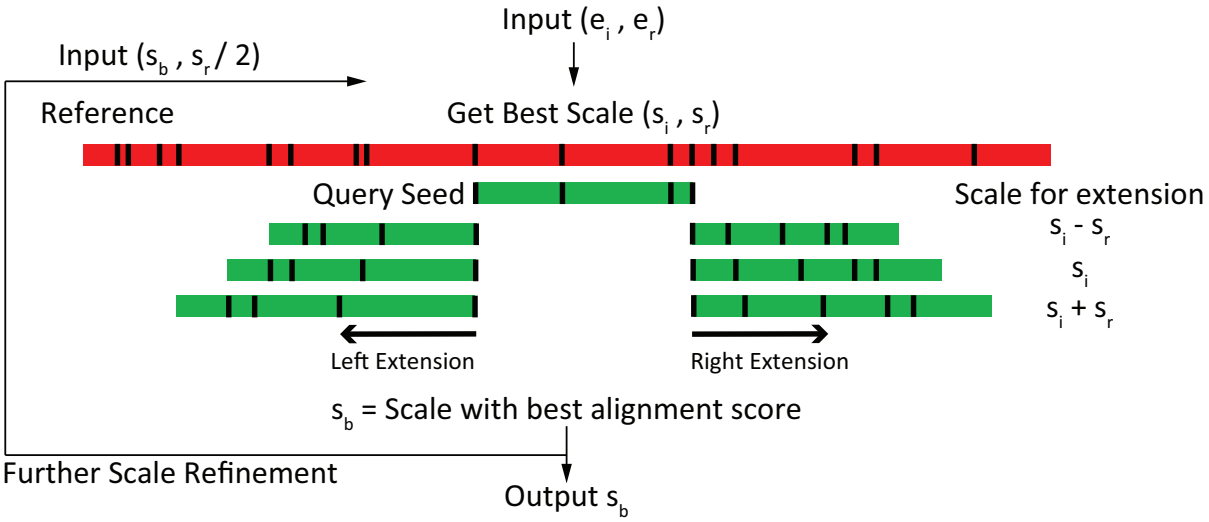


Figure S4: Overview of recursive refinement during extension.

As shown in Figure S4, a constant number of rounds of extensions is taken to refine the best scaling factors. In every round, given the initial scale s_i and a range of scale variations s_r, the seed is extended to the left and right separately using three scaling factors $s_i - s_r$, s_i and $s_i + s_r$. Each pair of left and right extensions is combined and the scale yielding the best extension score is taken as the best scale s_b for this round of extension. To further refine the scale, the best scale s_b is passed for another round of extension along with half of the range of scale variation.

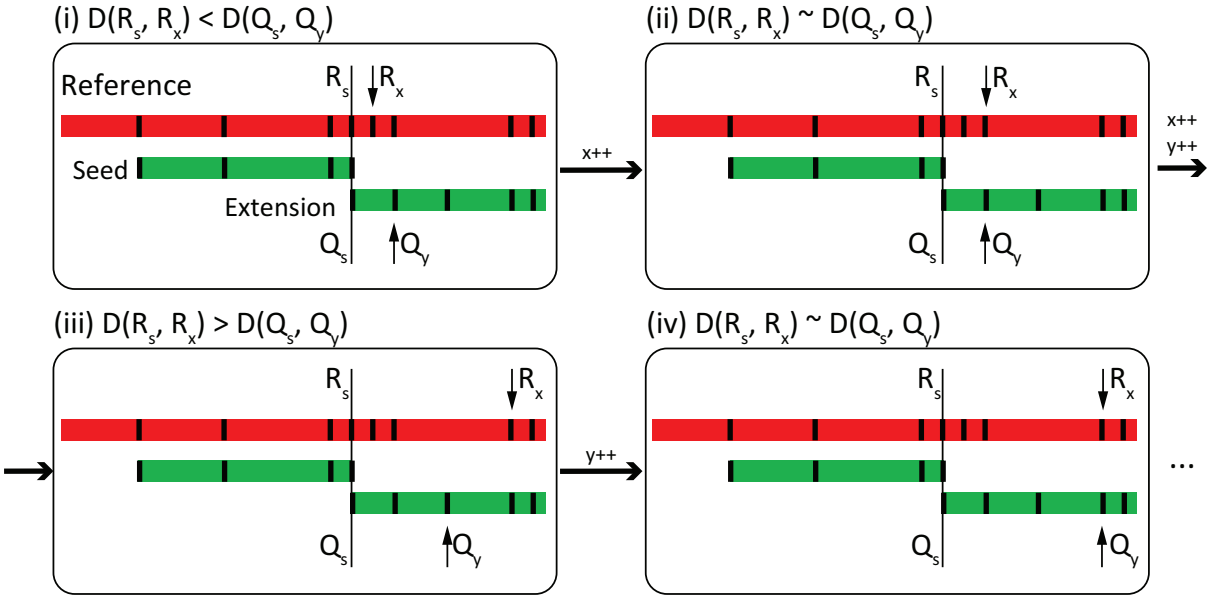


Figure S5: Illustration of steps in extension.

Supplementary Figure S5 provides further details in the extension process. The distance between two signals S_i and S_j is defined as $D(S_i, S_j)$. Before extension, the starting reference signal R_s and starting query signal Q_s forms a signal matching signal pair as an anchor point. During extension, this module checks if signal R_x matches Q_y. If the distance $D(R_s, R_x)$ and

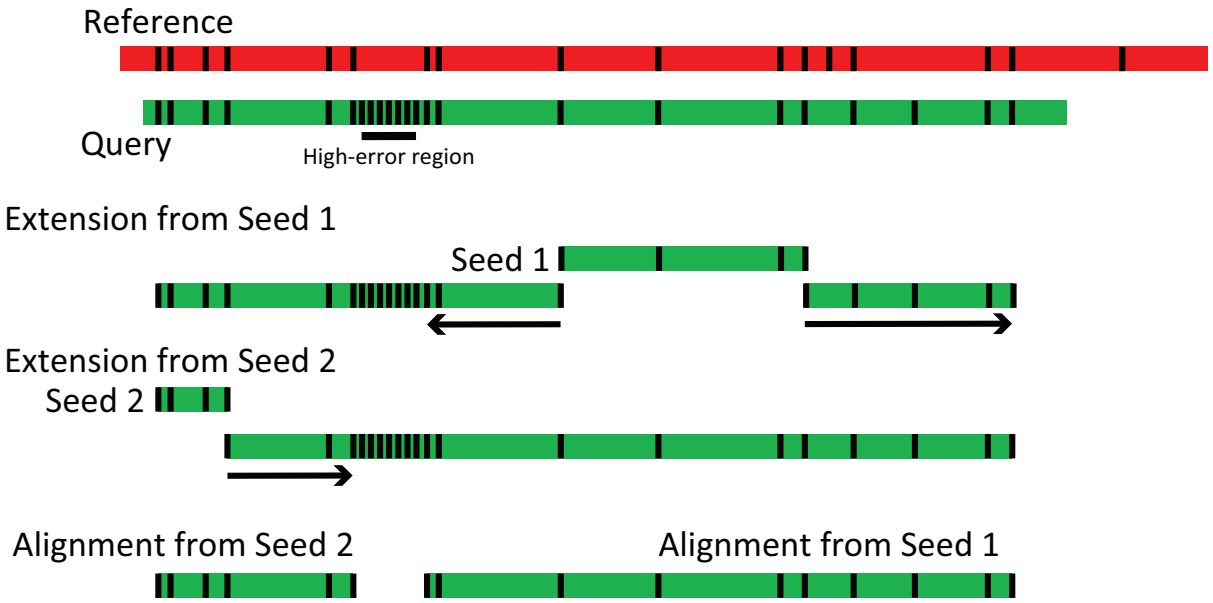


Figure S6: An example of the extension of the query with high-error region.

$D(Q_s, Q_y) \times scale$ are similar according to Equation (1), R_x matches Q_y and form a matching signal pair R_xQ_y, followed by the increment of x and y by 1 (Cases ii and iv). If the distances do not match, x is incremented by 1 if $D(R_s, R_x) < D(Q_s, Q_y) \times scale$ (Case i), or y is incremented by 1 if $D(R_s, R_x) > D(Q_s, Q_y) \times scale$ (Case iii). This method guarantees to find the best number of matches in the whole extension but not the best alignment details. The extension ends when either R_x or Q_y is the last available signal on the reference or query respectively. To further facilitate the extension process, a heuristic approach is taken to end the extension when consecutive extra or missing sites are encountered. Under some conditions, this heuristic blocks further extension when a query contains a local high-error region, as illustrated in Supplementary Figure S6.

In this example, the query could be extended from Seed 1 and Seed 2, but the high-error region prevents further left and right extension on Seed 1 and Seed 2, respectively. Nevertheless, the two separate alignments which covers the confidently aligned regions can be joined by the alignment-joining module, to be described in Supplementary Section S2.3.

S2.2 Overlapping Alignment Merging

After extension, the alignments are merely *partial* alignments since they will match only some of the signals in the reference. Supplementary Figure S7 illustrates the difference between two types of overlapping alignments using two examples.

Case 1 (Supplementary Figure S7(A)) shows an example of two alignments with overlapping signals. The query's signals from Q_1 to Q_3 in alignment 1 overlaps with the same query's signals in alignment 2. Additionally, the reference's signal from R_2 to R_4 in alignment 1 overlaps with the reference's signals R_1 to R_3 in alignment 2. However, in this example, the matching signal pairs R_2Q_1, R_3Q_2, and R_4Q_3 in alignment 1 are clearly distinct from the matching signal pairs R_1Q_1, R_2Q_2, and R_3Q_3 in alignment 2.

Case 2 (Supplementary Figure S7(B)) shows an example of two alignments with overlapping matching signal pairs since the matching signal pair R_5Q_4 exists in both alignment 1 and alignment 2. Overlapping matching signal pair implies at least one query and one reference signal are shared between the two alignments. Therefore, these partial alignments always contain

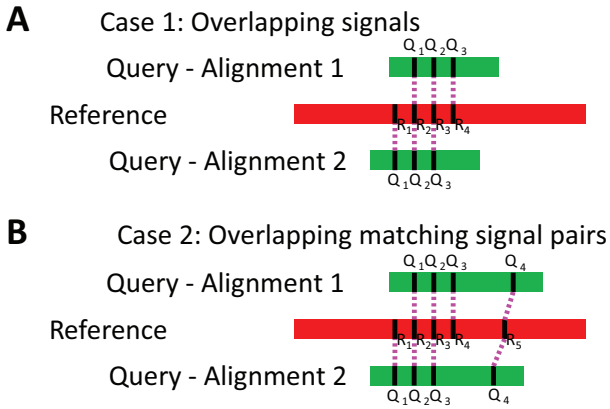


Figure S7: Difference between alignments with overlapping matching signal pairs and overlapping signals.

overlapping signals, even though the converse is not necessarily true.

Partial alignments extended from various seeds could lead to overlapping alignments. Supplementary Figure S8 shows an example of 3 partial alignments being overlapping. This module merges these partial alignments with overlapping matching signal pairs (Case 2) into one single concordant partial alignment. Matching signal pairs from all partial alignments are taken as nodes. Each partial alignment provides the information of edges on the connection of matching signal pairs. For example, nodes pointed to by the outgoing edges of R_1Q_1 include R_2Q_2, as suggested by partial alignments 1 and 3, and R_3Q_2, as suggested by partial alignment 2 (Supplementary Figure S8(A)). The weight of the edge is set to a certain score dependent on the number of extra or missing sites present between two matching pairs of signals. The weight is calculated based on the former part of Equation (3) $(t_m \times u_m - t_{es} \times u_{es} - t_{ms} \times u_{ms})$, where $u_m = 1$, $u_{ms} = |i_{x+1} - i_x| - 1$ and $u_{es} = |j_{x+1} - j_x| - 1$. Edges with or without extra or missing signals are represented in black and red arrows respectively.

A weighted directed acyclic graph is built Supplementary Figure S8(B) based on the nodes and edges in Supplementary Figure S8(A). The graph is acyclic because for every directed edge from $R_{x_1}Q_{y_1}$ to $R_{x_2}Q_{y_2}$, it is guaranteed $x_2 > x_1$ and $y_2 > y_1$. The path with highest total weight is obtained and the nodes (matching signal pairs) of this path are taken to form the concordant partial alignment.

S2.3 Alignment Joining

In this module, partial alignments are joined as the complete final alignment. We first describe the relationship between partial alignments. Despite the removal of partial alignments with overlapping matching signal pairs by the previous module (Case 2, Supplementary Figure S7(B)), some partial alignments still contain overlapping signals and require trimming (Case 1, Supplementary Figure S7(A)). A graph-based approach is then used for the best combination of partial alignments.

S2.3.1 Alignment Relationship

Any two non-overlapped partial alignments has any one of the following relationships (Supplementary Figure S9). Two partial alignments are close together on the reference if their distance is smaller than a user-defined value which is 250 kbp by default.

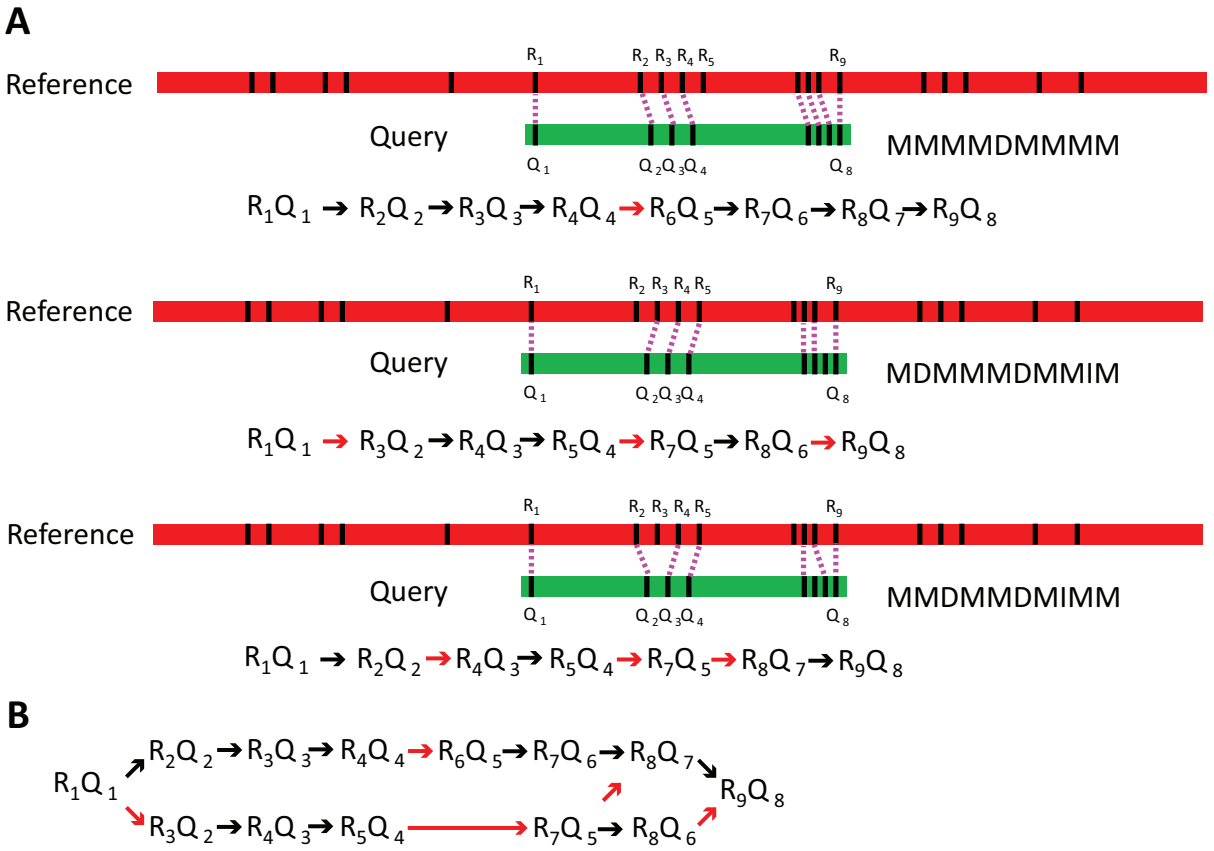


Figure S8: Example of merging overlapping alignments.

Insertion/Deletion Two partial alignments are close together on the reference with the same orientation

Inversion Two partial alignments are close together on the reference with different orientations. The orientation of both forward-reverse or reverse-forward pairs are valid.

S2.3.2 Overlapped Alignment Trimming

Two alignments that overlap with no shared matching signal pair can both be trimmed so that the information that they give are no longer contradictory. Within OMBlast, we have implemented an algorithm which trims partial alignments by a predetermined number of times (default as 5) before joining them up. If they cannot be trimmed, then the partial alignments cannot be joined together. Trimming proceeds a step at a time for each partial alignment.

For example, in Supplementary Figure S10, we have two alignments $A1$ and $A2$. Alignment $A1$ needs to be trimmed from the right, towards the starting position of the seed. Alignment $A2$, though, has its seed on the right; thus, it needs to be trimmed from the left. Consider $A1_x$ and $A2_y$ being the partial alignment 1 after x trimming events and partial alignment 2 after y trimming events respectively. We iteratively increment x and y (trim $A1$ and $A2$) until $A1$ and $A2$ are non-overlapping. We define the pair x', y' as a child of x, y if $x' \geq x$ and $y' \geq y$. We assume the fewer the trimming steps on an alignment the better. Therefore, if $A1_x$ and $A2_y$ are non-overlapping, any children of the pair x, y is not attempted. Here, $A1_5$ and $A2_1$ is non-overlapping. Any children pair like $x = 6$ and $y = 2$ is not attempted but $x = 6$ and $y = 0$ will be attempted. The end result is the joined alignment at the bottom of the figure, which is composed of $A1_x$, the left half of $A1$, and $A2_y$, the right half of $A2$.

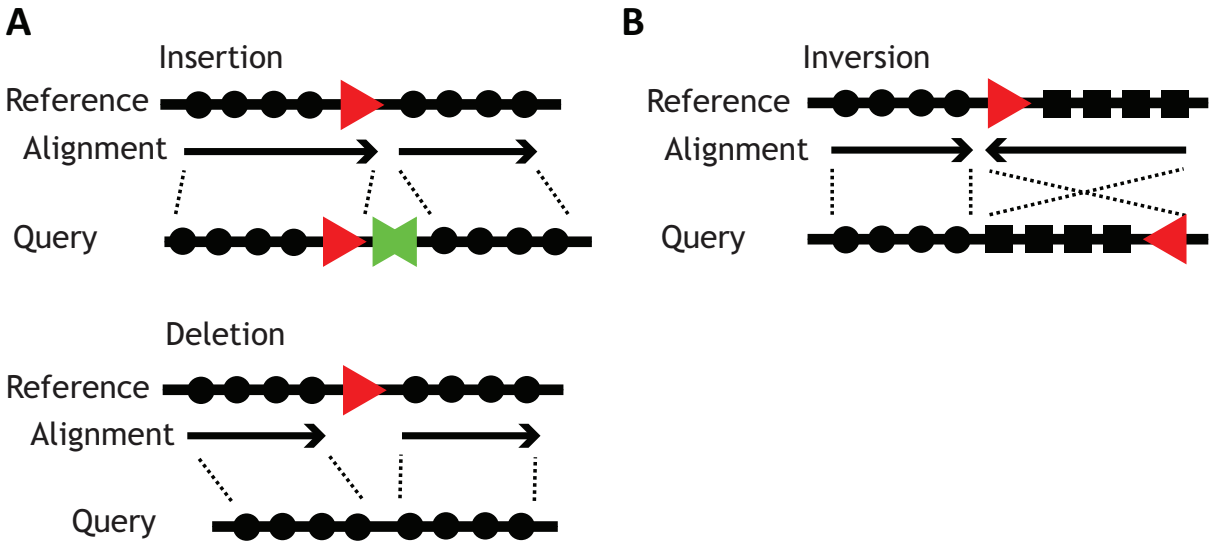


Figure S9: Example of partial alignment relation.

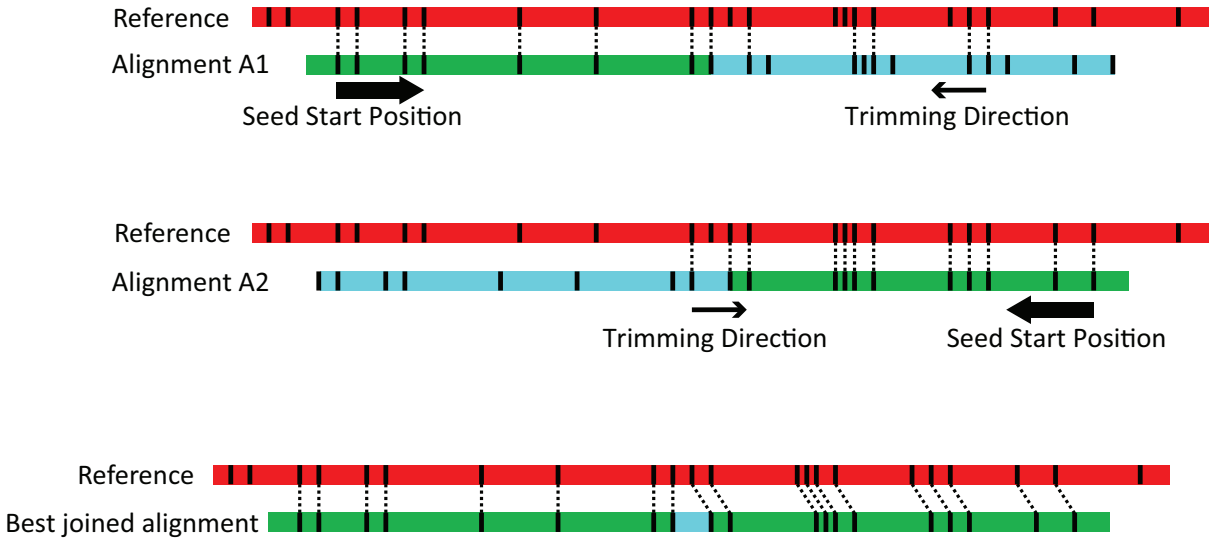


Figure S10: Example of trimming.

S2.3.3 Alignment Joining

In this example, only insertion, deletion and inversion relationship between alignments are considered. From Supplementary Figure S11(A), 5 parital alignments of a query are overlapping. To find the best combination of partial alignments, we need to build a table of relationships between any two partial alignments (Supplementary Figure S11(B)), such that each relationship is taken as a node in a graph. Next we need to determine the edge between all nodes. We loop through every node in the order of starting alignment followed by stopping alignment.

(i) Edges are formed between the alignment pair 1,2 and any node with starting element 2, i.e. 2,3 and 2,4.

(ii) Similarly, the alignment pair 1,3 has edges to 3,4 and 3,5.

(iii) In some cases, if a partial alignment contains too few matching signals and could not pass the user-defined threshold (which is default to 3) after trimming in both directions,

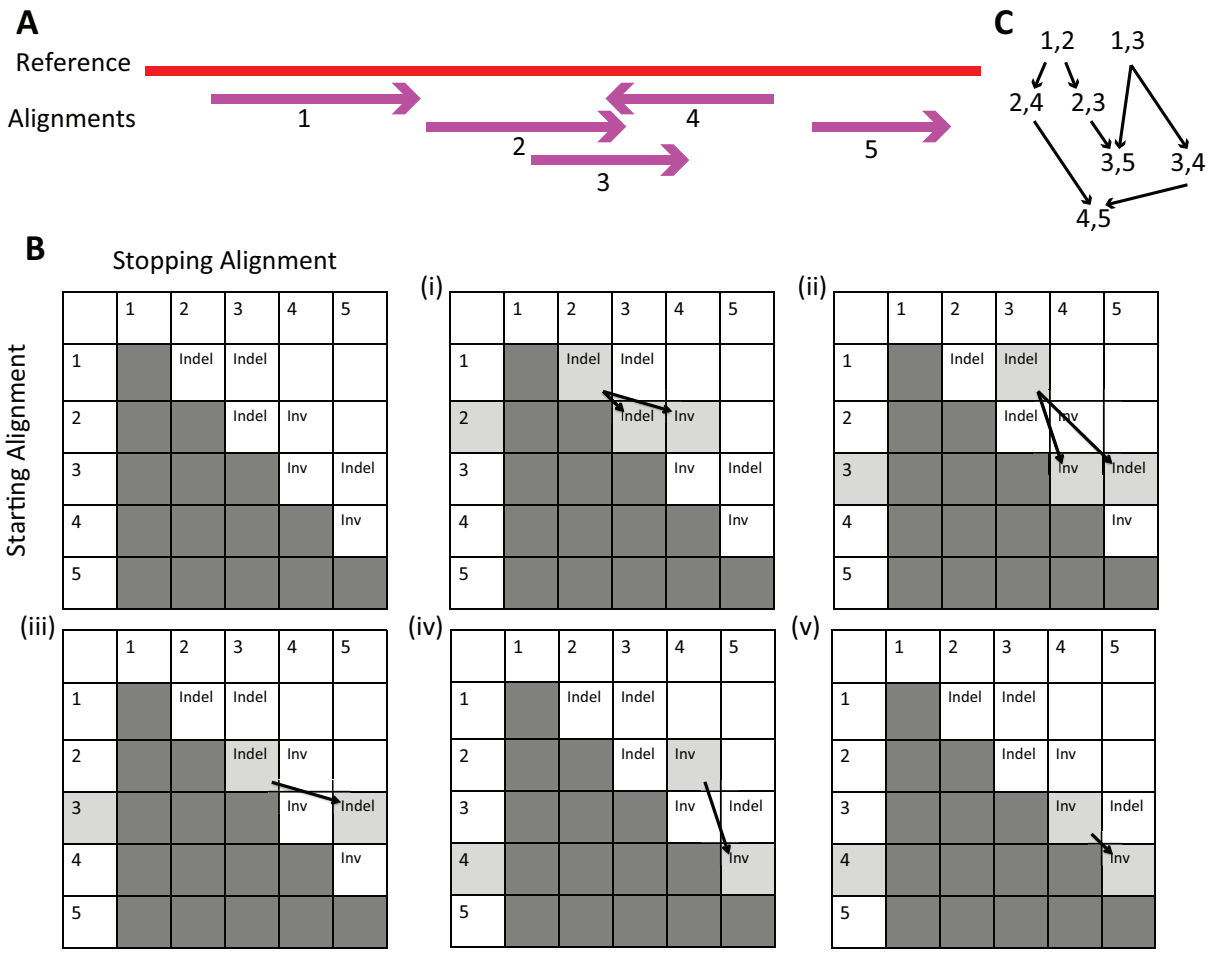


Figure S11: Example of alignment joining.

such a relationship is discarded. For example, the alignment pair 2,3 can only be linked to 3,5 but not 3,4. This is because partial alignment 3 is too short after being trimmed twice if an edge is built between 2,3 and 3,4. A visual relationship in Supplementary Figure S11(A) shows clear overlap between partial alignment 2, 3 and 4.

(iv) the alignment pair 2,4 have only one edge to 4,5.

(v) the alignment pair 3,4, similarly, have only one edge to 4,5.

After building the edges among the alignment pairs, a weighted directed graph is formed and the path with the best score is taken as the final alignment by using dynamic programming.

S2.3.4 Uniqueness of Alignment

As an alternative to the scoring system described in Section 3.4, the user can use *uniqueness*. Intuitively, uniqueness is an indicator for a particular alignment, normalized across all other alignments. Uniqueness indicates how an alignment stands out from the other alignments based on the score. Low uniqueness reflects either one or both of these factors: (1) the query can be aligned to more than one region confidently; and (2) the score of this alignment is similar to the alignment on other non-specific regions. Only the top 10 alignments with the highest score

are considered in the calculations. The uniqueness u of the final alignment i is defined as:

$$u_i = \begin{cases} \frac{1}{j} & : \text{all possible alignments share the same score} \\ \frac{o_i - o_{\min} + b}{\sum_j (o_j - o_{\min} + 1)} & : \text{otherwise} \end{cases} \quad (1)$$

where o_i denotes the score of the ith alignment, as described in Equation (4), and b denotes a base constant which defaults to 1.

S3 Experiment Framework

All analysis was run on an Intel Xeon CPU E5-4640 (2.40 GHz) with 512 GB of main memory running the Ubuntu 14.04.2 operating system with version 3.13 of the Linux kernel. The following software was used: Python 2.7.6, Java 1.7.0_55 and the GNU compiler collection 4.5.2. The five tools used in our study are presented in Supplementary Table S2.

Table S2: Software used in our study.

Software	Methodology	Version	Citation
OMBlast [1]	Seed-and-extend	1.0	This manuscript
RefAligner [2]	Dynamic programming, Hashing	r3520	Shelton *et al.* (2015)
Valouev [3]	Dynamic programming	N/A	Valouev *et al.* (2006)
SOMA [4]	Dynamic programming	2	Nagarajan *et al.* (2008)
TWIN [5]	FM-index	1.03rc	Muggli *et al.* (2014)

S3.1 Program Installation

Executable files for RefAligner and TWIN were downloaded directly. The Java archive file for OMBlast was run directly in a Java Virtual Machine. The following minor changes to the downloaded source codes were made prior to installation.

Valouev Line 43 was removed in `fit_mols_and_store_als.cc` to disable the erasure of short segments.

SOMA Lines 295–297 were removed in match.cc to allow multiple placements of molecules on the same region of a reference.

S3.2 Commands Executed

Supplementary Table S3 gives the commands executed for the experiments performed in our investigation. Since different alignment methods require various formats, we assume the prefix of reference, query and result file be `ref`, `data` and `result`, respectively.

S3.3 Method of Analysis

A molecule is defined to have aligned correctly if:

1. the alignment position and the simulated position is overlapping; and
2. the alignment strand matches the simulated strand (This criterion is skipped when checking the correctness of alignments on data with inversions).

Throughout our paper, we measured the relative performance of alignment methods in terms of the following three criteria:

[1] https://github.com/aldenleung/OMBlast
[2] http://www.bnxinstall.com/RefalignerAssembler/Linux/SSE/
[3] http://www-hsc.usc.edu/ valouev/papers/om_alignment/optmap_alignment.tar.gz
[4] http://www.cbcb.umd.edu/finishing/soma-v2.tar.gz
[5] http://www.cs.colostate.edu/twin/

OMBlast	`java --Xmx120G --jar OMBlast.jar --refmapin ref.ref --optmapin data.sdata --optresout result.omd --writeunmap false --filtermode 1 --postjoinmode 2 --clustermode 1 --thread 1`
OMBlast (Inversion)	`java --Xmx120G --jar OMBlast.jar --refmapin ref.ref --optmapin data.sdata --optresout result.omd --writeunmap false --filtermode 1 --postjoinmode 2 --clustermode 2 --thread 1`
RefAligner	`RefAligner -ref ref.cmap -i data.cmap -o result -maxthreads 1`
Valouev	`fit ref.valdata data.valdata > result.txt`
SOMA	`soma-v2 -m ref.opt -s data.silico -o result`
TWIN	`python om2bytes.py ref.opt ref.bin 1 *`
	`twin --opt_map ref.bin --silico_map data.silico --smallest_frag_length 0 > result.txt`
	`python twin2psl.py result.txt ref.bin result.psl *`

Table S3: Command-line options used for experiments.

Precision The percentage of molecules that were aligned correctly.
Recall The percentage of correctly aligned molecule out of total input molecules.
Time The amount of time spent by each program (also called "user time").

Precision and recall are depicted together as precision-recall graphs, as explained in Section 4.2.

Under default settings, RefAligner, Valouev and SOMA each output a single alignment. The split-mapped partial alignments output by OMBlast lie very close to the region in both insertion/deletion and inversion mode. If any of the partial alignments aligns correctly, the alignment itself is defined to align correctly. TWIN outputs multiple possible alignments lying on both nearby and distant regions. We define the possible alignments as being "nearby" if all of them overlap with others. A custom program in Java was written to retain alignments of TWIN if they are located at nearby regions. The first possible alignment is then taken as the representative alignment for verifying the precision and recall.

S4 Default Parameters

Most parameters in OMBlast can be user-defined. Through empirical analysis on various organisms, we have defined a list of default parameters for the best performance. The analysis of these important parameters is described in this section.

S4.1 Seeding Algorithm

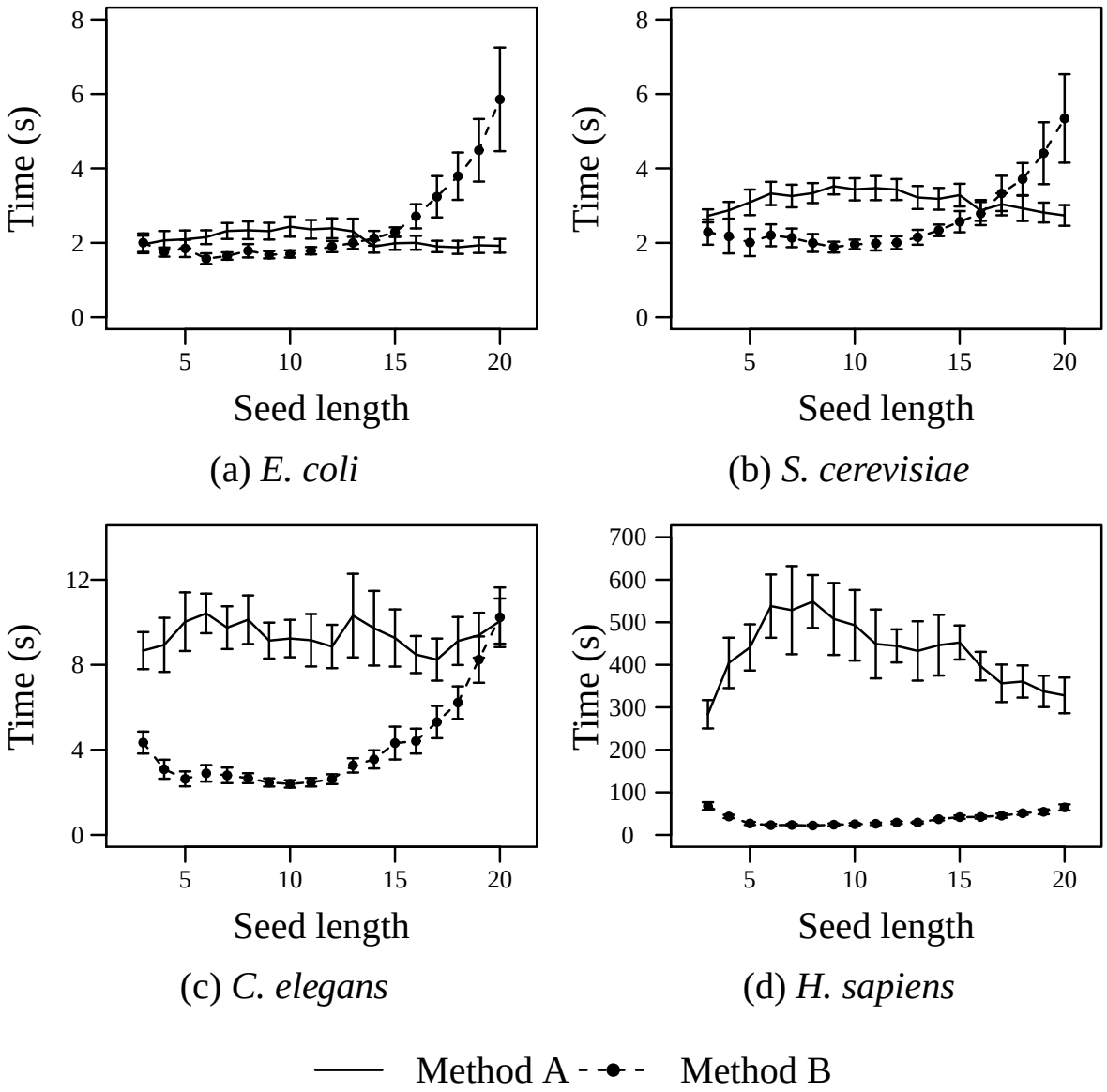


Figure S12: Analysis on the running time of two seeding methods.

As indicated earlier, both Method A (intersection of sorted lists) and Method B (binning and searching) for identifying seeds produce identical results (See Supplementary Figure S3 for an explanation of the underlying methods.). Thus, we only compare the running times of the two methods but not the downstream extension process.

Supplementary Figure S12 and Supplementary Figure S13 present the running time and memory usage of the two methods against various values for the seed length, respectively.

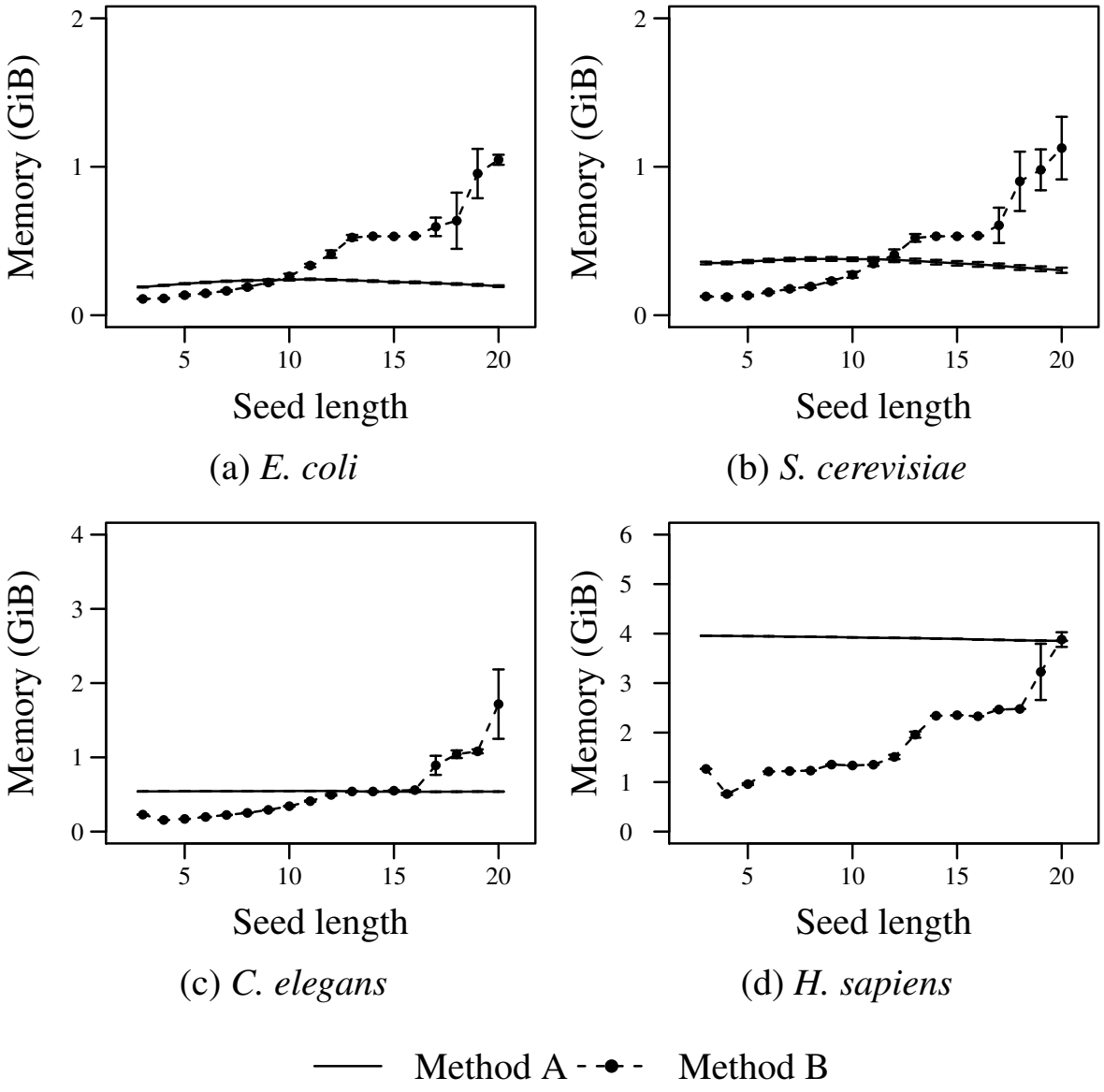


Figure S13: Analysis on the memory usage of two seeding methods.

Each panel represents synthetic optical mapping data derived from the genomes of one of the four different species. Experiments with the 3 replicate data sets for each species was repeated 10 times each to yield the standard deviations shown.

In terms of running time, excluding *H. sapiens* for the moment, Method B is faster than Method A, except when the seed length becomes large. Of course, the difference in time is only 8 or 12 seconds since these data sets come from smaller genomes. For the *H. sapiens* data set, the running time of Method B is several times faster than Method A, for all values of seed lengths considered.

The memory usage in Method A remains constant across various seed lengths. Excluding *H. sapiens*, Method B uses less memory for shorter seed lengths but more memory for longer seed lengths than Method A. For the *H. sapiens* data set, Method B uses less memory than Method A for all values of seed lengths considered. Since the default seed length of OMBlast is only 3, as deduced in Supplementary Section S4.2, we employ only Method B as the default algorithm for the remainder of our experiments.

S4.2 Seed Length

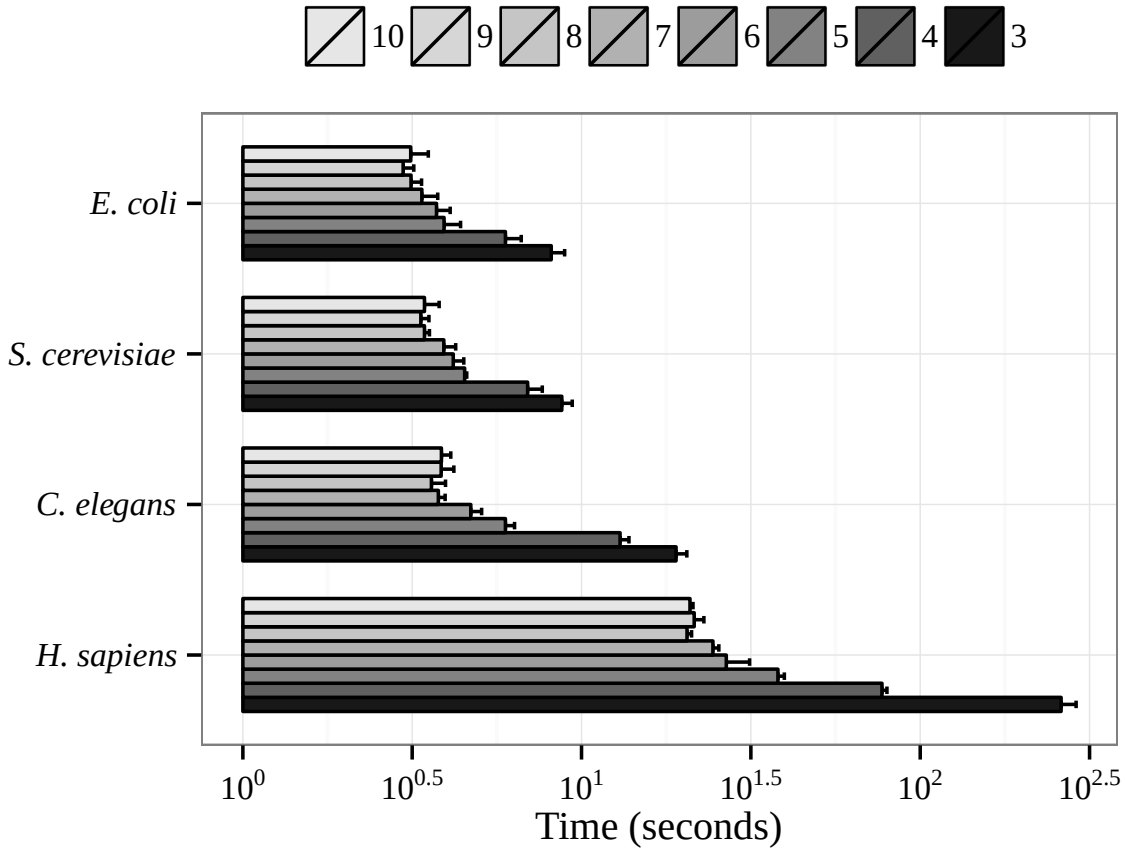


Figure S14: Running time (seconds) for various species with varying seed lengths.

Seed length determines the number of potential seeds for extension. Briefly, with a longer seed length, the target hit becomes more specific and faster seed search ensues (Supplementary Figure S14). However, the error tolerance decreases which can lead to inferior precision and recall even at low error, as shown with the human data set (Supplementary Figure S15). To have the best alignments, a default value of 3 is set for the default seed length.

S4.3 Match Scores

The value of a score for a match (t_m), extra signal (t_{es}) and missing signal (t_{ms}) are interconnected and relative. In our analysis that follows, t_m is fixed at 5, while penalty scores for extra and missing signals are iterated across a range of values from 0 to 5. No single pair of penalty scores work best among various species and error rates. We choose 2 and 2 as the default values for t_{es} and t_{ms}, respectively. Supplementary Figure S16 indicates the precision-recall relationship for the selected pair of default value in bold line, while other combinations are represented as grey lines.

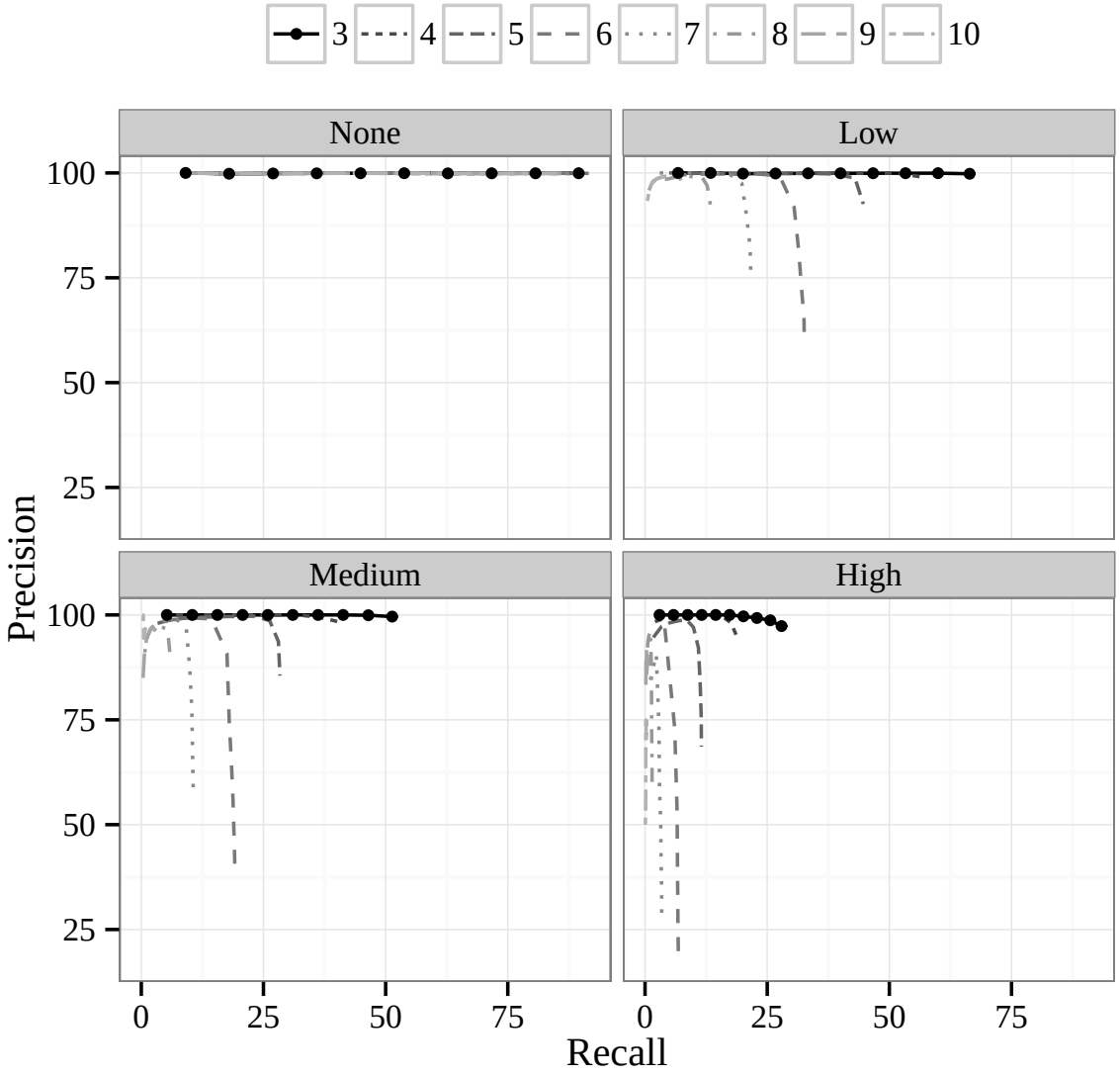


Figure S15: Precision-recall graphs for *H. sapiens* with varying seed lengths.

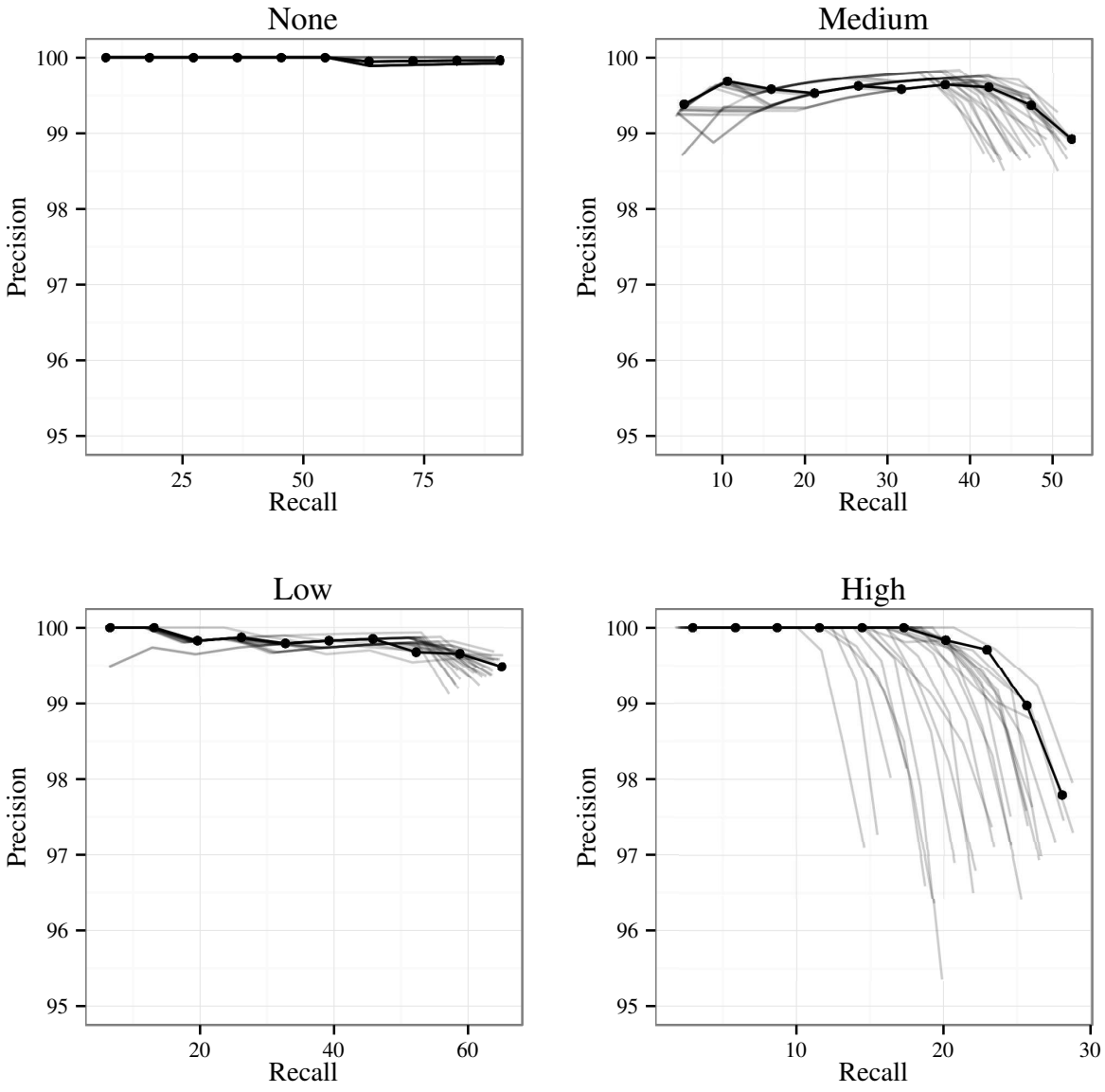


Figure S16: Precision-recall graphs for *H. sapiens* with varying match scores.

S4.4 Error Tolerance

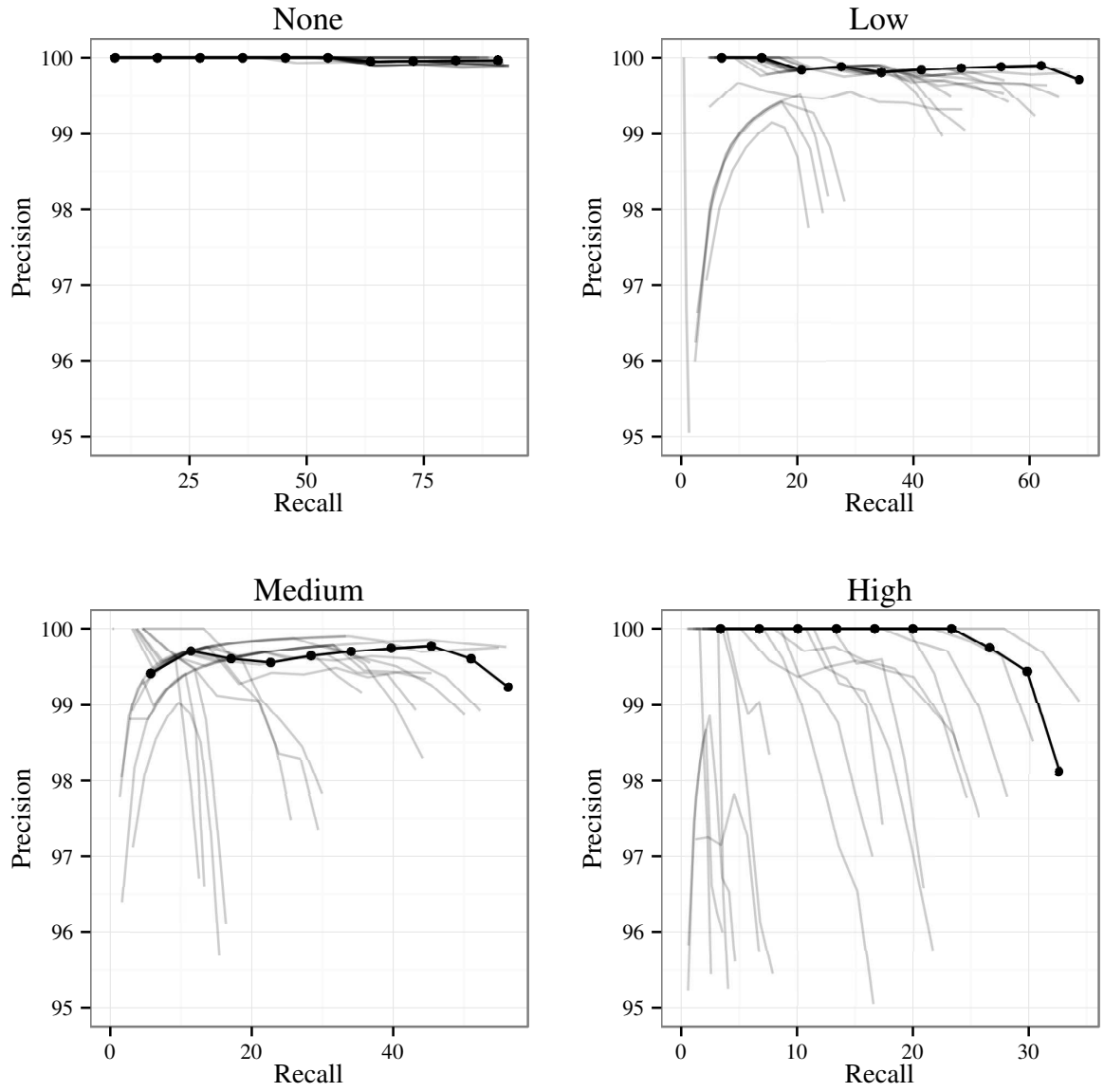


Figure S17: Precision-recall graphs for *H. sapiens* with varying error tolerances.

Having both too high or too low error tolerance is detrimental to the alignment results. With low error tolerance, data sets with higher error rate have a higher chance to be unaligned, and even worse, aligned to the wrong genomic region. With high error tolerance, running time increases as more seeds are extracted and more possible regions are extended (Supplementary Figure S18).

We considered various combinations of scaling and measurement error. We varied scaling error from 0 to 0.2, in increments of 0.05. We varied measurement error from 0 to 1250, in increments of 250. The bold line in the precision-recall graphs of Supplementary Figure S17 indicate the default value (0.05 in scaling error and 500 in measurement error) which works well at various error rates.

Meanwhile, Supplementary Figure S18 shows the running time for each of the combination of parameters. Along the horizontal axes is the running time in seconds on a logarithmic scale.

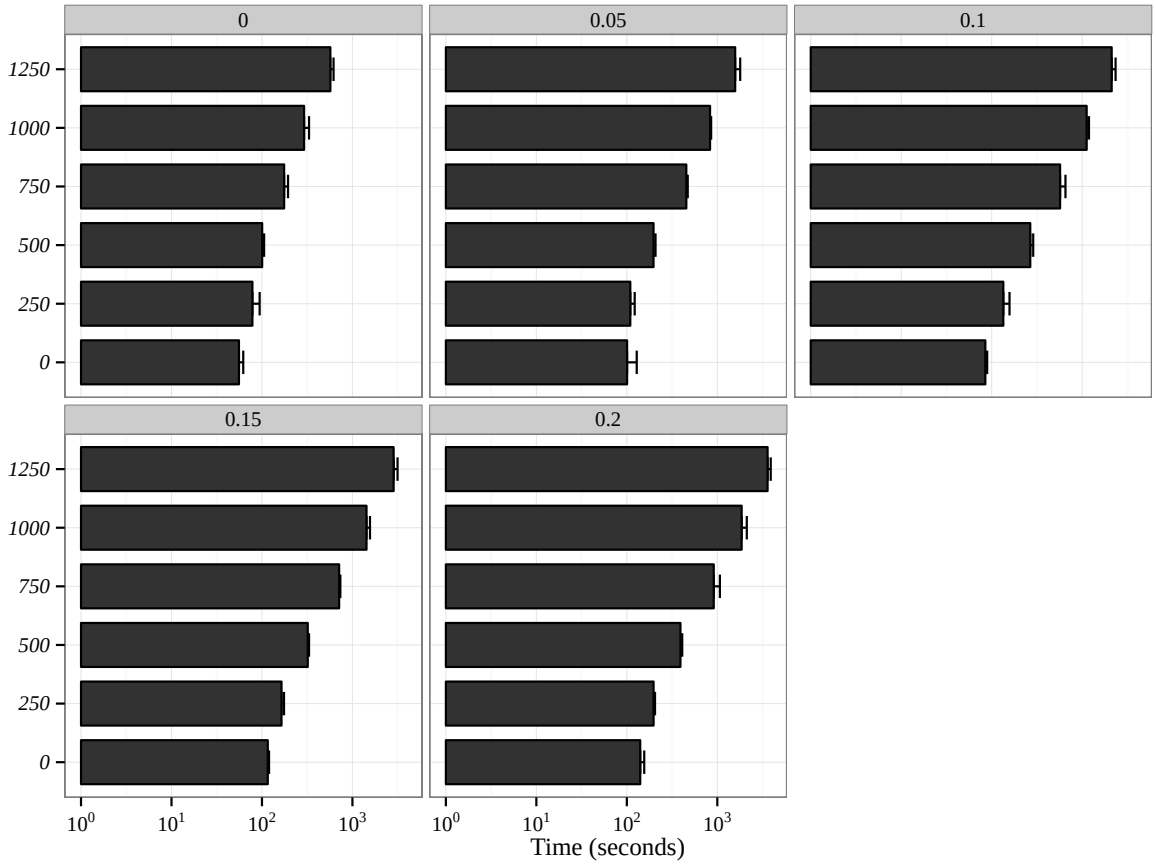


Figure S18: Running time for *H. sapiens* data set with medium error rate at varying error tolerances. The vertical axes represents measurement error while the horizontal axes represents running time in seconds. Each panel represents a different value for scaling error.

Along the vertical axes is the scaling error. Each panel represents different values for the measurement error.

S4.5 Alignment Joining Mode

Since the insertion/deletion mode can help OMBlast tolerate consecutive errors, leading to interrupted extension, enabling this module offers better accuracy than disabling this module. This is true for the data sets without SVs (Supplementary Figure S19) and with SVs (Supplementary Figure S20). The precision-recall graphs shown represent the *H. sapiens* data set with medium error rate.

Hence, enabling insertion/deletion mode suits most cases except inversion. Enabling inversion mode provides more freedom in alignment. It leads to a slightly diminished precision if the data set does not contain any inversions (Supplementary Figure S19). Therefore it is recommended that inversion mode is enabled when the data set contains inversions, or inversion is the focus of downstream analysis (Supplementary Figure S20).

In terms of running time, enabling insertion/deletion mode or inversion mode runs slightly slower comparing to disabling all alignment joining modules (Supplementary Figure S21).

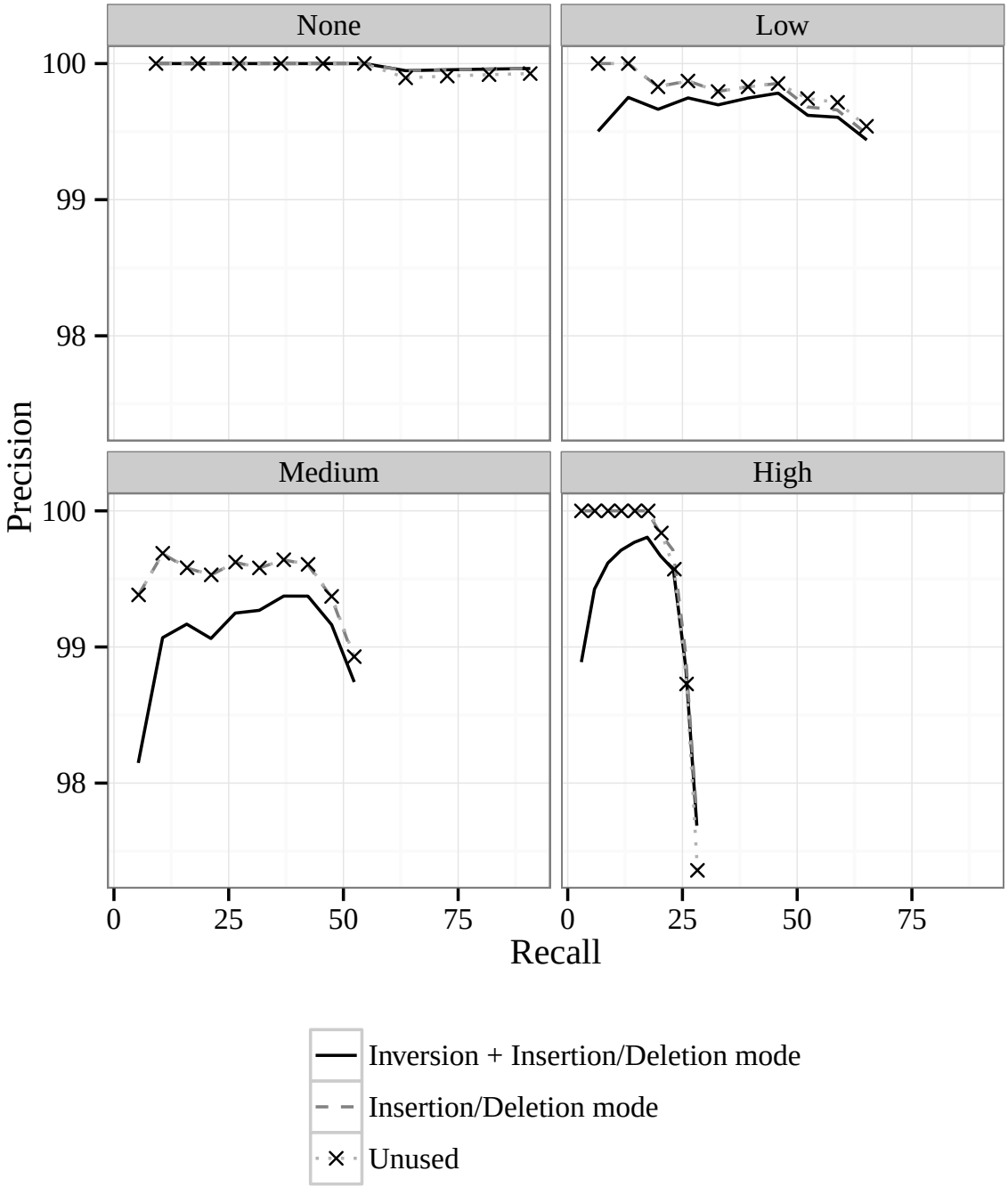


Figure S19: Precision-recall graphs for various alignment joining modes for the *H. sapiens* data set at medium error rate without SVs.

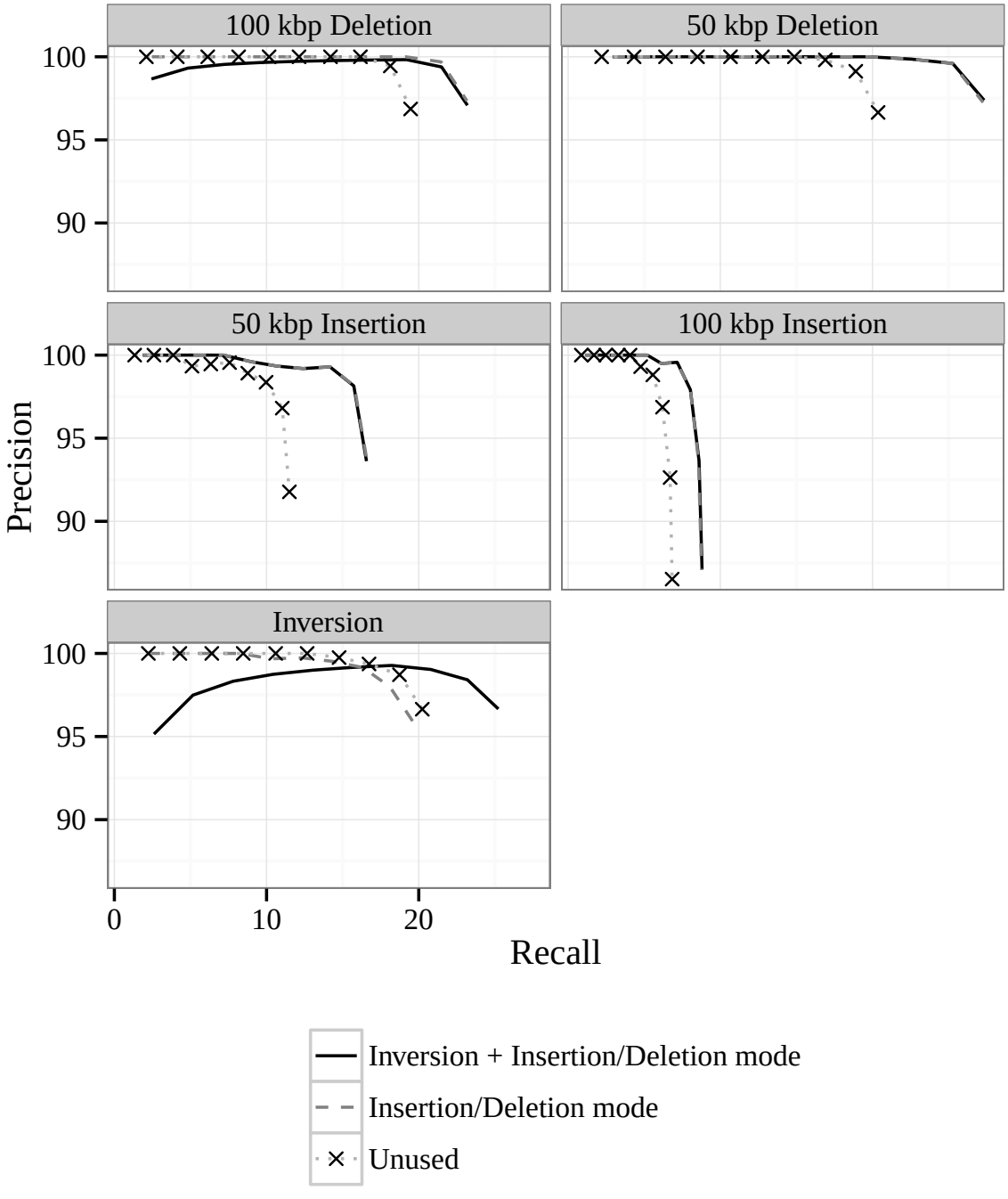


Figure S20: Precision-recall graphs for various alignment joining modes for the *H. sapiens* data set at medium error rate in the presence of SVs.

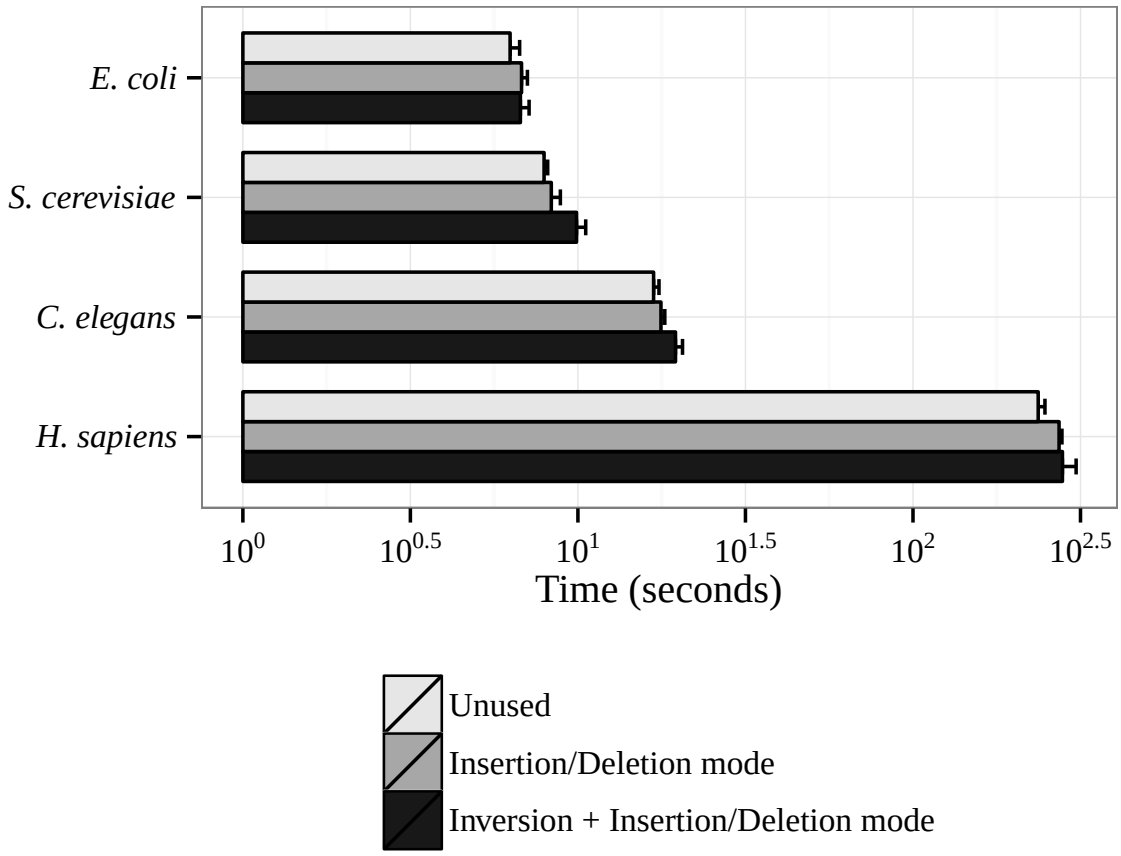


Figure S21: Running time with various alignment joining modes across all species.

S5 Simulation Data

S5.1 Reference Sequences

Table S4: Accession IDs of genomes downloaded from NCBI.

E. coli	*S. cerevisiae*	*C. elegans*	*H. sapiens*	
NC_000913.3	NC_001133.9	NC_003279.8	NC_000001.11	NC_000017.11
	NC_001134.8	NC_003280.10	NC_000002.12	NC_000018.10
	NC_001135.5	NC_003281.10	NC_000003.12	NC_000019.10
	NC_001136.10	NC_003282.8	NC_000004.12	NC_000020.11
	NC_001137.3	NC_003283.11	NC_000005.10	NC_000021.9
	NC_001138.5	NC_003284.9	NC_000006.12	NC_000022.11
	NC_001139.9		NC_000007.14	NC_000023.11
	NC_001140.6		NC_000008.11	NC_000024.10
	NC_001141.2		NC_000009.12	
	NC_001142.9		NC_000010.11	
	NC_001143.9		NC_000011.10	
	NC_001144.5		NC_000012.12	
	NC_001145.3		NC_000013.11	
	NC_001146.8		NC_000014.9	
	NC_001147.6		NC_000015.10	
	NC_001148.4		NC_000016.10	

Figure S22: Explanation of *in silico* data generation.

Reference sequences were downloaded from NCBI (ftp://ftp.ncbi.nlm.nih.gov/) on February 13, 2015. The accession IDs of the downloaded data are given in Supplementary Table S4.

These reference genomes were pre-processed for our simulation experiments as follows. Since not all of the programs supported multiple chromosomes as input, we concatenated all of the chromosomes of each genome into a single file, following the order in Supplementary Table S4. Then, all locations of the nicking enzyme sites were identified. Next, the reference sequences were converted to a sequence of "0" and "1" to represent the absence and presence of a signal, respectively. This procedure is depicted in Supplementary Figure S22.

S5.2 Reference Optical Maps

Table S5 shows some statistics about the reference optical maps that were generated, including the size of each genome, the total number of signals and the average number of bases between signals.

Table S5: Information about the optical maps from the *in-silico* digested reference genomes.

Organism	Genome Size (Mbp)	Total Signals	Average Bases Between Signals (kbp)
E. coli	4.6	683	6.8 ± 7.3
S. cerevisiae	12.1	1953	6.2 ± 6.7
C. elegans	100.3	14837	6.8 ± 8.0
H. sapiens	3088.3	377143	8.2 ± 83.2

S5.3 Generation of Simulated Data

Table S6: Error parameters of the simulated data.

Error Rate	None	Low	Medium	High
Extra Signal Rate	0	0.000005	0.00001	0.00002
Missing Signal Rate	0	0.05	0.1	0.2
Scaling	0	0.02	0.04	0.08
Measurement (bp)	0	500	500	500
Resolution (bp)	0	1200	1200	1200

From each of these reference genomes, 1000 molecules were extracted at random. In this simulation, we assumed all reference genomes are non-circular. Data generation for a certain setting was repeated 3 times each in order to obtain 3 replicate data sets. Various levels of errors were then introduced, which we have categorized as none, low, medium and high error rate. The meaning for each of these categories are defined in Supplementary Table S6. The justification for the error parameters and the respective distributions for data generation are reported elsewhere (manuscript in preparation). Data generation was done in 4 steps, which were ordered according to the experimental flow.

S5.3.1 Initial Molecule Position

A random molecule size was derived from an exponential distribution with the average size as 200 kbp. A random position on the reference was selected. If the end of the random molecule size exceeded the end of the reference genome, then the end of the molecule was truncated accordingly.

S5.3.2 SV Incorporation

Artificial data with structural variations (SV) were devised to evaluate the alignment methods. The SVs that were considered included insertions, deletions and inversions.

We used the following procedure for simulating insertions and deletions. Sequences of lengths 50 kbp or 100 kbp were either inserted or deleted. When generating the optical maps, a

sufficient length was sampled from the genome so that the lengths *after* the SV was introduced would be close to 200 kbp. Simulated data with insertion or deletion was generated using an average size of (Original Average Size − Insertion/Deletion Size). Deletions were performed in the middle of the optical map. As for insertions, we ensured that the probability of a signal in the inserted region was equal to the overall probability from the original reference genome. As with deletions, the inserted region was added to the middle of the optical map.

The procedure for generating synthetic data with inversions was relatively easier. The second half of each molecule was inverted so as to simulate an inversion breakpoint.

S5.3.3 Intrinsic Error in Molecules

Three sources of errors were included in simulated data generation. First, missing signals were removed from the molecules, such that each signal was assumed to have an independent chance of missing, as indicated in the row *missing signal rate* of Supplementary Table S6. Next, extra signals were introduced into the molecules. The number of extra signals was determined by a Poisson distribution with an average number taken as (the derived molecule size $\times$ extra signal rate), as indicated in the row *extra signal rate* of Supplementary Table S6. Finally, scaling error was introduced using a Cauchy distribution with the median defined as 1, and standard deviation indicated in the row *scaling* of Supplementary Table S6.

S5.3.4 Error in Molecule Imaging and Measurement

Other than the data set without any error, all error induced in this section remains constant because such errors are directly linked to the equipment quality. To simulate the resolution error set with res base pairs, we iterate from the beginning of the molecule the signal (sig_f). For every signal iterated from the end of the molecule sig_r to the signal sig_f, there was a chance c_m to merge all signals in between sig_f and sig_r according to the equation below:

$$c_m = \frac{1}{1 + e^{-0.01 \times (sig_r - sig_f - res)}} \tag{2}$$

where res is indicated in the row *resolution* of Supplementary Table S6.

Upon the merging of signals, a new signal was set as the mean position of all merged signals. To induce measurement error, a uniform distribution was employed to modify the position of all signals within a restricted range, as indicated in the row *measurement* of Supplementary Table S6.

S6 Without Structural Variations

S6.1 Simulated Data Preprocessing

In the current simulated data generation model, the size of the first and last segments of an optical map was usually smaller than the signal-to-signal length. In OMBlast, RefAligner and TWIN, the first and last segments were neglected during the alignment process. In Valouev and SOMA, these segments were all considered and led to poorer performance. Therefore, the first and last segments were removed for these two alignment methods.

TWIN ran fast in general but for more than 10 hours and terminated with an error in a few cases. We believe the presence of repetitive signal patterns was the cause of this. Thus, molecules with repetitive patterns were removed, as summarized in Supplementary Table S7, so that TWIN terminated normally. Examples of the molecules with repetitive patterns are shown in Supplementary Table S8. In the analysis, we assumed the removed molecules were "not aligned". By removing these molecules (which accounted for less than 2% of the entire data set) before alignment, TWIN ran much faster and achieved a slightly lower recall level. The results for TWIN in the manuscript and this supplementary document have had such pre-processing of the data performed.

Since Valouev, SOMA and TWIN do not support multiple chromosomes, in this simulation, we joined all of the chromosomes into a single reference for each species. The statistics of the genomes are shown in Table S5.

Table S7: IDs of removed molecules for TWIN.

Data set			IDs of removed optical maps
Species	Error rate	Replicate	
S. cerevisiae	none	1	56, 92, 134, 237, 395, 543, 566, 599, 753, 878, 925, 990, 999
S. cerevisiae	none	2	40, 94, 316, 343, 424, 470, 515, 563, 598, 920, 935, 995
S. cerevisiae	none	3	106, 116, 124, 179, 180, 227, 276, 422, 434, 642, 656, 743, 836, 907, 993
C. elegans	none	1	169, 360, 604, 889, 968
C. elegans	none	3	541, 622
H. sapiens	none	1	211

Table S8: Example repetitive signals that were removed to accommodate TWIN.

Data set				Segment details of the example optical maps
Species	Error rate	Replicate	Optical Map ID	
S. cerevisiae	none	1	134	3953;976;129;5483;11562;5481; 7166;**35;44;35;35;44;35;35;35;35; 44;35;44;35;44;44;35;44;35;44; 44;44;35;44;44;44;35;35;44;44; 44;35;44;44;134;44**;1537;1060; 1963;9656;2002;1277;170;5759; 3302;2729;13120;2716;1686;2325; 7472;2110;4653;6089;13684;6486; 2287;881
C. elegans	none	1	360	15646;5806;25786;2456;**183;367; 367;183;367;175;551;183;551;367; 175;735;367;183;183;367;183;183; 367;367;183;183;183;183;183;367; 183;183;183;183**;887;5426;8457; 20829;15200;11713;15422;12569
H. sapiens	none	1	211	37501;2790;5449;9995;5529;5624; 5574;5141;770;3780;3270;720;28773; 5316;3394;23704;9447;13839;1470; 16011;4532;2875;109;22;70;6208; **45;45;45;45;45;45;45;45;45;45;45; 45;45;45;45;45;45;45;45;45;45;45**;707; 4197;20;1580;**64;64;64;64;64;64; 64;64;64;64;64;64;64;64;64;64;64; 64;64;64;64;64;64;64;64;64;64;64; 64;64;64;64**;41495

S6.2 Additional Results

For the remainder of this section, we present results for species other than *H. sapiens* and without any SVs. These results have been omitted from the manuscript due to space limitations.

In the manuscript, the running time and memory usage for various species at medium error rate was shown in Figure 2 and Supplementary Figure S23. Here, we show the running time and memory usage results for all error rates, a species at a time. The running time results shown are for *E. coli* (Supplementary Figure S24(a)), *S. cerevisiae* (Supplementary Figure S24(b)), *C. elegans* (Supplementary Figure S24(c)) and *H. sapiens* (Supplementary Figure S24(d)). The memory usage results shown are for *E. coli* (Supplementary Figure S25(a)), *S. cerevisiae* (Supplementary Figure S25(b)), *C. elegans* (Supplementary Figure S25(c)) and *H. sapiens* (Supplementary Figure S25(d))

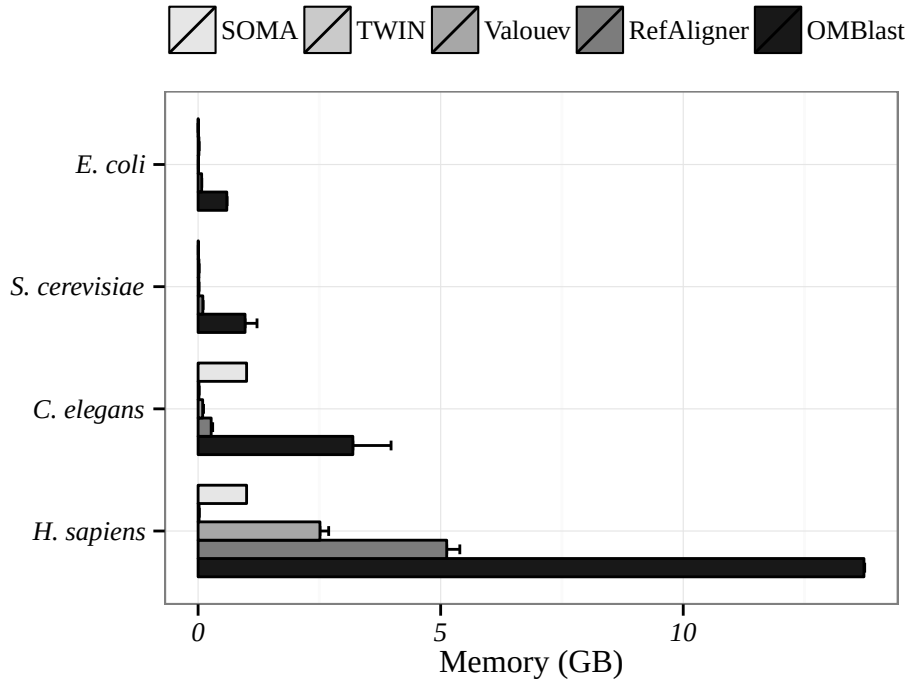
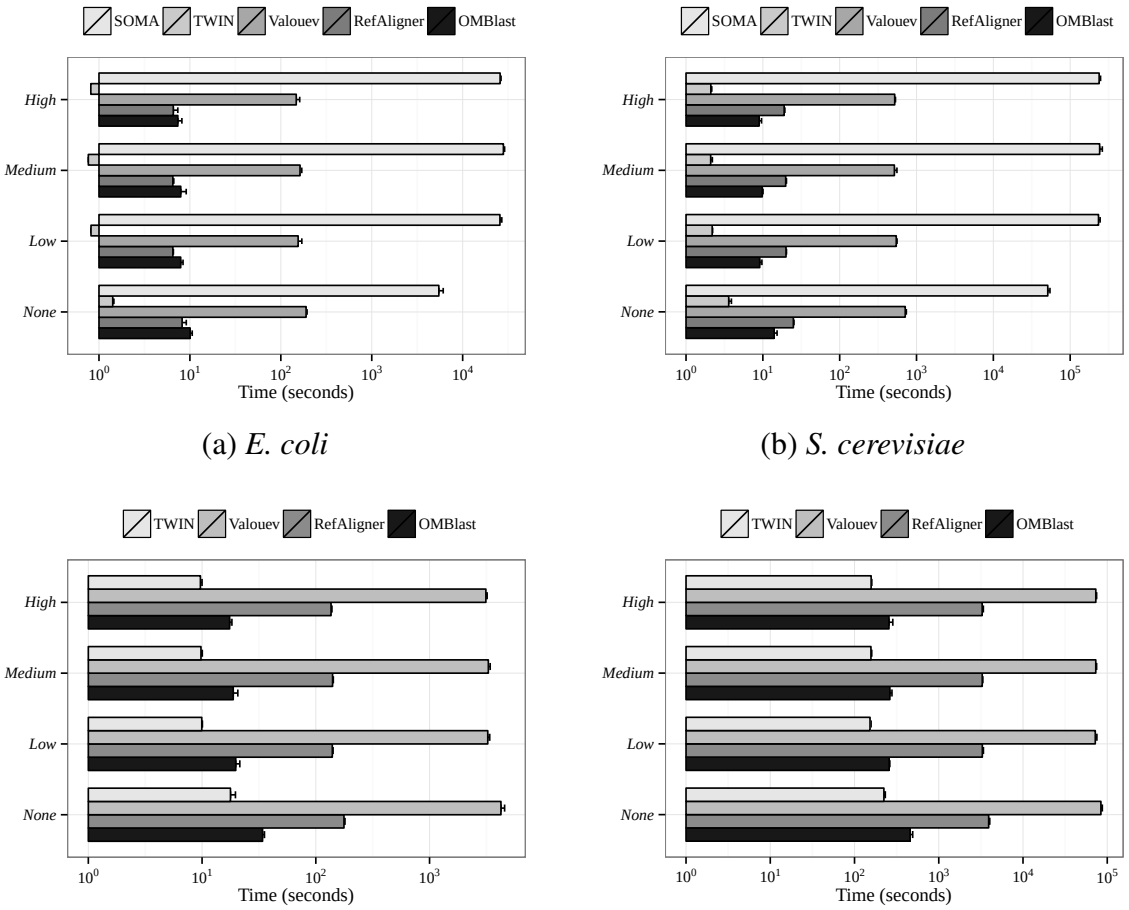


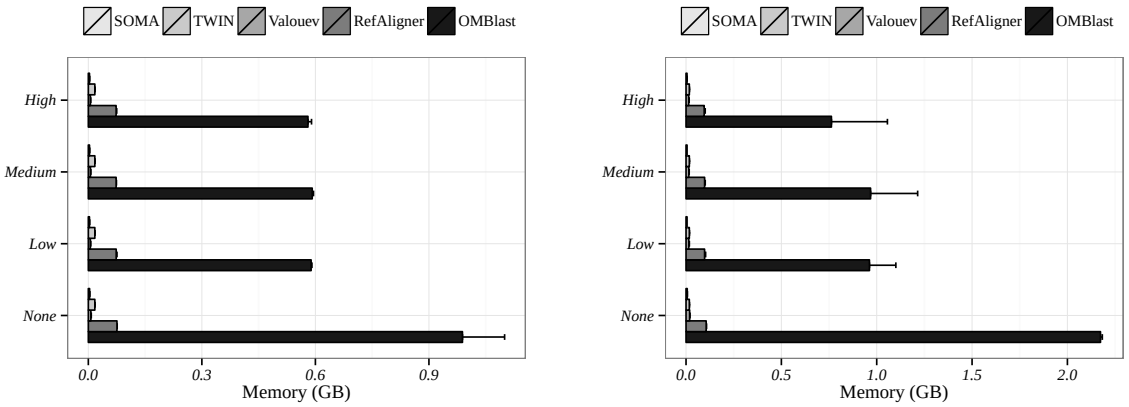
Figure S23: Memory usage of different alignment methods at medium error across four species, without any SVs. Along the horizontal axes is the memory usage in GB. (*) Results of SOMA on the *C. elegans* and *H. sapiens* data sets are missing in this graph.



(a) *E. coli* (b) *S. cerevisiae*

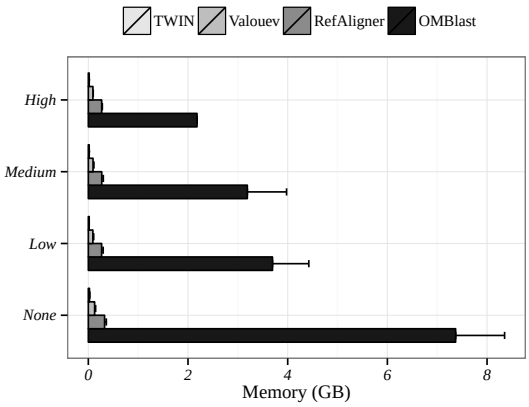
(c) *C. elegans* (d) *H. sapiens*

Figure S24: Alignment speed (user time) of different alignment methods without SVs in the data (all species).

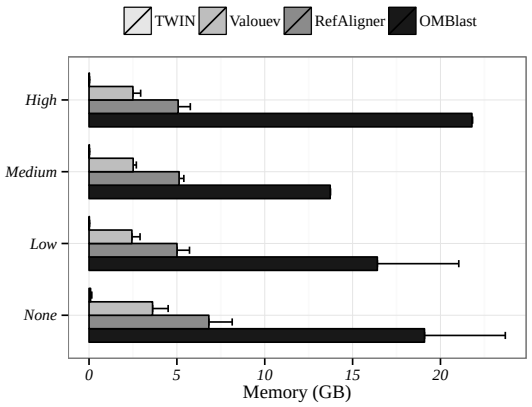


(a) *E. coli*

(b) *S. cerevisiae*



(c) *C. elegans*



(d) *H. sapiens*

Figure S25: Memory usage of different alignment methods without SVs in the data (all species).

To check the minimum memory requirements for OMBlast on the *H. sapiens* data set, we assigned the maximum memory available to the Java virtual machine by using the "-Xmx" parameter. Supplementary Figure S26 reports on the relationship between running time and maximum memory available to OMBlast on this data set with medium error rate. When less than 100 MB was available, an "OutOfMemoryError" exception was thrown by the program, indicating that OMBlast did not have enough memory for the alignment. As the memory available to the Java virtual machine increased, running time decreased. However, as the graph shows, once the available memory reaches 2 GB, no substantial improvements in running time were observed.

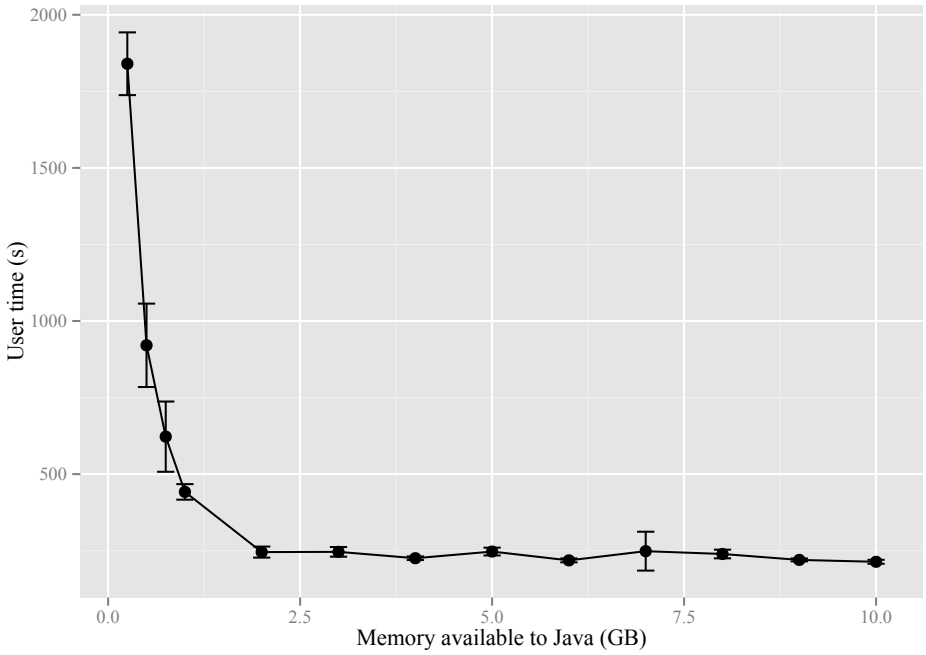


Figure S26: Running time in seconds with respect to the maximum memory assigned to OMBlast for aligning the *H. sapiens* data set at medium error rate. The errors bars represent standard deviations across 5 replicates.

The manuscript presented the precision-recall graphs for *H. sapiens* as Figure 3. Here, we present the precision-recall graphs for *E. coli* (Supplementary Figure S27), *S. cerevisiae* (Supplementary Figure S28) and *C. elegans* (Supplementary Figure S29).

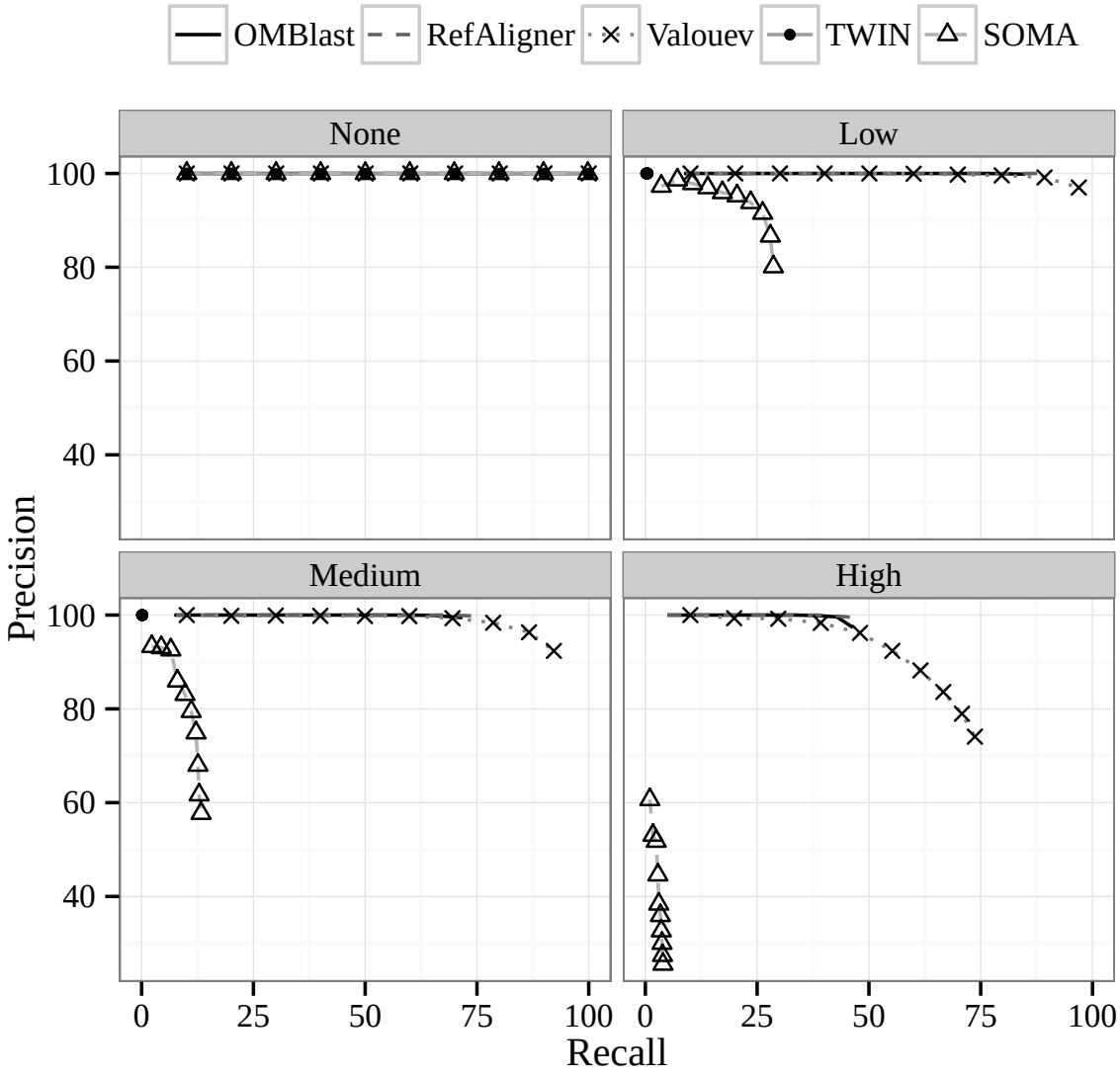


Figure S27: Precision-recall graphs for *E. coli* without any SVs.

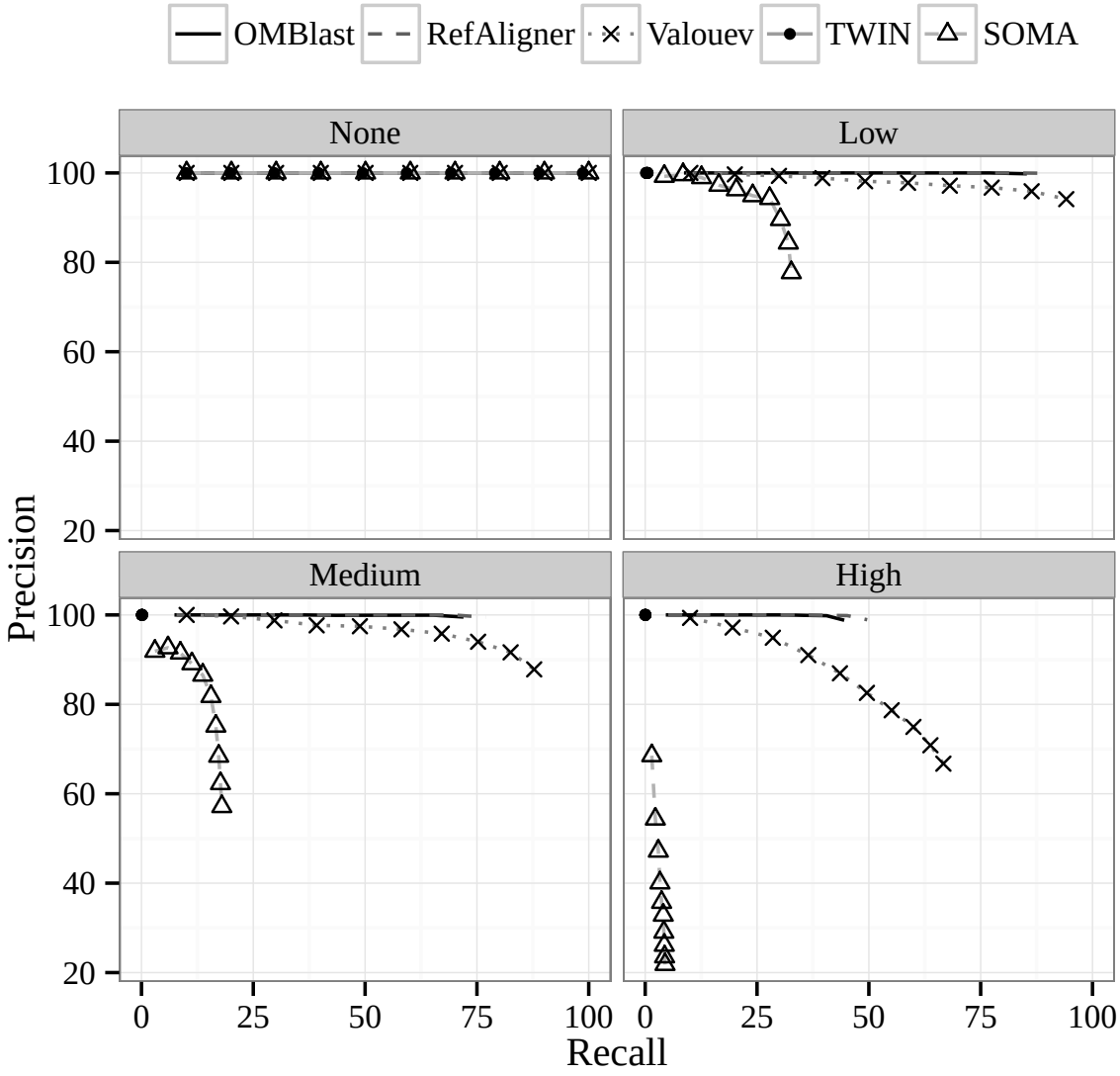


Figure S28: Precision-recall graphs for *S. cerevisiae* without any SVs.

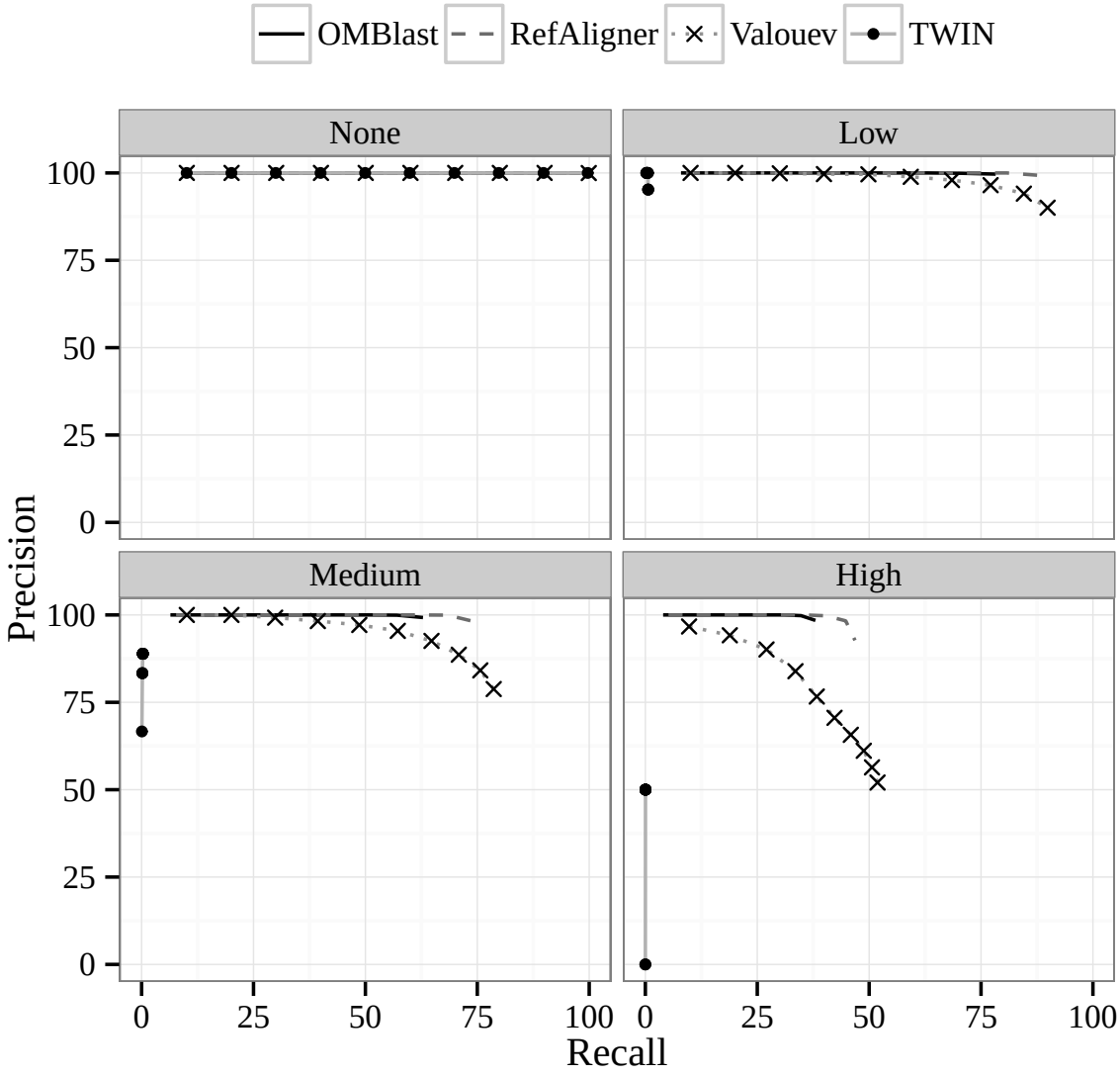


Figure S29: Precision-recall graphs for *C. elegans* without any SVs.

Here, we investigated the effects of extra and missing signals on alignment performance. As a baseline, only measurement and resolution error was applied to the data set. Extra or missing signals were then introduced independently in two data sets. For the last data set, both extra and missing signals were added, which was equivalent to the data set with medium error rate. The performance of OMBlast, RefAligner and Valouev was slightly affected by extra or missing signals, and the effects were additive, as shown by poorer performance when both extra and missing signals were present. Both precision and recall level of SOMA declined significantly upon introduction of extra and missing signals. Even at baseline, the recall level of SOMA remained low. We believe that this could be attributed to the stringent selection of unique matches from the alignment results by SOMA. Interestingly, we found that TWIN still yields a very low recall level even when no extra or missing signals were introduced. One potential explanation for the inferior performance of TWIN was that it required the whole optical map to be aligned, making it very susceptible to resolution error where close signals could be combined. The precision-recall graphs for *E. coli* (Supplementary Figure S30), *S. cerevisiae* (Supplementary Figure S31), *C. elegans* (Supplementary Figure S32) and *H. sapiens* (Supplementary Figure S33) are presented below.

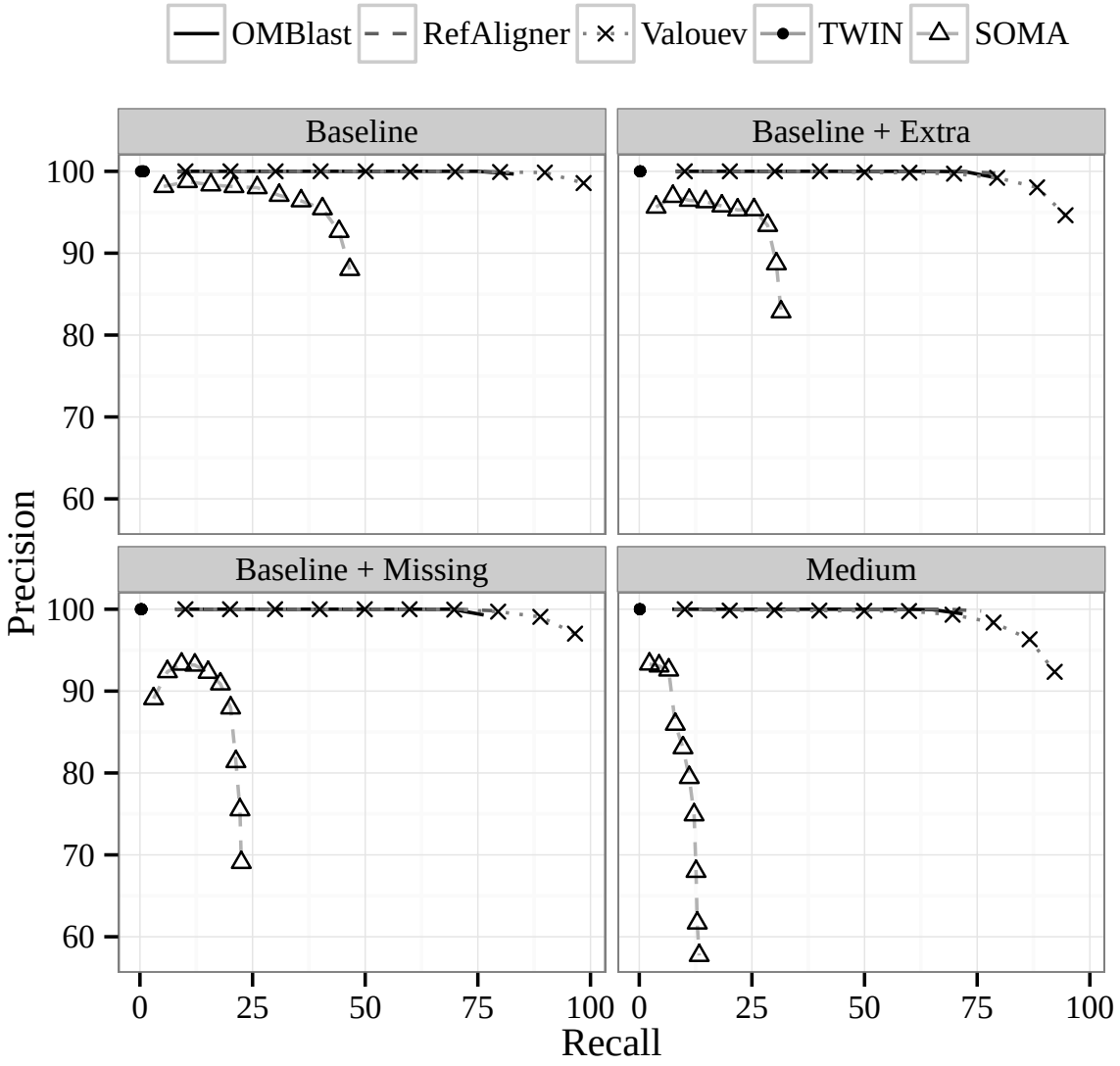


Figure S30: Precision-recall graphs for *E. coli* without any SVs.

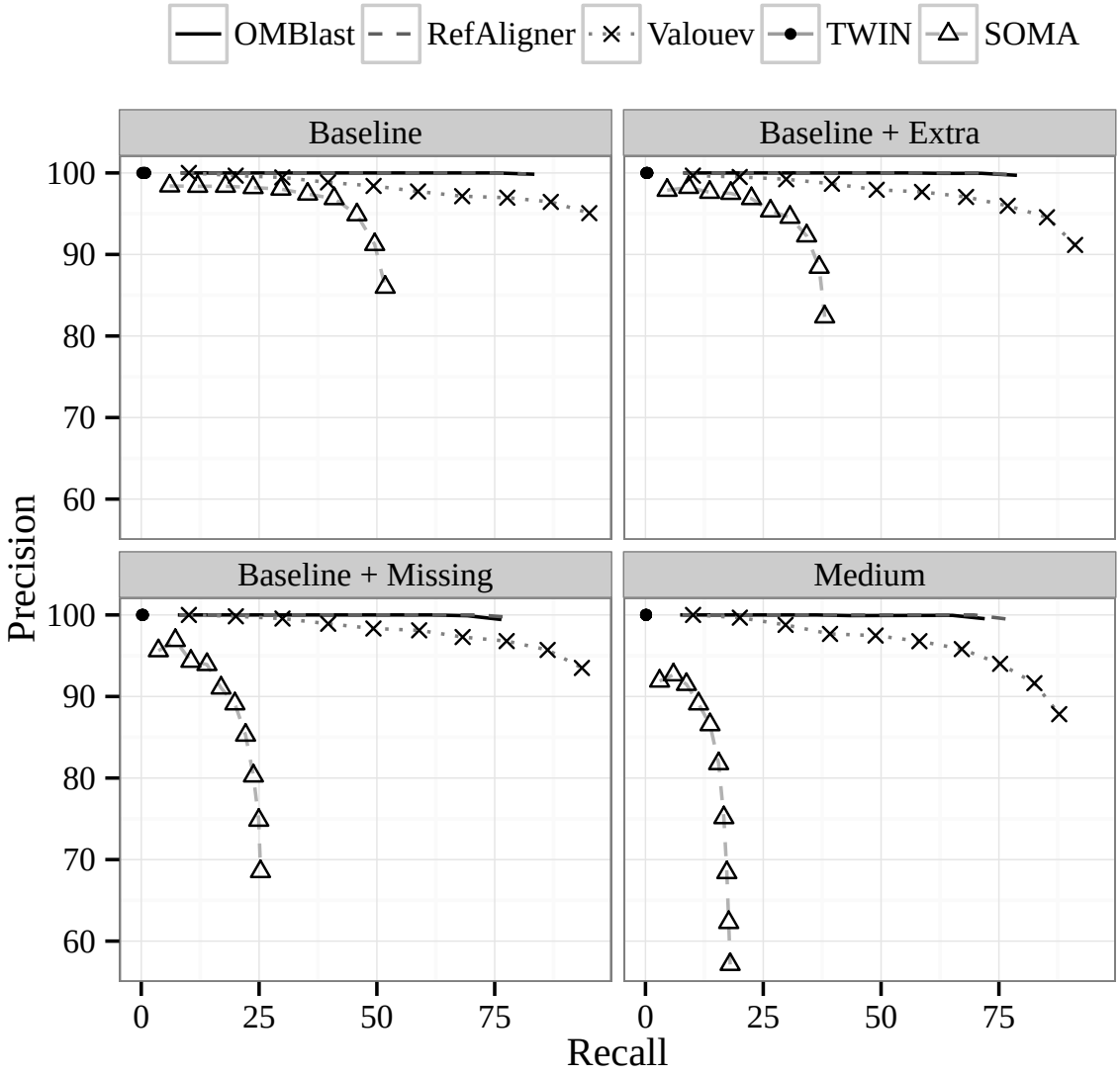


Figure S31: Precision-recall graphs for *S. cerevisiae* without any SVs.

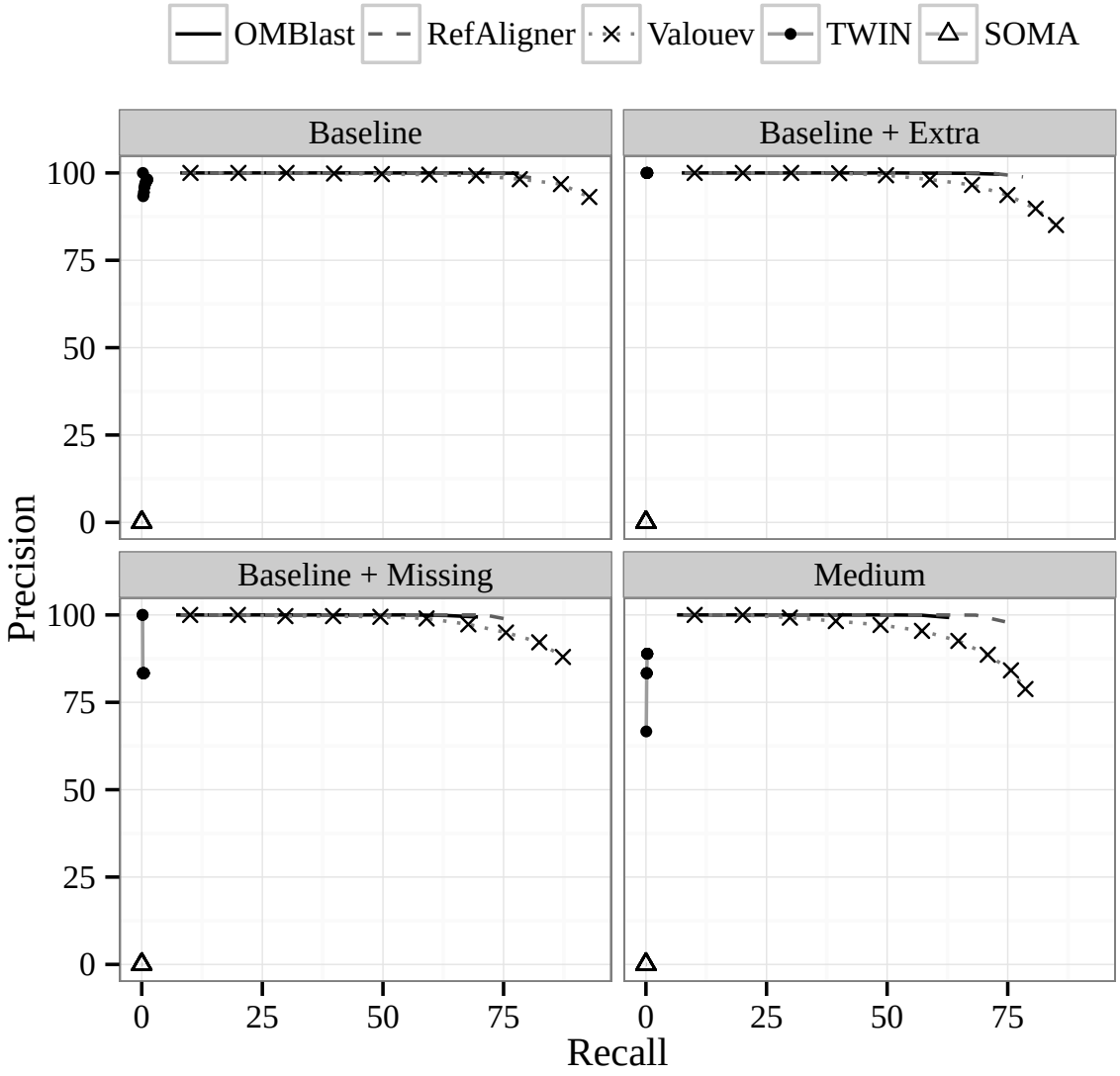


Figure S32: Precision-recall graphs for *C. elegans* without any SVs.

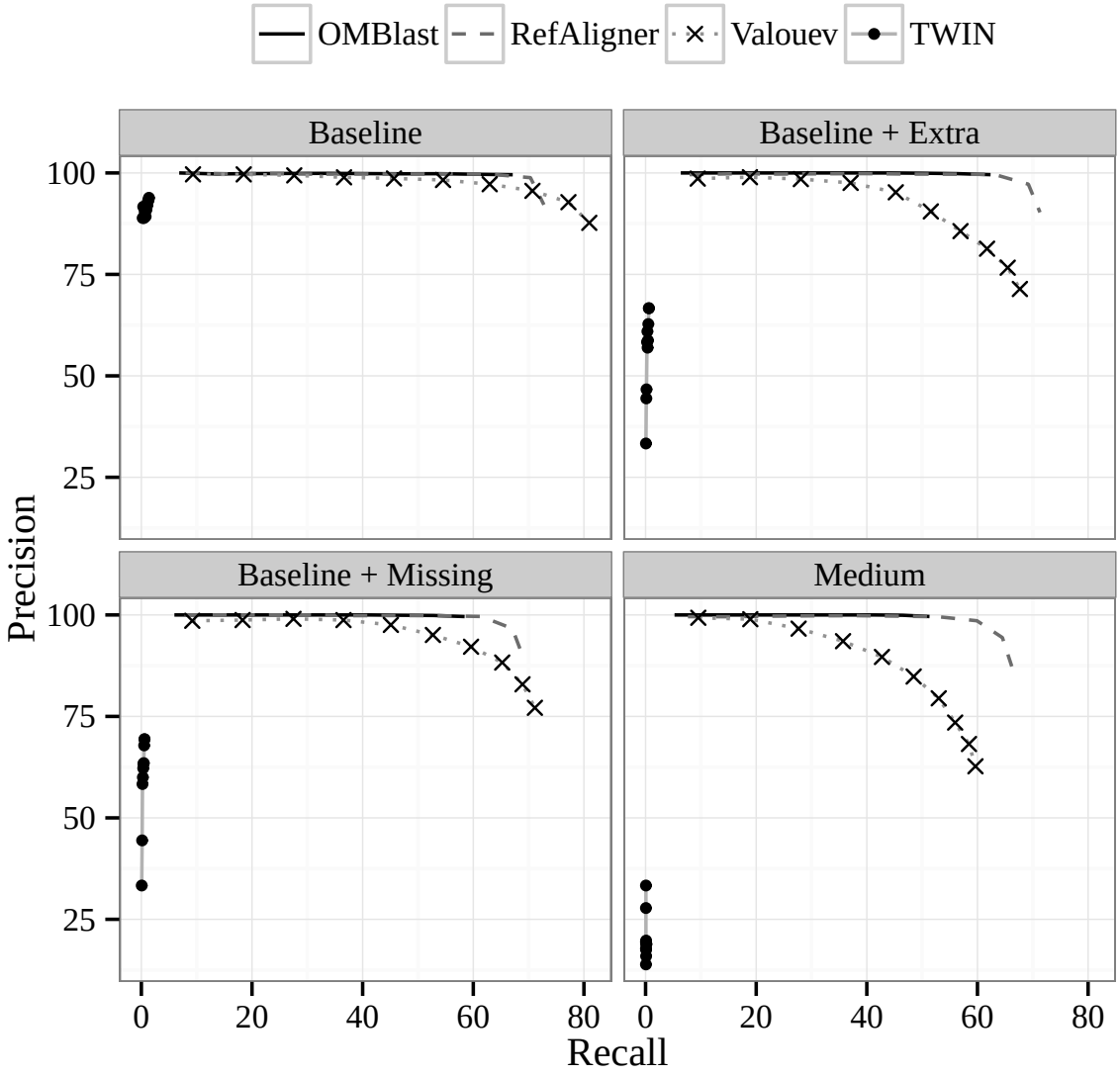
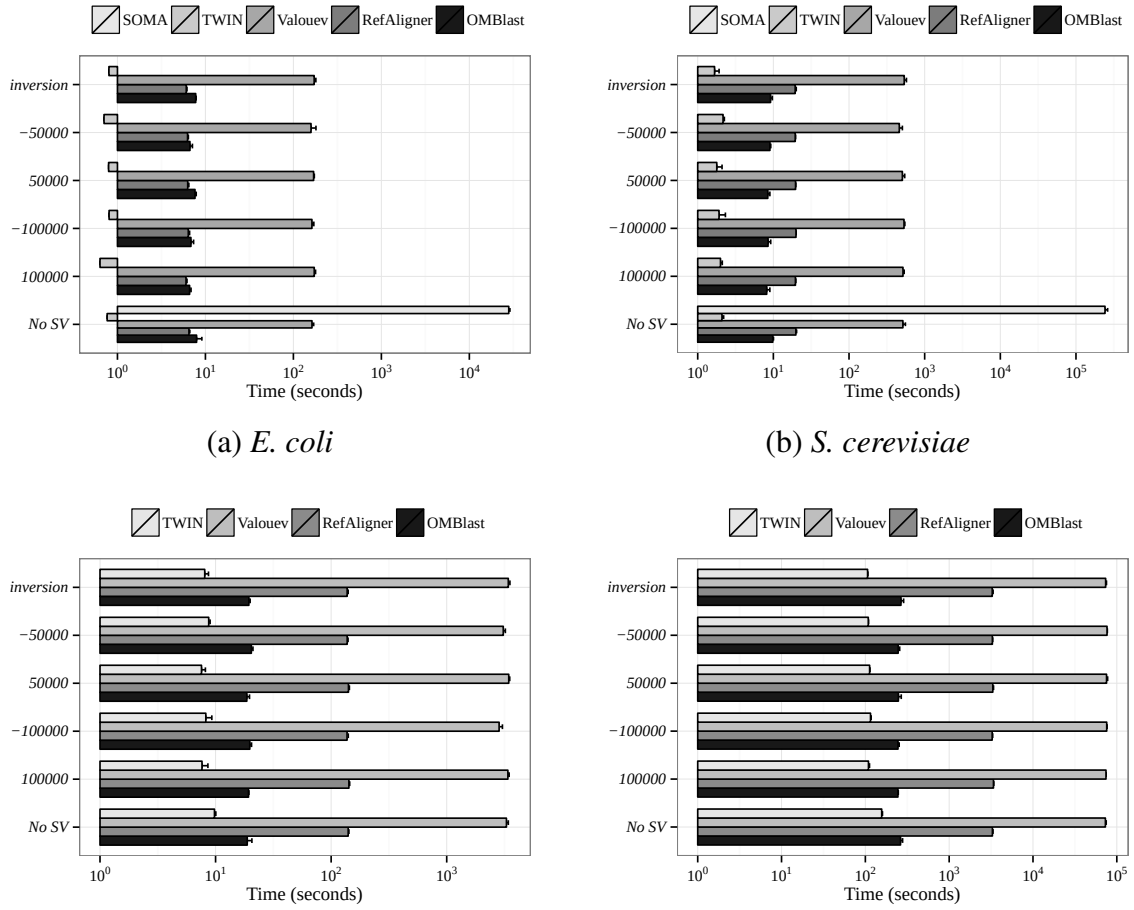


Figure S33: Precision-recall graphs for *H. sapiens* without any SVs.

S7 With Structural Variations

S7.1 Additional Results

The existence of SVs do not have a large impact on the running time and memory usage of these alignment methods. Here, we show the running time (Supplementary Figure S34) and memory usage results (Supplementary Figure S35) for all error rates, a species at a time.



(a) *E. coli*

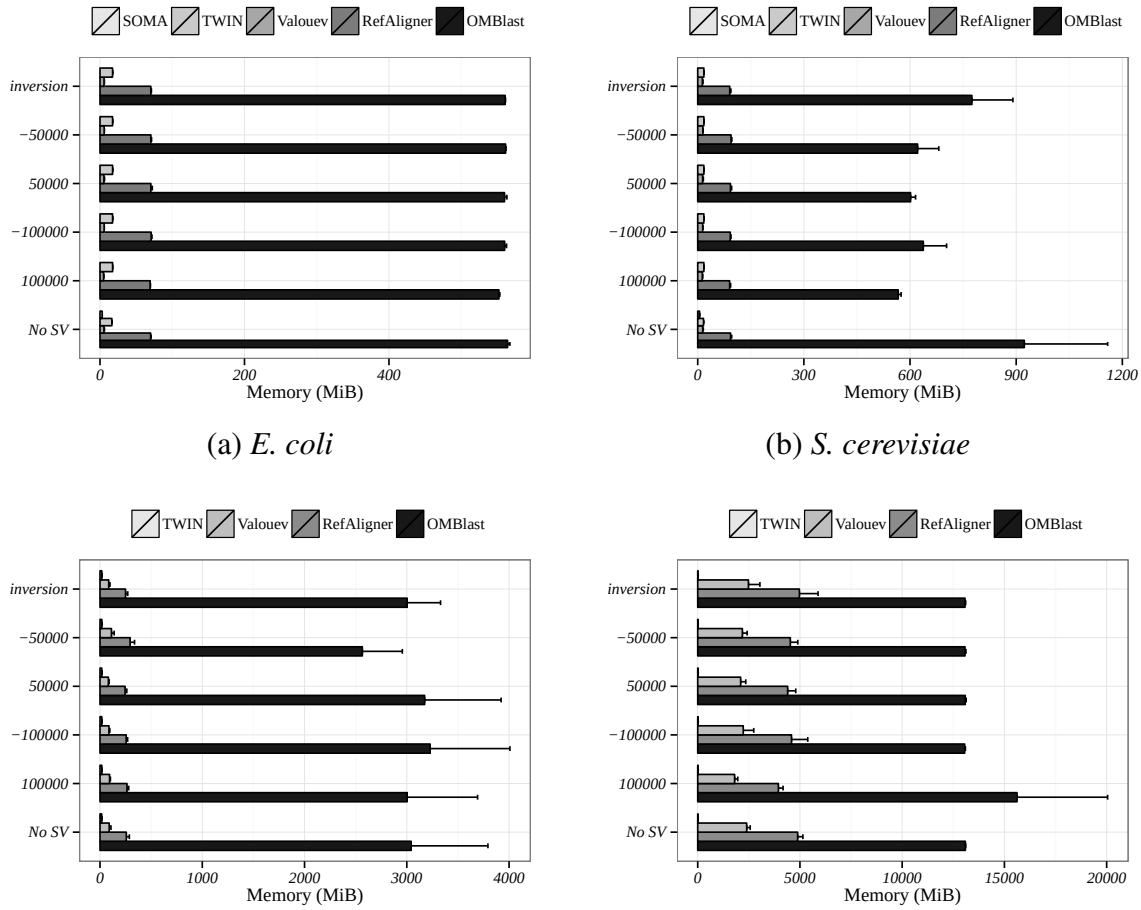
(b) *S. cerevisiae*

(c) *C. elegans*

(d) *H. sapiens*

Figure S34: Alignment speed (user time) of different alignment methods at medium error with various SVs in the data (all species).

In the manuscript, Figure 4 presented the precision-recall graphs for *H. sapiens* with insertions, deletions and inversions at medium error rate. Here, we present the precision-recall graphs at same experiment conditions but for *E. coli* (Supplementary Figure S36), *S. cerevisiae* (Supplementary Figure S37) and *C. elegans* (Supplementary Figure S38).



(a) *E. coli*

(b) *S. cerevisiae*

(c) *C. elegans*

(d) *H. sapiens*

Figure S35: Memory usage of different alignment methods at medium error with various SVs in the data (all species). Along the horizontal axes is the memory in GB.

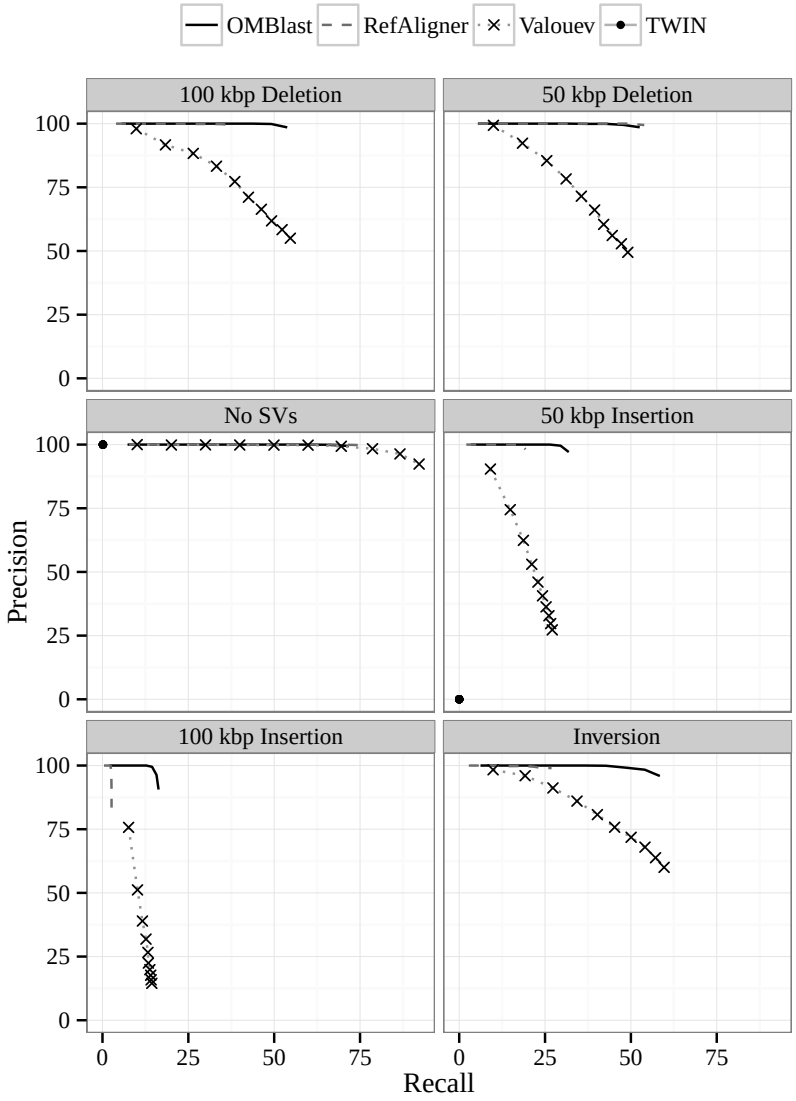


Figure S36: Precision-recall graphs for *E. coli* with SVs.

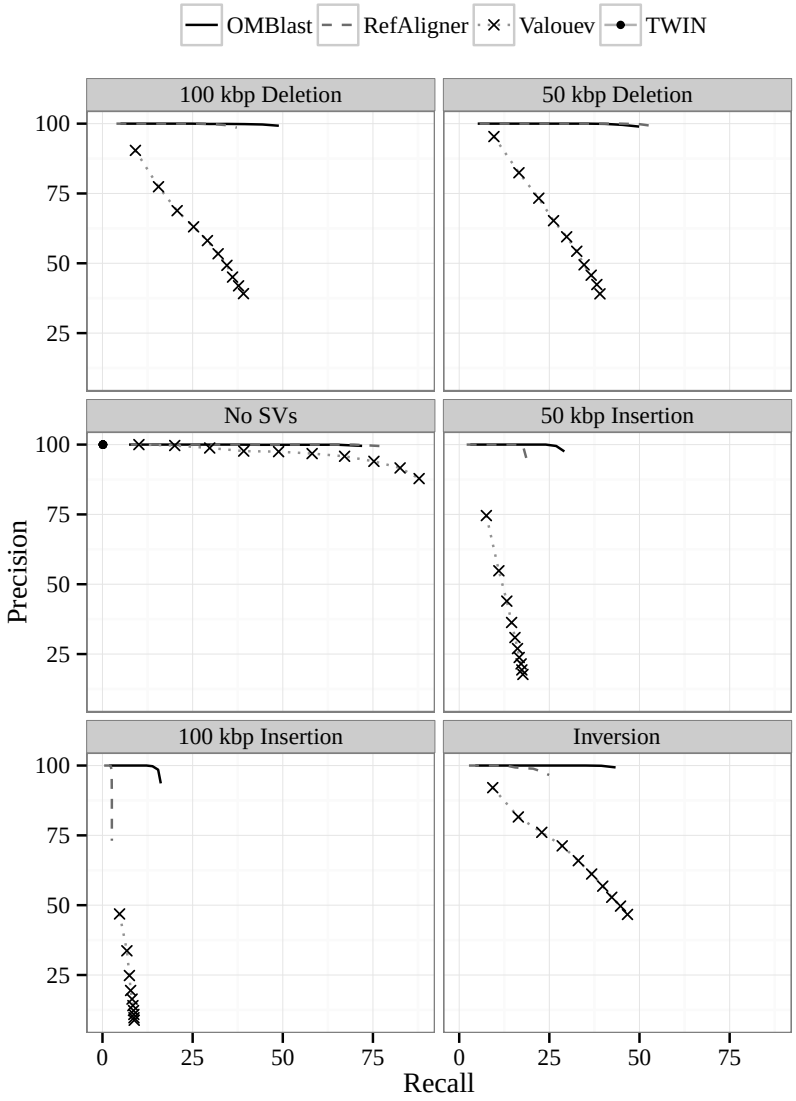


Figure S37: Precision-recall graphs for *S. cerevisiae* with SVs.

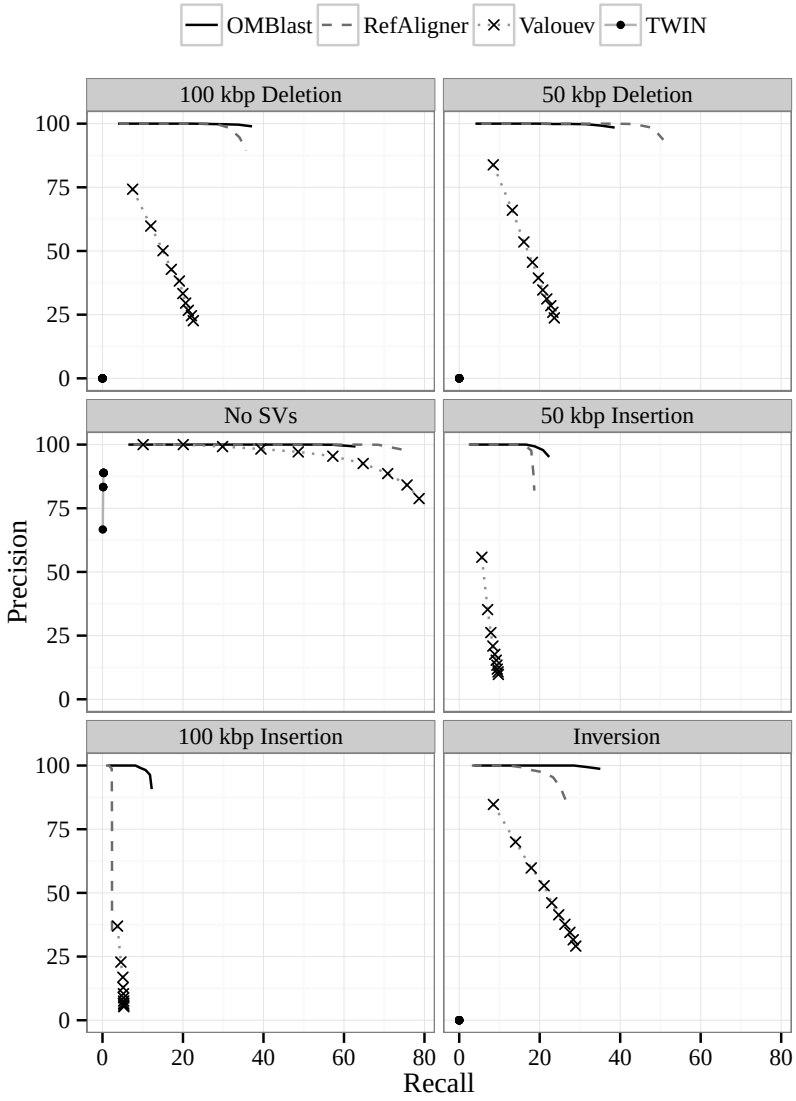


Figure S38: Precision-recall graphs for *C. elegans* with SVs.

S8 Real Data Experiments

S8.1 Data Generation

The DNA extraction and imaging method followed previously described protocols (Lam *et al.*, 2012). Briefly, DNA of high molecular weight extracted from the *A. baumannii* cells were digested with the Nt.BspQI nicking endonuclease and repaired with a fluorescent nucleotide. The backbone of DNA is labeled by YOYO-1. The digested DNA was passed through the nanochannel in IrysChip and photographed.

S8.2 Duplicated Optical Maps

Duplicated optical maps were observed in the YH data set (Cao *et al.*, 2014), where the duplicated optical maps have similar IDs. The first slot of duplication involves IDs from 8000000000 to 8099999999 and from 8100000000 to 8199999999. The second slot of duplication involves IDs from 8200000000 to 8299999999 and from 8300000000 to 8399999999. Because of this, the optical maps with IDs in the range of 8100000000-8199999999 and 8300000000-8399999999 were removed from analysis.

S8.3 Optical Maps Filtering

Short optical maps with few signals are usually aligned with low confidence, which impairs precision. Therefore, empirically, data filtering is performed to remove these optical maps. Optical maps with sizes smaller than 100 kbp or with less than 10 signals were removed from the analysis. We noted that RefAligner and Valouev could align a larger number of short optical maps than OMBlast, but from our experience, these alignments are usually of lower confidence.

S8.4 Consistency

We used *consistency* to measure the similarity in results between pairs of alignment methods. Let the two sets of results be the sets X and Y, respectively. Consistency gives the percentage of alignments that are overlapping – that is, if the corresponding regions on the reference and query coincide with one another.

Alternatively, we can express consistency as follows:

$$\text{consistency} = \frac{X \cap Y, \text{(overlapping)}}{X \cap Y} \times 100$$

Thus, query alignments in X that are not in Y are removed from the calculation of consistency (and vice-versa).

S8.5 Additional Results

The main manuscript provided results from using Bionano's Irys System, whose optical maps were aligned with *A. baumannii* 718532 (Yim *et al.*, 2015). Here, we demonstrate that the results are similar when the same set of molecules are aligned with the representative *A. baumannii* genome ATCC 17978 from GenBank (accession ID: NC 009085.1). As with *A. baumannii* 718532, we present the pair-wise consistency in Supplementary Table S9. The running times of each alignment method on the concatenated scaffold of *A. baumannii* 718532 from

Section 4.3.5 and *A. baumannii* genome ATCC 17978 are shown in Supplementary Table S10. The fraction of the genome covered by the aligned queries ranged from 75% to 100% for each alignment method (Supplementary Table S11). Note that TWIN did not output any alignments on both references.

Table S9: Consistency of pairs of alignment methods on the representative genome ATCC_17978. Percentages represent the ratio of consistent alignments to the total number of alignments, as reported by both alignment methods.

	OMBlast	RefAligner	Valouev	SOMA
OMBlast		99.6%	73.0%	52.1%
RefAligner			82.1%	49.6%
Valouev				32.7%

Table S10: Average running time (seconds per 1000 molecules) for the concatenated scaffold on *A. baumannii* 718532 and the representative genome ATCC_17978.

	718532	ATCC_17978
OMBlast	2.9	2.3
RefAligner	4.7	4.4
Valouev	67.0	68.1
SOMA	6488.0	7233.0
TWIN	0.3	0.3

Table S11: Fraction of the genome covered by the aligned queries as reported by each alignment method on the *A. baumannii* data set for the concatenated scaffold on *A. baumannii* 718532 and the representative genome ATCC_17978.

	718532	ATCC_17978
OMBlast	85.2%	82.6%
RefAligner	98.9%	97.0%
Valouev	100.0%	100.0%
SOMA	83.1%	75.7%
TWIN	0.0%	0.0%

In addition to *A. baumannii*, we also compare all alignment methods on a data set from a more well-established genome, *E. coli*. Our results showed similar consistency (Supplementary Table S12) and running time (Supplementary Table S13) as in the *A. baumannii* data set. However, we noticed that RefAligner had extremely low recall level even when the same parameters as *A. baumannii* were used (Supplementary Table S14). Almost all genome regions were covered by aligned queries, as reported by each alignment method, with the exception of TWIN (Supplementary Table S15).

Table S12: Consistency of pairs of alignment methods on the genome of *E. coli*. Percentages represent the ratio of consistent alignments to the total number of alignments reported by various alignment methods.

	OMBlast	RefAligner	Valouev	SOMA
OMBlast		93.9%	88.7%	56.7%
RefAligner			88.2%	52.4%
Valouev				43.1%

Table S13: Average running time (seconds per 1000 molecules) on the *E. coli* data set

	E. coli
OMBlast	0.8
RefAligner	5.9
Valouev	97.4
SOMA	23928.2
TWIN	0.5

Table S14: Number of alignments reported by each alignment method on the *E. coli* data set with 127961 filtered optical maps

	E. coli
OMBlast	68217
RefAligner	9783
Valouev	124976
SOMA	28845
TWIN	0

Table S15: Fraction of the genome aligned reported by each alignment method on the *E. coli* data set

	E. coli
OMBlast	99.8%
RefAligner	99.8%
Valouev	100.0%
SOMA	95.4%
TWIN	0.0%

S8.6 Demonstration of using OMBlast on OpGen data

To demonstrate that OMBlast could also be used to align OpGen data, an ostrich data set generated using OpGen technology was downloaded for alignment (Zhang *et al.*, 2014). Optical maps smaller than 350 bp were removed. The remaining optical maps were aligned against the *in silico* digested sequence assembly.

The alignment results on the ostrich data set are summarized in Supplementary Table S16. Briefly, OMBlast ran faster than RefAligner. In this data set, both alignment methods gave low recall levels (less than 20%), which could be attributed to the use of an incomplete scaffolds rather than a complete genome as the reference. Parameters needed to be further optimized for improvement on this aspect. An example alignment from OMBlast is shown in Supplementary Figure S39, where an optical map was aligned on "scaffold53" in the sequence assembly.

Table S16: Alignment statistics of OMBlast and RefAligner on the ostrich data set.

	OMBlast	RefAligner
Total optical maps	1989698	1989698
Total alignments	370667 (18.6%)	352521 (17.7%)
Consistency	98.2%	
Depth of Coverage	76.1	78.5
Fraction of genome aligned	96.5%	99.8%
Total running time (seconds)	395565	3572622
Average running time (seconds per 1000 molecules)	199	1796

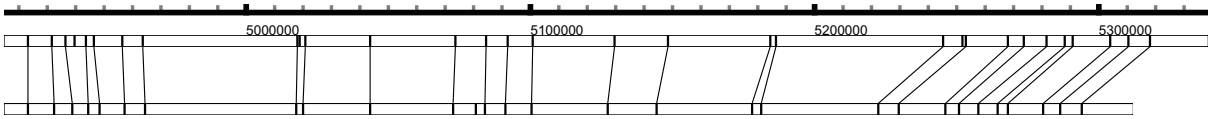


Figure S39: Example of an alignment by OMBlast of the ostrich's optical map against "scaffold53" in the sequence assembly

References

Altschul, S.F. et al (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403–410.

Cao, H.H. et al (2014). Rapid detection of structural variation in a human genome using nanochannel-based genome mapping technology. *GigaScience*, **3**(34).

Lam, E.T. et al (2012). Genome mapping on nanochannel arrays for structural variation analysis and sequence assembly. *Nature Biotechnology*, **30**(8), 771–776.

Muggli, M.D., Puglisi, S.J. and Boucher, C. (2014). Efficient indexed alignment of contigs to optical maps. In D. Brown and B. Morgenstern, editors, *Proc. 14th International Workshop on Algorithms in Bioinformatics (WABI 2014)*, volume 8701 of *Lecture Notes in Computer Science*, pages 68–81. Springer Berlin Heidelberg.

Nagarajan, N., Read, T.D. and Pop, M. (2008). Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics*, **24**(10), 1229–1235.

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.

Valouev, A. et al (2006). Alignment of optical maps. *Journal of Computational Biology*, **13**(2), 442–462.

Yim, A.K.Y. et al (2015). Draft genome sequence of extensively drug-resistant Acinetobacter baumannii strain CUAB1 from a patient in Hong Kong, China. *Genome Announcements*, **3**(3).

Zhang, G. et al (2014). Genomic data of the ostrich (struthio camelus australis). *GigaScience Database*.