Published online 6 December 2016 Nucleic Acids Research, 2017, Vol. 45, No. 6 **e43**
doi: 10.1093/nar/gkw1191

The combination of direct and paired link graphs can boost repetitive genome assembly

Wenyu Shi†, Peifeng Ji† and Fangqing Zhao*

Computational Genomics Lab, Beijing Institutes of Life Science, Chinese Academy of Sciences, Beijing 100101, China

Received August 08, 2016; Revised November 07, 2016; Editorial Decision November 15, 2016; Accepted November 17, 2016

ABSTRACT

Currently, most paired link based scaffolding algorithms intrinsically mask the sequences between two linked contigs and bypass their direct link information embedded in the original *de Bruijn* assembly graph. Such disadvantage substantially complicates the scaffolding process and leads to the inability of resolving repetitive contig assembly. Here we present a novel algorithm, inGAP-sf, for effectively generating high-quality and continuous scaffolds. inGAP-sf achieves this by using a new strategy based on the combination of direct link and paired link graphs, in which direct link is used to increase graph connectivity and to decrease graph complexity and paired link is employed to supervise the traversing process on the direct link graph. Such advantage greatly facilitates the assembly of short-repeat enriched regions. Moreover, a new comprehensive decision model is developed to eliminate the noise routes accompanying with the introduced direct link. Through extensive evaluations on both simulated and real datasets, we demonstrated that inGAP-sf outperforms most of the genome scaffolding algorithms by generating more accurate and continuous assembly, especially for short repetitive regions.

INTRODUCTION

As an integral step in any genome sequencing project, *de novo* assembly reconstructs the genome from scratch using a collection of reads (1,2). An accurate and complete assembly is crucial, as downstream analyses would be severely affected by low-quality assemblies (3). As the most widely used assembly strategy, *de Bruijn* graph based algorithms (4–8) split sequenced reads into kmers and then infer the sequence by decomposing the graph and finding the Eulerian path with relatively low computational cost (9,10). Although *de Bruijn* graph based approaches have intrinsic high computational efficiency, the complex repeats will result in a large number of fragmented contigs owing to the limited kmer size (11). Moreover, when the repeat length exceeds the read length, it will increase the difficulty of assembly (12,13).

One possible solution is to employ long read sequencing technologies (i.e. PacBio) to tackle with repetitive genome regions (14,15). But the relatively high sequencing cost and error rate have limited their wide application in *de novo* genome sequencing projects (16,17). An alternative approach is to utilize paired links between contigs produced by Illumina sequencing and then to perform the scaffolding process, in which the contigs will be ordered, oriented and connected. Genome scaffolding, however, usually faces two obstacles, noisy paired links from mapping biases or errors and repetitive regions in the genome. Traditional scaffolding algorithms generally employ two different strategies (Supplementary Figure S1A and S1B): (i) to choose seed contigs and extend from both ends through a specified minimum overlap with a limited number of mismatches (18); (ii) to take contigs as vertexes and paired links as edges to build paired link graph, and then linearize the graph (19–21). Both strategies are implemented in a post-assembly manner, and mainly focus on eliminating paired link noises and extending contigs. SSPACE (18), for example, hierarchically selects the longest contig and extends this contig by searching appropriate consistent sequences with a series of flexible parameters. SCARPA (21) proposes two denoising models, the odd cycle transversal model and feedback arc set model, to orientate and order contigs. SOPRA (22) takes several sophisticated models to denoise paired link, determine the direction of contigs using paired link and remove misassemblies. OPERA (23) creates scaffolds by gradually adding adjacent contigs sharing the most reliable paired links and removes noises through dynamic programing. Collectively, these paired link-based algorithms have made extensive efforts to establish criteria to abandon redundant contigs and to reduce paired link noises. However, repetitive contigs have been largely neglected by these tools. For example, Bambus2 (24), OPERA and SOPRA simply mask these repetitive contigs to simplify the assembly process, while some scaffolders can assemble a small

*To whom correspondence should be addressed. Tel: +86 10 84504172; Fax: +86 10 6488 0586; Email: zhfq@biols.ac.cn
†These authors contributed equally to this work as first authors.

fraction of them that are strongly associated with unique contigs but discard the remainders. For example, as shown in Supplementary Figure S1C–E, repetitive contig R_4 and R_5 are abandoned because of their forward forks, whereas R_1, R_2 and R_3 can be successfully connected at the end of the scaffold, owning to the sharing paired links between them and non-repetitive contigs N_1 and N_2.

Paired link, as an indirect linkage, intrinsically masks the sequences between two linked contigs and complicates the scaffolding process. Moreover, above paired link based approaches bypass the direct linkage embedded in the original *de Bruijn* graph, which are generated by splicing the forks during the stage of decomposing the assembly graph into contigs. Obviously, incorporating direct link will greatly increase the connectivity of the scaffolding graph and simplify the layout of contigs. Such advantage will improve the efficiency of denoising and more importantly facilitate the assembly of repetitive contigs. Recently, SPAdes (25) has been developed to assemble single-cell and multi-cells bacterial datasets, in which it employs a paired assembly graph for genome assembly. ExSPAnder (26), a module of SPAdes, creates pairing information on the assembly graph and extends scaffolds along the high scored edges. Although SPAdes was designed for single-cell sequencing, it outperforms other assemblers by producing increased assembly length and improved accuracy on isolated bacterial genomes or mini-metagenomes (25). However, it still cannot resolve complex regions including numerous short repetitive contigs, partly due to the fact that these regions are lacking of read pairs support and exhibit severe mapping biases. This situation is even more fundamentally skewed considering that short repetitive sequences represent the majority of the assembled contigs. Although most assemblers, such as SOAPdenovo2 (8) and ALLPATHS (4), have incorporated original connection information in the *de Bruijn* graph into the scaffolding process, the challenge of assembling repetitive regions is still unsolved.

Here, we propose a novel algorithm, inGAP-sf, based on the combination of direct link and paired link graphs to address above scaffolding obstacles. inGAP-sf employs direct link to provide extra routes and decreases the complexity of repetitive contigs enriched regions. With the supervision of paired link, true routes are determined and short repetitive contigs are specifically grouped by allocating into their corresponding routes. In addition, we construct a comprehensive model based on read pairs support estimation to eliminate the noise routes. By testing on simulated and real datasets, we demonstrated that inGAP-sf can significantly improve assembly continuity and genome coverage, as well as genome accuracy.

METHODS

Overview of inGAP-sf

The workflow of inGAP-sf can be divided into three main steps (Figure 1): (i) Direct link graph construction. Direct link is created based on the overlapped subsequence between two contigs. (ii) Paired link creation and pre-scaffold construction. Paired link is created between two contigs according to a bundle of paired reads mapped to these contigs. Unique contigs with exclusive paired link support are assembled into pre-scaffolds. After this step, the sketch of the genome is established. The pre-scaffolds containing gaps are filled and repetitive regions are resolved by DProute in the next step. (iii) DProute creation and extension (Figure 2). Firstly, DProutes are created by traversing the direct link graph under the guidance of paired link and are then filtered using a statistic-based estimation (SBE) model. Secondly, DProutes are extended and merged based on their overlapping paths and are split at the fork. Finally, the merged DProutes are used to fill gaps and to connect the pre-scaffolds that are broken by repetitive regions. inGAP-sf is freely available together with full documentation at https://sourceforge.net/projects/ingap-sf.

Direct link graph creation and simplification

We define direct link as a direct connection between two adjacent contigs decomposed from the *de Bruijn* assembly graph, sharing a certain length of overlaps with each other. Most of these directly connected contigs share a kmer-sized overlap, whereas a small number of them may have fewer or even no overlap due to the denosing process when decomposing the *de Bruijn* assembly graph. Therefore, to build a comprehensive direct link graph from pre-assembled contigs, the first step is to determine the kmer size used in constructing the *de Bruijn* graph. If the kmer size is not provided by the user, inGAP-sf utilizes a scanning approach to determine it. Briefly, subsequences with length ranging from 128 to 30 bp, are chopped from two ends of each contig, as well as their complementary sequences. A hash is built with the key representing the subsequence and the value representing its frequency. Kmer size is determined based on the change point of the frequency of hashed subsequences (Supplementary Figure S2A). For the assemblies generated by different assemblers (e.g. IDBA and SPAdes), we found that this kmer scanning strategy could also determine their actual kmer size (Supplementary Figure S3). After obtaining kmer size, direct link is created when a pair of contigs sharing kmer-sized overlap (Supplementary Figure S2B). Moreover, the remaining pairs of contigs, sharing overlaps shorter than kmer size, are also connected by gradually reducing the length of subsequences (Supplementary Figure S2C). Here, we define direct link graph as $G_D = (V, E)$, $V = \{c_i\}$, $E = \{c_i, c_j, dg_{ij}\}$, where c_i denotes contig i, and dg_{ij} is the gap size between contig c_i and c_j. This value is negative and generally equals to $-k$.

Once the direct link graph is constructed, bubble structures are merged to reduce the complexity of the direct link graph. To achieve this, inGAP-sf employs a key contig detection strategy. As shown in Supplementary Figure S2D, a key contig is defined as the removal of this contig will break the connectivity among the vertexes in a range of traversing depth (default 5). After detecting all key contigs, the routes between two adjacent key contigs are merged if both the difference and length of these routes are smaller than the parameters, *maxDiff* (default difference coverage rate is 0.3 and difference bases are 5) and *maxBubbleLength* (default one fold insert length). The former represents the threshold of divergence among the routes in the bubble, and the later

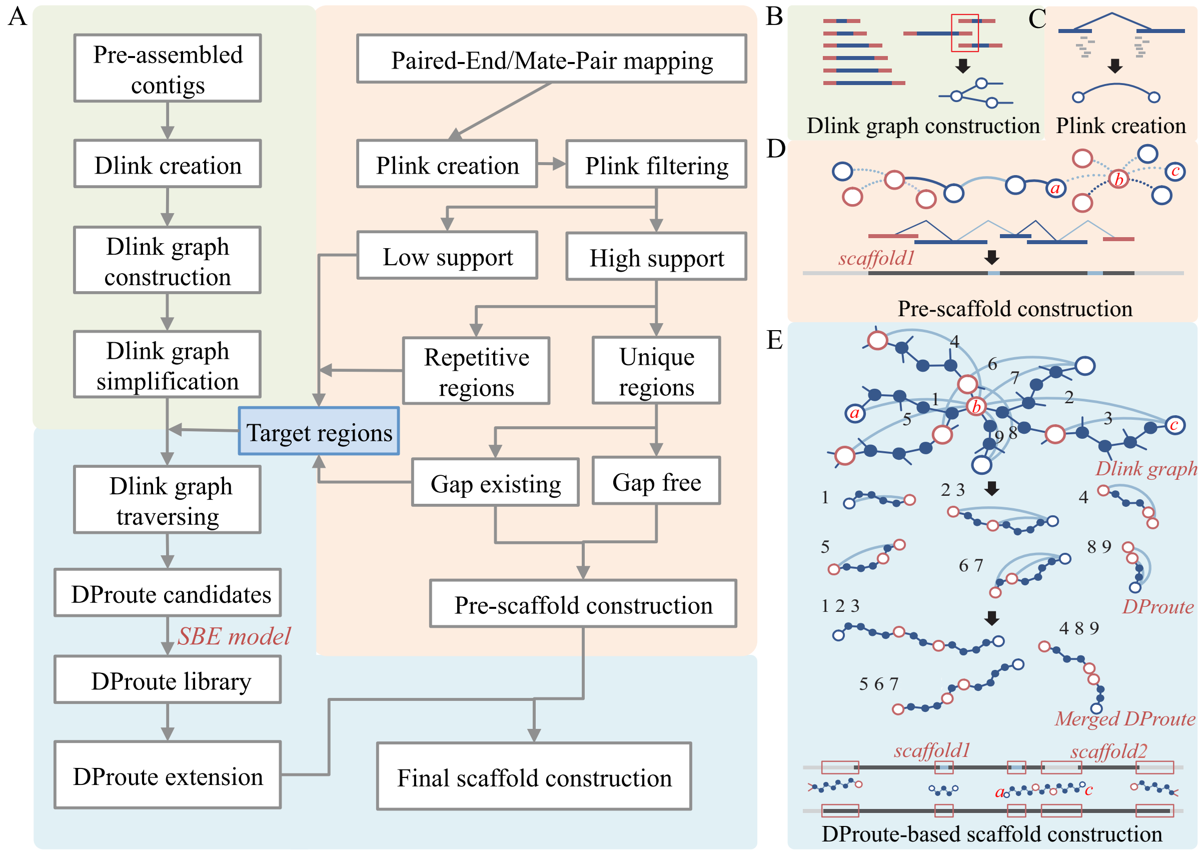

Figure 1. Overview of the inGAP-sf framework. (**A**) The workflow of inGAP-sf including direct link (Dlink) graph construction (light green), paired link (Plink) creation and pre-scaffold construction (light orange), and DProute-based scaffold construction (light blue). (**B–E**) A brief description of the three key steps implemented in inGAP-sf. (**B**) Direct link graph construction. Subsequences (red lines) are chopped from two ends of each contig. Direct link is created between two contigs if they share an overlapped subsequence. In the direct link graph, nodes denote the contigs and the line between two nodes refers to the direct link. (**C**) Paired link creation. Paired link is created between two contigs based on a bundle of pairing reads from paired-end or mate pair library. The curve and polyline denote the paired link. (**D**) Pre-scaffold construction. The layout of contigs can be determined by paired link. Conflict-free regions are assembled into pre-scaffold, whereas repetitive contigs are left unassembled. Red nodes/lines and blue nodes/lines denote repetitive and unique contigs, respectively. (**E**) DProute-based scaffold construction. DProutes are obtained by traversing on the direct link graph under the guidance of paired link (blue arcs). Then DProutes are merged or extended according to the consistent path. Finally, DProutes are used to fill the gaps in pre-scaffolds and to connect adjacent pre-scaffolds. The red quadrangles indicate newly assembled regions by the incorporation of DProutes, where the gap in the scaffold$_1$ is closed, and scaffold$_1$ and scaffold$_2$ are connected by the extended DProutes.

represents the threshold of length of the routes in the bubble.

Paired link creation and pre-scaffold construction

Paired link represents an indirect connection between two contigs, which are supported by a bundle of mapped paired-end or mate-pair reads. It is defined as a tuple $c_i, c_j, pg_{ij}, support_{ij}$ (c_i, c_j as abbreviation), where $support_{ij}$ is the number of mapped read pairs between contig c_i and c_j, and pg_{ij} is the gap size between these two contigs. The mean value of pg_{ij} is estimated by the following equation:

$$E(pg_{ij}) = \mu_{ins} + \overline{map\,Pos_i} + \overline{map\,Pos_j} - length(c_i) - length(c_j),$$

and the standard deviation is $SD(pg_{ij}) = \sigma_{ins}/\sqrt{support_{ij}}$, where $\overline{map\,Pos_i}$ is the average position of mapped reads on the contig c_i, μ_{ins} and σ_{ins} is the mean and standard deviation of library insert length, respectively. Considering that the bias in the real data may lead to a thicker tail or multiple peaks in the insert length distribution, we integrate a bias correction model into our paired link creation process, which contains three steps (Supplementary Figure S4). (i) Data cleaning and insert length estimation. Abnormal mapping with overwhelming mapping indels, extremely large insert length or false mapping direction, is detected and discarded. The remaining mapping information will be used to estimate the insert length distribution. Firstly, the distribution of library insert length is taken as a Gaussian mixture distribution. Expectation-maximum (EM) algorithm is used to recognize these Gaussian distribution, with a cluster number k ranging from 1 to 3. Secondly, according to Bayesian Information Criterion (BIC), the best cluster number k is further validated using Maximum Likelihood Estimation (MLE). Thirdly, the estimated insert length is determined and other Gaussian distribution are treated as noise, of which the proportion is calculated. The dataset *Burkholderia* sp. serves as an example, in which the best cluster number k is 3 and the EM result is: *Mean* = (762, 209, 427), and *SD* = (21, 56, 149). Be-

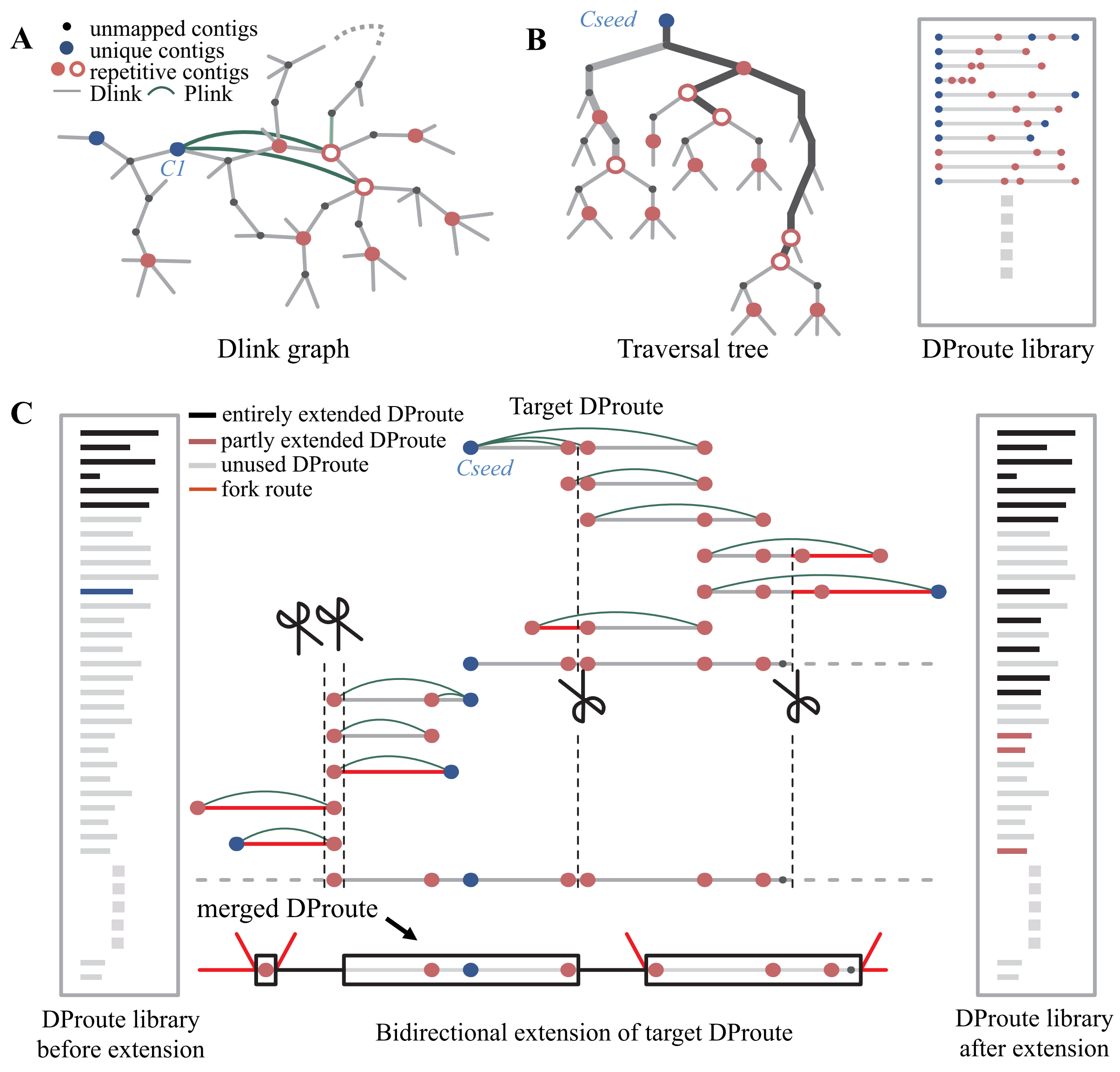

Figure 2. The algorithms of DProute creation and extension. (**A**) The direct link graph of a KIR region in the *S. aureus* assembly. The dashed lines represent the omitted direct link graph. (**B**) The traversal tree from the seed contig c_{seed}. If the distance between a node and c_{seed} falls outside the range of estimated gap size of paired link, this node is removed. Subsequently, the route score of each DProute is calculated using the SBE model and any DProute with route score below the given threshold is removed. The bold branches denote the traversal results which meet the gap size restriction and the black branches refer to the results which meet the SBE constraint. These DPoutes are put into library and will be extended in the next step. (**C**) Iteratively bidirectional extension of DProutes. The unused and partly extended DProutes are extended toward two directions using overlapped routes. If fork (red line) occurs which is caused by conflicting routes, the inconsistent parts on these routes are chopped. The chopped DProutes are reput into the DProute library and the fork position is recorded. This step is executed until all the DProutes are extended.

cause this dataset was produced using a library length of 800 bp, the real insert length is approximated to $N(762, 21^2)$ and the other peaks were treated as lower noise, which accounted for a proportion of 51% (Supplementary Figure S4A). (ii) Bias correction. After estimating the insert length and the proportion of noises in the library, the bundle of read pairs mapped to two contigs is extracted. For each bundle, the sum value, $map\,Pos_i + map\,Pos_j$, for a pair of reads is recorded into the set $SUMPEM$. Considering that these read pairs are sampling from the library, they should follow the same distribution with the library insert length. Therefore, we eliminate the noise with the same proportion from the set $SUMPEM$, and the remainders of read pairs in this bundle are used to estimate the gap size with the equation mentioned above (Supplementary Figure S4B). (iii) Repetitive contig splitting. In case that the standard deviation of $SUMPEM$ σ_{set} is far more larger than σ_{ins}, which indicates a repetitive contig occurs multiple times within insert length, we split $SUMPEM$ into several subgroups based on the ratio of σ_{set} to σ_{ins} (Supplementary Figure S4C). After performing these steps, paired link $c_i, c_j, pg_{ij}, support_{ij}$ is created based on the read pairs mapped to contig c_i and c_j', where c_j' refers to the reverse complement sequence of contig c_j.

In order to reduce computational complexity, unique regions are determined by the topology structure of direct link graph and contig depth, and the contigs in these regions are assembled into pre-scaffolds by using paired link. To achieve this, we first extract the contig pairs containing at least one large contig and sharing paired link. Then, the paired link is bypassed if the paired link support is more than triple estimated read pairs support standard deviation which is determined by the latter model. Finally, the contigs are assembled into pre-scaffolds using a greedy algorithm if there is no competition fork. After this process, the resulting pre-scaffolds are divided into three groups: pre-scaffolds without gaps, with gaps and with overhang regions caused by fork. Moreover, direct link will be employed in the following step to fill the gaps and to resolve the remaining unassembled contigs under the supervision of paired link.

Accurate gap size estimation is essential in determining the layout of the contigs, which are connected by paired link. In most cases, the estimated gap size can efficiently supervise the traversing to generate correct routes. However, the estimated gap size among contigs alone is unable to filter noise routes in complex regions. To address this problem, we develop a new model to estimate read pairs support, as well as a Bayesian scoring function depending on the SBE model to filter these routes.

Statistic-based estimation (SBE) model

The SBE model is designed to estimate the read pairs support between two contigs based on their length, distance and the distribution of insert length. In detail, inGAP-sf first classifies all contigs into three categories according to their length, large (above insert length), medium (between insert length and double kmer size) and small (below double kmer size) contigs. The insert length and double kmer size are used as thresholds, because the mapping distance of read pairs are limited by insert length and two directly connected contigs usually share a kmer-sized overlap. Next, the SBE model is developed based on the combination of different sized contigs (Supplementary Figure S5):

1) Combination of two large contigs ($L_1 > L_2 > ins$). As the paired-end reads are sequenced from both strands of insert fragments, the start position st and end position en of an insert fragment from each 5'-end are the same as the mapping position of the paired reads. Assume that the start positions of insert fragments are subject to uniform distribution and the insert length is subject to normal distribution. Given the start position st, if $st + ins > gap + m$, meaning that the reads mapping across the gap and the read pairs support will be accumulated by 1, where the ins, gap and $m/2$ represent the length of insert fragment, the gap between two mapped contigs and the minimum match length of the aligner, respectively (Supplementary Figure S5A). If all the insert fragments are taken as a whole, the read pairs support subjects to binomial distribution $B(n, p)$. The sample size n is $L_1 \cdot paircov$ and the probability p equals to $\frac{1}{L_1}\sum_{x=0}^{L_1} P(ins > gap + m + x)$, where $paircov$ refers to reads pairs coverage and L_1 is the length of contig c_1.

The probability p can be calculated as $p = -\frac{1}{2L_1}gap + b$, where $b = \frac{1}{L_1}(-\frac{\sigma}{\sqrt{2\pi}} + \frac{\mu - m}{2})$, μ and σ refer to the mean value and standard deviation of the insert length, respectively. The sample size n is reduced to $ins \cdot paircov$. For binomial distribution, the expectation $E(sup) = np$ and the deviation $SD(sup) = np(1 - p)$.

The average of read pairs support between two large contigs has a significant linear correlation with their gap size. In most instances, the deviation of the read pairs support decreases with the increasing of gap size. To explain these result, we consider the mapping position of paired reads as a Gaussian mixture distribution and the mean value of these Gaussian distributions have equal difference when the reads are sequenced uniformly (Supplementary Figure S5B). Thus, probability density function (PDF) of the mixture distribution (the mapping position of paired reads) can be determined while one read is mapped to contig c_1. As shown in Supplementary Figure S5C (upper), the probability p reduced linearly with the increasing gap size, but this reduction rate shifts slowly when the gap size increased enough (to *insertlength-minmatch*). To demonstrate this point, we used a Monte Carlo simulation ($gapmin = -81$, $gapmax = 500$, $miu = 500$, $sigma = 45$, $minmatch = 80$, $samplefrequence = 0.5$) and found that the simulation performed as expected (Supplementary Figure S5E). In addition, we assembled the simulated dataset *Ec-skewed* into contigs and extracted the gap size and the read pairs support among contigs which shared the same paired link. In most cases, the correlation between the gap size and the read pairs support fitted this model perfectly. Interestingly, when the gap size was negative, the read pairs support decreased slightly compared with expectation. The primary factor responsible for this result should be the mapping bias on overlapped contigs (Supplementary Figure S5F).

2) Combination of one large and one medium contigs ($L_1 > ins > L_2 > 2kmer$). Following the assumptions mentioned above, the distribution of end position en can be calculated. As shown in Supplementary Figure S5C (lower), the probability of paired reads falls on the contig c_2, p^*, decreases with the decreasing of contig c_2 length. Approximately, the reduction rate ($r_{reduction}$) is:

$$\begin{cases} r_{reduction} = \frac{L_2}{\mu - m - gap} & L_2 < \mu - m - gap \\ r_{reduction} = 1 & L_2 \geq \mu - m - gap \end{cases},$$

where L_2 refers to the length of contig c_2. Given $r_{reduction}$ and gap, the probability p^* is calculated with an extremely small approximation error: $p^* = p \cdot r_{reduction}$. Supplementary Figure S5F shows the correlation between gap size and read pairs support in simulated data. $r_{reduction}$ can be expressed as the slope and also fitted this model perfectly.

3) Combination of two medium contigs ($ins > L_1 > L_2 > 2kmer$). Most of the calculation processes are the same with the situation 2, except the sample size n^* is replaced by $L_1 \cdot paircov$ according to the length of contig c_1.

4) Combination of one small contig and one contig with arbitrary length ($2kmer > L_1 > L_2$ or $L_1 > 2kmer > L_2$).

The calculation of read pairs support used by above situations is no longer applicable, because the approximate error is too large to be ignored. Considering that contig c_2 is such short that the positions of mapped reads are limited in a small range, we simplify the start position st with a constant. As a result, $st + ins$ is subject to Gaussian distribution and thus, the probability p^+ is easily calculated as $F(\frac{gap+L_1-\mu+m}{\sigma}) - F(\frac{L_1-\mu+m}{\sigma})$, where F refers to the cumulative density function (CDF) of Gaussian distribution, and $gap + L_1 - \mu + m$ and $L_1 - \mu + m$ are the end and start positions of contig c_1, respectively. The sample size n^+ is $L_2 \cdot paircov$ according to the length of contig c_2 (Supplementary Figure S5D).

DProute creation and extension

A DProute starting from seed contig c_{seed} refers to a route, which is a list of connected contigs with determined gaps. It is generated by connecting the marker contigs, which share paired link with the seed contig c_{seed}, on the direct link graph (Figure 2A). Here, two types of contigs are selected as seed contigs: (i) large contigs (above insert length), because the paired link connecting these contigs has a high level of reliability; (ii) traversed contigs which have reads mapped, because the KIR regions can be resolved by the DProutes starting from these contigs. After selecting the seed contigs, an exhaustive strategy is employed to create DProute for seed contig c_{seed} and this process can be described as three major steps. Firstly, a heuristic search method is used to obtain DProutes. This approach starts at c_{seed} and uses contig c_k sharing paired link with c_{seed} as landmarks to reduce search space using the gap size based score, $gscore_{seed\ k}$, to limit search direction. Once $gscore_{seed\ k}$ is out of range (default 0.004), the search branch of c_k is cut entirely. For a pair of contigs: c_i and c_j, we define $gscore_{ij} = f(\frac{E(pg_{ij})-gap_{ij}}{SD(pg_{ij})})$, where gap_{ij} refers to the traversing distance between c_i and c_j, f refers to the Gaussian density function. The calculation of $E(pg_{ij})$ and $SD(pg_{ij})$ is mentioned in paired link creation part. Subsequently, traversal results are treated as DProute candidates (Figure 2B). Secondly, after obtaining these candidates, the distance between each contig in these DProutes on the route is determined. Then, the read pairs support based score, $sscore$, is calculated for the seed contig c_{seed} with each contig c_t on the DProutes. For a pair of contigs: c_i and c_j, we define $sscore_{ij} = f(\frac{support_{ij}-E(sup_{ij})}{SD(sup_{ij})})$, where $support_{ij}$ is the real read pairs support between contig c_i and c_j. The calculation of $E(sup_{ij})$ and $SD(sup_{ij})$ are mentioned in SBE model part. Especially, if there is no paired link between contig c_i and c_j, $support_{ij}$ is equal to 0 and $E(sup_{ij})$ and $SD(sup_{ij})$ are calculated based on contig length and the traversing distance. As read pairs support based score is more accurate than the gap size based score, the route score, which feasibly represents the probability of the DProute being correct, can be calculated as $\sum \frac{L_t-kmer}{L_{route}} sscore_{seed\ t}$, where L_t is the length of contig c_t and L_{route} refers to the length of DProute. Notably, when scaffolding with multiple libraries, the $gscore$ and $sscore$ are calculated for each library separately and the maximum score is reserved. Although being less accurate, the gap size based score is still employed prior to the read pairs support based score. The main reason responsible for this is that the read pairs support based score must be calculated through the determined distance between seed contig and every contig on the DProute. Moreover, calculating the read pairs support based score for a large number of DProutes in complex regions is a substantial computational challenge. In contrast, the gap size based score can be easily calculated based on the paired link during the process of traversing the direct link graph, which can efficiently reduce the amount of candidate DProutes and computational cost. Thirdly, DProutes starting from the same seed contig are extracted and the routes with high route read pairs support score are put into the DProute library (Figure 2B). In case of the coverage of c_{seed} is one fold, only one DProute starting from this contig will be reserved. Thus, DProute expressed as $(c_{r_1}, c_{r_2}, \ldots, c_{r_{m-1}}, c_{r_m})$ is created, where contig c_{r_1} is the seed contig and two adjacent contigs c_{r_i} and $c_{r_{i+1}}$ share direct link.

After the DProute library is created, DProute extension process is performed based on the overlapped contigs between DProutes. Given a target DProute $(c_{r_1}, c_{r_2}, \ldots, c_{r_i}, c_{r_i+1}, \ldots, c_{r_{m-1}}, c_{r_m})$, if another DProute $(c_{r_i}, c_{r_i+1}, \ldots, c_{r_{m-1}}, c_{r_m}, c_{r_{m+1}}, \ldots, c_{r_{m+n}})$ shares a consistent path $(c_{r_i}, c_{r_i+1}, \ldots, c_{r_{m-1}}, c_{r_m})$ with the target DProute, the extension process can be executed. The DProute which is used to extend the target DProute is called extended DProute. If more than two DProutes, for example, $(c_{r_i}, c_{r_i+1}, \ldots, c_{r_{m-1}}, c_{r_m}, c_{r_{m+1}}, \ldots, c_{r_{m+n}}, c_{r_p}, c_{r_{p+1}}, \ldots, c_{r_{p+j}})$ and $(c_{r_i}, c_{r_i+1}, \ldots, c_{r_{m-1}}, c_{r_m}, c_{r_{m+1}}, \ldots, c_{r_{m+n}}, c_{r_q}, c_{r_{q+1}}, \ldots, c_{r_{q+l}})$, share consistent path with the target DProute, but the inconsistent paths $(c_{r_p}, c_{r_{p+1}}, \ldots, c_{r_{p+j}})$ and $(c_{r_q}, c_{r_{q+1}}, \ldots, c_{r_{q+l}})$ exist between them, the extension process will be stopped at the fork (mark after $c_{r_{m+n}}$). The detailed route extension process can be described as follows (Figure 2C): Firstly, the DProute starting from the longest seed contig is chosen to be the target DProute, because the DProute starting from long contigs tend to be more reliable. Secondly, the DProutes in library sharing consistent path with target DProute are used to extend the target DProute forward and backward. The extension process continues until the fork occurs in extended DProutes. The resulting routes are termed as merged DProutes. Thirdly, entirely extended DProutes including target DProute are removed from the DProute library, whereas partly extended DProutes with recorded fork are put back into the DProute library and will be extended again. This whole DProute extension process is executed until DProute library is empty. At last, all merged DProutes are checked to merge bubble routes (redundant assembly sequence), remove tip routes (noise assembly sequence), and resolve circle routes (repetitive sequence).

Finally, these merged DProutes are used to fill the gaps within pre-scaffolds and to connect adjacent pre-scaffolds.

Assembly continuity evaluation based on counting breakpoints

To evaluate the continuity of an assembly, all breakpoints between adjacent contigs along the reference genome were detected. Firstly, a contigs list, representing the order of all the contigs on the genome linked by the direct link, was created by aligning the contigs to the reference genome. The unambiguously aligned contigs were recorded and used to determine the location of conflicting repetitive contigs based on their direct link connections. Next, the combined sequence derived from the contig list was aligned back to the reference genome to correct false placements in the contig list. Finally, spliced diplex and triplet contigs list were aligned to the assembly to determine breakpoints. Obviously, scaffolding results with fewer assembly breakpoints are better.

Test datasets and parameter settings

All simulated datasets were generated by a sequencing data simulator pIRS v111 (27). The error rate was set to 0.5%, and other parameters were set to default values. Firstly, to test the dependency of short library in genome scaffolding, a pair of datasets, *Sa-multi* and *Sa-single*, were simulated from *Staphylococcus aureus* (GCF_000027045.1). The former contained two libraries with insert length of 180 and 500 bp, respectively. The later contained a library with insert length of 500 bp. Secondly, to test the performance on biased insert length, a pair of datasets, *Ec-norm* and *Ec-screwed*, were simulated from *Escherichia coli* (GCF_000005845.2). Dataset *Ec-screwed* had a lower tail compared with dataset *Ec-normal*. Thirdly, to test the performance of paired-end contamination in the mate pair library, a pair of datasets, *Rs-confree* and *Rs-contaminate*, were simulated from *Rhodobacter sphaeroides* (GCF_000012905.2). The former contained one paired-end library with insert length of 300 bp and one mate pair library with insert length of 3 Kbp. Whereas, the later contained an additional 25% of paired-end reads in the mate pair library. Besides, to test the performance on large genomes, *Sc-sim* and *Pf-sim* were simulated from *Saccharomyces cerevisiae* (GCF_000146045.2) and *Plasmodium falciparum* (GCF_000002765.3), respectively. Detailed description of these datasets was shown in Supplementary Table S1.

Eight real sequencing datasets were downloaded from the NCBI SRA database, with SRR522165 and SRR522163 for *Escherichia coli*, DRR008626 and DRR008628 for *Burkholderia* sp., ERR422403 and ERR163029 for *P. falciparum*, SRR018008, SRR018009 and SRR018012 for *Grosmannia clavigera*, respectively. Detailed insert length, sequencing coverage, read number and read length were shown in Table Supplementary Table S3. For each dataset, the reads were first trimmed by Sickle v1.210 (https://github.com/najoshi/) and then corrected by using SOAPec v2.01 (http://soap.genomics.org.cn/) on the trimmed reads.

For each dataset except *P. falciparum* and *G. clavigera*, pre-assembled contigs were generated using SOAPdenovo2 (with arguments "all -R -F -E -w -u -s -D 1 -d 1 –M 1"). For these two exceptions, the reads were pre-assembled using SOAPdenovo2 (with modified arguments "map -k 31"). Note that the multiple kmer module of SOAPdenovo2 was used when assembling these datasets. Five widely used scaffolders, OPERA, SCARPA, SOPRA, SSPACE and *scaf* module implemented in SOAPdenovo2, were executed on these datasets with their default parameters. As for inGAP-sf, to save computing time, it performed scaffolding based on the combination of contigs and scaffolds generated by SOAPdenovo2. After obtaining the scaffold sequences, all the assembled sequences larger than 1kbp are evaluated by QUAST v4.3 (28).

RESULTS

The introduction of direct link graph can improve genome scaffolding

Compared with the paired link graph, the direct link graph exhibits significantly decreased graph complexity. To demonstrate such deduction and quantify their difference, we used sequencing reads from *E. coli* to compare the properties of both types of graphs. This dataset was assembled into contigs using SOAPdenovo2 (8), and linkages among the resulting contigs were used to construct the direct link and paired link graphs separately. We first used the ratio of edges to vertexes and degree of vertexes to evaluate the graph complexity. As shown in Supplementary Figure S6, the ratio of edges to vertexes for the paired link graph was 5.49, and 1.33 for the direct link graph, which contained 857 additional vertexes compared to the paired link graph. In addition, >15% of the vertexes on the paired link graph had a degree >10. In contrast, nearly all the vertexes on the direct link graph had a low level of degree, ranging from 2 to 4. Such significantly decreased complexity in the direct link graph will greatly facilitate the graph traversing and route selection. To obtain a more intuitive understanding of the direct link graph complexity, we explored the kmer-sized interspersed repetitive (KIR) contigs in the direct link and paired link graphs, which represented the most dominant type of repetitive contigs in the assembly graph. As shown in Supplementary Figure S6, it is evident that on the paired link graph, the vertexes of KIR contigs were tangled together and they had an average degree up to 12. Thus, the scaffolding algorithms solely relying on the paired link tend to misassemble these vertexes into chimeras or to break them into fragmented assemblies. In contrast, most of the vertexes in the direct link graph exhibited a substantial reduced topological complexity.

During the scaffolding process, the assembly continuity should benefit from the reduced complexity of direct link graph, especially when resolving repetitive contigs. To validate this point, inGAP-sf, SOAPdenovo2, OPERA, SSPACE, SOPRA and SCARPA were used to scaffold a pre-assembled dataset *E. coli* (7,032 contigs with mean length of 729 bp assembled from 150 × paired-end reads with 540 bp insert length). For each tool, the assembled scaffolds were aligned to the reference to detect breakpoints using spliced contig order list, which could reflect the continuity of the assembly. As shown in Supplementary Figure S7, inGAP-sf outperformed all other tools, with the smallest number of breakpoints, indicating that the introduction of direct link graph in inGAP-sf can remarkably improve the assembly. Since KIR regions are widely present in all kinds of genomes, we further tested the ability of each tool on decomposing these regions. We detected all KIR regions on the reference genome using a kmer-based scanning approach and checked whether these regions were associated with the breakpoints of the assembly by each scaffolder. As shown in Supplementary Figure S8C, all paired link based scaffolders failed to assemble most of these KIR regions. In contrast, inGAP-sf correctly ordered nearly all the repetitive contigs (>90%) in these regions, except those with a length larger than insert size (technically *insertlength-minmatch*). To get a comprehensive understanding on how inGAP-sf resolves such KIR regions, we surveyed the KIR contigs that could be assembled by inGAP-sf and explored their associated direct link and paired link. We found that the KIR contigs can be specifically allocated by paired link into different routes through integrating with direct link, which connected single KIR contig. As shown in Figure 3A and B, all the given contigs were connected by direct link and several possible routes in this region could be obtained. N_2, R_3 and R_1, R_4 grouped KIR contigs R_1, R_2 and R_3 together. R_1, R_7 and R_2, R_8 separated R_1, R_2 and R_2, R_3 into different groups, respectively. As a result, the KIR region from N_2 to R_4 was successfully assembled with the supervision of all the paired link in this region (Figure 3C). Similarly, the other two complex regions were also resolved. Two examples of assembled KIR regions using the dataset *E. coli* were illustrated, where Figure 3D showed a long tandem repeat and Figure 3E showed an interspersed repeat with three mixed routes located on the *E. coli* genome at around 737K, 1530K and 3623K, respectively. For both cases, inGAP-sf could make an accurate assembly.

Taken together, the introduction of direct link graph can greatly improve genome scaffolding, especially for KIR regions. Besides the direct link graph, such improvement can also be attributed to precise gap estimation and the SBE model implemented in inGAP-sf, which will be evaluated in next sections.

Precise gap size estimation

We compared the accuracy of inGAP-sf on gap size estimation with other five tools using two simulated datasets, *Ec-normal* and *Ec-skewed* (Supplementary Table S1). The insert length of the former followed a normal distribution, whereas the later possessed a skewed lower tail. For both data sets, we used the above six tools to scaffold pre-assembled contigs, and compared the contigs against both the reference genome and the resulting scaffolds to measure the distance between adjacent contigs. As shown in Supplementary Figure S8A, compared with other tools, inGAP-sf exhibited a decreased standard deviation of estimated gap size and the smallest difference between the estimated and true gap size on both data sets. Considering that gap size estimation can severely affect the continuity, accuracy and the length of uncertain bases, we compared the assemblies of inGAP-sf with those of other tools. For the dataset *Ec-skewed*, all tools except inGAP-sf showed much worse performance on NGA50, assembly accuracy and the number of uncertain bases compared with the *Ec-normal* dataset. In contrast, the NGA50 and accuracy of inGAP-sf only dropped slightly in the skewed dataset, suggesting its advanced performance on gap size estimation.

We further tested these tools on a real dataset *Burkholderia* sp., and as expected, inGAP-sf achieved the best performance on gap size estimation (Supplementary Figure S8B). We calculated all gap sizes in the scaffolds, which was inferred from aligning the paired linked contigs to assembled scaffolds, and compared them to the true gap sizes. As shown in Table 1, inGAP-sf outperformed the other tools at all levels of accuracy. Moreover, most of the detected gaps by inGAP-sf matched the ground truth. Notably, the other five tools either produced poor gap estimations on repetitive contigs or failed to assemble these contigs into scaffolds (Supplementary Figure S8C). inGAP-sf remarkably increased both the accuracy of gap estimation and the number of assembled repetitive contigs, indicating its substantial advantage over other tools.

The SBE model facilitates DProute filtering and traversal

Accurate gap estimation is necessary but not sufficient for selecting correct DProutes. As an essential improvement, inGAP-sf leveraged a newly developed mathematical model to further screen out correct routes by estimating the read pairs support between two linked contigs. To examine the accuracy of this model, we used the assembly result of the dataset *E. coli* as a benchmark. First, we extracted the referential gap size which was inferred by aligning contigs back to the reference genome and paired link support. Next, we classified all contigs into three categories according to their length (above insert length, below double *kmer size* and the remainder), and then the combination of these categories of contigs resulted in four situations (see *Methods*). For each situation, we evaluated the accuracy of the SBE model by measuring the difference between estimated and referential read pairs support, as well as the estimation deviation. We found that this model was consistently reliable across all four situations (Figure 4A–H and Supplementary Figure S9). The expectation of estimated read pairs support based on the referential gap size was equal to the regression of referential read pairs support, with most referential read pairs support cases in the range of estimation. In the first three situations, the estimation deviation followed a normal distribution ($P > 0.1$, Shapiro-Wilk test). We also revealed that this excellent performance should not be attributed to a large standard deviation of estimated read pairs support (Figure 4C). Regarding to the last but most complicate situation, at least one of the adjacent contigs was shorter than double *kmer size* and in most cases it was a KIR contig. Therefore, we used an entirely different strategy to estimate the read pairs support, where the estimation deviation approximately followed a normal distribution and only 124 (6.31%) and 67 (3.41%) data points were out of the triple and quadruple standard deviation, respectively (Supplementary Figure S9F). It should be noted that these data points could be corrected in the latter DProute creation and extension. When applying the SBE model to the dataset *P. falciparum* (Supplementary Figure S10), we also observed an excellent performance on read pairs support estimation, indicating a high level of robustness of the SBE model.

To further evaluate the SBE model on DProute selection, we compared read pairs support based score with the gap size based score using the dataset *E. coli*. After constructing the direct link graph, the DProutes were extracted from the KIR regions and the correctness of each DProute was determined by aligning the corresponding sequence of

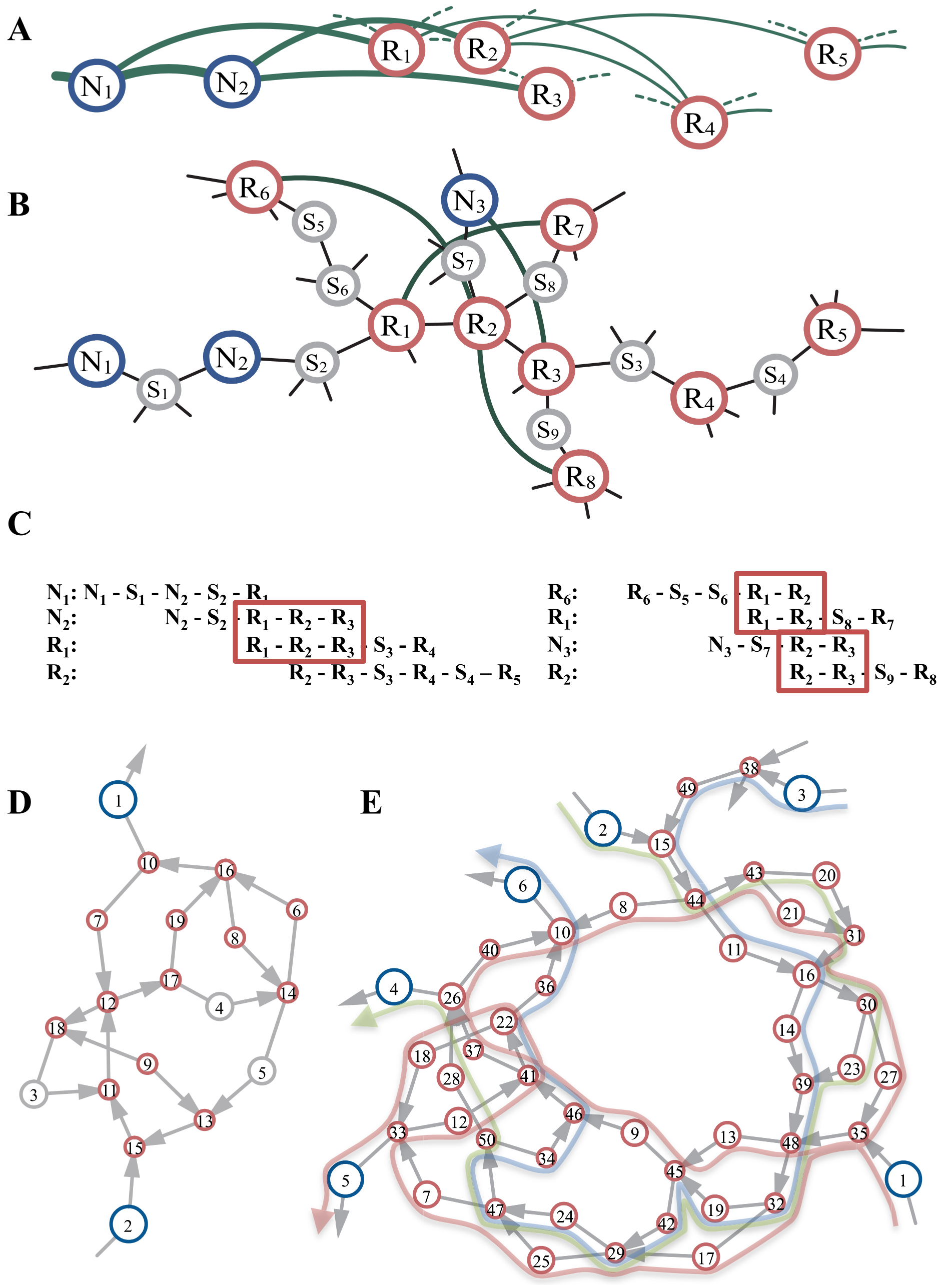
A
B
C
N1: N1 - S1 - N2 - S2 - R1
N2: N2 - S2 - R1 - R2 - R3
R1: R1 - R2 - R3 - S3 - R4
R2: R2 - R3 - S3 - R4 - S4 – R5
R6: R6 - S5 - S6 - R1 - R2
R1: R1 - R2 - S8 - R7
N3: N3 - S7 - R2 - R3
R2: R2 - R3 - S9 - R8
D
E

Figure 3. The assembly of KIR regions by introducing the direct link graph. (**A**) The paired link layout of a KIR region in the dataset *Sa-multi*. Thick green curves denote the paired links that are used for generating scaffolds by most paired link based algorithms, whereas thin green curves represent the paired links that are excluded because of the occurrence of the fork. Dashed curves refer to the paired links which connect the other part of the genome. The repetitive contigs, R_4 and R_5, are failed to be assembled because the downstream paired links of R_1, R_2 and R_3 are conflicting. (**B**) The direct link graph of the same KIR region. (**C**) Improved assembly of the KIR region by the combination of direct link and paired link graphs. R_1, R_2 and R_3 are connected by direct links and are grouped together by the paired link, N_2, R_3 and R_1, R_4. Because R_1, R_2 and R_2, R_3 are grouped together by other paired links, R_4 is successfully assembled into the scaffold. R_5 is also assembled in the same way. (**D** and **E**) Two examples of KIR regions in the *E. coli* assembly. The long tandem repeat (**D**) and the interspersed repeat (**E**) are pre-assembled into fragmented contigs but they can be assembled into scaffolds by inGAP-sf. Light red, blue and green routes refer three mixed routes which are resolved by inGAP-sf, located on the *E. coli* genome at around 737K, 1530K and 3623K, respectively.

Table 1. Performance comparison between inGAP-sf and other five assemblers on four real datasets

Dataset	Size (Mb)	Tools	Largest contig (kbp)	NGA50 (kbp)	>1kbp Scaffolds	Total Length (kbp)	Large MA	Local MA	Indel (kbp)	N (kbp)	GF (%)
E. coli (220 bp + 540 bp)	4.7	OPERA	209.2	61.6	126	4505	5	7	9.5	8.2	97.0
		SCARPA	178.8	84.7	124	4562	2	40	4.4	26.6	97.6
		SOAPdenovo2	411.1	146.3	62	4662	0	751	0.3	159.3	96.9
		SOPRA	328.7	117.3	74	4533	4	30	11.4	4.3	97.6
		SSPACE	273.6	58.9	143	4649	2	394	49.6	52.5	97.6
		inGAP-sf	414.6	171.5	56	4591	1	3	0.1	0.7	98.0
E. coli (540 bp)	4.7	OPERA	209.2	61.6	126	4505	5	7	9.5	8.2	97.0
		SCARPA	61.1	12.6	518	4504	2	6	1.7	11.6	96.6
		SOAPdenovo2	413.4	133.1	64	4615	2	209	0.2	48.8	97.8
		SOPRA	328.7	112.2	82	4532	6	19	11.4	4.6	97.7
		SSPACE	167.1	59.4	134	4628	0	605	0.3	102.3	97.4
		inGAP-sf	414.6	169.3	57	4585	1	3	0.1	0.8	98.0
Burkholderia sp. (300 bp + 800 bp)	7.1	OPERA	603.5	245.7	35	6898	15	32	9.6	8.3	98.9
		SCARPA	673.8	309.7	57	6921	1	24	9.0	15.0	99.1
		SOAPdenovo2	1274.0	728.5	26	6936	0	64	2.0	21.4	99.2
		SOPRA	838.2	373.7	58	6911	1	15	9.4	3.5	99.1
		SSPACE	730.8	299.7	38	6983	5	53	37.7	32.7	99.3
		inGAP-sf	1835.3	686.8	25	7051	2	0	0.1	1.9	99.4
P. falciparum (500 bp + 3 kbp)	23.6	OPERA	37.9	4.4	2687	21149	3856	177	88.3	59.7	89.2
		SCARPA	215.0	31.6	1530	22147	75	1,739	157.6	937.3	90.1
		SOAPdenovo2	37.9	3.3	6301	19869	23	210	17.3	56.9	84.1
		SOPRA	78.4	14.1	2662	21772	64	1,456	202.3	567.4	89.5
		SSPACE	52.4	7.0	4052	21059	27	1,857	27.0	1392.4	83.5
		inGAP-sf	109.7	17.9	2537	22170	59	925	95.6	79.2	89.6
G. clavigera (200 bp + 700 bp)	29.7	OPERA	241.8	41.9	1008	25747	109	1,807	185.5	341.6	84.6
		SCARPA	760.5	127.0	941	27526	158	203	87.0	350.6	90.0
		SOAPdenovo2	481.3	108.3	988	28681	79	4,002	103.2	1907.5	89.1
		SOPRA	617.4	80.7	1086	27372	373	799	136.0	287.2	89.1
		SSPACE	248.0	61.8	1453	28500	111	1,618	159.5	730.8	89.9
		inGAP-sf	496.7	105.1	1017	28420	165	179	97.5	287.5	89.2

Dark gray and light gray highlight the first and second rank for each indicator, respectively. LargeMA, LocalMA, Indel, N and GF are the indicators which are given by QUAST, which refer to the number of large misassembly, the number of local misassembly, indel length, uncertain bases and genome fraction, respectively.

this route back to the reference genome. Subsequently, two scores were obtained by measuring each DProute. Then, the DProutes sourced from the same seed contig were ranked according to their calculated scores. We firstly surveyed the accuracy of each rank obtained by using these two models. As shown in Figure 4I, the accuracy of both models showed a tendency towards deduction from the first rank to the last rank, suggesting that the higher the score, the more feasible the DProute was. Secondly, we compared the first rank accuracy between the two models and found that the SBE model outperformed the gap size estimation model, with the accuracy increased from 74.8% to 93.3%. Such enhancement was essential to the continuity of the assembly, because DProutes in the first rank were always chosen as true routes. Notably, for KIR regions, there should be more than one true DProutes. Therefore, it was crucial for the model to screen out these DProutes in each rank. To examine the ability of the SBE model on DProute selection, we evaluated its sensitivity and specificity by comparing selected DProutes to true DProutes. As shown in Figure 4J, the SBE model

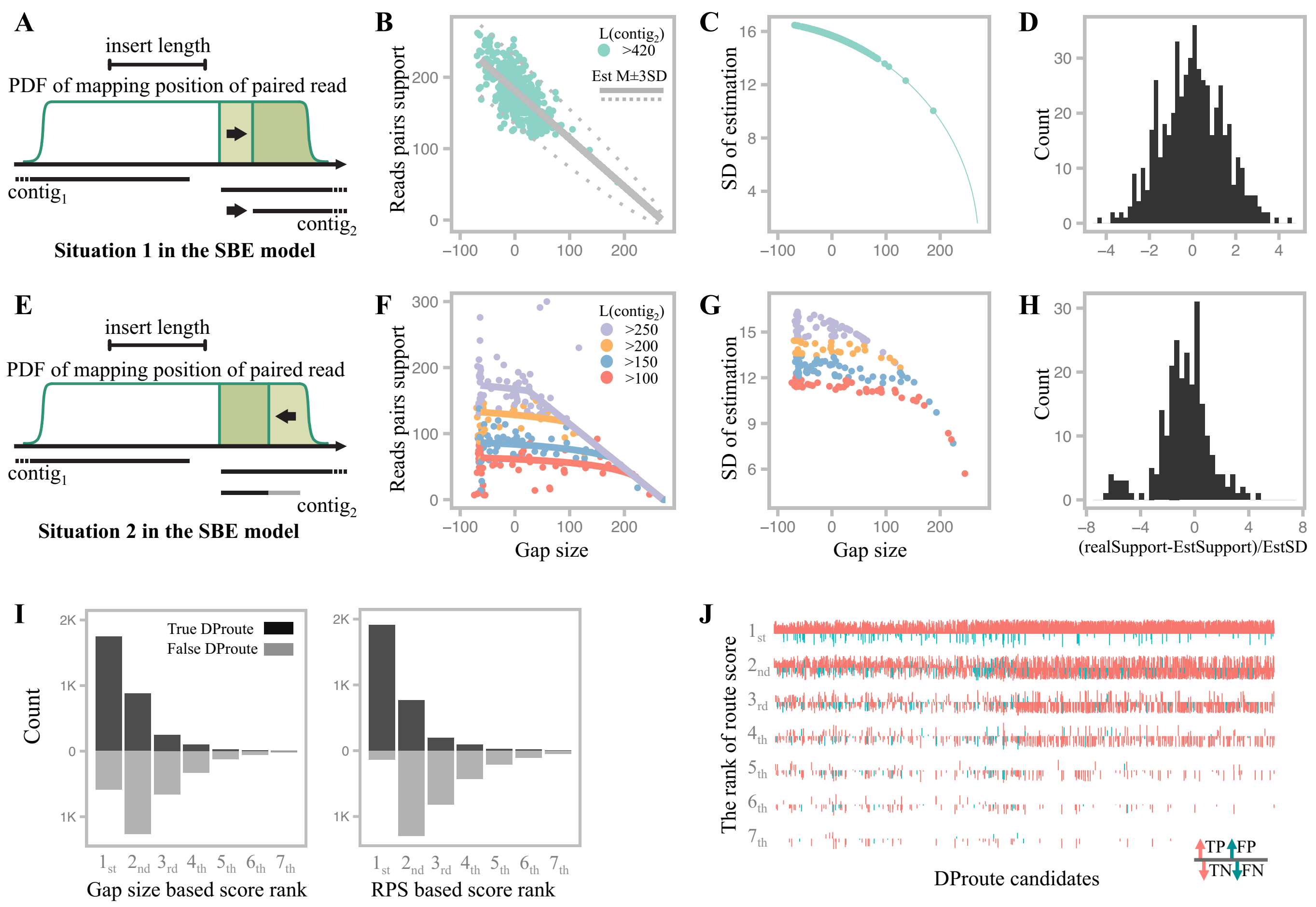

Figure 4. Performance evaluation of the SBE model on DProute creation and selection. The performance of the SBE model on the first two situations, L(contig$_1$) > L(contig$_2$) > *insertlength* (**A–D**) and L(contig$_1$) > *insertlength* > L(contig$_2$) > 2**k* (**E–H**), were evaluated. (**A**, **E**) The sketch of the SBE model. The probability of the event (paired reads mapped across the gap), *p*, decreases with increasing gap size between contigs (**A**) and decreasing contig length (**E**). Light green and green regions represent decreased *p* and remaining *p*, respectively. (**B**, **F**) Relation between the gap size and the paired link support. After estimating read pairs support by the SBE model and the gap size model, most of the paired link support, 96.65% for the first situation (**B**) and 91.57% for the second situation (**F**), fell within the range of triple standard deviation of estimation. Notably, for the first situation, the estimation of read pairs support was calculated based on the gap size model. Since the estimation of read pairs support was determined by the gap size and the length of contig$_2$, for the second situation, the estimation was calculated separately according to the mean length of each interval. (**C**, **G**) Relation between gap size and the standard deviation of the estimation. (**D**, **H**) Distribution of the estimation biases for the two situations. (**I**) Comparison between gap size based score and read pairs support based score for DProutes. DProutes sourced from the same seed contig were ranked according to the calculated scores. Obviously, the read pairs support based scoring system was more efficient to select the correct DProute than gap size based scoring system. (**J**) DProute filtrations. The read pairs support based scores of DProutes from the same seed contig were in the same column and ranked. The DProute, of which the score below the cutoff, was abandoned and expressed as down arrow. Conversely, the remaining DProutes were put into the DProute library and expressed as up arrow. The length of the arrow refers to the score. True positive and true negative were shown in red color, which accounted for 92.56%. False positive and false negative were shown in green color.

achieved substantially high sensitivity (98.7%) by detecting most of the true DProutes with sacrificing some specificity (85.8%). These false routes can be easily filtered during the route extension process (see *Methods* and Supplementary Figure S11), because these routes may form tip structures.

Benchmarking with simulated datasets

To comprehensively evaluate the performance of inGAP-sf, we constructed eight simulated datasets (Supplementary Table S1). The sequencing library simulation was designed to test the influence of short insert length on scaffolders. The bias simulation was to investigate the impact of biased library insert length on scaffolding. The contamination simulation was designed to examine whether the scaffolder could avoid the influence of PE-contamination in mate-pair libraries. The species simulation was to investigate the ability of scaffolders on assembling large eukaryotic genomes. All these datasets were firstly assembled into contigs using SOAPdenovo2, and then the contigs were assembled into scaffolds by inGAP-sf and five other tools. We evaluated the performance of these tools on the continuity, misassemblies and genome coverage. It should be noted that NGA50 and uncertain bases represent the continuity of genome assembly, but aggressively greedy algorithm implemented in most scaffolding tools tends to boost NGA50 by sacrificing assembly accuracy. Hence, we surveyed the number of large and local misassemblies, as well as indel length, in the assembled scaffolds.

As shown in Supplementary Table S2, for all simulated data sets, inGAP-sf produced an improved or comparable performance with increased NGA50 and genome coverage, as well as decreased number of misassemblies and uncertain bases. The library simulation showed that owing to the utility of the direct link graph and the SBE model, inGAP-sf could overcome the drawbacks of other tools that were dependent on multiple short-read libraries with different insert sizes to fill gaps and to increase the linkage density on the paired link graph. The bias simulation indicated that owing to its highly accurate gap estimation, inGAP-sf was robust to biased sequencing data with non-normal insert size distribution, which may cause significantly increased uncertain bases in other tools. Similarly, inGAP-sf could also distinguish paired-end reads contamination from mate pair sequencing data using precise gap size estimation. The species simulation showed that the difficulty of scaffolding grows with increasing genome size. For the dataset *Sc-sim*, inGAP-sf outperformed all other tools in all compared aspects, whereas for the dataset *Pf-sim*, it ranked the second after OPERA on all aspects except the number of uncertain bases.

These results collectively indicated that by applying accurate gap estimation and the SBE model, inGAP-sf can enhance genome assembly in the aspects of continuity, accuracy and completeness. Moreover, by introducing the direct link graph, inGAP-sf achieved the best genome coverage, especially on repetitive regions, across all simulated datasets. Such improved genome assembly will profoundly facilitate downstream analyses.

Performance evaluation on real datasets

We further evaluated the performance of inGAP-sf and the other five tools using four real datasets with different genome sizes (Table 1). Similar to the results on simulated datasets, inGAP-sf exhibited much better performance on the two small genomes than the other five tools, with increased NGA50 length and genome coverage, as well as decreased number of misassemblies. For the two datasets *P. falciparum* and *G. clavigera* with 23.6 and 29.8 Mb genome size, respectively, inGAP-sf achieved a comparable performance with respect to the other tools. We found that the primary factor responsible for this result was the fragmented direct link graph in these two datasets, in which the ratio of edges to vertexes was as low as 0.64 and 0.46, respectively. In contrast, in the other two datasets, the ratio of edges to vertexes was more than 1.6. If there is no sufficient direct link information among the pre-assembled contigs, inGAP-sf has to use the paired link graph for scaffolding and thus the results should be comparable to other tools.

SPAdes performed assembly by finding a genomic path in the assembly graph based on read pairs mapping (25). Different with most scaffolders, SPAdes generated both contigs and scaffolds through paired assembly graph. We further compared the performance of inGAP-sf with SPAdes on three real datasets (Supplementary Figure S12). For each dataset, inGAP-sf scaffolded the contigs pre-assembled by SOAPdenovo2. Regarding to SPAdes, each dataset was assembled into scaffolds from scratch. After contig generation, we found that the NGA50 of SPAdes was at least 1.8-fold larger than that of SOAPdenovo2. However, after scaffolding, inGAP-sf significantly increased the continuity of each assembly and outperformed SPAdes in the *Burkholderia* sp. and *P. falciparum* genome assemblies, and yielded a comparable result in the dataset *E. coli*. In particular, inGAP-sf produced a NGA50 >2- and 1.5-fold larger than that of SPAdes in the datasets *P. falciparum* and *G. clavigera*, respectively. Notably, the initial contig NGA50 length used for scaffolding of SPAdes was at least 2-fold larger than that of inGAP-sf, indicating that although SPAdes introduced the direct link into genome scaffolding, inGAP-sf obtained more integrated scaffolds by resolving complicated repetitive regions.

Running time and memory usage

To evaluate the efficiency of inGAP-sf, the running time, memory usage and CPU utilization rate were compared with other five scaffolders using four datasets. As shown in Figure 5, for smaller genomes with fewer KIR contigs, inGAP-sf achieved a comparable time consumption and memory requirement compared with all other scaffolders. But the memory and CPU time cost of inGAP-sf dramatically increased accompanied with the increase of KIR contigs and it exhibited a medium level of performance as compared with other tools (Supplementary Figure S13). It should be noted that both SOAPdenovo2 and SSPACE performed read mapping during the scaffolding process, which may lead to an overestimation of running time in all cases, whereas the other scaffolders started from parsing read mapping files. SOPRA had the highest time consumption and the factor attributed to this result was that it employed several sophisticated but time-consuming modules in scaffolding and resolving misassemblies (22).

DISCUSSION

Paired link based scaffolders abandoned the original connectivity of assembled contigs and the scaffolding was treated as an independent process. However, the lost connectivity information has precluded these methods from assembling repetitive contigs, especially the KIR regions. Considering that the length of assembled contigs follows a power law distribution, KIR contigs account for a significant proportion in assembled sequences. Moreover, such small-sized repetitive contigs were widely distributed along the genome, including coding regions. For example, when assembling the *Escherichia coli* genome by SOAPdenovo2 using a *kmer size* of 63, 37% of the genes were fragmented by KIR contigs (Data not shown). Although small in size, the high fraction of KIR contigs should severely affect downstream genome analyses. To address this challenge, we present a novel algorithm, inGAP-sf, for effective KIR contigs resolving and efficient genome scaffolding. The main advantage of inGAP-sf is that it introduces the direct link graph to cluster and link KIR contigs and also the SBE model to screen out correct routes from numerous noise routes in repetitive regions. Through extensive evaluations on both simulated and real datasets, we demonstrated that

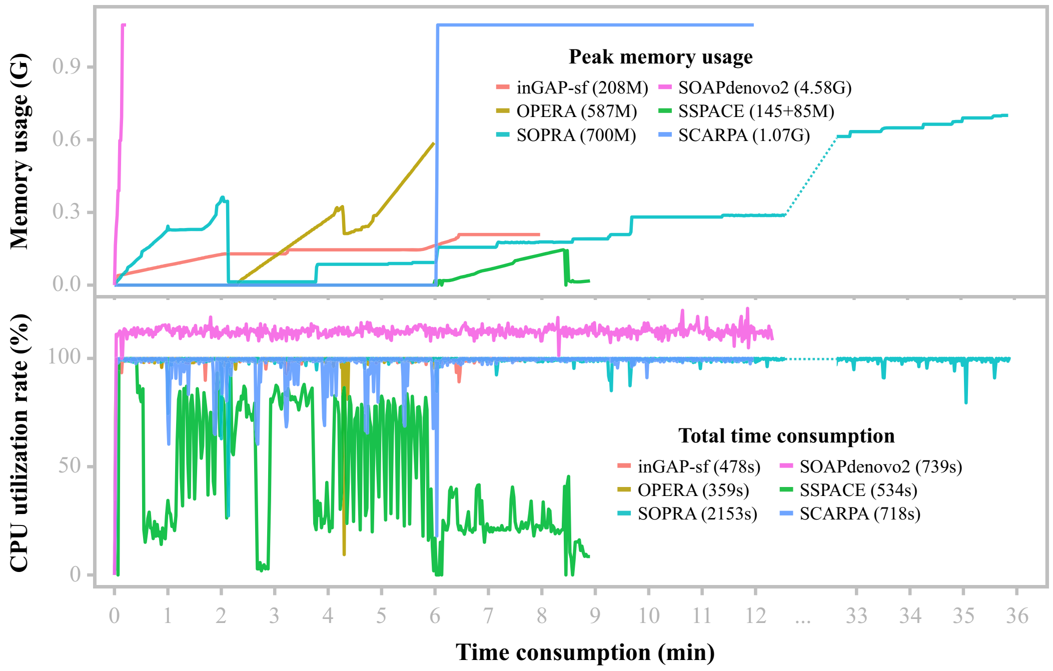

Figure 5. Comparison of time consumption, real-time tracking of memory usage and CPU utilization rate among six methods on the dataset *Burkholderia* sp.

inGAP-sf has improved continuity, accuracy and completeness of genome assembly than other known scaffolders.

Compared with the paired link graph, the direct link graph exhibits decreased graph complexity in the aspects of the ratio of edges to vertexes and the degree of vertexes. In addition, this type of graph contains remarkably simplified topology structures on KIR regions compared with the paired link graph. More importantly, the layout of KIR contigs on the direct link graph can be directly and accurately determined by direct link, and they can further be specifically allocated into the corresponding DProutes using paired link. Therefore, most of the KIR regions within insert length can be resolved by integrating direct link with paired link graphs. Collectively, these features of the direct link graph remarkably facilitate improved continuity, accuracy and completeness of genome assembly. Similar to inGAP-sf, SPAdes performed assembly by finding a genomic path in the assembly graph based on the mapping position of read pairs and this assembler also included a scoring function to resolve repeats (25). SPAdes has been successfully used to assemble single-cell genomes and mini-metagenomes (29,30). However, the regions containing multiple connected KIR contigs are bypassed during scaffolding by SPAdes, because these regions are lacking of paired link support, which may result in failures of its scoring system. In contrast, inGAP-sf addresses this problem by utilizing the SBE model, which is independent to paired link supports.

inGAP-sf is a stand-alone software and can serve as a drop-in replacement for current *de Bruijn* graph based assemblers. The input files for inGAP-sf include the contigs and the read mapping results, which can be generated by using BWA (31), BWA-mem, Bowtie (32,33) or SOAPaligner (34). inGAP-sf parses the read mapping results to generate the paired link graph, the efficiency of which varies among different read mappers. For example, due to their inherent limitation on mapping reads to contig ends, both BWA and Bowtie exhibit considerably decreased paired link supports compared with SOAPaligner and BWA-mem. Especially for small contigs, this situation is even worse for BWA or Bowtie, which usually leads to a severe paired link loss on the KIR regions. Take the assembly of dataset *Burkholderia* sp. for example, the paired link supports derived from BWA mapping results were only ∼40% of those by SOAPaligner (Supplementary Figure S14), which resulted in a significantly increased number of assembly breakpoints. The direct link creation of inGAP-sf largely relies on the provided contigs, which may vary among different *de Bruijn* graph-based assemblers. When the overlapped sequences between two adjacent contigs contains mismatches, hashing based strategy will fail to create the direct link for them. To address this problem and ensure the applicability to various assemblers, inGAP-sf leverages a direct link recovery strategy to improve the connectivity of the direct link graph by finding overlaps between any two contigs from the region within a paired link. In most cases, this strategy exhibits an excellent performance. For example, as shown in Supplementary Figure S15, after applying the direct link recovery process on the *E. coli* datasets (ERX002508, ERX008638), the connectivity of the direct link graph significantly increased. However, if the pre-assembled contigs have been extensively filtered when decomposing the *de Bruijn* graph, the direct link graph will not be improved by using this strategy.

In summary, we present an efficient scaffolder for assembling short repetitive contigs. More importantly, we propose a new strategy that combines the direct link and paired link graphs to achieve accurate genome assembly and to improve the continuity of genome scaffolding. Further efforts could be made to improve the running time and the efficiency of assembling large genomes by partitioning the graph into many small overlapped subgraphs. We believe that inGAP-sf is a significant improvement over current genome scaffolding algorithms and provides novel insights into *de novo* assembly algorithm development.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

FUNDING

Natural Science Foundation of China (NSFC) [31671364, 91531306]; Strategic Priority Research Program of the Chinese Academy of Sciences [XDB13000000] and National Key R&D Program [2016YFC1200804]. Funding for open access charge: NSFC.
Conflict of interest statement. None declared.

REFERENCES

1. Nagarajan,N. and Pop,M. (2013) Sequence assembly demystified. *Nat. Rev. Genet.*, **14**, 157–167.
2. Miller,J.R., Koren,S. and Sutton,G. (2010) Assembly algorithms for next-generation sequencing data. *Genomics*, **95**, 315–327.
3. Denton,J.F., Lugo-Martinez,J., Tucker,A.E., Schrider,D.R., Warren,W.C. and Hahn,M.W. (2014) Extensive error in the number of genes inferred from draft genome assemblies. *PLoS Comput. Biol.*, **10**, e1003998.
4. Butler,J., MacCallum,I., Kleber,M., Shlyakhter,I.A., Belmonte,M.K., Lander,E.S., Nusbaum,C. and Jaffe,D.B. (2008) ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Res.*, **18**, 810–820.
5. Simpson,J.T., Wong,K., Jackman,S.D., Schein,J.E., Jones,S.J.M. and Birol,I. (2009) ABySS: a parallel assembler for short read sequence data. *Genome Res.*, **19**, 1117–1123.
6. Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.
7. Li,R.Q., Zhu,H.M., Ruan,J., Qian,W.B., Fang,X.D., Shi,Z.B., Li,Y.R., Li,S.T., Shan,G., Kristiansen,K. *et al.* (2010) De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res.*, **20**, 265–272.
8. Luo,R., Liu,B., Xie,Y., Li,Z., Huang,W., Yuan,J., He,G., Chen,Y., Pan,Q. and Liu,Y. (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, **1**, 1–6.
9. Idury,R.M. and Waterman,M.S. (1995) A new algorithm for DNA sequence assembly. *J. Comput. Biol.*, **2**, 291–306.
10. Pevzner,P.A., Tang,H. and Waterman,M.S. (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U.S.A.*, **98**, 9748–9753.
11. Li,Z., Chen,Y., Mu,D., Yuan,J., Shi,Y., Zhang,H., Gan,J., Li,N., Hu,X., Liu,B. *et al.* (2012) Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and de-bruijn-graph. *Brief. Funct. Genomics*, **11**, 25–37.
12. Schatz,M.C., Delcher,A.L. and Salzberg,S.L. (2010) Assembly of large genomes using second-generation sequencing. *Genome Res.*, **20**, 1165–1173.
13. Ukkonen,E. (1992) Approximate string-matching with q-grams and maximal matches. *Theoret. Comput. Sci.*, **92**, 191–211.
14. Chaisson,M.J., Huddleston,J., Dennis,M.Y., Sudmant,P.H., Malig,M., Hormozdiari,F., Antonacci,F., Surti,U., Sandstrom,R. and Boitano,M. (2015) Resolving the complexity of the human genome using single-molecule sequencing. *Nature*, **517**, 608–611.
15. Berlin,K., Koren,S., Chin,C.-S., Drake,J.P., Landolin,J.M. and Phillippy,A.M. (2015) Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat. Biotechnol.*, **33**, 623–630.
16. Koren,S. and Phillippy,A.M. (2015) One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. *Curr. Opin. Microbiol.*, **23**, 110–120.
17. Quick,J., Quinlan,A.R. and Loman,N.J. (2014) A reference bacterial genome dataset generated on the MinION portable single-molecule nanopore sequencer. *Gigascience*, **3**, 22.
18. Boetzer,M., Henkel,C.V., Jansen,H.J., Butler,D. and Pirovano,W. (2011) Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics*, **27**, 578–579.
19. Sahlin,K., Vezzi,F., Nystedt,B., Lundeberg,J. and Arvestad,L. (2014) BESST–efficient scaffolding of large fragmented assemblies. *BMC Bioinformatics*, **15**, 281.
20. Mandric,I. and Zelikovsky,A. (2015) ScaffMatch: scaffolding algorithm based on maximum weight matching. *Bioinformatics*, **31**, 2632–2638.
21. Donmez,N. and Brudno,M. (2013) SCARPA: scaffolding reads with practical algorithms. *Bioinformatics*, **29**, 428–434.
22. Dayarian,A., Michael,T.P. and Sengupta,A.M. (2010) SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, **11**, 345.
23. Gao,S., Sung,W.K. and Nagarajan,N. (2011) Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *J. Comput. Biol.*, **18**, 1681–1691.
24. Koren,S., Treangen,T.J. and Pop,M. (2011) Bambus 2: scaffolding metagenomes. *Bioinformatics*, **27**, 2964–2971.
25. Bankevich,A., Nurk,S., Antipov,D., Gurevich,A.A., Dvorkin,M., Kulikov,A.S., Lesin,V.M., Nikolenko,S.I., Pham,S. and Prjibelski,A.D. (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.*, **19**, 455–477.
26. Prjibelski,A.D., Vasilinetc,I., Bankevich,A., Gurevich,A., Krivosheeva,T., Nurk,S., Pham,S., Korobeynikov,A., Lapidus,A. and Pevzner,P.A. (2014) ExSPAnder: a universal repeat resolver for DNA fragment assembly. *Bioinformatics*, **30**, i293–i301.
27. Hu,X., Yuan,J., Shi,Y., Lu,J., Liu,B., Li,Z., Chen,Y., Mu,D., Zhang,H. and Li,N. (2012) pIRS: Profile-based Illumina pair-end reads simulator. *Bioinformatics*, **28**, 1533–1535.
28. Gurevich,A., Saveliev,V., Vyahhi,N. and Tesler,G. (2013) QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, **29**, 1072–1075.
29. Magoc,T., Pabinger,S., Canzar,S., Liu,X., Su,Q., Puiu,D., Tallon,L.J. and Salzberg,S.L. (2013) GAGE-B: an evaluation of genome assemblers for bacterial organisms. *Bioinformatics*, **29**, 1718–1725.
30. McLean,J.S., Lombardo,M.J., Badger,J.H., Edlund,A., Novotny,M., Yee-Greenbaum,J., Vyahhi,N., Hall,A.P., Yang,Y., Dupont,C.L. *et al.* (2013) Candidate phylum TM6 genome recovered from a hospital sink biofilm provides genomic insights into this uncultivated phylum. *Proc. Natl. Acad. Sci. U.S.A.*, **110**, E2390–2399.
31. Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, **25**, 1754–1760.
32. Langmead,B. and Salzberg,S.L. (2012) Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, **9**, 357–359.
33. Langmead,B., Trapnell,C., Pop,M. and Salzberg,S.L. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.
34. Gu,S., Fang,L. and Xu,X. (2013) Using SOAPaligner for short reads alignment. *Curr. Protoc. Bioinformatics*, doi:10.1002/0471250953.bi1111s44.