OPERA-LG: Efficient and exact scaffolding of large, repeat-rich eukaryotic genomes with performance guarantees

Song Gao[1,2,3*], Denis Bertrand[1*], Niranjan Nagarajan[1]

[1]*Computational and Systems Biology, Genome Institute of Singapore, Singapore 138672, Singapore*

[2]*NUS Graduate School for Integrative Sciences and Engineering, Singapore 117456, Singapore*

[3]*South Australian Health & Medical Research Institute, North Terrace Adelaide 5000 SA, Australia*

[*]*Contributed equally*

Abstract

The assembly of large, repeat-rich eukaryotic genomes continues to represent a significant challenge in genomics. While long-read technologies have made the high-quality assembly of small, microbial genomes increasingly feasible, data generation can be prohibitively expensive for larger genomes. Fundamental advances in assembly algorithms are thus essential to exploit the characteristics of short and long-read sequencing technologies to consistently and reliably provide high-qualities assemblies in a cost-efficient manner. Here we present a scalable, exact algorithm (OPERA-LG) for the scaffold assembly of large, repeat-rich genomes that exhibits almost an order of magnitude improvement over the state-of-the-art programs in both correctness (>5X on average) and contiguity (>10X). This provides a systematic approach for combining data from different sequencing technologies, as well as a rigorous framework for scaffolding of repetitive sequences. OPERA-LG represents the first in a new class of algorithms that can efficiently assemble large genomes while providing formal guarantees about assembly quality, providing an avenue for systematic augmentation and improvement of 1000s of existing draft eukaryotic genome assemblies.

Introduction

The field of sequence assembly has witnessed a significant amount of mathematical and algorithmic study of the problem (e.g. (Kececioglu and Myers 1993) and (Nagarajan and Pop 2009)) and yet most assembly programs in use (particularly for scaffold assembly) do not have a clear objective function that they optimize, and rely on heuristics and/or manually-tuned parameters at the heart of their algorithms to piece genomes together with variable degrees of success (Nagarajan and Pop 2013). With an almost endless array of heuristics and parameter choices to try, the right combination that works well across a range of datasets may not always be apparent and new assembly tools run the risk of being tuned for the datasets on which they are benchmarked. Recent assembly competitions such as GAGE (Salzberg et al. 2012), Assemblathon (Earl et al. 2011) and Assemblathon2 (Bradnam et al. 2013) have thus played an important role in galvanizing the community and in highlighting the drawbacks of existing assemblers, including the fact that no single assembler typically tends to outperform other assemblers on all datasets, thus making the choice of an assembler a matter of guesswork for new genome assembly projects.

The prevalence of heuristic approaches in assembly owes its origins partly to several well known early results regarding its computational complexity (Peltola et al. 1983; Kececioglu and Myers 1993) (and further confirmed by recent studies (Medvedev et al. 2007; Nagarajan and Pop 2009)) that suggest that most formal definitions of various assembly problems (such as contiging and scaffolding) are computationally intractable (NP-hard). Notably though, most complexity results have been limited to worst-case analysis and relatively little has been said about average-case or parametric complexity of various assembly problems (Nagarajan and Pop 2009). For example,

while the problem of constructing contigs from read data (typically formulated as a path-finding problem) has been shown to be intractable (NP-hard) in terms of worst-case complexity (Medvedev et al. 2007; Nagarajan and Pop 2009), in practice, the problem is usually under-constrained in the absence of ultra-long reads, and trivially-computable, fragmented contig assemblies are the best we can do (Nagarajan and Pop 2009). The use of paired-end and mate-pair reads to scaffold contigs thus plays a vital role in assembly projects to significantly boost assembly quality (Pop et al. 2004; Boetzer et al. 2011; Gao et al. 2011). While worst-case analysis for the scaffolding problem also suggests that it could be computationally expensive to solve exactly, surprisingly, it is possible to design exact algorithms that require runtime polynomial in the size of the scaffold graph (Gao et al. 2011). These algorithms guarantee a scaffold assembly that minimizes discordance with the input data and thus provide an internal "quality-control" for their results. They also avoid the possibility of arbitrary and heuristic design decisions significantly impacting assembly quality in datasets dissimilar to the one on which the method was designed. Where feasible, exact algorithms that have a natural optimization criterion thus provide users with a method that always works well (when input specifications are met), without the need for extensive benchmarking to confirm their validity. As shown in Gao et al. (Gao et al. 2011) using experiments on small genomes, an exact algorithm for scaffolding simultaneously leads to more accurate as well as contiguous assemblies.

Scalable exact algorithms or hybrid-exact algorithms (those that use approximations or heuristics only where needed) may therefore potentially better address the needs of the assembly community by providing a *paradigm shift* in performance. Here we show that this is indeed feasible through algorithmic and engineering refinements that enable us to extend applicability of an exact algorithm

(Gao et al. 2011) to large genomes in a time and memory-efficient manner (referred to here as OPERA-LG) without sacrificing formal properties. We show by benchmarking against state-of-the-art scaffolders and assembly pipelines (SOAPdenovo (Li et al. 2010), ALLPATHS-LG (Gnerre et al. 2011) and SSPACE (Boetzer et al. 2011)) that this indeed provides a notable improvement in performance – several-fold increase in contiguity metrics and an order of magnitude reduction in errors – in many cases, instead of the incremental boost that is often seen.

In addition to being the first, scalable scaffolding and assembly algorithm [1] with proven performance guarantees (other recent works on scaffolding are either reported to be not scalable or do not have formal guarantees (Dayarian et al. 2010; Salmela et al. 2011; Lindsay et al. 2012)), OPERA-LG incorporates several additional features specifically tailored for producing high quality draft assemblies for large, repeat-rich genomes. These include the ability to simultaneously use data from multiple libraries (a first for stand-alone scaffolding programs and originally implemented in the Celera Assembler at the mate-pair level (Myers et al. 2000)) as is typically needed in large assembly projects, an improved edge-length estimation algorithm and an exact extension for scaffolding the repetitive sequences that typically confound assembly tools. Based on our experiences with recent genome assembly projects, we also highlight OPERA-LG's ability to be sequencing platform independent – including a module to scaffold with third-generation long-read sequencing data (e.g. from Pacific Biosciences: http://www.pacificbiosciences.com) – and its use as

[1] Note that we use the term assembly here to refer to all steps of sequence assembly, which is often done in a multi-step process involving steps such as contiging, scaffolding and gap-filling.

an assembly augmentation and refinement tool to take an existing assembly and improve it in a systematic manner (leveraging new sequencing data where available).

Results

Overview

The algorithmic core of OPERA-LG is adapted from the approach described in Gao et al. (Gao et al. 2011) (summarized in **Supplementary Figure 1**) and is based on (i) a memoized-search to find a scaffold that minimizes the number of discordant read-derived-links connecting contigs (**Supplementary Figure 1a**), (ii) a graph contraction approach that allows for localizing the search for an optimal scaffold without losing the guarantee of a globally optimal scaffold (**Supplementary Figure 1b**) and (iii) a quadratic programming formulation to compute gap sizes that best match mate-pair derived distance constraints (**Supplementary Figure 1c**; this is similar in principle to the approach used in the Celera Assembler (Myers et al. 2000)). To enable it to produce long and accurate scaffolds for large, repeat-rich genomes, OPERA-LG incorporates several key novel features and improvements including (a) Optimized data-structures and a hybrid-exact algorithm to improve its scalability (b) Refined edge-length estimation and the ability to simultaneously use multiple-libraries to improve scaffolding accuracy and (c) Extensions that allow for the scaffolding of repeat sequences. Algorithmic details for each of these features and improvements in OPERA-LG can be found in the **Methods** section.

Scalability and Multi-library Scaffolding

Scalability. Runtime and memory optimizations in OPERA-LG (including the hybrid-exact mode;

see **Methods**) are key to its scalability as shown in **Table 1**. In particular, while the method in Gao et al. (Gao et al. 2011) was unable to scaffold the full *D. melanogaster* dataset due to excessive memory usage, OPERA-LG takes a few seconds and a few hundred megabytes of memory (largely for storing read mapping information). For genomes where the previous method was feasible, OPERA-LG is typically more than 10 times faster and requires significantly lesser memory (roughly 1/2 to 1/20th of the requirements for Gao et al.). Across datasets, OPERA-LG's runtime (excluding the read mapping step) was found to be comparable, but slower, than SSPACE (a popular and efficient, heuristic scaffolder) (Boetzer et al. 2011) and significantly less than the runtimes for SOAPdenovo2 and ALLPATHS (**Figure 1**). In addition, for small genomes OPERA-LG's runtime is dominated by the preprocessing step, while for larger genomes the runtime for scaffolding (excluding the preprocessing step) may not necessarily increase proportional to genome size and is more likely to be affected by the intrinsic features of the genome (e.g. repeat lengths and distribution; **Figure 1**). Overall, OPERA-LG's runtime was less than 1 day (on a single processor) using <60 GB of memory for all the datasets tested here (including the human genome), establishing its feasibility for scaffolding large genomes and retaining the potential for further improvements with parallelization.

Edge Length Estimation and Multi-library Analysis. OPERA-LG was redesigned to simultaneously use data from multiple "jumping" libraries for scaffolding, a process that can be critical for improving assembly contiguity and correctness in large genome assembly projects. We directly evaluated these improvements to establish their utility in OPERA-LG. Firstly, scaffold edge length estimates from OPERA-LG were compared to the known true edge lengths for the synthetic

datasets and were found to be in excellent agreement overall (**Figure 2b** and **Figure 2d**). In addition, OPERA-LG's estimates were found to be more accurate than a commonly used naïve estimation procedure (**Figure 2a** and **Figure 2c**), which was found to consistently under-estimate edge lengths for longer edges, though the bias observed here was not as severe as was observed previously (Sahlin et al. 2012). Secondly, OPERA-LG's ability to handle multiple libraries simultaneously was found to provide a clear benefit over the commonly employed hierarchical approach as seen in **Figure 2e** and **Figure 2f**. The *simultaneous* approach not only led to fewer assembly errors (**Figure 2f**) but also provided improved assembly contiguity (measured by corrected N50; see **Methods**) as a by-product (**Figure 2e**). Note that the results for *C. elegans* may be more indicative of the performance boost that can be expected as the *E. coli* and *D. melanogaster* datasets had high-quality assemblies to begin with.

Improvements in Assembly Contiguity and Correctness

Benchmarking with Synthetic Datasets. To evaluate the performance of OPERA-LG, we first benchmarked it on several synthetic datasets as these provide the critical flexibility to vary parameters and assess their effect on the assembler (**Table 2**). We also assessed OPERA-LG at two levels, a) for its scaffold quality by comparing against the most popular (in terms of citations) stand-alone scaffolder, SSPACE (SS), and the scaffolding stage in SOAPdenovo2 (S2) on the same set of input contigs from SOAPdenovo and b) for overall assembly quality (contiging and scaffolding), using SOAPdenovo as a contig assembler and OPERA-LG as a scaffolder (OP), when compared to ALLPATHS-LG (AP) and SOAPdenovo2 (S2) as representatives of state-of-the-art assembly pipelines (i.e. using their contiging and scaffolding modules; contig and scaffold size

statistics for all programs can be found in **Supplementary Table 1**).

At the scaffold level, while the corrected N50 for SSPACE and SOAPdenovo2 were typically comparable, OPERA-LG produced assemblies that were 10 times more contiguous, regardless of the genome being assembled (corrected N50; **Figure 3a**). In addition, scaffolds produced by OPERA-LG also contained significantly fewer errors (<1/5th that of other scaffolders on average; **Figure 3b**). Manual inspection of scaffolding errors from OPERA-LG suggested that they were largely due to local ordering errors in regions of the scaffold graph that were not sufficiently constrained by scaffold edges (relocations) and gap size estimation errors due to lower read coverage (indels). In particular, OPERA-LG had few translocation errors, where distant regions of the genome were incorrectly brought together (**Figure 3b** and **Supplementary Table 2**). In contrast, results from SSPACE and SOAPdenovo2 suggest that they can lead to a large number of translocation errors in the assembly - more than 10 times what is obtained using OPERA-LG (**Figure 3b**). Another striking feature of OPERA-LG is the lack of inversion errors, where the orientation of contigs is incorrectly determined, suggesting that the global optimization employed in OPERA-LG effectively eliminates such errors. Overall, OPERA-LG had ≤1 inversion or translocation error for the *D. melanogaster* and *C. elegans* assemblies and less than a 100 such errors for the *H. sapiens* genome.

These results were also replicated at the overall assembly level. While ALLPATHS-LG and SOAPdenovo2 had comparable or larger original N50s, corrected assembly N50s were significantly lower for them (<1/6th) than those obtained with the combination of SOAPdenovo (contiging only) and OPERA-LG (**Figure 3c**). This was despite the fact that SOAPdenovo contigs were typically

more fragmented and provided a more challenging starting point for the scaffolder. Again, a likely explanation for the improvements seen here is the generation of significantly fewer assembly errors compared to ALLPATHS-LG and SOAPdenovo2 (**Figure 3d** and **Supplementary Table 3**; see **Supplementary Table 4** for a similar experiment with longer reads).

In comparison to other assembly tools, OPERA-LG introduced fewer errors as a function of increasing information in the quality and amount of mate-pair libraries provided to it as input (**Supplementary Figure 2a**). For example, incorporation of a 3 kbp mate-pair library in addition to a 10 kbp library was seen to consistently help eliminate local assembly errors in under-constrained regions of the assembly. Correspondingly, both original and corrected assembly N50s showed an improvement as OPERA-LG was provided with more data (roughly determined by the largest library-size provided as input) – a trend that was not so strongly seen for other assemblers (**Figure 3e**). A similar trend was also seen with increased sequencing depth where OPERA-LG was able to successfully leverage higher coverage to produce longer and more accurate assemblies (**Figure 3f** and **Supplementary Figure 2b**). In contrast, other assemblers either produced unchanged or marginally improved assemblies (ALLPATHS-LG) or, paradoxically in some cases, worse assemblies (SSPACE and SOAPdenovo2) on these datasets.

Finally, we also explored the robustness of various assemblers to variability in the quality of the sequencing library, as measured by the standard deviation in the library size (where low standard deviation is desired). Our results suggest that within a reasonable range, most assemblers produce very similar assemblies in terms of assembly contiguity (**Supplementary Figure 3a**) but with an increase in the number of assembly errors (**Supplementary Figure 3b**). OPERA-LG's assemblies,

however, have fewer errors and better contiguity in the worst-case then even the best-case scenario for the other methods tested here. These results further highlight the relative robustness of OPERA-LG in dealing with low-quality input data.

Results from sequenced genomes. While evaluation of assembly performance on real datasets can be influenced by the lack of gold-standard references in many cases, our results confirm that significant assembly improvements are also seen using OPERA-LG on real datasets. For example, for the extensively sequenced parrot genome datasets (*Melopsittacus undulatus*, estimated genome size = 1.2 Gbp), which were assembled as part of the Assemblathon2 competition (Bradnam et al. 2013), the corrected N50 obtained using OPERA-LG (with SOAPdenovo contigs) was found to be >2 times that of its closest competitor (ALLPATHS-LG; **Figure 4a**). This was despite the original N50 for OPERA-LG being slightly smaller than that for other programs such as ALLPATHS-LG and SOAPdenovo, and is likely due to the dramatic reduction in assembly errors seen with it. It is important to note here that OPERA-LG was run with default parameters while the results for other programs were the best submissions by their respective teams. In addition, OPERA-LG's results were based on the more conservative contigs from SOAPdenovo and could benefit further from improved contig assembly.

Similar results were also obtained for the assembly of a sweet orange genome (*C. sinensis*) (Xu et al. 2013), where OPERA-LG's corrected N50 was >2.5 times that of SOAPdenovo2 (178 vs 68 kbp) and with less than a third of its scaffolding errors (1492 vs 5455). Finally, in the case of a well-assembled yeast genome (*P. stipitis*), OPERA-LG's scaffolds were more than 3 times as contiguous as SSPACE's scaffolds (corrected N50 of 320 vs 92 kbp) and slightly larger than

SOAPdenovo2's scaffolds (299 kbp), but with significantly fewer scaffold errors (1 vs 61). In all datasets, OPERA-LG was significantly better than the competing standalone scaffolder SSPACE (>20 fold increase in corrected N50 for *C. sinensis* and >200 fold reduction in assembly errors on *P. stipitis*), highlighting the difference in performance using heuristic and exact approaches.

Application I: Scaffolding of Repeat Sequences

Assembly of repeat sequences is typically the most error-prone stage of many assembly pipelines as was observed in the results for the GAGE (Salzberg et al. 2012) and Assemblathon (Earl et al. 2011) competitions (e.g. the gap-filling stage in SOAPdenovo) and is often handled in a post-scaffolding, gap-filling stage. The algorithmic extensions in OPERA-LG that allow it to simultaneously scaffold unique and non-unique regions of the genome provide it with the ability to more appropriately assemble repeat-rich genomes – a feature that is not available in other competing scaffolders. Evaluation of scaffold correctness for OPERA-LG in the presence of repeat contigs suggested that this was indeed the case with >80% of scaffold gaps (regions with no unique contigs) being filled with repeat contigs in the correct order (as suggested in the reference genome; **Figure 5a**). In addition, typically more than 90% of repeat contigs in gap regions were placed in OPERA-LG scaffolds (**Figure 5a**), highlighting the completeness of scaffolds despite the conservative placement of repeats in OPERA-LG (see **Methods**). Finally, we noted that the repeat contigs placed allowed for large gaps in the assembly to be filled (for *D. melanogaster*, about half of the filled gaps were longer than 5 kbp; **Figure 5b**), a process that is otherwise challenging for gap-filling programs (Gao et al. 2012).

Application II: Assembly Augmentation and Hybrid Assembly

With the increasing availability of new sequencing technologies and additional sequencing datasets at reduced costs, improvement of older draft genome assemblies in a systematic fashion is an area of increasing interest in the field. As the underlying algorithms in OPERA-LG are conservative and intended to minimize discordance with data, the use of OPERA-LG as an *assembly augmentation* tool is attractive and we explored this idea further in two recent genome assembly projects. In the first project, for the assembly of a sweet orange genome (*C. sinensis*) (Xu et al. 2013), we used all available Illumina sequencing datasets with SOAPdenovo to generate a preliminary draft assembly (N50 of ~430 kbp). This assembly was then augmented using OPERA-LG, with SOAPdenovo scaffolds as starting sequences and by reusing the larger mate-pair libraries (10 kbp and 20 kbp). Notably, the final N50 obtained by this process was four times larger (~1.6 Mbp; **Figure 4b**) and the assembly was validated to be of high-quality using BAC-end sequences, genetic linkage maps and cytogenetic analysis (Xu et al. 2013). In another project, we augmented an existing reference genome assembly for a Chinese hamster ovary cell line (CHO-K1) (Xu et al. 2011), with newly generated Illumina sequencing datasets (300 bp library at 61X and 10 kbp library at 16X base pair coverage). The resulting assembly boosted N50 6-fold (1.2 Mbp to 6.9 Mbp; **Figure 4b**) with >90% of the genome assembled into ~500 scaffolds. The scaffolds were also used to close gaps *in silico*, filling >130,000 gaps and scaffold correctness was confirmed using alignments of the transcriptome to the assembly (Yusufi et al., in preparation).

As an example of a novel direction for hybrid assembly with OPERA-LG (in addition, to its ability to mix paired-read libraries and assemblies from different technologies), we also explored an approach to use long reads from third-generation sequencing technologies for scaffolding (by

generating mate-pair libraries *in silico*; see **Methods**). In particular, we experimented with reads generated on the Pacific Biosciences platform (PacBio) for the *M. undulates* assembly (Koren et al. 2012), though a similar approach should be applicable to data generated on the Oxford Nanopore platform as well (https://www.nanoporetech.com/). Despite the low overall coverage of this dataset (~7X; median length 1.3 kbp) and even lower coverage from long reads (~3X for reads longer than 2 kbp), OPERA-LG was able to boost corrected N50 by a factor of 10 (from 3 kbp to 30 kbp). Overall runtime for OPERA-LG was less than 10 minutes while analysis with a long-read version of SSPACE (SSPACE-Long Read) (Boetzer and Pirovano 2014) was unable to run to completion in a month. Similar results were also obtained for the *D. melanogaster* and *C. elegans* datasets (SOAPdenovo assembly), where OPERA-LG was able to improve corrected N50 from 50 and 9 kbp respectively, to 143 and 75 kbp using 7X coverage in simulated PacBio reads (see **Methods**). Inspection of mate-pair conflicts in OPERA-LG's assemblies suggested that many of these were likely due to mapping errors and thus improvements in read mapping (to cope with the high-frequency of indel errors i.e. ~1 in 10 bases) or the use of error-corrected reads could significantly improve assembly, despite the limited long-read coverage available.

Discussion

For many bioinformatics problems (and correspondingly the tools that solve them), it is either hard to formalize a clear objective for algorithm design, or the formalized objective is believed to be computationally intractable. Benchmarking of a novel algorithm is, therefore, a widely accepted norm in bioinformatics for demonstrating the algorithm's utility and advance over the state-of-the-art. Due to resource limitations, however, benchmarks are typically limited in nature

and this is also the case in this work. Moreover, consensus on a standardized and comprehensive benchmark can be elusive in many areas – assembly being one such field (though recent assembly competitions have provided valuable resources in this direction) – and this leads to a confusing landscape of choices for end-users.

Where feasible, algorithms that have a clear optimization criterion provide a way out of this conundrum. With a clear optimization criterion and accompanied by a formal proof of an algorithms' ability to optimize this objective, a user has to only be convinced that the criterion is valid to believe that the algorithm will work on an as-yet-unseen dataset. Of course, software programmers are fallible and benchmarking still serves to assure users that the algorithm has largely been translated correctly into code. This is the paradigm adopted in OPERA-LG, where the algorithm is guaranteed to produce a minimal-repeat scaffold that globally minimizes the number of discordant scaffold edges[2]. While this is by no means the only optimization criterion that one can choose, it is a simple and intuitive one and has the property that the scaffolder attempts to maximize agreement with input data. It also serves as a check against the possibility that the algorithm produces unusually poor results on an unexpected input dataset or has been over-optimized to work well on only a few datasets.

A common concern and criticism for exact algorithms for assembly and scaffolding, in particular, has been that they are slow, not practical for large datasets and cannot handle complicated cases (e.g.

[2] Note that OPERA-LG still requires a preprocessing stage that is heuristic in how it bundles reads into scaffold edges and uses read coverage to identify repeat sequences.

for scaffolding repeats). An important contribution of this work is to show that these challenges are surmountable with appropriate algorithm design and engineering. Our results show that large and repeat-rich genomes are amenable to assembly using exact algorithms and that the efficiency of OPERA-LG is comparable (and in some cases better) to heuristic algorithms. Furthermore, switching from heuristic to exact algorithms provides almost an order of magnitude (rather than incremental) and consistent improvements in assembly quality.

As characteristics of sequencing technologies continue to improve, particularly in read length, assembly quality is also expected to benefit as evidenced by recent work (Bashir et al. 2012; Koren et al. 2012). However, sequence assembly will continue to be a trial-and-error process until assembly algorithms fully exploit the power of the data and provide users with formal performance guarantees. Assembly errors continue to plague downstream biological sequence analysis, affecting almost every aspect of modern bioinformatics, and a full assessment of their impact has yet to be performed (Salzberg and Yorke 2005). Our analysis suggests that commonly used assembly tools such as SOAPdenovo and SSPACE may be introducing hundreds to thousands of assembly errors due to heuristic decisions during scaffolding that should be avoidable using an exact scaffolder such as OPERA-LG (inversion and translocation errors, in particular). Each of these programs has been used to construct hundreds of draft eukaryotic genomes to-date which could benefit significantly from re-assembly and assembly augmentation using newer assembly tools. As we move to an era where assembled sequences could potentially guide medically-relevant decisions, tolerance for assembly errors is likely to be even more limited. A paradigm-shift in assembly is thus needed such that fully-phased and complete, quality-guaranteed assemblies are the norm for future assemblers.

Methods

Scaffolding of Repeat Sequences

OPERA-LG relies on the read coverage of contigs to identify potentially repetitive contigs in the assembly (as well as polyploid sequences, as detailed below) (Gao et al. 2011). More sophisticated approaches that use the graph structure to do this (as is done in the Celera Assembler) could potentially yield more accurate and sensitive results (especially in low coverage settings) and will be explored in future versions of OPERA-LG. Since coverage estimates may not always be provided directly by the assembler (e.g. ALLPATHS (Gnerre et al. 2011)) or can be inaccurate (e.g. SOAPdenovo k-mer coverage values are bounded at 63), OPERA-LG computes these directly from user-provided read mappings. For haploid genomes, OPERA-LG identifies sequences with coverage less than a multiple (1.5 by default) of the genomic average as *unique*. The search procedure in Gao et al. (Gao et al. 2011) is then designed to find an optimal scaffold for such non-repeat sequences. Here, we describe how the search procedure can be extended to simultaneously scaffold repeat sequences as well. There are two assumptions for handling repeats in our algorithm: (1) Repeats cannot be used to extend scaffolds – this is to avoid ambiguous extensions to the scaffold and (2) The concordance of edges between repeats is ignored as these cannot be verified without assuming that all repeat instances have been scaffolded.

Definitions

A *scaffold* is given by a signed permutation of the contigs (in which repeat contigs are allowed to occur multiple times) as well as a list of gap sizes between adjacent contigs. For the concordance of a scaffold edge, we slightly extend the definition in Gao et al. (Gao et al. 2011), as follows: an edge

$e =< c_1, c_2 >$ is a *concordant* edge if any pair of instances of c_1 and c_2 in the scaffold can satisfy the distance and orientation constraints imposed by e and otherwise e is *discordant*. For a scaffold graph $G = (V, E)$, where V is a set of contigs and E is a set of scaffold edges representing multiple read-pairs that suggest a consistent order and orientation between the adjacent contigs (see section **Refined Edge Length Estimation and Multi-library Scaffolding** for details on how they are constructed in OPERA-LG), and a given partial scaffold S', the *dangling edge set* $D(S')$ is the set of edges from unique contigs in S' to all contigs in $V - S'$. When the distance between a repeat r and the tail of S' (measured as sum of contig lengths in the partial scaffold) is larger than the upper-bound of the paired-read library, r is said to be *confirmed* (in the sense that it will not be removed from the partial scaffold constructed by OPERA-LG). Then we define the *active region* $A(S')$ as the shortest suffix of S' (including all unconfirmed repeats) such that all concordant dangling edges are adjacent to a contig in $A(S')$. Also, we use the notation $u(G)$ and $u(S)$ to refer to the subgraph of G and the subset of S composed of only the unique contigs.

To constrain the number of occurrences of repeat contigs in scaffolds, we use a simple parsimony criterion to redefine our notion of an optimal scaffold:

Definition 1 (Minimal-repeat Optimal Scaffold). *Given a scaffold graph $G = (V, E)$ and a scaffold S that minimizes the number of discordant edges in the graph, S is considered a "minimal-repeat optimal scaffold", if removing any occurrence of a repeat from S will increase the number of discordant edges.*

Correspondingly, we have the following updated formulation of the Scaffolding Problem with

Repeats:

Definition 2 (Scaffolding Problem with Repeats). *Given a scaffold graph G, find a minimal-repeat optimal scaffold S of the contigs.*

Note that the criterion for including repeats is inherently conservative and would, for example, favor the placement of a single copy of a tandem repeat, where feasible. Suitable post-processing scripts would therefore be needed to estimate and expand-out copies of a tandem repeat region. We next describe how OPERA-LG's search procedure is designed to guarantee a scaffold that minimizes discordance with paired-read derived scaffold edges while parsimoniously including repeat contigs in the scaffold.

Construction of a minimal-repeat optimal scaffold

Unlike the analysis in Gao et al. (Gao et al. 2011), but without loss of generality, in OPERA-LG, partial scaffolds are only extended to the right of the active region, but all contigs are tried as potential starting points. Also, as before, the search in OPERA-LG is limited to an updated equivalence class of partial scaffolds as follows:

Lemma 1. Given a scaffold graph G and two valid partial scaffolds S'_1 and S'_2 with k_1 and k_2 discordant edges respectively, if $(A(S'_1),\ D(S'_1),\ k_1) = (A(S'_2),\ D(S'_2),\ k_2)$, then (1) S'_1 and S'_2 contain the same set of unique contigs; and (2) both or neither of them can be extended to a solution with equal or less than k discordant edges ($\forall k \in \mathbb{N}$).

Proof. For (1), since the dangling edge set defines a cut in $u(G)$, $D(S'_1) = D(S'_2)$ define the

same cut and since $A(S'_1) = A(S'_2)$, S'_1 and S'_2 must be on the same side of this cut and thus contain the same set of unique contigs.

For (2), let S'' be any scaffold extension of S'_1 such that $S'_1\,S''$ has $\leq k$ discordant edges. Then $S'_2\,S''$ would also be a valid scaffold as $u(S'') = u(V - S'_1) = u(V - S'_2)$. Also, since the active regions are identical, any newly discordant edge in $S'_1\,S''$ (i.e. not discordant in S'_1) that is adjacent to a contig in S'' will also be newly discordant in $S'_2\,S''$ (i.e. not discordant in S'_2) and *vice versa*. Corresponding the number of discordant edges in $S'_2\,S''$ is $\leq k_2 + (k - k_1) = k$.

□

During the memorized search in OPERA-LG (see **Supplementary Figure 1a**), a minimal-repeat optimal scaffold is obtained based on the following definition and lemma:

Definition 3 (Essential Repeat Instance). A repeat instance r in a partial scaffold is considered essential if no extension of the partial scaffold can be optimal if r is removed.

Lemma 2. (a) A repeat instance r in a partial scaffold S is essential *iff* (b) removing r increases the number of discordant edges when r is being confirmed (i.e. in the process of being tested to see if it should be marked confirmed).

Proof. Let the partial scaffold S' be obtained by removing r from S. (1) To prove $(a) \Rightarrow (b)$, for the sake of contradiction, let a repeat instance r in a partial scaffold S be essential s.t. removing r does not increase the number of discordant edges when r is being confirmed. Then, if there exists an optimal extension T of S with p discordant edges, the scaffold $S'T$ also has at most p discordant edges (as the status of edges connecting to T cannot change from ST to $S'T$, from the

definition of a confirmed repeat) and is therefore also optimal. Hence r is not essential, giving a contradiction. (2) To prove $(b) \Rightarrow (a)$, suppose removing a repeat instance r increases the number of discordant edges when r is being confirmed. Then for any extension T of S' s.t. $S'T$ has p discordant edges, the scaffold ST will have less than p discordant edges (as before, the status of edges connecting to T cannot change) and hence, $S'T$ can never be optimal. By Definition 3, therefore, r is essential in S.

□

Note that by definition, non-essential repeat instances should not be included in a minimal-repeat optimal scaffold. Correspondingly, based on Lemma 2, the algorithm presented in **Supplementary Figure 4** is guaranteed to report a minimal-repeat optimal scaffold.

The runtime complexity for the algorithm in **Supplementary Figure 4** is established in the following theorem.

Theorem 1. Consider a scaffold graph $G = (V, E)$. Let p be the maximum allowed number of discordant edges and w be the maximum number of contigs in the active region. The algorithm ScaffoldWithRepeat runs in $O(p|V|^w|E|^{p+1})$ time.

Proof. As in Gao et al. (Gao et al. 2011), the set of possible active regions is $O((2|V|)^w)$ and there are at most $O(2^{w^2})$ possible sets of concordant dangling edges for any given active region. In addition, there are at most $O(|E|^p)$ possible sets of discordant dangling edges. So, there are at most $O(|E|^p 2^{w^2}) = O(|E|^p)$ sets of dangling edges. The number of equivalence classes is therefore bounded by $O((2|V|)^w|E|^p p) = O(p|V|^w|E|^p)$. For each equivalence class, confirming

repeats, updating the active region, the dangling set and the number of discordant edges in steps 6-11 takes $O(|E|)$ time.

□

As in Gao et al. (Gao et al. 2011), we use the concept of fenced subgraphs and graph contraction to improve the runtime of OPERA-LG in practice, without affecting its guarantee of finding an optimal scaffold. Note that since repeats are not used for extending scaffolds, we can construct fenced subgraphs on the unique subgraph $u(G)$ in the same manner as in Gao et al. (Gao et al. 2011). Then, the *fenced subgraph with repeats* can be obtained by adding back repeat contigs with edges to unique contigs in a subgraph. The results in Gao et al. (Gao et al. 2011) can then be extended trivially to show that minimal-repeat optimal scaffolds on fenced subgraphs with repeats lead to a global optimum on the entire graph.

Correcting for polyploidy and aneuploidy

Most assembly programs are designed with assumptions specific to haploid genomes and correspondingly applying them to non-haploid genomes can lead to unexpected results and greater fragmentation of assembly than is typically induced by repetitive sequences. Also, for aneuploid genomes (as is frequently seen in cancer tissues and cell lines), the straightforward approach described previously to identify unique contigs may be too conservative in identifying non-repeat sequences (where repeats are defined as being sequences with multiple distinct locations in the genome).

For assembling polyploid or aneuploid genomes, OPERA-LG uses the haploid coverage of the

genome (H, user-specified) to estimate the copy number for each contig as $\max(1, round\left(\frac{C}{H}\right))$ (where C is the average coverage of the contig). Note that with sufficient coverage and the availability of single-copy contigs, more sophisticated, model-fitting-based methods can be used to directly estimate haploid genome coverage from coverage statistics and we plan to investigate these in future versions of OPERA-LG. Based on the copy number of contigs, OPERA-LG clusters and scaffolds contigs with the same copy number (also removing edges between contigs whose average coverage differs $> H/2$) assuming that all contigs with copy number less than the maximum ploidy (user-specified, but can obtained as before from coverage statistics) are classified as unique (for diploid genomes, the --hybrid-scaffold option allows haploid and diploid contigs to be scaffolded together). While this assumption can lead to scaffold errors when, for example, a two-copy repeat from a haploid region is linked by a scaffold edge to a unique contig from the diploid genome, such cases are likely be relatively infrequent. In balance, this assumption allows OPERA-LG to correctly scaffold unique contigs (and repeat contigs) from polyploid chromosomes, which can form a significant fraction of the genome to be assembled. Furthermore, estimation of copy number for scaffold edges can enable even more reliable scaffolding of polyploidy genomes. However, this would require development of improved models that account for mapping biases and is beyond the scope of this work.

Scalability and Optimized Data Structures

Due to the computational intensity of the search procedure employed in OPERA-LG (Gao et al. 2011), engineering issues such as code optimizations and data structures play an important role in enabling its application to large genomes. To allow for greater control over runtime and memory

optimizations, OPERA-LG is implemented in C++ with custom data structures designed to allow for a smaller memory and runtime footprint by exploiting the depth-first structure of the search (Gao et al. 2011). Specifically, as shown in **Supplementary Figure 1a**, the search procedure in OPERA-LG needs to keep track of a partial solution S during its search by remembering the corresponding active region $A(S)$ and a set of discordant edges $X(S)$ (Gao et al. 2011). Since partial scaffolds that are related in the search tree (say S_1 and S_2) can overlap in their respective active regions and discordant edge sets, OPERA-LG avoids duplication of this information, by only recording the difference in these sets (i.e. $A(S_1) \vartriangle A(S_2)$ and $X(S_1) \vartriangle X(S_2)$). In addition, as shown in **Supplementary Figure 1a**, the search avoids unnecessary computation by keeping track of partial scaffolds that have already been explored (memoization). While this can be done in a straightforward way using a hashtable, memory requirements for this approach can be significant. In OPERA-LG we implemented a *prefix tree* data structure to store active regions (these are lists by definition) and corresponding discordant edges (an arbitrary order was imposed to convert these sets into lists) as shown in **Supplementary Figure 5**. This allowed for a significant reduction in the memory footprint of OPERA-LG (**Table 1**) while allowing for lookups in time proportional to the size of the active region and the discordant edge set.

While the runtime requirements for OPERA-LG were typically found to be modest and in particular aided by the graph contraction step shown in **Supplementary Figure 1b**, the search time for some sub-graphs can be significantly longer than average. Correspondingly OPERA-LG allows the user to bound the number of partial scaffolds enumerated on any one sub-graph (default value is 1 million), switching to solving the problem with an increased edge size threshold (default step size of

1) when the maximum is reached. This hybrid-exact option in OPERA-LG (default setting) allows for the user to benefit from an exact algorithm for most of the assembly (>90% of the genome in all datasets tested here) while relying on a reasonable heuristic (edges with few supporting reads are less reliable) when an exact approach is not directly feasible. Though the rate of incorrect scaffold joins for such sub-graphs was found to be comparable to the rest of the graphs (0.5% in both cases), OPERA-LG conservatively flags such scaffolds to the user for further investigation.

Finally, to improve correctness on large genomes with shallow read coverage, the search procedure in OPERA-LG explores contig extensions in order of their estimated distance from the end of a partial scaffold (line 4 in **Supplementary Figure 4**). To estimate these distances, it employs a breadth-first-search (visiting each scaffold edge only once) and uses a weighted mean (down-weighting by the number of edges) for a contig that can be reached by more than one path. This ordering of contig extensions does not impact the performance guarantees in OPERA-LG (i.e. its ability to find an optimal scaffold) but typically generates more accurate scaffolds in regions with ambiguous extensions.

Refined Edge Length Estimation and Multi-library Scaffolding

In order to simultaneously use data from multiple "jumping" libraries for scaffolding, OPERA-LG uses a three-staged process to combine library information in the scaffold graph. Firstly, instead of relying on user input for library properties such as mean and standard deviation of insert sizes and read orientation, by default, OPERA-LG directly estimates these from read mappings on large sequences (> 2 times the insert size, as estimated from a set of 1000 read-pairs). The read

orientation of a library is set by default to the majority orientation of all read-pairs mapped to the same contig and these are then used to calculate the mean insert length and standard deviation. To avoid biases due to outliers (from mis-assemblies, mis-mapping or sequencing errors), read-pairs with distance lesser than $Q_1 - 3 \times IQR$ or greater than $Q_3 + 3 \times IQR$ were ignored, where Q_1 and Q_3 are the first and third quartiles respectively of distances between read-pairs on a contig and $IQR = Q_3 - Q_1$.

Secondly, OPERA-LG combines information from read-pairs in a single library to get a library specific estimate of edge length for each scaffold edge (Gao et al. 2011). To account for the fact that the observed read-pairs are from a truncated distribution (Sahlin et al. 2012) (in the range $[g, C]$ as shown in **Supplementary Figure 6**), we use a reverse lookup table to estimate g (gap size) from the observed mean of $\hat{S}$ (defined as the length of contig sequences that overlap with the insert region of the paired-reads; see **Supplementary Figure 6**). As noted in Sahlin et al. (Sahlin et al. 2012), this is an important step to avoid biases in edge length estimation that could lead to incorrect ordering of contigs and we propose a novel approach for bias correction. Specifically, given g and the overall distribution of insert lengths I (as determined by the read mapping on large sequences), we compute $E(\hat{S})$ as $E(I_{[g,C]}) - g$ (where $I_{[g,C]}$ is the random variable for the truncated distribution indicated in **Supplementary Figure 6b**) for every value of g in the range $[0, L]$ (where L is an upper bound on the insert size for the library (Gao et al. 2011)). To reduce runtime, OPERA-LG pre-computes such a lookup table for l (= sum of contig lengths) in the range $[g, Q_3 + 3 \times IQR]$ at 500bp intervals and returns the value of g that corresponds to the value of $E(\hat{S})$ closest to the observed mean of $\hat{S}$ (using the appropriate lookup table for the current value

of l). Note that the pre-computation for a lookup table is quite efficient as it can be done in linear time (as a function of the library size) using cumulative sums.

Thirdly, scaffold edges obtained from different libraries are combined to form a unified scaffold graph in OPERA-LG (this is distinct from the approach used in the Celera Assembler where libraries are combined directly at the read-pair level). To do this, OPERA-LG uses the mean (as obtained above) and standard-deviation (Huson et al. 2002) estimates for all edges connecting a pair of contigs (in the same orientation) to cluster and merge edges, starting at each stage with the edge with the largest standard deviation and identifying other edges whose means are within k ($= 6$ by default) standard deviations of this edge to merge into a single edge. In the case where more than one edge remains at the end of the process, OPERA-LG uses the edge supported by the most paired-reads and discarded all other edges (for assembly of polyploid sequences all edges will be discarded and for repeat contigs all edges will be retained).

Since libraries with very different insert sizes (say 200 bp vs 20 kbp) provide largely orthogonal information for scaffolding, it is possible to consider them in groups of similar insert sizes (this also provides tighter control on border contig sizes (Gao et al. 2011)). OPERA-LG therefore also allows the user to combine libraries in a *staged* fashion (< 1kbp, 1-10kbp and > 10kbp, by default) for greater runtime efficiency.

Hybrid Assembly and Scaffolding with Long Reads

Similar to other scaffolders that are not restricted to a specific mapper or assembler (Boetzer et al. 2011), OPERA-LG allows users to combine contig assemblies and paired reads from different

sequencing technologies. This feature of OPERA-LG has been exploited in several projects including the assembly of CHO cell lines using SOLiD mate-pair and Illumina paired-end datasets (Yusufi et al., manuscript in preparation). In addition, the availability of 'third-generation' sequencing technologies that directly produce longer reads (e.g. median lengths in the range 2-8 kbp from PacBio Systems (Bashir et al. 2012)) has provided another avenue for significantly improving sequence contiguity during genome assembly. For example, in the case of the *M. undulates* assembly, the authors reported significant improvement in contig N50s (30-fold to 100 kbp) by including PacBio reads (Koren et al. 2012). The use of PacBio reads to aid scaffolding is an alternative approach and we tested a straightforward way to incorporate this information for scaffolding with OPERA-LG. Specifically, we aligned PacBio reads to contigs using BLASR (Chaisson and Tesler 2012) with default parameters. Reads aligned to multiple contigs were then used to construct synthetic mate-pairs connecting every pair of contigs aligned to the read. Only alignments containing more than 90% of the contig or where read and contig ends overlapped were considered for this analysis. Synthetic mate-pairs with estimated distances in the range [0-300], [300-1,000], [1,000-2,000], [2,000-5,000] and [5,000-15,000] were then provided as synthetic libraries for OPERA-LG to scaffold with.

Evaluation on Synthetic Datasets

As in Gao et al. (Gao et al. 2011), all paired-end and mate-pair read libraries for synthetic datasets were generated using Metasim with default Illumina sequencing settings (Richter et al. 2008). In addition to the three small genomes (*E. coli*, *S. cerevisiae* and *D. melanogaster* chromosome X) analyzed in Gao et al. (Gao et al. 2011), we generated libraries and benchmarked on three larger

genomes as well i.e. *D. melanogaster*, *C. elegans* and *H. sapiens* (reference genomes were obtained from the NCBI website and details can be found in **Table 2**). PacBio reads were generated using PBSIM v1.03 (Ono et al. 2013) (using --data-type CLR, i.e. long reads with high error rate) to mimic a library of 3 kbp mean read length (maximum read length of 25 kbp) and give ~7X coverage of the *D. melanogaster* and *C. elegans* genomes.

For the large synthetic datasets, the GAGE pipeline was used to evaluate the final assemblies (Salzberg et al. 2012) for contig errors (indels longer than 5bp, inversions, relocations and translocations) and scaffold errors. Corrected assemblies were produced by splitting the final assemblies produced by all programs at contig and scaffold errors (for evaluating the overall assembly) or only scaffold errors (for scaffold-level evaluation) to compute the *corrected N50* (N50 is defined here as the fragment length such that more than 50% of the genome is in fragments of equal or longer length). For handling scaffolds with repeats, the mappings produced by the GAGE pipeline were further analyzed. A mapping position was considered correct if both coverage and identity was found to be greater than 90%. For each repeat, all correct positions were considered for correctness and completeness analysis.

Benchmarking on Sequenced Genomes

Publicly available data from three sequenced and published genomes were further used to benchmark scaffolders and assemblers in this study: a yeast genome (*Pichia stipitis*, 15.0 Mbp) (Chapman et al. 2011), a sweet orange genome (*Citrus sinensis*, 0.3 Gbp) (Xu et al. 2013) and a parrot genome (*Melopsittacus undulatus*, 1.2 Gbp) (Bradnam et al. 2013).

Due to the availability of a high-quality reference, assemblies for *P. stipitis* were evaluated using the GAGE pipeline. For the other genomes, the assemblies were evaluated for errors using REAPR (Hunt et al. 2013) (used as part of the evaluation pipeline for Assemblathon2 (Bradnam et al. 2013)). For *C. sinensis* and *M. undulatus*, all reads from the 10 and 20 kbp mate-pair libraries, respectively, were used for evaluation with REAPR (mapped using –i 15000). For *M. undulatus*, PacBio reads were mapped onto the assemblies to recognize contig errors, defined as regions not covered by *any* PacBio reads and having more than three split-mapped reads within 100 bp of the putative breakpoint. Reads were aligned to the assembly using Nucmer v3.23 (with –maxmatch), and the best mapping position for each read was selected using delta-filter v3.23 with default parameters. This approach resulted in only 1 false-positive break on a simulated dataset with 7X coverage of the SOAPdenovo contig assembly for *M. undulatus*. Corrected assemblies were produced for all datasets by splitting the final assemblies produced by all programs at contig and scaffold errors (for evaluating the overall assembly) or only scaffold errors (for scaffold-level evaluation) to compute the corrected N50.

Parameter Settings

Parameter settings for the benchmarked scaffolders and assemblers were as follows – SSPACE (SSPACE-BASIC-2.0): default values, SOAPdenovo2 (version 2.04): -K 41, -d 1 and ALLPATHS-LG (version 43019): PATCH_SCAFFOLDS=false, KPATCH=false (to skip the gap filling stage). Contigs produced by SOAPdenovo (with -K 41 -d –D) were provided as inputs for SSPACE and OPERA-LG. Read mappings for OPERA-LG were generated using BWA (0.7.10-r789) (samse -n 1) (Li and Durbin 2009) and OPERA-LG was run using default parameters

(contigs smaller than max(500, 2 × paired-end insert-size) bp were not scaffolded and contigs with coverage greater than 1.5 times the average coverage were treated as repeats). Assemblies from Assemblathon2 produced by Meraculous, BCM-HGSC and Newbler-454 for *M. undulatus*, were used for comparison as they were reported to be among the best assemblies based on several criteria (Bradnam et al. 2013). For runtime efficiency, all genomes >200 Mbp were scaffolded using the staged approach in OPERA-LG.

Data Access

Source code and executables for OPERA-LG are freely available from its Sourceforge website: http://sourceforge.net/projects/operasf/. Simulated datasets and assemblies presented (except the ones that were downloaded from Assemblathon 2) are available at the following ftp site: ftp://ftp2.gis.a-star.edu.sg/opera-lg/. Published sequencing datasets were accessed based on information from the following websites (1) *Pichia stipitis*: sequencing data was downloaded from ftp://ftp.jgi-psf.org/pub/JGI_data/meraculous/ (2) *Citrus sinensis*: sequencing data was downloaded from http://citrus.hzau.edu.cn/orange (3) *Melopsittacus undulatus*: Accession numbers from http://www.gigasciencejournal.com/content/supplementary/2047-217x-2-10-s5.xlsx were used to download data from the Sequence Read Archive (http://www.ncbi.nlm.nih.gov/sra).

Acknowledgements

This work was supported by the IMaGIN platform (project No. 102 101 0025), through a grant from the Science and Engineering Research Council as well as funding to the Genome Institute of Singapore from the Agency for Science, Technology and Research (A*STAR), Singapore.

Figure Legends

Figure 1: **Runtime as a function of genome size.** Note that both the x and y-axes are log-scaled and the results for various genomes are indicated at the corresponding genome size. Runtime for just the scaffolding module in OPERA-LG is shown separately to highlight the fact that it can take a fraction of the overall runtime and is influenced more by the repeat complexity of the genome than the size of the genome. Due to its library-size restrictions, ALLPATHS-LG could be run on only a few of the datasets shown here.

Figure 2: **Improvements in multi-library scaffolding.** Subfigures (a)-(d) show the improved correlation between empirical estimates and true edge lengths when using the procedure in OPERA-LG in comparison to the Naïve estimation that is commonly used (results reported are for the10 kbp libraries). Subfigures (e)-(f) depict the improvements in corrected assembly N50 and reduction in corresponding assembly errors when using the multi-library scaffolding implemented in OPERA-LG and in comparison to a commonly used hierarchical scaffolding that considers libraries independently and in order of insert size.

Figure 3: **Boosting assembly contiguity and correctness with OPERA-LG.** Subfigures (a)-(b) show scaffold contiguity and correctness for various scaffolders on different genomes starting from common sets of contigs (generated by SOAPdenovo). The final assemblies were only corrected for scaffold errors. Subfigures (c)-(d) show corresponding overall assembly metrics for various assembly pipelines that were provided all read libraries as input. The final assemblies were corrected for both contig and scaffold errors here to allow for a fair comparison. Arrows highlight fold improvement in corrected N50 using OPERA-LG. ALLPATHS-LG analysis on the human dataset had an abnormal exit after >10 days of runtime but appears to have produced a valid assembly. Subfigures (e)-(f) depict overall assembly contiguity as a function of mate-pair libraries and sequencing depth, provided as input. Results shown are for the *C. elegans* dataset and are qualitatively similar for other datasets as well (data not shown).

Figure 4: **Assembly improvements on sequenced genomes using OPERA-LG.** (a) for the *M. undulates* genome, comparing OPERA-LG against the best assemblers in the ASSEMBLATHON competition. Note that while OPERA-LG's results are similar to that of the other assemblers in the original N50, its correct N50 is more than 2 times that of the other assemblers. (b) Assembly augmentation using OPERA-LG for the *C. sinenis* and CHO-K1 genomes significantly boosts assembly contiguity.

Figure 5: **Scaffolding of repeat sequences with OPERA-LG.** (a) Correctness and completeness of scaffolding with repeat contigs. A gap is considered correctly filled if all scaffolded repeat contigs belong there and are in the right order and orientation. For completeness, we consider all repeat contigs (longer than 500bp) in valid gaps (gaps where the adjacent contigs are in the right order and orientation) and with edges in the scaffold graph. (b) Length distribution of filled gaps in various genomes.

Tables

Table 1: **Scalability comparison between the exact method in Gao et al. and OPERA-LG.**

		E. coli	*S. cerevisiae*	*D. melanogaster* Chromosome X	*D. melanogaster*
Gao et al.	Runtime (s)	1.1	1.3	2.6	N/A
	Memory (Mb)	95	62	265	N/A
OPERA-LG	Runtime (s)	<0.1	0.1	0.1	2.4
	Memory (Mb)	5	34	26	635

*Runtime reported is the scaffolding time for each method

Table 2: **Statistics for synthetic datasets.**

	D. melanogaster (RefSeq Assembly ID: GCF_000001215.2)	*C. elegans* (RefSeq Assembly ID: GCF_000002985.5)	*H. sapiens* (hg19)
Genome Size (Gbp)	0.1	0.1	3.2
Chromosomes	7	6	24
Paired-end Libraries*	80 bp, 140±10 bp, 40X		
Additional Paired-end Libraries*	N/A	N/A	80 bp, 700±70 bp, 10X
Mate-pair Libraries*	50 bp, 3±0.3 kbp, 2X 50 bp, 10±1 kbp, 2X 50 bp, 20±2 kbp, 2X		
Additional Mate-pair Libraries*	50 bp, 10±1 kbp, 10X 50 bp, 10±1.5 kbp, 2X 50 bp, 10±2 kbp, 2X	50 bp, 10±1 kbp, 10X	N/A

*Library details are specified in the format: read length, insert size, base pair coverage

References

Bashir A, Klammer AA, Robins WP, Chin CS, Webster D, Paxinos E, Hsu D, Ashby M, Wang S, Peluso P et al. 2012. A hybrid approach for the automated finishing of bacterial genomes. *Nature biotechnology* **30**(7): 701-707.

Boetzer M, Henkel CV, Jansen HJ, Butler D, Pirovano W. 2011. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics* **27**(4): 578-579.

Boetzer M, Pirovano W. 2014. SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. *BMC bioinformatics* **15**: 211.

Bradnam KR, Fass JN, Alexandrov A, Baranay P, Bechner M, Birol I, Boisvert S, Chapman JA, Chapuis G, Chikhi R et al. 2013. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience* **2**(1): 10.

Chaisson MJ, Tesler G. 2012. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC bioinformatics* **13**: 238.

Chapman JA, Ho I, Sunkara S, Luo S, Schroth GP, Rokhsar DS. 2011. Meraculous: de novo genome assembly with short paired-end reads. *PloS one* **6**(8): e23501.

Dayarian A, Michael TP, Sengupta AM. 2010. SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC bioinformatics* **11**: 345.

Earl D, Bradnam K, St John J, Darling A, Lin D, Fass J, Yu HO, Buffalo V, Zerbino DR, Diekhans M et al. 2011. Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome research* **21**(12): 2224-2241.

Gao S, Bertrand D, Nagarajan N. 2012. FinIS: improved in silico finishing using an exact quadratic programming formulation. *Algorithms in Bioinformatics Lecture Notes in Computer Science* **7534**: 314--325.

Gao S, Sung WK, Nagarajan N. 2011. Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *Journal of computational biology : a journal of computational molecular cell biology* **18**(11): 1681-1691.

Gnerre S, Maccallum I, Przybylski D, Ribeiro FJ, Burton JN, Walker BJ, Sharpe T, Hall G, Shea TP, Sykes S et al. 2011. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences of the United States of America* **108**(4): 1513-1518.

Hunt M, Kikuchi T, Sanders M, Newbold C, Berriman M, Otto TD. 2013. REAPR: a universal tool

for genome assembly evaluation. *Genome biology* **14**(5): R47.

Huson DH, Reinert K, Myers EW. 2002. The greedy path-merging algorithm for contig scaffolding. *Journal of the ACM* **49**(5): 603-615.

Kececioglu JD, Myers EW. 1993. Combinatorial algorithms for DNA sequence assembly. *Algorithmica* **13**: 7-51.

Koren S, Schatz MC, Walenz BP, Martin J, Howard JT, Ganapathy G, Wang Z, Rasko DA, McCombie WR, Jarvis ED et al. 2012. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology* **30**(7): 693-700.

Li H, Durbin R. 2009. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**(14): 1754-1760.

Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K et al. 2010. De novo assembly of human genomes with massively parallel short read sequencing. *Genome research* **20**(2): 265-272.

Lindsay J, Salooti H, Zelikovsky A, M I, #259, ndoiu. 2012. Scalable genome scaffolding using integer linear programming. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pp. 377-383. ACM, Orlando, Florida.

Medvedev P, Georgiou K, Myers G, Brudno M. 2007. Computability of Models for Sequence Assembly. *Algorithms in Bioinformatics* **4645**: 289-301.

Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, Kravitz SA, Mobarry CM, Reinert KH, Remington KA et al. 2000. A whole-genome assembly of Drosophila. *Science* **287**(5461): 2196-2204.

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

Nagarajan N, Pop M. 2013. Sequence assembly demystified. *Nature reviews Genetics* **14**(3): 157-167.

Ono Y, Asai K, Hamada M. 2013. PBSIM: PacBio reads simulator--toward accurate genome assembly. *Bioinformatics* **29**(1): 119-121.

Peltola H, Soderlund H, Tarhio J, Ukkonen E. 1983. Algorithms for Some String Matching Problems Arising in Molecular Genetics. *IFIP Congress*: 59-64.

Pop M, Kosack DS, Salzberg SL. 2004. Hierarchical scaffolding with Bambus. *Genome research* **14**(1): 149-159.

Richter DC, Ott F, Auch AF, Schmid R, Huson DH. 2008. MetaSim: a sequencing simulator for genomics and metagenomics. *PloS one* **3**(10): e3373.

Sahlin K, Street N, Lundeberg J, Arvestad L. 2012. Improved gap size estimation for scaffolding algorithms. *Bioinformatics* **28**(17): 2215-2222.

Salmela L, Makinen V, Valimaki N, Ylinen J, Ukkonen E. 2011. Fast scaffolding with small independent mixed integer programs. *Bioinformatics* **27**(23): 3259-3265.

Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, Treangen TJ, Schatz MC, Delcher AL, Roberts M et al. 2012. GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome research* **22**(3): 557-567.

Xu Q, Chen LL, Ruan X, Chen D, Zhu A, Chen C, Bertrand D, Jiao WB, Hao BH, Lyon MP et al. 2013. The draft genome of sweet orange (Citrus sinensis). *Nature genetics* **45**(1): 59-66.

Xu X, Nagarajan H, Lewis NE, Pan S, Cai Z, Liu X, Chen W, Xie M, Wang W, Hammond S et al. 2011. The genomic sequence of the Chinese hamster ovary (CHO)-K1 cell line. *Nature biotechnology* **29**(8): 735-741.

Figure 1

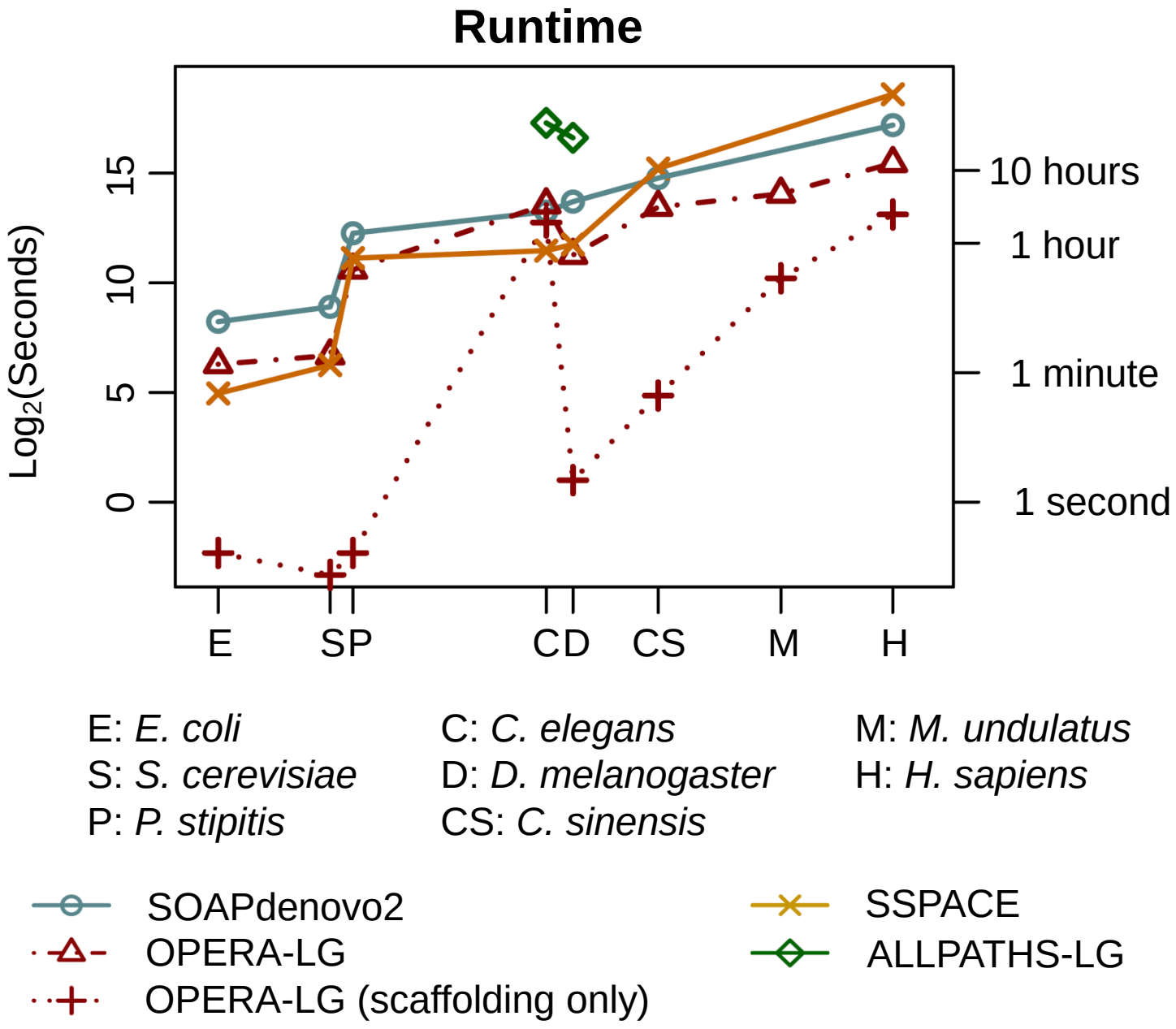

Runtime
Log2(Seconds)
0
5
10
15
10 hours
1 hour
1 minute
1 second
E
SP
CD
CS
M
H
E: E. coli
S: S. cerevisiae
P: P. stipitis
C: C. elegans
D: D. melanogaster
CS: C. sinensis
M: M. undulatus
H: H. sapiens
SOAPdenovo2
OPERA-LG
OPERA-LG (scaffolding only)
SSPACE
ALLPATHS-LG

Figure 2

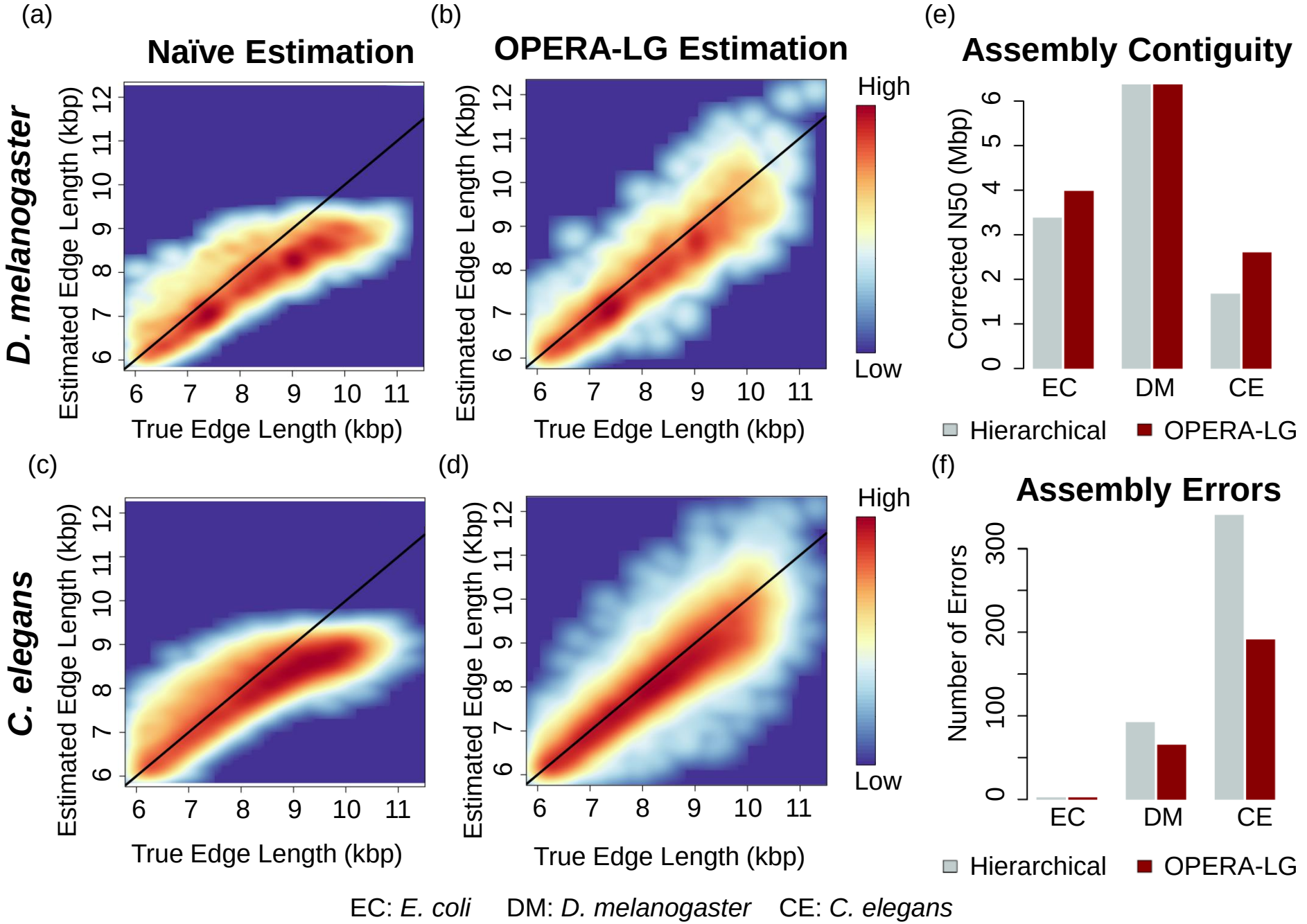

(a)
Naïve Estimation
(b)
OPERA-LG Estimation
(e)
Assembly Contiguity
D. melanogaster
Estimated Edge Length (kbp)
True Edge Length (kbp)
Estimated Edge Length (Kbp)
High
Low
Corrected N50 (Mbp)
EC
DM
CE
Hierarchical
OPERA-LG
(c)
(d)
(f)
Assembly Errors
C. elegans
Number of Errors
EC: E. coli DM: D. melanogaster CE: C. elegans

Figure 3

DM: *D. melanogaster*
CE: *C. elegans* HS: *H. sapiens*

SS: SSPACE S2: SOAPdenovo2
OP: OPERA-LG AP: ALLPATHS-LG

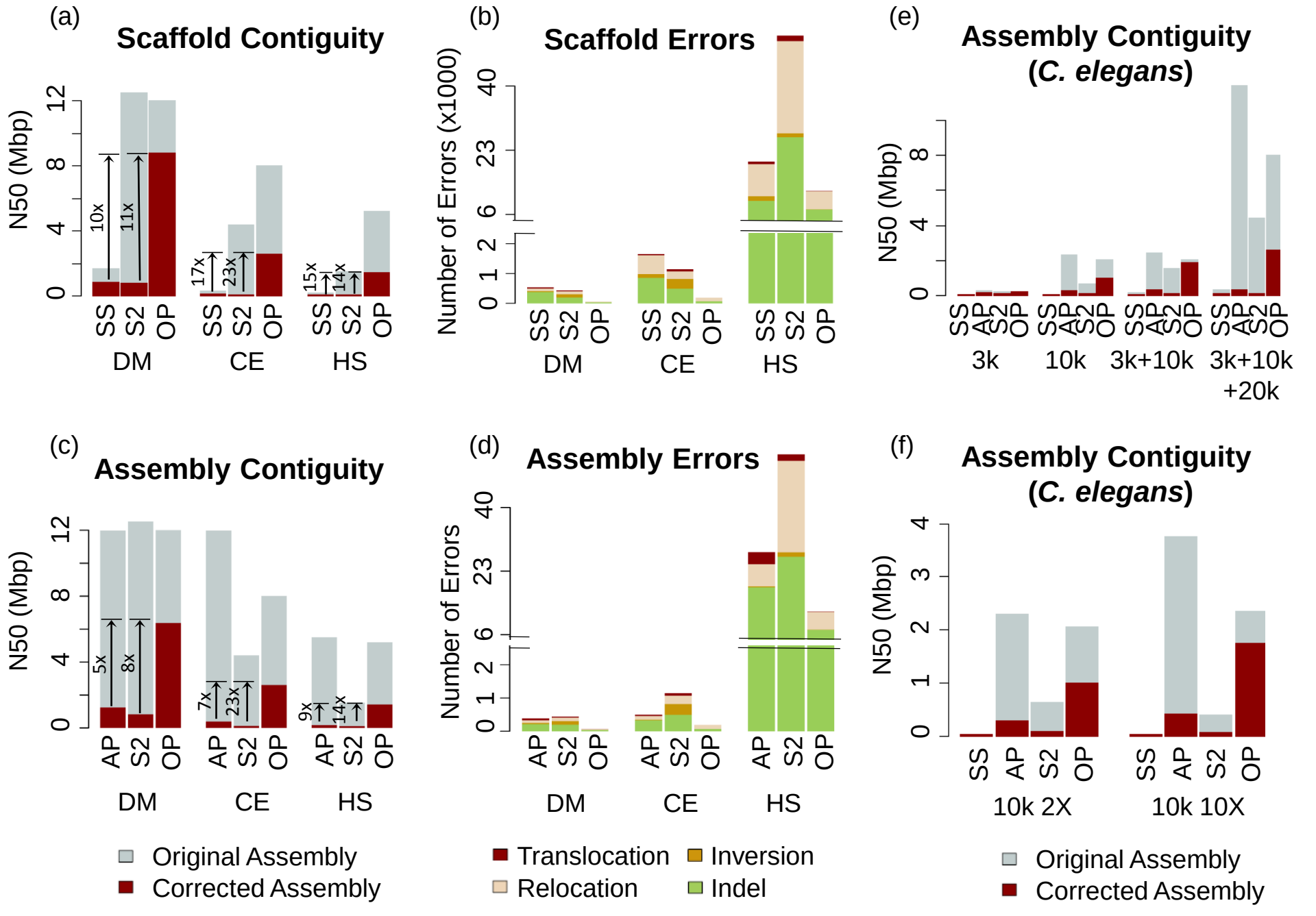


Figure 4

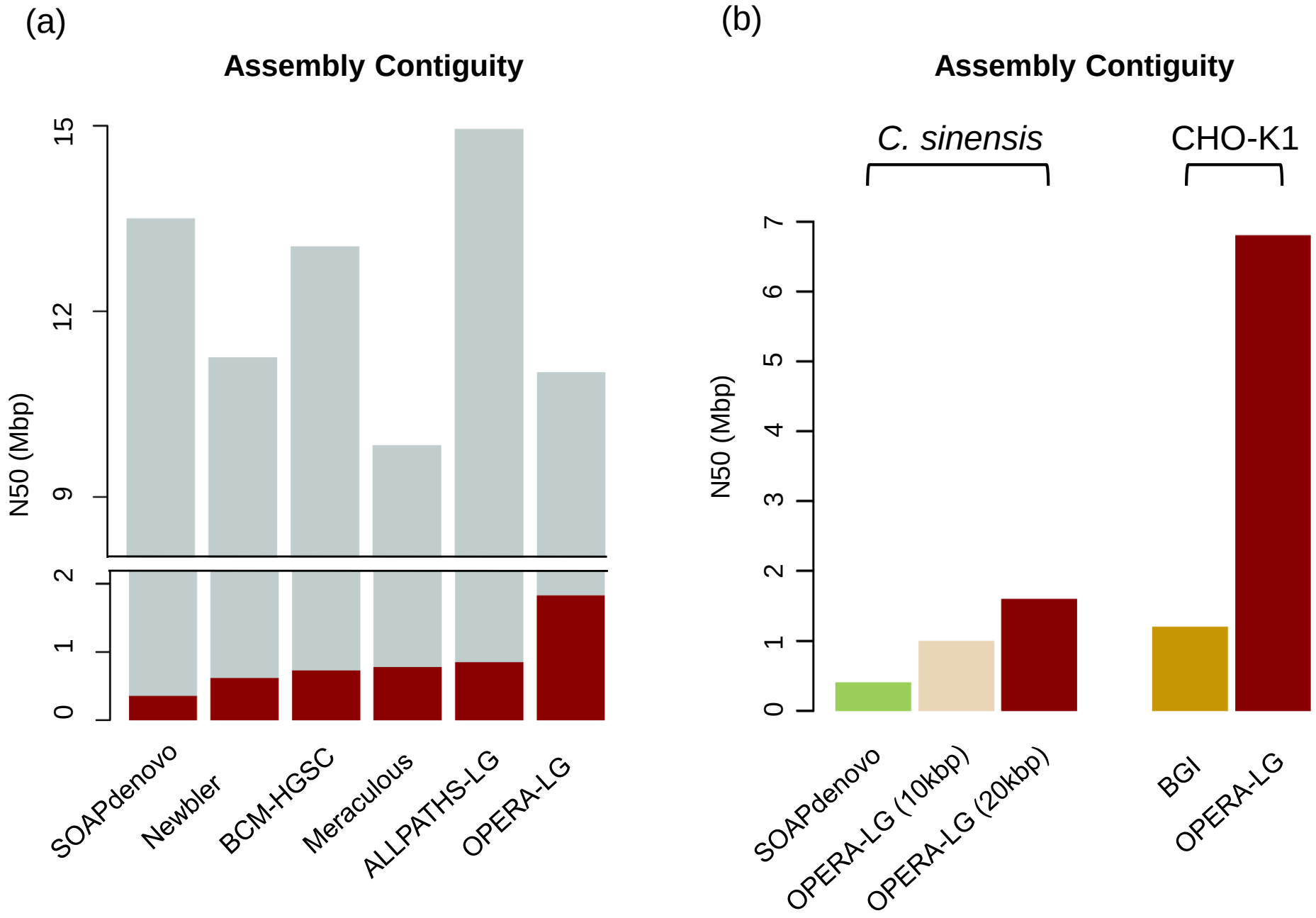
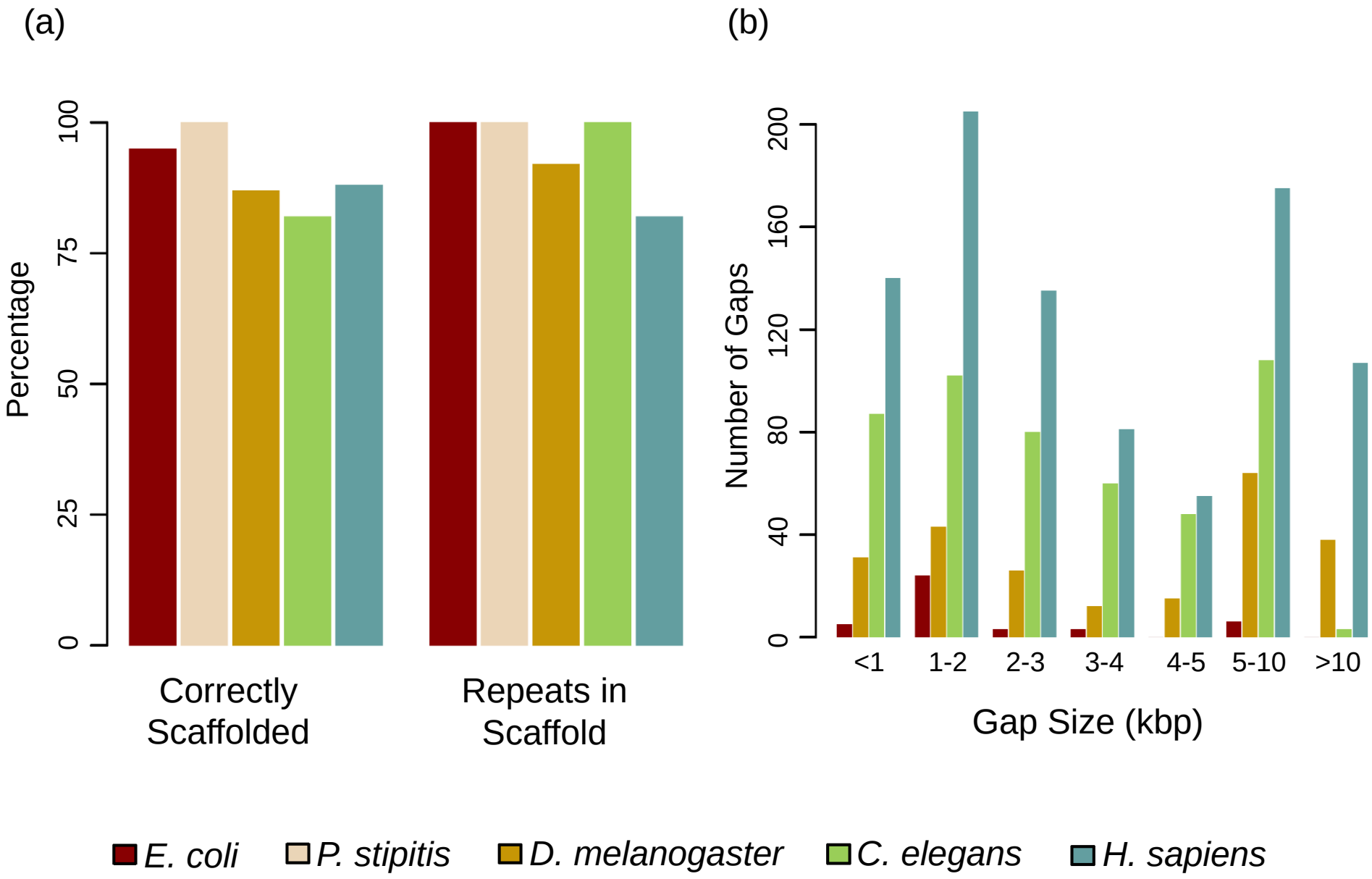

(a)
Assembly Contiguity
N50 (Mbp)
0
1
2
9
12
15
SOAPdenovo
Newbler
BCM-HGSC
Meraculous
ALLPATHS-LG
OPERA-LG
(b)
Assembly Contiguity
C. sinensis
CHO-K1
N50 (Mbp)
0
1
2
3
4
5
6
7
SOAPdenovo
OPERA-LG (10kbp)
OPERA-LG (20kbp)
BGI
OPERA-LG

Figure 5


(a)
Percentage
0
25
50
75
100
Correctly Scaffolded
Repeats in Scaffold
(b)
Number of Gaps
0
40
80
120
160
200
<1
1-2
2-3
3-4
4-5
5-10
>10
Gap Size (kbp)
E. coli
P. stipitis
D. melanogaster
C. elegans
H. sapiens