UCLA
UCLA Previously Published Works

Title
Genome reassembly with high-throughput sequencing data

Permalink
https://escholarship.org/uc/item/66p3401c

Journal
BMC Genomics, 14(Suppl 1)

ISSN
1471-2164

Authors
Parrish, Nathaniel
Sudakov, Benjamin
Eskin, Eleazar

Publication Date
2013-01-21

DOI
http://dx.doi.org/10.1186/1471-2164-14-S1-S8

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Parrish et al. BMC Genomics 2013, 14(Suppl 1):S8
http://www.biomedcentral.com/1471-2164/14/S1/S8

BMC Genomics

PROCEEDINGS Open Access

Genome reassembly with high-throughput sequencing data

Nathaniel Parrish[1], Benjamin Sudakov[2], Eleazar Eskin[1*]

From The Eleventh Asia Pacific Bioinformatics Conference (APBC 2013)
Vancouver, Canada. 21-24 January 2013

Abstract

Motivation: Recent studies in genomics have highlighted the significance of structural variation in determining individual variation. Current methods for identifying structural variation, however, are predominantly focused on either assembling whole genomes from scratch, or identifying the relatively small changes between a genome and a reference sequence. While significant progress has been made in recent years on both *de novo* assembly and resequencing (read mapping) methods, few attempts have been made to bridge the gap between them.

Results: In this paper, we present a computational method for incorporating a reference sequence into an assembly algorithm. We propose a novel graph construction that builds upon the well-known de Bruijn graph to incorporate the reference, and describe a simple algorithm, based on iterative message passing, which uses this information to significantly improve assembly results. We validate our method by applying it to a series of 5 Mb simulation genomes derived from both mammalian and bacterial references. The results of applying our method to this simulation data are presented along with a discussion of the benefits and drawbacks of this technique.

Introduction

Within a species, individual genomes differ from one another by a certain amount of genetic variation. These variations exist at different scales, ranging from single nucleotide variants (SNVs), to small-scale insertions and deletions (indels), up to large structural variations (SVs) of kilo- to mega-base scale. Many studies in genomics are focused on characterizing the content of these variations and identifying associations with diseases or other phenotypes [1,2]. While SNVs have been widely studied in recent years, larger-scale structural variations have been more dificult to characterize. Despite this, studies have shown a strong correlation between SVs and genetic disorders, including Crohn's disease and Down's syndrome [3-5].

In recent years, the development of high-throughput sequencing (HTS) technologies has made it possible to sequence an individual genome rapidly and at low cost. However, the problem of how to interpret this sequencing data remains. Traditionally, one of two approaches is taken. In *de novo* assembly, we consider the target (or donor) genome in isolation, using no information from prior assemblies. The output of a *de novo* assembly is a set of short sub-sequences, or contigs, representing the donor genome. Modern *de novo* assemblers typically employ a de Bruijn graph [6,7], and may take advantage of additional information from paired-end sequencing data [8,9] or multiple sequencing technologies [8] to join contigs into longer sequences.

De novo assembly contrasts with the alternative approach of resequencing. In this approach, we assume that the donor genome differs only by SNVs and indels from some reference genome. Resequencing, also known as read-mapping, takes advantage of the reference genome to map the sequencing reads to some position on the reference and identify the variations from the consensus of all mapped reads. Recent implementations of resequencing algorithms may also utilize paired-end sequencing data to disambiguate reads that map to multiple locations on the reference genome [10-12]. Some information derived from the read mappings, such as discordant read pairs (paired-end reads

* Correspondence: eeskin@cs.ucla.edu
[1]Department of Computer Science, University of California Los Angeles, Los Angeles, California, USA
Full list of author information is available at the end of the article

BioMed Central
© 2013 Parrish et al.; licensee BioMed Central Ltd. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

which map to conflicting positions on the reference), may be employed to detect the presence and content of structural variation. This is discussed further below.

It is helpful to consider these two approaches, *de novo* assembly and resequencing, in terms of their prior assumptions. *De novo* assembly methods assume no prior knowledge of the genome being assembled and instead treat each genome as if it represents a novel organism. Conversely, resequencing techniques assume the existance of only small variations from the reference genome. In many cases, both of these assumptions may be unrealistic. In particular, many genomic studies are focused on identifying the differences between genomes which are largely similar, but which may also contain large structural variations. For example, it is estimated that between two human individuals the total genetic variation may be as much as 8 Mb of sequence content [13]. In such cases neither *de novo* assembly nor resequencing adequately capture the correct assumptions and as a result may fail to identify the full range of variations present in the sequencing data. In particular, *de novo* assemblers generate large number of contigs, and provide little information about their relative ordering in the genome, making them unsuitable for identifying specific variations between individuals. Resequencing algorithms work very well for identifying SNVs in unique (non-repeat) regions of the genome that are largely conserved between the donor and the reference, but do not provide information on the larger structural variations.

To address this problem, a number of methods have been developed to both identify the loci of larger SVs [14-16] and estimate their content. These methods can be thought of as post-processing steps that take as input the data produced in resequencing and apply additional computation to those sequencing reads which are not consistent with the resequencing assumptions. Different methods exist, some focusing on characterizing *copy-number* variations (CNVs) [17], while others focus on large-scale insertions and deletions [18,19]. We believe these methods are limited in two key ways. First, they rely heavily on the results of resequencing, which can be unreliable in the presence of repeats, translocations, and inversions. Second, the methods tend to be highly specialized, focusing on a single type of mutation. Obtaining a complete picture of the genome thus remains difficult.

Because of this difficulty, many studies continue to rely on *de novo* assembly, even for organisms for which a high-quality reference exists. This is both computationally inefficient, typically requiring more than 100 GB of memory to compute, as well as undesirable in its results, which while unbiased, do not leverage the prior work that has been done in creating high-quality reference sequences.

A number of software packages have been developed in recent years with the aim of utilizing a set of reference genomes to produce a more optimized scaffolding, or layout, of the contigs produced in *de novo* assembly. OSLay [20] uses a maximum-weight matching algorithm to identify likely neighboring contigs. Treecat [21] builds a fully connected graph of the contigs, with edges weighted by the distance between syntenic regions in the reference, and attempts to find a minimum-weight Hamiltonian path through the graph using a greedy heuristic. Finally, PGA [22] uses a genetic algorithm to search the space of possible contig orderings. By relying on the contigs produced through *de novo* assembly, however, these methods may not take full advantage of the reference genome.

Our aim in this paper is to propose a novel model for the assembly of a donor genome which uses the reference as a guide, and to show how this approach improves assembly results over pure *de novo* assembly. Towards this goal, we formulate a novel graph construction capturing the similarities between the two genomes, and present the genome reassembly problem as a means of finding the valid set of paths through this graph. We present the results of our work on simulation data generated from both mammalian and bacterial genomes, and discuss the benefits and challenges of applying our method.

Results

Here we present the results of our work, beginning with a brief overview of our method. We follow this with a discussion of our simulation results and the implications for the feasibility of our method.

Method overview

As an example of how a reference sequence can aid in assembly, consider the de Bruijn graph of a donor genome "ATAGAGGCAATGAGCGTGGAGTTC" in Figure 1a. Note that this graph has two possible Eulerian tours, one in which the lower branch is taken first, and one in which the upper branch is taken first. Only one of these tours spells the original donor genome, and a *de novo* assembly can not distiguish between them. If, however, we are given the reference sequence "ATAGCAATCGTGTTC," then it may be possible to discerne the correct tour. Figure 1c depicts the original de Bruijn graph augmented with the reference sequence, represented by red edges. In this new graph, it is now possible to choose the tour that is most *parsimonious* with the reference sequence. Stripping away the red edges that have no parallel blue edge, as shown in Figure 1d makes this more clear. We can now see that the tour following the upper branch first touches the reference edges 0, 3, 4, 8, and 11 in sequential order, with novel content in between. This indicates that the donor genome spelled by this tour represents a donor with only three insertions relative to the reference. The tour following the lower branch first touches the reference edges out of order in the sequence 0-8-3-4-11, indicating three insertions as

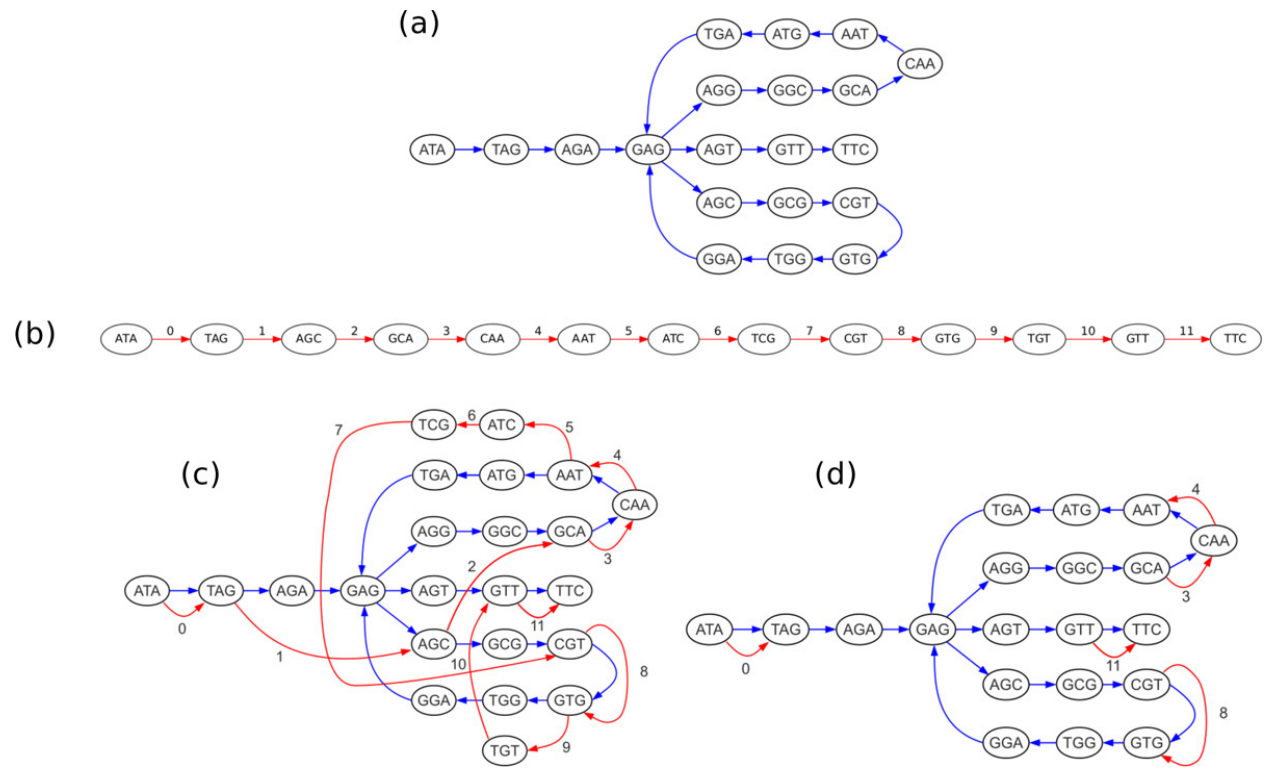

Figure 1 A motivating example demonstrating how the use of a reference can help discover the most parsimonious traversal of the de Bruijn graph. (a) de Bruijn graph of the donor sequence "ATAGAGGCAATGAGCGTGGAGTTC". (b) de Bruijn graph of the reference sequence "ATAGCAATCGTGTTC", including edge index labels. (c) Graph combining the donor and reference sequences. (d) Graph stripped of red edges with no parallel blue edge.

well as a translocation. By appealing to parsimony, we can therefore conclude that the tour taking the upper branch first is the more likely of the two.

With this idea in mind, our method begins by building a graph of the contigs in the donor sequence. The construction of these contigs is flexible, and they may be derived from the sequencing reads through either *de novo* assembly or a hybrid process using both resequencing and assembly. Similar to the example above, in which parallel red edges were used to indicate local alignment between the donor and reference sequences, here each contig is compared to the reference genome and annotated with information denoting the local alignments of the contig to the reference. Our goal now is to find a tour of the graph, which corresponds to an ordering of the contigs, such that the size of "gaps" between aligned subsequences is limited by some value τ, which will be a parameter to our method. The problems with generating such a tour are two-fold. First, there may be spurious alignments caused by repeat or translocated regions, which will confound naïve attempts to traverse the graph. Second, there may be contigs with no alignments, representing sequences that are novel in the donor genome. We resolve these problems through a two-phased process of propagation and pruning, in which each contig first receives the set of alignments at contigs that are reachable within a distance of τ, then progressively eliminates spurious alignments by attempting to match them against alignments in adjacent contigs. The result is a graph in which much of the desired tour can be logically determined by inspection of the remaining alignment annotations. In those cases where the contig order can be determined, the adjacent contigs are merged as a means of simplifying the graph and promoting further elimination of spurious alignments.

It is important to note that while we use the alignment information in our method, it is never assumed that any specific alignment is correct. We believe this is a strength compared to other methods that more heavily rely on read mapping and as a result may be more biased towards the reference.

Simulation results

In order to validate our method, we design a simulation framework using two reference genomes; the O157:H7 strain of the *E. coli* bacterial genome (NCBI NC011353.1), and chromosome 1 of the reference mouse genome (NCBIM37). The E. coli genome, at roughly 5 Mb in length, is used in its entirety, while we generate a

simulation reference from the mouse genome by sampling a 5 Mb subsequence from the chromosome.

For each reference genome, we generate simulated donor genomes by applying a series of mutations to the reference, including insertions, deletions, duplications, and translocations. We vary the average size of the mutation events from 5 Kb to 50 Kb, such that these events comprise roughly 15% of donor genome. We further apply a set of SNV mutations at a rate of 0.1%.

We generate simulated paired-end sequencing data from each donor genome using a read length of 100 bp and fixed insert size of 500 bp. In all cases we assume error-free reads and uniform coverage (a read from every position). While these assumptions are unrealistic in practice, correcting for read errors and variable coverage are orthogonal problems which have been studied independently [23-25]. Although our method will need to be extended to account for the types of data encountered in real-world studies, we believe these preliminary results show significant promise.

For each data set, we perform paired-end assembly using Velvet [9] as a performance baseline, using a k-mer size of 99 bp. We then apply our own method and demonstrate that we are able to acheive significant improvements in both the number and size of assembled contigs. We validate that our contigs remain accurate by aligning them back against the simulation donor genome and observe that on average fewer than 1% are misassembled. This indicates that our method is relatively conservative, and does not bias excessively towards the reference. Refer to Table 1 for the results of our experiments on three different simulated donor sequences derived from the mouse reference.

We further evaluate our method by comparing two different strains of the E. coli bacteria, using the O157: H7 strain as a reference, and the K-12 strain as a donor. The K-12 strain is significantly shorter in length than the O157 strain, indicating the presence of large-scale deletions. This is supported by our assembly results, which indicate mutation events up to 120 Kb in size. The full results of our simulations on the E. coli reference are reported in Table 2.

It is important to note that while comparisons against *de novo* assemblers such as Velvet provide a valuable baseline for performance metrics, our method incorporates a significant source of additional information (the reference genome). Direct comparisons are therefore inherently unfair. Our results are instead intended to show the possible extent to which *de novo* results could be improved upon through the incorporation of existing reference sequences and reasonable assumptions.

Methods

Let R be a reference genome and D be our donor genome. We define S_R and S_D to be multisets (allowing repeats) of subsequences of the reference and donor genomes, respectively. In all cases, S_R represents the spectrum of k-mers, a single k-mer sampled from every position in R. S_D is dependent on our sequencing technology but in all cases is an approximation of the spectrum. We proceed first by defining a series of graph constructions that facilitate our method, then describe a message-passing formulation of our method that is concise and simple to implement.

Reference/donor graphs

Given the multisets of k-mers S_R and S_D, we can construct de Bruijn graphs $G_R = \{V_R, E_R\}$ and $G_D = \{V_D, E_D\}$, where V_R (V_D) is the union of all (k - 1)-mers in S_R (S_D) and E_R (E_D) is the multiset sum of all k-mers in S_R (S_D). In more simple terms, we are given a set of k-mers, either sampled directly from the reference or generated by breaking up reads of length l in the donor sequencing data into l - k + 1 k-mers, and we construct a graph such that every k-mer is represented by an edge. Because we are interested in the similarities beteween the donor and the reference, it is helpful to combine both the reference and the donor in a single graph as described below.

Definition: a **reference/donor graph** G_{RD} is the superposition of de-Bruijn graphs G_R and G_D such that $G_{RD} = \{V_R \cup V_D, E_R \uplus E_D\}$, where the $\uplus$ operator indicates multiset sum. In order to maintain distinction between reference and donor edges, we assign an edge color of red to E_R and blue to E_D in the construction of G_{RD}. Each reference edge $e \in E_R$ is annotated with an integer index *e.pos* equal to the corresponding position of the edge in the reference genome.

Refer to Figure 2 for a simple example of such a graph. While we are interested in constructing tours of

Table 1 Results of running both Velvet and our method on simulated mouse chromosomes.

	Velvet			Our method			
Donor genome	# Contigs	N50	Max contig	# Contigs	N50	Max contig	Accuracy
Mouse, 5 Kb	1014	14315	56677	**352**	**73042**	**288172**	**99.7%**
Mouse, 25 Kb	773	19038	102858	**386**	**88473**	**227406**	**99.7%**
Mouse, 50 Kb	705	21721	98684	**410**	**117127**	**336208**	**99.2%**

Each simulated chromosome is 5 Mb in length, containing roughly 15% mutated content, using mutation event sizes of 5, 25, and 50 kb. In each case, contig statistics are given for the Velvet assembly and for our method, along with the accuracy of our computed contigs. Accuracy is calculated as the percentage of computed contigs that align back to the donor genome.

Table 2 Results of running Velvet and our method on *E.coli*-based genomes.

	Velvet			Our method			
Donor genome	**# Contigs**	**N50**	**Max contig**	**# Contigs**	**N50**	**Max contig**	**Accuracy**
E. Coli O157, 5 Kb	1034	25477	158013	**422**	**56750**	**274293**	**99.5%**
E. Coli O157, 25 Kb	870	71194	286061	**727**	**96535**	**285958**	**99.6%**
E. Coli K12	166	125649	327149	**33**	**429486**	**734812**	**97.0%**

In the first three cases, simulated donor genomes are derived from the O157 reference by applying a series of mutations (insertions, deletions, and SNVs). In the final case, the K12 *E. coli* strain is used as a donor and compared against the O157 reference.

this reference/donor graph, the traditional definition of an Euler tour in which each edge is visited exactly once no longer applies. We are only intersted in tours which exclusively touch donor edges, as these are the only tours that represent the sequencing data. This leads to the following simple definition.

Definition: a **donor tour** of a reference/donor graph is a complete tour that includes only blue (donor) edges.

In other words, a donor tour of G_{RD} is equivalent to an Euler tour of G_D.

We are now interested in a concise way to characterize the similarities between the reference and the donor. We start by considering those cases in which a red and blue edge are parallel in the graph (denoted by the || operator). The notation and terminology we will use in discussing these cases is defined as follows.

Definition: a donor edge $e \subseteq G_{RD}$ is considered **reference-parallel** if there exists a reference edge $R(e) \in G_{RD}$ such that $R(e) \mid\mid e$. A sequence of donor edges $E = \{e_1, e_2, \ldots, e_n\}$ is considered reference-parallel if for every pair of consecutive edges $e_i, e_{i+1} \in E$ there exist parallel reference edges $R(e_i), R(e_j) \in G_{RD}$ such that $R(e_j).pos = R(e_i).pos + 1$. A sequence of donor edges $E = \{e_1, e_2, \ldots, e_n\}$ is considered **novel** if it is not reference-parallel.

Definition: the reference indexes of a reference-parallel sequence are the values $R(e_1).pos, R(e_2).pos, \ldots, R(e_n).pos$ and are concisely represented as the pair $m = (R(e_1).pos, R(e_n).pos)$, referred to as a **reference marker**. The beginning and end of the reference-parallel sequence are referred to as *m.start* and *m.end*, respectively. Given an edge e, the set of all reference markers associated with the edge is denoted $markers(e)$.

Refer to Figure 2c for an example of a complete reference/donor graph with a reference-parallel sub-sequence, along with the associated reference marker. Having defined this notion of reference markers, it is natural to consider the associations *between* them. In particular, we are interested in pairs of reference markers that are within a certain distance in the graph, and which represent nearby segments of the reference genome.

Definition: given two reference markers m_i and m_j, we say that m_i **connects to** m_j within distance d if $m_j.start - m_i.end < d$ and the markers are separated by at most d edges in the graph. This relationship is indicated by $m_i \xrightarrow{d} m_j$.

With this graph construction, we can now de ne the genome reassembly problem.

The genome reassembly problem: given a reference/donor graph G_{RD}, generate a donor tour of the graph which maximizes the summed lengths of all reference-parallel subsequences.

We note here that this is an extremely large combinatorial problem, and as such a solution is impractical. We therefore formulate a new problem, imposing an

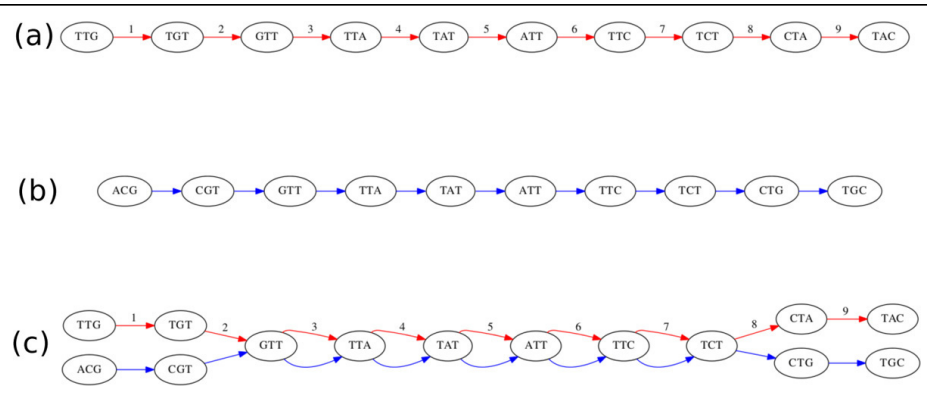

Figure 2 A most basic example of a reference/donor graph, constructed from the superposition of the two original graphs. (a) Reference graph G_R. (b) Donor graph G_D. (c) The graph representing the reference sequence "TTGTTATTCTAC" and donor sequence 'ACGTTATTCTGT'. The donor subsequence "GTTATTCT" is reference-parallel, with reference marker $R[3: 7]$, indicating that it aligns to position 3-7 in the reference sequence.

assumption on the size of any single variation event in the donor genome.

The τ-gap genome reassembly problem: given a reference/donor graph G_{RD}, generate a donor tour of the graph such that for any novel subsequence X of the tour, $|X| < \tau$ and X is bounded by reference-parallel subsequences with reference markers at most τ apart.

Condensed reference/donor graph

In order to reduce the computation and storage demands of the algorithm, we first transform the reference/donor graph G_{RD} into a *condensed* reference/donor graph G_{CRD}. The condensed graph is produced by concatenating linear (non-branching) sequences of donor edges into single edges representing a contig. Rather than storing the reference edges, we store only the reference markers associated with any reference-parallel subsequences in a given contig. Refer to Figure 3 for an example of a condensed graph. Note that while each edge in the reference/donor graph had an implicit edge length of 1, each edge in the condensed graph has a length equal to the length of its contig.

With the condensed reference/donor graph in mind, we may think of a valid traversal as one which touches a sequence of reference markers, one from each edge in the path. For this traversal to be valid, each adjacent pair of reference markers in this sequence must be separated by a distance of at most τ. We can therefore encode the set of valid traversals by maintaining the list of reference markers attached to each edge, and pose the problem as follows.

The τ-gap genome reassembly problem on a condensed reference/donor graph: given a condensed reference/donor graph G_{CRD}, generate a tour of the graph consisting of a sequence of edges $T = \{e_1, e_2, \ldots, e_m\}$, such that for every adjacent pair of edges e_i, e_{i+1}, there exist reference markers $m_i \in markers(e_i)$, $m_j \in markers(e_{i+1})$ with $m_j.start - m_i.end \leq \tau$.

Note, however, that this formulation requires at least one reference marker at *every* edge in the graph. Initially, the graph will likely not satisfy this requirement, as there will be many edges which have no analogue in the reference. This is not an indication that our assumption has been violated, but simply that these edges carry no information. We resolve this problem through a method referred to as *marker propagation*, which aims to update the reference markers at each edge with information from neighboring edges.

Message passing algorithm

Having constructed the reference/donor graph, our aim is to encode within the graph exactly those traversals which satisfy our initial assumptions on τ, the maximum size of any variation event. Specifically, we would like to know for any pair of adjacent edges, whether a path through the two edges is valid. In order to know this, we must know not only what reference markers exist at a given contig, but also the set of markers at contigs reachable along some directed path within a distance of τ. We solve this problem by propagating information along edges in the graph. This process leaves many spurious markers, however, so we then apply a pruning phase to eliminate these. Refer to Figure 4 for an example.

Propagation

As previously described, each edge in the condensed reference/donor graph stores a list of the reference markers associated with any reference-parallel subsequences within that edge's contig. The first phase of the message-passing algorithm propagates this information throughout the graph, such that each edge additionally stores a list of reference markers at edges that are *reachable*

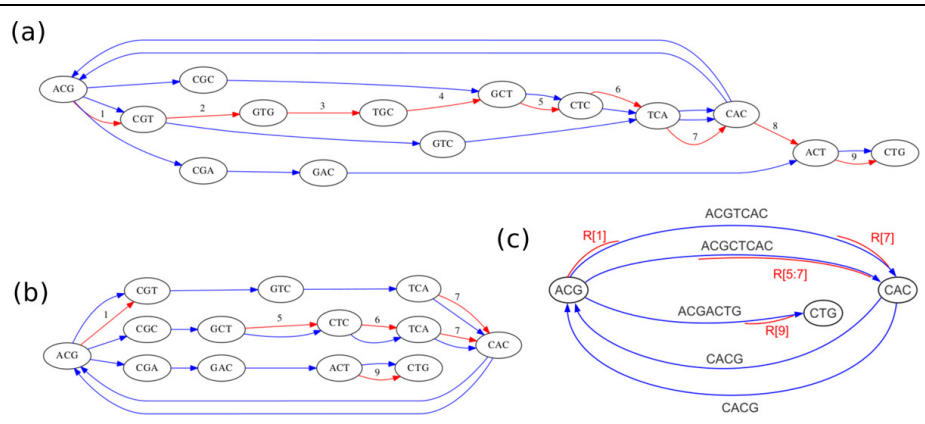

Figure 3 Construction of a condensed reference/donor graph in a more complex case. In the first pass, any reference edges with no parallel donor edge are removed. In the second pass, linear subpaths are condensed to single edges, and the parallel reference edges are summarized using reference markers. (a) Initial reference/donor graph. (b) Intermediate graph with isolated reference edges removed. (c) Final condensed reference/donor graph, with reference markers shown.

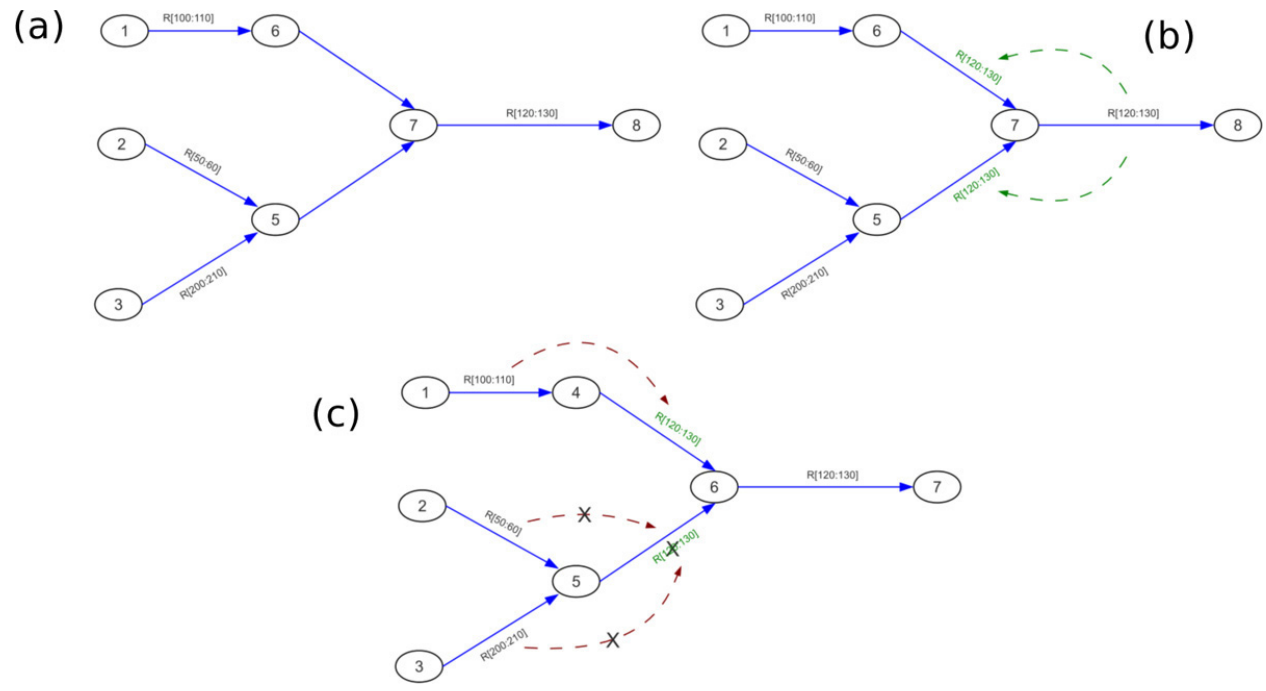

Figure 4 Showing the process of propagation and pruning, assuming τ = 15. All edges shown have an edge length of 10. Initially, we have the graph in (a), in which the edges (6,7) and (5,7) have no reference markers. The marker attached to edge (7,8) is propagated along incoming edges up to a distance of τ, resulting in the graph shown in (b). Note that the marker has been added to edges on both the top and bottom branch. Finally, in the pruning phase, each marker is checked for connectedness. On the top branch, the newly added marker is connected, while on the bottom branch, no adjacent marker connects, so the marker is pruned.

along directed edges in the graph, within a distance of τ. While there are different methods for computing this list, we provide the message-passing formulation here as the most concise.

A message in this propagation phase consists of a set of pairs, each pair (m, d), consisting of a reference marker m and distance d (in terms of total edge length) that the marker has been propagated so far. On receiving a message, an edge e checks each pair in the set, incrementing the distance values by $e.length$. Any pairs with $d > \tau$, or which have already been added to the list, are eliminated, while the rest are stored in the edge's local list and then propagated on as new messages to each incoming neighbor. If no markers from an incoming message are added to the local list, no new messages are generated. The propagation phase ends when there are no messages remaining in the graph. Algorithm 1 demonstrates the message handler for the propagation phase.

Algorithm 1 Propagation_ReceiveMessage(edge, message)

1: $added \leftarrow \emptyset$
2: **for** (m, d) ⌊ message **do**
3: $d \leftarrow d$ + edge.*length*
4: **if** $d < \tau$ **then**
5: edge.*distant_markers* ← edge.*distant_markers* ∪ m
6: $added \leftarrow added \cup (m, d)$
7: **end if**
8: **end for**
9: **if** $added \neq \emptyset$ **then**
10: *Propagation_SendMessage*(*in_neighbors*(*edge*), *added*)
11: **end if**

Pruning

Following the propagation phase, each edge in the graph must have at least one reference marker in its list (if any edge does not, then our assumption on τ has been violated). There will also be many edges which have *excess* reference markers. That is, reference markers which are touched by no valid tour of the graph, yet which may confound or complicate our attempts to generate such tours. We iteratively prune these excess markers through a second message-passing phase.

In the second phase, each edge on receiving a message inspects the markers in its list, categorizing each as either "connected" or " orphaned." A connected marker m is one for which there are associated markers m_{in} and m_{out}, belonging to some incoming and outgoing edge, respectively, such that $m_{in} \xrightarrow{\tau} m \xrightarrow{\tau} m_{out}$. An orphaned marker is any marker that is not connected, and each orphaned marker is removed from the edge's list. Whenever an edge

removes some reference marker from its list, a message is sent to each neighboring edge. The phase begins by sending a message to each edge in the graph, and ends when there are no messages remaining. Algorithm 2 provides a more concrete example of a message handler for this phase.

Algorithm 2 Pruning_ReceiveMessage(edge)

1: *removed* ← ∅
2: **for** $m_e \in$ edge.*markers* **do**
3: *connected* ← **false**
4: **for** *outgoing* ⊏ *outgoing_neighbors* (edge) **do**
5: **for** $m_o \in$ *outgoing.markers* **do**
6: **if** $m_e \xrightarrow{\tau} m_o$ **then**
7: *connected* ← **true**
8: **end if**
9: **end for**
10: **end for**
11: **if** ¬ *connected* **then**
12: edge.*markers* = edge.*markers*/m_e
13: *removed* ← *removed* ∪ m_e
14: **end if**
15: **end for**
16: **if** *removed* ≠ ∅ **then**
17: *Pruning_SendMessage*(*neighbors*(*edge*))
18: **end if**

Merging and iteration

During the pruning phase, we will observe many cases in which there is only one possible path for a traversal to follow. In these cases, we can merge the adjacent edges and recompute their respective markers. Each such merge operation will reduce complexity of the graph, further allowing reference markers to be eliminated. Refer to Figure 5 for an example. Each merge therefore produces a new round of pruning, and this repeats iteratively until nothing more can be done.

Implementation notes

The condensed reference/donor graph is an annotated version of the condensed de-Bruijn graph, and can be easily constructed as such. In recent years, a number of methods have been proposed to construct the condensed de-Bruijn (contig) graph. The simplest method was given in [6] and constructs a de-Bruijn graph using single reads (no paired-end information). This method has since been extended to incorporate paired-end [26], and number of available assemblers implement some variation on this idea [8,9,27]. For our purposes, any method is sufficient, though methods which produce longer contigs are obviously desirable. Methods which utilize the reference in assembly may also be used, provided they generate overlapping contigs which form a graph. In our implementation, we construct the contig graph following the method in [28] using paired-end information, and find local alignments using a custom method based on the read-mapping tool BWA [11]. Other tools such as BLAST [29] may be used to compute the alignments.

Our simulations were performed using a single-threaded implementation running on a 3.2 GHz processor with 16 GB of memory, and demonstrated a worst-case running time of approximately 1 hour. The time complexity of the algorithm is $O(n^3)$ in the worst case, where n is the number of contigs. This is driven by an initial computation of the all-pairs shortest distances, but in general this can be highly optimized as we do not care about distances larger than τ. Each phase of the algorithm can also be parallelized with minimal changes.

Discussion

The goal of any genome sequencing project is to characterize the full genomic content of an individual organism. With the steeply declining cost of genome sequencing in recent years, there has been significant focus on new and improved methods for both *de novo* assembly and resequencing. Despite this focus, however, there have been few methods developed to bridge the gap between these areas. While progress has been made in discovery and assembly of structural variation, the tools remain highly specialized. In this study, we proposed a novel graph construction that concisely represents the similarities between a reference and donor genome, and developed a method using the graph to disambiguate contig ordering. Through simulation, we demonstrated that this method can be effective when working with related bacterial genomes, but significant challenges remain.

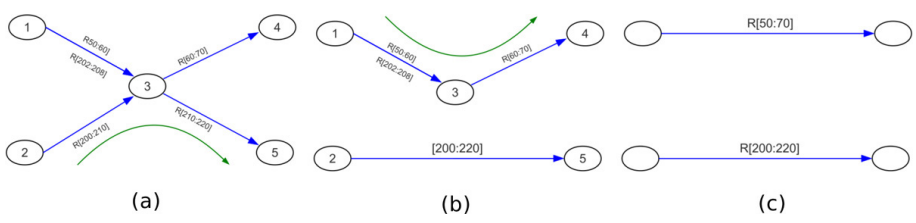

Figure 5 Showing the process of merging edges when there is only one possible path. In (a), edge (1,3) could be followed by either (3,4) or (3,5) based on the reference markers. Edge (2,3) however can only be followed by (3,5). In (b), (2,3) and (3,5) are merged, which makes it possible to eliminate a marker from edge (1,3), and to merge it with (3,4). (c) shows the final state.

One such challenge is that as input to our method we require an estimate of the maximum mutation event size. In practice, this value is not known, and this is currently a significant drawback of our method. It is possible, however, that methods could be developed to estimate this parameter. For example, iterative application of our method with successively larger or smaller values could help discover the true maximum size. Alternatively, the parameter could be estimated directly from the alignment data. Notably, we have also not discussed the application of our method in the presence of read errors. The effect of these data imperfections can be mitigated to an extent by the application of preprocessing methods to correct the errors prior to assembly. Recent studies have shown that read errors can be significantly reduced even in the presence of non-uniform coverage [24,25]. However, further experiments should be performed to validate the effectiveness of our method under less ideal conditions.

Despite these remaining challenges, we believe our method presents a novel approach to the challenge of genome assembly that takes advantage of the increasing availability of reference sequences. It is our hope that this work can help motivate future research into unified reassembly methods.

Acknowledgements
N.P. and E.E. are supported by National Science Foundation grants 0513612, 0731455, 0729049, 0916676 and 1065276, and National Institutes of Health grants K25-HL080079, U01-DA024417, P01-HL30568 and PO1-HL28481. B.S. was supported by NSF grant DMS-1101185, by AFOSR MURI grant FA9550-10-1-0569 and by a USA-Israel BSF grant.

Author details
[1]Department of Computer Science, University of California Los Angeles, Los Angeles, California, USA. [2]Department of Mathematics, University of California Los Angeles, Los Angeles, California, USA.

Authors' contributions
N.P. developed and implemented the algorithm, generated and analyzed results, and authored this manuscript. B.S. and E.E. provided the initial problem framework and algorithmic approach, insights and suggestions on the development of the algorithm, and critiques of the manuscript.

Declarations
The publication costs for this article were funded by the corresponding author's institution.
This article has been published as part of *BMC Genomics* Volume 14 Supplement 1, 2013: Selected articles from the Eleventh Asia Pacific Bioinformatics Conference (APBC 2013): Genomics. The full contents of the supplement are available online at http://www.biomedcentral.com/bmcgenomics/supplements/14/S1.

Competing interests
The authors declare that they have no competing interests.

Published: 21 January 2013

References

1. Goldstein BDavid: **Common genetic variation and human traits.** *The New England Journal of Medicine* 2009, **360(17)**:1696-8.
2. Hardy John, Singleton Andrew: **Genomewide association studies and human disease.** *The New England Journal of Medicine* 2009, **360**:1759-1768.
3. Sharp JAndrew, Hansen Sierra, Selzer RRebecca, Cheng Ze, Regan Regina, Hurst AJane, Stewart Helen, Price MSue, Blair Edward, Hennekam CRaoul, Fitzpatrick ACarrie, Segraves Rick, Richmond ATodd, Guiver Cheryl, Albertson GDonna, Pinkel Daniel, Eis SPeggy, Schwartz Stuart, Knight JLSamantha, Eichler EEvan: **Discovery of previously unidentified genomic disorders from the duplication architecture of the human genome.** *Nature Genetics* 2006, **38(9)**:1038-1042.
4. Korbel OJan, Tirosh-Wagner Tal, Urban Eckehart Alexander, Chen Xiao-Ning, Kasowski Maya, Dai Li, Grubert Fabian, Erdman Chandra, Gao CMichael, Lange Ken, Sobel MEric, Barlow MGillian, Aylsworth SArthur, Carpenter JNancy, Dawn Clark Robin, Cohen YMonika, Doran Eric, Falik-Zaccai Tzipora, Lewin OSusan, Lott TIra, McGillivray CBarbara, Moeschler BJohn, Pettenati JMark, Pueschel MSiegfried, Rao WKathleen, Shaffer GLisa, Shohat Mordechai, Van Riper JAlexander, Warburton Dorothy, Weissman Sherman, Gerstein BMark, Snyder Michael, Korenberg RJulie: **The genetic architecture of Down's syndrome phenotypes revealed by high-resolution analysis of human segmental trisomies.** *Proceedings of the National Academy of Sciences of the United States of America* 2009, **106(29)**:12031-12036.
5. McCarroll ASteven, Huett Alan, Kuballa Petric, Chilewski DShannon, Landry Aimee, Goyette Philippe, Zody CMichael, Hall LJennifer, Brant RSteven, Cho HJudy, Duerr HRichard, Silverberg SMark, Taylor DKent, Rioux DJohn, Altshuler David, Daly JMark, Xavier JRamnik: **Deletion polymorphism upstream of IRGM associated with altered IRGM expression and Crohn's disease.** *Nature Genetics* 2008, **40(9)**:1107-1112.
6. Pevzner PA, Tang H, MS Waterman: **An Eulerian path approach to DNA fragment assembly.** *Proceedings of the National Academy of Sciences of the United States of America* 2001, **98(17)**:9748-53, August.
7. De Bruijn NG: **A combinatorial problem.** *Koninklijke Nederlandse Akademie v Wetenschappen* 1946, **49**:758-764.
8. Butler Jonathan, MacCallum Iain, Kleber Michael, Shlyakhter Allya, Belmonte KMatthew, Lander SEric, Nusbaum Chad, Jaffe BDavid: **ALLPATHS: *de novo* assembly of whole-genome shotgun microreads.** *Genome Research* 2008, **18(5)**:810-820.
9. Zerbino RDaniel, Birney Ewan: **Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs.** *Genome Research* 2008, **18(5)**:821-829.
10. Langmead Ben, Trapnell Cole, Pop Mihai, Salzberg LSteven: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.** *Genome Biology* 2009, **10(3)**:R25.
11. Li Heng, Durbin Richard: **Fast and accurate short read alignment with BurrowsWheeler transform.** *Bioinformatics* 2009, **25(14)**:1754-1760, 9.
12. Hach Faraz, Hormozdiari Fereydoun, Alkan Can, Hormozdiari Farhad, Birol Inanc, Eichler EEvan, Cenk Sahinalp S: **mrsFAST: a cache-oblivious algorithm for short-read mapping.** *Nat Methods* 2010, **7**:576-577.
13. Li Ruiqiang, Li Yingrui, Zheng Hancheng, Luo Ruibang, Zhu Hongmei, Li Qibin, Qian Wubin, Ren Yuanyuan, Tian Geng, Li Jinxiang, Zhou Guangyu, Zhu Xuan, Wu Honglong, Qin Junjie, Jin Xin, Li Dongfang, Cao Hongzhi, Hu Xueda, Blanche Helene, Cann Howard, Zhang Xiuqing, Li Songgang, Bolund Lars, Kristiansen Karsten, Yang Huanming, Wang Jun, Wang Jian: **Building the sequence map of the human pan-genome.** *Nat Biotechnol* 2010, **28**:57-63, advance on.
14. Lee Seunghak, Cheran Elango, Brudno Michael: **A robust framework for detecting structural variations in a genome.** *Bioinformatics* 2008, **24(13)**: i59-i67.
15. Chen Ken, Wallis WJohn, Mclellan DMichael, Larson EDavid, Kalicki MJoelle, Pohl SCraig, Mcgrath DSean, Wendl CMichael, Zhang Qunyuan, Locke PDevin, Shi Xiaoqi, Fulton SRobert, Ley JTimothy, Wilson KRichard, Ding Li, Mardis RElaine: **BreakDancer: an algorithm for high-resolution mapping of genomic structural variation.** *Nature Methods* 2009, **6(9)**:677-681.
16. Hormozdiari Fereydoun, Hajirasouliha Iman, Dao Phuong, Hach Faraz, Yorukoglu Deniz, Alkan Can, Eichler EEvan, Cenk Sahinalp S: **Next-generation VariationHunter: combinatorial algorithms for transposon insertion discovery.** *Bioinformatics* 2010, **26(12)**:i350-7, June.
17. Alkan Can, Kidd MJeffrey, Marques-Bonet Tomas, Aksay Gozde, Antonacci Francesca, Hormozdiari Fereydoun, Kitzman OJacob, Baker Carl, Malig Maika, Mutlu Onur, Cenk Sahinalp S, Gibbs ARichard, Eichler EEvan: **Personalized copy number and segmental duplication maps using next-generation sequencing.** *Nature Genetics* 2009, **41(10)**:1061-1067.
18. Parrish Nathaniel, Hormozdiari Farhad, Eskin Eleazar: **Assembly of non-unique insertion content using next-generation sequencing.** *BMC Bioinformatics* 2011, **12(Suppl 6)**:S3.

19. Hajirasouliha Iman, Hormozdiari Fereydoun, Alkan Can, Kidd MJeffrey, Birol Inanc, Eichler EEvan, Cenk Sahinalp S: **Detection and characterization of novel sequence insertions using paired-end next-generation sequencing.** *Bioinformatics* 2010, **26(10)**:1277.
20. Richter CDaniel, Schuster CStephan, Huson HDaniel: **OSLay: optimal syntenic layout of unfinished assemblies.** *Bioinformatics* 2007, **23(13)**:1573-1579, July.
21. Husemann Peter, Stoye Jens: **Phylogenetic comparative assembly.** *Algorithms for Molecular Biology* 2010, **5(1)**:3.
22. Zhao Fangqing, Zhao Fanggeng, Li Tao, Bryant ADonald: **A new pheromone trail-based genetic algorithm for comparative genome assembly.** *Nucleic acids research* 2008, **36(10)**:3455-62, June.
23. Song SYun: ***De novo* error-correction algorithms for short-read sequencing.** *Statistics* 2010.
24. Medvedev Paul, Scott Eric, Kakaradov Boyko, Pevzner Pavel: **Error correction of high-throughput sequencing datasets with non-uniform coverage.** *Bioinformatics* 2011, **27(13)**:i137-i141.
25. Yang Xiao: **Error correction and clustering algorithms for next generation sequencing.** *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum* 2011, 2101-2104.
26. Pevzner PA, Tang H: **Fragment assembly with double-barreled data.** *Bioinformatics* 2001, **17(Suppl 1)**:S225-S233, Suppl 1.
27. Medvedev Paul, Brudno Michael: ***Ab initio* whole genome shotgun assembly with mated short reads.** *Methods* 2008, 1-11.
28. Medvedev Paul, Pham Son, Chaisson Mark, Tesler Glenn, Pevzner Pavel: **Paired de Bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers.** *Research in Computational Molecular Biology* Springer; 2011, 238-251.
29. R F, Zhang Jinghui, Zhang Zheng, Miller Webb, Lipman JDavid: **BLAST Basic Local Alignment Search Tool.** *Distribution* 2008, **215(3)**:4-5.

doi:10.1186/1471-2164-14-S1-S8
Cite this article as: Parrish *et al.*: Genome reassembly with high-throughput sequencing data. *BMC Genomics* 2013 **14**(Suppl 1):S8.

Submit your next manuscript to BioMed Central and take full advantage of:

- **Convenient online submission**
- **Thorough peer review**
- **No space constraints or color figure charges**
- **Immediate publication on acceptance**
- **Inclusion in PubMed, CAS, Scopus and Google Scholar**
- **Research which is freely available for redistribution**

Submit your manuscript at
www.biomedcentral.com/submit

BioMed Central