Scaffolding Large Genomes using Integer Linear Programming

James Lindsay
University of Connecticut
371 Fairfield Road, Unit 2155
Storrs, CT 06269
james.lindsay@engr.uconn.edu

Alex Zelikovsky
Georgia State University
34 Peachtree Street
Atlanta, GA 30303
alexz@cs.gsu.edu

Hamed Salooti
Georgia State University
34 Peachtree Street
Atlanta, GA 30303
hsalooti@cs.gsu.edu

Ion Măndoiu
University of Connecticut
371 Fairfield Road, Unit 2155
Storrs, CT 06269
ion@engr.uconn.edu

ABSTRACT

The rapidly diminishing cost of genome sequencing is driving renewed interest in large scale genome sequencing programs such as *Genome 10K* (G10K). Despite renewed interest the assembly of large genomes from short reads is still an extremely resource intensive process. This work presents a scalable algorithms to create scaffolds, or ordered and oriented sets of assembled contigs, which is one part of a practical assembly. This is accomplished using integer linear programming (ILP). In order to process large mammalian genomes we employ non-serial dynamic programming (NSDP) and a hierarchical strategy. Both existing and novel quantitative metrics are used to compare scaffolding tools and gain deeper insight into the challenges of scaffolding. The code is available at: https://bitbucket.org/jrl03001/silp

1. INTRODUCTION

Since the completion of the human genome, rapid advances in high-throughput sequencing (HTS) technologies have resulted in orders of magnitude higher throughput and lower cost compared to classic Sanger sequencing [20]. The precipitous drop in sequencing costs has generated much enthusiasm for very large scale genome sequencing initiatives, such as the *Genome 10K* (G10K) project, that calls for sequencing over 10,000 of the approximately 60,000 extant vertebrate species. Genome sequencing on such a scale would be unthinkable using Sanger sequencing, at a cost of tens of millions of dollars per genome. It would also be prohibitively expensive using HTS technologies that deliver reads of length comparable to that of Sanger reads (Roche's 454 and Pacific Biosciences' single molecule real time sequencer), which have sequencing reagent costs of tens to hundreds of dollars per megabase [6]. In contrast, Illumina's HiSeq 2000 and Life Technologies' SOLiD 5500xl can both deliver 100× sequencing coverage of a typical vertebrate genome in a single run, with a total reagent cost of a few tens of thousands of dollars. However, assembling high-quality genome sequences from the short reads (currently around 100bp) generated by these technologies represents a formidable computational challenge that has yet to be met (see [3, 17, 21, 23, 24, 29] for recent reviews and benchmarking results).

While feasibility of *de novo* draft assembly entirely from short HTS reads has been demonstrated for several vertebrate genomes, including human (using both ABySS [32] and SOAPdenovo [14]), chicken (using SOAPdenovo [34]) and Panda (using SOAPdenovo [16]), short read assemblies are much more fragmented compared to high-quality draft Sanger assemblies. For example, the dog genome assembly [18], generated from 7.5× coverage by Sanger reads, has an N50 contig length of 180Kb. In contrast, the ABySS assembly of the human genome in [32] has N50 contig length of 1.5Kb. SOAPdenovo assemblies have somehow longer contigs – but still much shorter than those of Sanger assemblies – with reported N50 contig lengths of 5.9Kb respectively 7.4Kb for the African and Asian human assemblies in [14], 12Kb for the Illumina-based chicken genome in [34], and 36.7Kb for the Panda genome [16]. By using 100bp Illumina paired-end reads with a mix of tightly controlled insert lengths and total coverage depth of over 100×, ALLPATHS-LG [7] obtained assemblies with N50 contig lengths of 24Kb for human and 16Kb for mouse.

Short contig lengths, typically between 2-4Kb, are also the characteristic of tens of draft genome assemblies generated from low-coverage (2×) Sanger reads over the past decade, e.g., as part of the Mammalian Genome Project [1]. To increase the utility of such fragmented assemblies, additional long-range linkage information is used to orient contigs relative to one another and order them in larger structures referred to as *scaffolds*. For conventional Sanger assemblies, long-range linkage information is obtained by sequencing both ends of clones of up to hundreds of kilobases. HTS

platforms can also generate paired reads, albeit with lower insert length – commonly limited to a few Kb due to the fact that current library preparation protocols involve a DNA circularization step whose efficiency decreases for long inserts. Linkage information provided by HTS pairs is also much less reliable than that provided by clone-based Sanger protocols due both to chimeric pairs resulting from library preparation artifacts and erroneous mapping of reads originating from repeats. These difficulties, along with the sheer number of HTS pairs and contigs that must be handled, render scaffolding methods developed for Sanger pairs such as [10, 25] ineffective on HTS data. While recent algorithmic advances [2, 5, 27] have led to improved scaffolding accuracy from such "noisy" HTS paired reads, scaling these methods to datasets consisting of hundreds of thousands to millions of contigs and hundreds of millions of read pairs, as expected for a vertebrate genome, remains a significant challenge.

Since the scaffolding problem is known to be NP-hard problem [10], most practical methods such as [10, 25] adopt greedy heuristics. SOPRA [2]and MIP [27] both use a partitioning scheme to reduce the size of the problem. SLIQ [26] employs inequalities derived from the geometry of contigs on the line predicting the orientation and ordering of adjacent contigs. In contrast to these methods, Opera [5] uses a dynamic programming algorithm for finding a feasible scaffold with the minimum number of inconsistent pairs of contigs for HTS pairs.

In this paper we develop a highly scalable method for accurate scaffolding of large genomes. Our method uses integer linear programming (ILP) to find orientations and linear orderings of contigs most consistent with the available paired reads. However, solving the scaffolding ILP directly for very large problem instances is often impractical. To achieve scalability we take advantage of the sparsity of the underlying scaffolding graph by independently scaffolding its tri-connected components, generated in linear time using the SPQR-tree datastructure, and then optimally combining them using non-serial dynamical programming (NSDP) [30]. For cases when even tri-connected components are too large to be handled directly we adopt a hierarchical scaffolding scheme that solves multiple ILPs for increasingly denser subgraphs of G obtained by filtering low confidence edges. Experimental results show that our method scales further than previous methods with similar accuracy.

2. METHODS

This work solves the scaffolding problem by first finding the best orientation and pairwise ordering of the contigs using an ILP. Next the scaffolding path is extracted using bipartite matching. Finally the algorithm is made practical by utilizing NSDP, and if additional scalability is required a hierarchical strategy is employed which uses high confidence information first. The flow of the complete scaffolding pipeline is as follows;

1. map reads to contigs
2. filter mapped pairs
3. find orientation and pairwise order using ILP
4. find scaffold path using bipartite matching

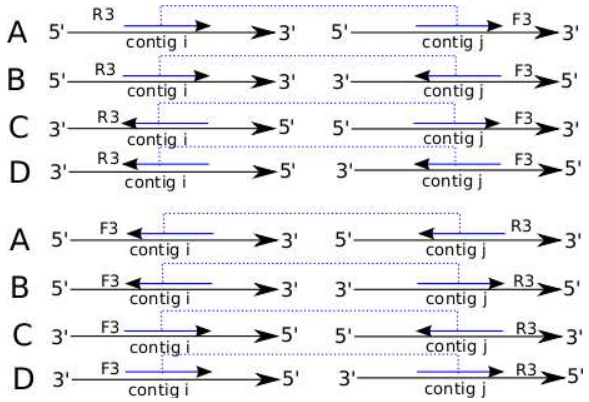

Figure 1: All possible states for a read pair.

2.1 Integer Linear Program

The scaffolding ILP model is best described using the scaffolding graph $G = (V, E)$, where vertices represent contigs and edges are derived from linkage information provided by paired HTS reads. Paired sequencing libraries are constructed such that the order and orientation of the reads relative to their originating fragment is known. For this work we will use the mate-pair style of reads where the pairs come from the same strand and are in the same orientation. In a pair, each read can be mapped on the 5' or 3' strand of each contig. Therefore, there are 4 possible orientations of two contigs i, j connected by a paired read. There is another 4 possible states if the first read is mapped to contig j. All possible combinations are described in Figure1.

We define several boolean variables to assist in creating the model. First $S_i = \{0, 1\}\ \forall i \in N$ is used to indicate the orientation of each contig; $S_i = 0$ indicates the contig orientation does not change. Then $S_{ij} = \{0, 1\}\ \forall (i, j) \in E$ tells if the contigs have the same orientation ($S_{ij} = 0$) or if one is flipped ($S_{ij} = 1$). Four state variables, $A_{ij}, B_{ij}, C_{ij}, D_{ij} = \{0, 1\}\ \forall (i, j) \in E$ represent the 4 possible orders and orientations of adjacent contigs; they are mutually exclusive. Each state has an associated weight $A^w_{ij}, B^w_{ij}, C^w_{ij}, D^w_{ij}$ which are the sum of the weight of corresponding pairs. The objective is to maximize the number of concordant edges.

$$\text{Max} \sum_{(i,j)\in E} A^w_{ij} \cdot A_{ij} + B^w_{ij} \cdot B_{ij} + C^w_{ij} \cdot C_{ij} + D^w_{ij} \cdot D_{ij}$$

The following constraints enforce the behavior of the orientation variables.

$$S_{ij} \leq S_i + S_j \qquad S_{ij} \leq 2 - S_i - S_j$$
$$S_{ij} \geq S_j - S_i \qquad S_{ij} \geq S_i - S_j$$

Using the ILP framework allows us to explicitly forbid certain scenarios which are known to be impossible. Since eukaryotic genomes are linear, then cycles in the scaffolding graph should be impossible. Therefore we have chosen to explicitly forbid all two cycles

$$B_{ij} + C_{ij} \leq S_{ij} \qquad A_{ij} + D_{ij} \leq 1 - S_{ij}$$

An additional 8 constraints forbid three cycles in all in-

stances where 3 contigs are connected.

$$A_{ij} + A_{jk} + D_{ki} \leq 2 \qquad A_{ij} + B_{jk} + C_{ki} \leq 2$$
$$B_{ij} + C_{jk} + D_{ki} \leq 2 \qquad B_{ij} + D_{jk} + C_{ki} \leq 2$$
$$C_{ij} + A_{jk} + B_{ki} \leq 2 \qquad C_{ij} + B_{jk} + A_{ki} \leq 2$$
$$D_{ij} + C_{jk} + B_{ki} \leq 2 \qquad D_{ij} + D_{jk} + A_{ki} \leq 2$$

It would be impractical to enumerate all larger cycles, if we find a cyclic solution then the cycle is broken heuristically. The solution to this ILP gives the orientation of every contig, and the pairwise ordering of compatible connected contigs. It is still necessary to find the linear path that corresponds to the scaffold.

2.1.1 Weights, Filters and Bundles

Only pairs for which both reads mapped uniquely to the genome are used for scaffolding. However these unique reads may map to regions annotated by external tools or multiple mapped reads as repetitive. A large amount of overlap between a unique mapped read and a repetitive region corresponds to a smaller probability p_k. The probability that a single read p_k^1 is correct is 1-(# bases overlap with repeat)/(# bases in read). Since each edge in the scaffolding graph consists of two mapped reads, the probability the pair is correct is $p_k = p_k^1 * p_k^2$. Each pair of adjacent contigs can have support for each of the four possible states and these weights are denoted as $A_{ij}^w, B_{ij}^w, C_{ij}^w, D_{ij}^w$.

Some linkage information can be thrown out before the scaffolding process begins because it violates logical constraints or has very little support. We can place a bound on the expected distance between two contigs given any state. If a read pair spans two contigs and the implied minimum distance of any supported state is larger than the insert size plus 3 standard deviations then the pair is ignored. Also if 90% of any read in a pair overlaps with a repeat annotated region then that read pair is ignored because it may be unreliable. If there are several paired reads between two contigs supporting the same state and similar distance between contigs, then these reads are collapsed in a single *bundle*. We follow a common practice of filtering out connections between contigs supported only by bundles of size less than a threshold B (see e.g., [2, 5]). Finally for any two connected contigs, if multiple states are supported and the one state is not weighted at least two times more than the next highest that pair is ignored.

2.2 Bipartite Matching

The solution to the ILP does not yield a scaffold, which is a linear graph. The ILP instead only gives the pairwise order and orientation of the contigs. To find the scaffold path we use bipartite matching. A bipartite graph $B = (V^1 \cup V^2, E)$ is defined where each vertex in V^1 corresponds to the 5' end of a contig, and each vertex in V^2 to the 3' end of a contig. Edges are defined by the solution to the ILP. Each edge is weighted according to the weight of the chosen state from the ILP. Matching in bipartite graphs has been extensively studied under several objects including maximum cardinality, maximum weight or a combination of the two [4]. All three objectives were tested and it was found that maximum weight yielded the best results. A maximum weight bipartite matching algorithm from [4] was used to find the scaffold path.

2.3 Non-serial Dynamic Programming

It is impractical to solve a scaffold ILP for a large mammalian genome as the number of variables and constraints is simply too large. To overcome this hurdle and solve the problem optimally we adopted the non-serial dynamic programming (NSDP) paradigm. This optimization technique exploits the sparsity of the scaffolding graph, which should be a bounded-width graph [5]. The solution is computed in stages, each stage using the results of a previous one to efficiently solve the problem. Please note that a completely independent component will have no influence on other components. Furthermore it was pointed out in [27] that biconnected components can be solved independently since the orientation of a component can be completely flipped if the orientation of the articulation contig is not the same in adjacent biconnected components. Therefore we assume that the scaffolding graph is biconnected.

A key concept in NSDP is the notion of a dependency graph which models the relationship between variables and constraints. An dependency graph contains a vertex for each variable and an edge is added between vertices if they appear in the same constraint or component of the objective function. NSDP is a process which eliminates variables in such a way that adjacent variables can be merged together [30]. The first step in applying NSDP is identifying weakly connected components of interaction graph. Specifically we wish to identify tri-connected components, each component will interact with another component with exactly two variables. We use efficient algorithms to find the tri-connected components of our dependency graph. Then an elimination order must be found so that each component can be solved independently in such a way that the solutions can be merged to find the global solution. The decomposition order for tri-connected components is given by the SPQR-tree [9] data structure.

The solution to each component of the interaction graph is found using a bottom up traversal. In general during the traversal the ILP for each component is solved 4 times. Once for each possible orientation of the common nodes. The objective value of each case is encoded in the objective of the components parent. After solving all components, top down DFS starting from the same root is performed to apply the chosen solution for each component. This concept has been previously applied in the area of layout decomposition in [19]. A sketch of the bottom up procedure for the tri-connected components is detailed in algorithm 1. In this algorithm the *skeleton*() function returns the set of nodes corresponding to that particular tri-connected component. The weights of each solution $sol_{00}, sol_{10}, sol_{01}, sol_{11}$ are used to modify the objective of the parent component by adding the following,

$$sol_{00} + \frac{sol_{11} + sol_{10} - sol_{01} - sol_{00}}{2} S_i \quad +$$
$$\frac{sol_{11} - sol_{10} + sol_{01} - sol_{00}}{2} \cdot S_j \quad +$$
$$\frac{-sol_{11} + sol_{10} + sol_{01} - sol_{00}}{2} \cdot S_{ij}$$

Algorithm 1 tri-component NSDP

```
stack = [root of SPQR-tree]
visited = empty
while stack is not empty do
  p = stack.pop()
  for each child q of p do
    if p not in visited then
      stack.push(p)
    end if
  end for
  if p is root then
    sol_final = SOLVE(p.skeleton())
  else
    (s, t) = getcut(p, parent(p))
    sol_00 = SOLVE(p.skeleton(), s = 0, t = 0)
    sol_01 = SOLVE(p.skeleton(), s = 0, t = 1)
    sol_10 = SOLVE(p.skeleton(), s = 1, t = 0)
    sol_11 = SOLVE(p.skeleton(), s = 1, t = 1)
    weight (s, t) in parent(p) appropriately
  end if
end while
```

2.4 Hierarchical Approach

It was observed that the number of paired edges p, that span contigs has a large effect on accuracy and scalability of the scaffolding program. In our work we attempted to solve the scaffolding problem without setting this parameter explicitly but in complex genomes it was observed that occasionally the ILP would be too large even after the complete decomposition procedure. In order to address this, we developed the following hierarchical approach to solving the scaffolding problem.

The problem is first solved with edges corresponding to higher bundle size $B+1$, then the resulted scaffold chains are fixed, i.e., replaced with pseudo-contigs, and the modified problem is solved for lower bundle size B (see Figure 2). After each iteration, utilized pairs are removed from consideration, and any edge in the scaffolding graph that is not compatible with a previous stage scaffold is removed. If any added pair has both ends on the scaffold and satisfies the orientation constraints of the sub-contigs, it is merged into the pseudo-contig, otherwise, is removed. If the added edge has just one end on the pseudo-contig and satisfies the insert size constraints, it is applied into the pseudo-contig, otherwise, is removed. Obviously using pseudo-contigs for smaller bundle sizes greatly reduces the size of the scaffolding problem.

There is another important advantage of the hierarchical approach. Note that edges and scaffolds with higher bundle size are much more likely to be correct than with the smaller size. Indeed, from table 3 we can see that decrease in bundle size reduces positive predicted value (PPV). On the other hand, higher bundle sizes disregard more information resulting in reduction of sensitivity. Therefore, the hierarchical approach gradually improves sensitivity by trading PPV and does not lose correct edges found with higher bundle sizes.

3. RESULTS

In order to asses the quality of our scaffolding tool we developed a testing framework which closely mimics a real world scaffolding problem. We utilized the Staphylococcus aureus and Rhodobacter sphaeroides genomes from the GAGE [28] assembly comparison to test the accuracy of ours and other leading scaffolding tools. These two genomes were assembled from 90x coverage, 50-100 bp reads from the Illumina platform. Each genome was assembled by the following de novo assembly tools: Velvet [35], Allpaths-LG [8], SOAPdenovo [15], MSR-CA, Bambus2 [11], SGA [31], ABySS [33], ABySS2. We also created a larger test case which was derived from a de novo draft assembly of an individual human [12]. The set of Sanger style reads used to create this finished version of the draft genome were sub sampled to 4x base coverage. A total of 11200000 reads were placed into the Celera 6.1 [22] assembler and assembled using the recommended parameters for large mammalian genomes. The assembler generated 422837 contigs with an N50 of 7704bp. In both the GAGE and human test sets, the contigs were mapped against the finished reference version of the genome using BWA-SW [13], only uniquely aligned, non-overlapping contigs with 90% identity were utilized. From this alignment a reference scaffold was created to serve as the test cases.

The accuracy and scalability of our ILP based methods (SILP, SILP-H) are compared against three of the leading external scaffolding tools, MIP [27], OPERA [5], and BAMBUS2 [11]. Unfortunately BAMBUS2 was not run on the Venter genome due to time constraints. Additionally OPERA was run at a bundle size of 7 in the GAGE datasets after it failed to complete on several datasets after an hour. The other methods were run with bundle size 2. A scaffolded genome can be thought as a linear directed graph and the act of scaffolding is the attempt to predict directed edges between adjacent contigs. We treat scaffolding as a binary classification test where methods attempt to predict true adjacencies in the test dataset. The accuracy and sensitivity can be directly measured according to table 1. While these measures are natural to a computer scientist they are not as useful to a biologist because the content of the contigs is ignored. A biologist typically asses a scaffold by the N50, or weighted median statistic such that 50% of the entire assembly is contained in scaffolds equal to or larger than this value. Unfortunately this measure does not reflect the accuracy of the scaffolds and rewards aggressive merging. Therefore we utilize the true positive N50, which we call TP50, where scaffolds are broken when an adjacency was falsely predicted, a similar measure was used in [28]. Additionally we also present the percentage of genome contained within correct scaffolds, excluding gaps.

3.1 Accuracy

The results of the comparison experiment are detailed in tables 2, 3. It was observed in the Staphylococcus and Venter genomes that SILP incorporates a larger percentage of the genome into the corrected scaffolds. SILP also has a

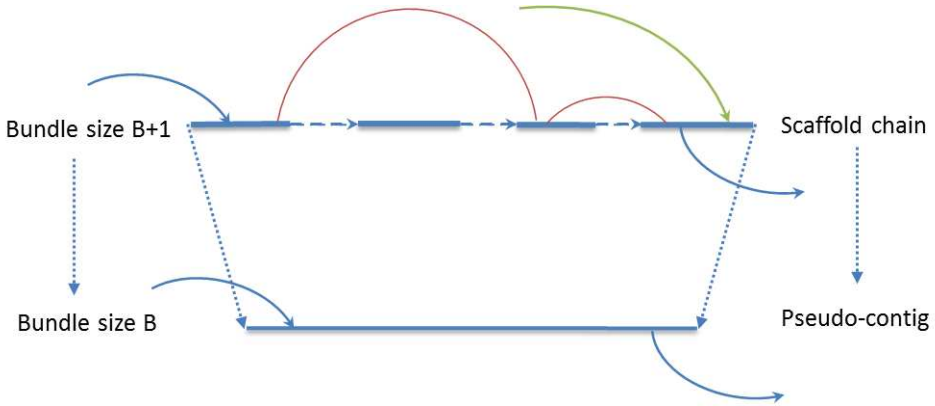

Figure 2: The scaffold obtained from solving problem with bundle size B+1 and the corresponding collapsed pseudo-contig for the problem with bundle size B : Paired reads with both ends in the scaffold (in red) are dropped . Paired reads with a single end in the scaffold are removed if the insert size constraints are not satisfied (green), otherwise paired reads are kept for bundle size B (blue).

Table 1: Scaffolding Contingency Table

	true adjacent	false adjacent
predicted adjacent	TP	FP
predicted non-adjacent	FN	TN

higher TP50 than the other methods except for BAMBUS2 on the Rhodobacter dataset. SILP has the highest sensitivity and the second highest PPV of all the methods. OPERA has a higher PPV in all 3 genomes, however because of its lower sensitivity it does not translate into a higher TP50 or percentage of genome. This could be due to the fact that OPERA have a active read filtering step that does not rely on thresholds to identify incorrect paired reads.
Interestingly there is a large difference in the PPV and sensitivity between the GAGE and the Venter assemblies. In the Venter genome we see a much higher degree of accuracy when compared to the GAGE genomes. This difference is reflected in all the methods tested. The GAGE genomes were assembled using short, high coverage, ultra high-throughput platforms while the Venter genome was assembled using low coverage, long, Sanger style sequencing.

3.2 Scalability

The main objective of this work is to produce a scaffolding tool that is capable of accurately scaffolding large mammalian genomes. In the end we were able to produce a method SILP-H that was able to scaffold the draft assembly of the human genome with no bundle size filter. Both OPERA and MIP failed to produce any results for small bundles sizes given 120 hours. We chose 120 hours, or 5 days as our threshold because we believe it is sufficient amount of time for each tool. Longer times would represent an unreasonable amount of time to wait for an scaffold given that the assembly took approximately 5 days. Even though the accuracy measures indicate that SILP-H was no more accurate than OPERA, the scalability of our method should prove valuable to biologists.

The runtime of SILP(-h) on the smaller GAGE genomes indicate that it is competitive with all of the methods tested on small genomes. We speculate that SILP(-H) runtime suffers on the large genomes due to the greater reliance on the decomposition code which has not yet been optimized for performance. Also, it should be noted that the scalability of OPERA is dependent on the presence *border* contigs which are longer than the insert size of the paired read libraries used. In the Venter genome test case OPERA runs very quickly because the contigs are larger than the insert size. However on the more fragmented assemblies in the GAGE genomes that is not the case.
The bundle size parameter is universal to all three methods. It was observed to has a large effect on the sensitivity of each method, as seen in table 3. Reducing the minimum accepted bundle size will increase the size of the problem but allow for larger correct scaffolds. Since SILP-H was able to scaffold the test dataset using no threshold on the bundle size we feel it represents to most scalable and accurate scaffolding solution currently available.

4. CONCLUSIONS

Scaffolding in an important step in the de novo assembly pipeline. Biologists rely on an accurate scaffold to perform many types of analysis. The larger the scaffold the more useful it will be to them. Recent advances in de novo assemblers has made it feasible to create draft assemblies for large mammalian genomes. We believe the SILP, coupled with the most recent scalable assemblers will produce the largest and most complete assemblies. This is made possible utilizing our robust ILP, Non-Serial Dynamic Programming and our hierarchical heuristic.

5. ACKNOWLEDGMENTS

This work has been partially supported by Agriculture and Food Research Initiative Competitive Grant no. 201167016-30331 from the USDA National Institute of Food and Agriculture and NSF awards IIS-0916401 and IIS-0916948.

6. REFERENCES

[1] Mammalian Genome Project, `http://www.broadinstitute.org/`

Table 2: GAGE Results Table

genome	method	runtime (s)	sensitivity (%)	PPV (%)	N50 (bp)	TP50 (bp)	% genome
staph	SILP	0.37	51.76	61.30	156,297	103,784	69.96
staph	BAMBUS2	0.49	45.35	53.53	145,928	66,455	42.94
staph	OPERA	39.86	32.34	65.90	49,982	51,163	40.42
staph	MIP	58.49	7.51	50.42	45,663	44,002	11.18
rhodo	SILP	1.32	33.94	58.46	65,898	57,438	42.53
rhodo	BAMBUS2	1.35	38.47	48.96	114,032	57,775	46.55
rhodo	OPERA	84.21	27.10	66.58	44,480	42,761	39.36
rhodo	MIP	18.71	4.91	50.75	40,213	39,431	10.12

Table 3: Venter Results Table

method	bundle size	runtime (s)	sensitivity (%)	PPV (%)	N50 (bp)	TP50 (bp)	% genome
SILP	5	51,538	76.27	97.96	24,780	24,370	94.41
SILP	4	59,553	77.64	97.79	25,562	25,060	95.35
SILP	3	18,657	79.2	97.51	26,515	25,982	96.38
SILP-H	3	18,657	79.2	97.51	26,515	25,982	81.39
SILP-H	2	26,663	79.41	97.1	26,636	26,019	81.46
SILP-H	1	31,266	79.49	93.68	27,216	26,028	81.48
OPERA	5	625	76.21	98.99	24,532	24,342	85.62
OPERA	4	671	77.57	98.85	25,288	25,044	86.53
OPERA	3	618	79.03	98.47	26,275	25,902	87.50
MIP	5	2020	60.04	98.37	19,375	19,186	83.35

`scientific-community/science/projects/mammals-models/mammalian-genome-project`.

[2] A. Dayarian, T. Michael, and A. Sengupta. SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11(1):345+, 2010.

[3] P. Flicek and E. Birney. Sense from sequence reads: methods for alignment and assembly. *Nature Methods*, 6(11s):S6–S12, 2009.

[4] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.*, 18(1):23–38, Mar. 1986.

[5] S. Gao, N. Nagarajan, and W.-K. Sung. Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. In *Proc. 15th Annual international conference on Research in computational molecular biology*, pages 437–451, 2011.

[6] T. C. Glenn. Field guide to next-generation DNA sequencers. *Molecular Ecology Resources*, 2011.

[7] S. Gnerre, I. MacCallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, G. Hall, T. P. Shea, S. Sykes, A. M. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. S. Lander, and D. B. Jaffe. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.

[8] S. Gnerre, I. MacCallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, G. Hall, T. P. Shea, S. Sykes, A. M. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. S. Lander, and D. B. Jaffe. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.

[9] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.

[10] D. H. Huson, K. Reinert, and E. W. Myers. The greedy path-merging algorithm for contig scaffolding. *J. ACM*, 49(5):603–615, 2002.

[11] S. Koren, T. J. Treangen, and M. Pop. Bambus 2: Scaffolding metagenomes. *Bioinformatics*, 2011.

[12] S. Levy *et al.* The diploid genome sequence of an individual human. *PLoS Biology*, 5(10):e254+, 2007.

[13] H. Li and R. Durbin. Fast and accurate long-read alignment with burrowsâĂŞwheeler transform. *Bioinformatics*, 26(5):589–595, 2010.

[14] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, 2009.

[15] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, 2010.

[16] R. Li et al. The sequence and de novo assembly of the giant panda genome. *Nature*, 463(7279):311–317, 2010.

[17] Y. Lin, J. Li, H. Shen, L. Zhang, C. J. Papasian, and H.-W. Deng. Comparative Studies of de novo Assembly Tools for Next-generation Sequencing Technologies. *Bioinformatics*, 2011.

[18] K. Lindblad-Toh et al. Genome sequence, comparative analysis and haplotype structure of the domestic dog. *Nature*, 438(7069):803–819, 2005.

[19] W.-S. Luk and H. Huang. Fast and lossless graph division method for layout decomposition using spqr-tree. In *Computer-Aided Design (ICCAD), 2010*

IEEE/ACM International Conference on, pages 112 –115, nov. 2010.

[20] M. L. Metzker. Sequencing technologies - the next generation. *Nature reviews. Genetics*, 11(1):31–46, 2010.

[21] J. R. Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, 2010.

[22] E. W. Myers, G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H. H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G. M. Rubin, M. D. Adams, and J. C. Venter. A whole-genome assembly of Drosophila. *Science (New York, N.Y.)*, 287(5461):2196–2204, Mar. 2000.

[23] K. H. Paszkiewicz and D. J. Studholme. *De novo* assembly of short sequence reads. *Briefings in Bioinformatics*, 11(5):457–472, 2010.

[24] M. Pop. Genome assembly reborn: recent computational challenges. *Briefings in Bioinformatics*, 10(4):354–366, 2009.

[25] M. Pop, D. S. Kosack, and S. L. Salzberg. Hierarchical scaffolding with Bambus. *Genome research*, 14(1):149–159, 2004.

[26] R. S. Roy, K. C. Chen, A. M. Segupta, and A. Schliep. Sliq: Simple linear inequalities for efficient contig scaffolding. *arXiv:1111.1426v2[q-bio.GN]*, Nov. 2011.

[27] L. Salmela, V. Mäkinen, N. Välimäki, J. Ylinen, and E. Ukkonen. Fast scaffolding with small independent mixed integer programs. *Bioinformatics*, 27(23):3259–3265, Dec. 2011.

[28] S. L. Salzberg, A. M. Phillippy, A. Zimin, D. Puiu, T. Magoc, S. Koren, T. J. Treangen, M. C. Schatz, A. L. Delcher, M. Roberts, G. MarÃğais, M. Pop, and J. A. Yorke. Gage: A critical evaluation of genome assemblies and assembly algorithms. *Genome Research*, 22(3):557–567, 2012.

[29] M. C. Schatz, A. L. Delcher, and S. L. Salzberg. Assembly of large genomes using second-generation sequencing. *Genome Research*, 20(9):1165–1173, 2010.

[30] O. Shcherbina. Nonserial dynamic programming and tree decomposition in discrete optimization. In *OR*, pages 155–160, 2006.

[31] J. T. Simpson and R. Durbin. Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*, 22(3):549–556, 2012.

[32] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and I. Birol. ABySS: a parallel assembler for short read sequence data. *Genome research*, 19(6):1117–1123, 2009.

[33] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and Ä. Birol. Abyss: A parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009.

[34] L. Ye, L. Hillier, P. Minx, N. Thane, D. Locke, J. Martin, L. Chen, M. Mitreva, J. Miller, K. Haub, D. Dooling, E. Mardis, R. Wilson, G. Weinstock, and W. Warren. A vertebrate case study of the quality of assemblies derived from next-generation sequences. *Genome Biology*, 12(3):R31, 2011.

[35] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, 18(5):821–9, 2008.