Efficient Indexed Alignment of Contigs to Optical Maps

Martin D. Muggli[1], Simon J. Puglisi[2], and Christina Boucher[1]

[1] Department of Computer Science,
Colorado State University, Fort Collins, CO
{muggli,cboucher}@cs.colostate.edu
[2] Department of Computer Science,
University of Helsinki, Finland
puglisi@cs.helsinki.fi

Abstract. Since its emergence almost 20 years ago (Schwartz et al., Science 1995), optical mapping has undergone a transition from laboratory technique to commercially available data generation method. In line with this transition, it is only relatively recently that optical mapping data has started to be used for scaffolding contigs and assembly validation in large-scale sequencing projects — for example, the goat (Dong et al., Nature Biotech. 2013) and amborella (Chamala et al., Science 2013) genomes. One major hurdle to the wider use of optical mapping data is the efficient alignment of *in silico* digested contigs to an optical map. We develop Twin to tackle this very problem. Twin is the first index-based method for aligning *in silico* digested contigs to an optical map. Our results demonstrate that Twin is an order of magnitude faster than competing methods on the largest genome. Most importantly, it is specifically designed to be capable of dealing with large eukaryote genomes and thus is the only non-proprietary method capable of completing the alignment for the budgerigar genome in a reasonable amount of CPU time.

1 Introduction

With the cost of next generation sequencing (NGS) continuing to fall, the last decade has been witness to the production of draft whole genome sequences for dozens of species. However, *de novo* genome assembly, the process of reconstructing long contiguous sequences (*contigs*) from short sequence reads, still produces a substantial number of errors [25,1] and is easily misled by repetitive regions [26].

One way to improve the quality of assembly is to use secondary information (independent of the short sequence reads themselves) about the order and orientation of contigs. Optical mapping, which constructs ordered genome-wide high-resolution restriction maps, can provide such information. Optical mapping is a system that works as follows [4,10]: an ensemble of DNA molecules adhered to a charged glass plate are elongated by fluid flow. An enzyme is then used

© Springer-Verlag Berlin Heidelberg 2014

to cleave them into fragments at loci where the enzyme's recognition sequence occurs. Next, the remaining fragments are highlighted with fluorescent dye and digitally photographed under a microscope. Finally, these images are analyzed to estimate the fragment sizes, producing a molecular map. Since the fragments stay relatively stationary during the aforementioned process, the images captures their relative order and size [23]. Multiple copies of the genome undergo this process, and a consensus map is formed that consists of an ordered sequence of fragment sizes, each indicating the approximate number of bases between occurrences of the recognition sequence in the genome [2].

The raw optical mapping data identified by the image processing is an ordered sequence of fragment lengths. Hence, an optical map with x fragments can be denoted as $\ell = \{\ell_1, \ell_2, \ldots, \ell_x\}$, where ℓ_i is the length of the ith fragment in base pairs. This raw data can then be converted into a sequence of locations, each of which determines where a restriction site occurs. We denote the converted data as follows: $L(x) = \{L_0 < L_1 < \cdots < L_n\}$, where $\ell_i = L_i - L_{i-1}$ for $i = 1, \ldots, n$, and L_0 and L_n are defined by the original molecule as a segment of the whole genome by shearing. This latter representation is convenient for algorithmic descriptions. The approximate mean and standard deviation of the fragment size error rate for current data [31] are zero and 150 bp, respectively. See Figure 1 for an illustration of the data produced by this technique. Each restriction enzyme recognizes a specific nucleotide sequence so a unique optical map results from each enzyme, and multiple enzymes can be used in combination to derive denser optical maps. Optical maps have recently become commercially available for mammalian-sized genomes[1], allowing them to be used in a variety of applications.

Although optical mapping data has been used for structural variation detection [28], scaffolding and validating contigs for several large sequencing projects — including those for various prokaryote species [24,32,33], *Oryza sativa* (rice) [35], maize [34], mouse [9], goat [11], *Melopsittacus Undulatus* (budgerigar) [16], and *Amborella trichopoda* [8] — there exist few non-proprietary tools for analyzing this data. Furthermore, the currently available tools are extremely slow because most of them were specifically designed for smaller, prokaryote genomes.

Our Contribution. We present the first index-based method for aligning contigs to an optical map. We call our tool TWIN to illustrate the association between the assembly and optical map as two representations of the genome sequence. The first step of our procedure is to *in silico* digest the contigs with the set of restriction enzymes, computationally mimicking how each restriction enzyme would cleave the short segment of DNA defined by the contig. Thus, *in silico digested contigs* are miniature optical maps that can be aligned to the much longer (sometimes genome-wide) optical maps. The objective is to search and align the *in silico* digested contigs to the correct location in the optical map. By using a suitably-constructed FM-Index data structure [12] built on the optical map, we

[1] OpGen (`http://www.opgen.com`) and BioNano (`http://www.bionanogenomics.com`) are commercial producers of optical mapping data.

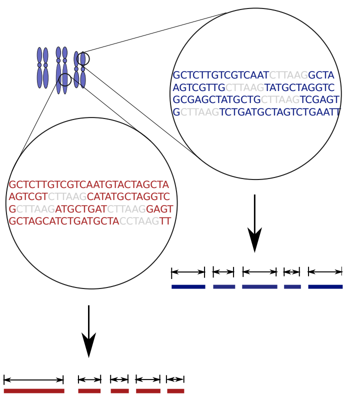
GCTCTTGTCGTCAATCTTAAGGCTA
AGTCGTTGCTTAAGTATGCTAGGTC
GCGAGCTATGCTGCTTAAGTCGAGT
GCTTAAGTCTGATGCTAGTCTGAATT
GCTCTTGTCGTCAATGTACTAGCTA
AGTCGTCTTAAGCATATGCTAGGTC
GCTTAAGATGCTGATCTTAAGGAGT
GCTAGCATCTGATGCTACCTAAGTT

Fig. 1. An illustration of the data produced by optical mapping. Optical mapping locates and measures the distance between restriction sites. Analogous to sequence data, optical mapping data is produced for multiple copies of the same genome, and overlapping single molecular maps are analyzed to produce a map for each chromosome.

show that alignments between contigs and optical maps can be computed in time that is faster than competing methods by more than two orders of magnitude.

Twin takes as input a set of contigs and an optical map, and produces a set of alignments. The alignments are output in Pattern Space Layout (PSL) format, allowing them to be visualized using any PSL visualization software, such as IGV [29]. Twin is specifically designed to work on a wide range of genomes, anything from relatively small genomes, to large eukaryote genomes. Thus, we demonstrate the effectiveness of Twin on *Yersinia kristensenii*, rice, and budgerigar genomes. Rice and budgerigar have genomes of total sizes 430 Mb and 1.2 Gb, respectively. *Yersinia kristensenii*, a bacteria with genome size of 4.6 Mb, is the smallest genome we considered. Short read sequence data was assembled for these genomes, and the resulting contigs were aligned to the respective optical map. We compared the performance of our tool with available competing methods; specifically, the method of Valouev et al. [30] and SOMA [22]. Twin has superior performance on all datasets, and is demonstrated to be the only current method that is capable of completing the alignment for the budgerigar genome in a reasonable amount of CPU time; SOMA [22] required over 77 days of machine time to solve this problem, whereas, Twin required just 35 minutes. Lastly, we verify our approach on simulated *E. coli* data by showing our alignment method found correct placements for the *in silico* digested contigs on a simulated optical map. Twin is available for download at `http://www.cs.colostate.edu/twin`.

Roadmap. We review related tools for the problem in the remainder of this section. Section 2 then sets notation and formally lays the data structural tools we make use of. Section 3 gives details of our approach. We report our experimental results in Section 4. Finally, Section 5 offers reflections and some potentially fruitful avenues future work may take.

Related Work. The most recent tools to make use of optical mapping data in the context of assembly are AGORA [19] and SOMA [22]. AGORA [19] uses the optical map information to constrain de Bruijn graph construction with the aim of improving the resulting assembly. SOMA [22] is a scaffolding method that uses an optical map and is specifically designed for short-read assemblies. SOMA requires an alignment method for scaffolding and implements an $O(n^2m^2)$-time dynamic programming algorithm. Gentig [2], and software developed by Valouev et al. [30] also use dynamic programming to address the closely related task of finding alignments between optical maps. Gentig is not available for download. BACop [34] also uses a dynamic programming algorithm and corresponding scoring scheme that gives more weight to contigs with higher fragment density. Antoniotti et al. [3] consider the unique problem of validating an optical map by using assembled contigs. This method assumes the contigs are error-free. Optical mapping data was produced for Assemblathon 2 [6].

2 Background

Strings. Throughout we consider a string $\mathsf{X} = \mathsf{X}[1..n] = \mathsf{X}[1]\mathsf{X}[2]\ldots\mathsf{X}[n]$ of $|\mathsf{X}| = n$ symbols drawn from the alphabet $[0..\sigma-1]$. For $i = 1,\ldots,n$ we write $\mathsf{X}[i..n]$ to denote the *suffix* of X of length $n-i+1$, that is $\mathsf{X}[i..n] = \mathsf{X}[i]\mathsf{X}[i+1]\ldots\mathsf{X}[n]$. Similarly, we write $\mathsf{X}[1..i]$ to denote the *prefix* of X of length i. $\mathsf{X}[i..j]$ is the *substring* $\mathsf{X}[i]\mathsf{X}[i+1]\ldots\mathsf{X}[j]$ of X that starts at position i and ends at j.

Optical Mapping. From a computational point of view, optical mapping is a process that takes two strings: a genome $\mathsf{A}[1,n]$ and a restriction sequence $\mathsf{B}[1,b]$, and produces an array (string) of integers $\mathsf{M}[1,m]$, such that $\mathsf{M}[i] = j$ if and only if $\mathsf{A}[j..j+b] = \mathsf{B}$ is the ith occurrence of B in A.

For example, if we let $\mathsf{B} = act$ and

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A	*a*	*t*	*a*	*c*	*t*	*t*	*a*	*c*	*t*	*g*	*g*	*a*	*c*	*t*	*a*	*c*	*t*	*a*	*a*	*a*	*c*	*t*

then we would have

$$\mathsf{M} = 3, 7, 12, 15, 20.$$

It will also be convenient to view M slightly differently, as an array of fragment sizes, or distances between occurrences of B in A (equivalently differences between adjacent values in M). We denote this *fragment size domain* of M, as the array $\mathsf{F}[1,m]$, defined such that $\mathsf{F}[i] = (\mathsf{M}[i] - \mathsf{M}[i-1])$, with $\mathsf{F}[1] = \mathsf{M}[1] - 1$. Continuing with the example above, we have

$$\mathsf{F} = 2, 4, 5, 3, 5.$$

Suffix Arrays. The suffix array [20] SA_{X} (we drop subscripts when they are clear from the context) of a string X is an array $\mathsf{SA}[1..n]$ which contains a permutation of the integers $[1..n]$ such that $\mathsf{X}[\mathsf{SA}[1]..n] < \mathsf{X}[\mathsf{SA}[2]..n] < \cdots < \mathsf{X}[\mathsf{SA}[n]..n]$. In other words, $\mathsf{SA}[j] = i$ iff $\mathsf{X}[i..n]$ is the j^{th} suffix of X in lexicographical order.

SA Intervals. For a string Y, the Y-interval in the suffix array SA_{X} is the interval $\mathsf{SA}[s..e]$ that contains all suffixes having Y as a prefix. The Y-interval is a representation of the occurrences of Y in X. For a character c and a string Y, the computation of $c\mathsf{Y}$-interval from Y-interval is called a *left extension.*

BWT and backward search. The Burrows-Wheeler Transform [7] $\mathsf{BWT}[1..n]$ is a permutation of X such that $\mathsf{BWT}[i] = \mathsf{X}[\mathsf{SA}[i] - 1]$ if $\mathsf{SA}[i] > 1$ and \$ otherwise. We also define $\mathsf{LF}[i] = j$ iff $\mathsf{SA}[j] = \mathsf{SA}[i] - 1$, except when $\mathsf{SA}[i] = 1$, in which case $\mathsf{LF}[i] = I$, where $\mathsf{SA}[I] = n$.

Ferragina and Manzini [12] linked BWT and SA in the following way. Let $\mathsf{C}[c]$, for symbol c, be the number of symbols in X lexicographically smaller than c. The function $\mathsf{rank}(\mathsf{X}, c, i)$, for string X, symbol c, and integer i, returns the number of occurrences of c in $\mathsf{X}[1..i]$. It is well known that $\mathsf{LF}[i] = \mathsf{C}[\mathsf{BWT}[i]] + \mathsf{rank}(\mathsf{BWT}, \mathsf{BWT}[i], i)$. Furthermore, we can compute the left extension using C and rank. If $\mathsf{SA}[s..e]$ is the Y-interval, then $\mathsf{SA}[\mathsf{C}[c] + \mathsf{rank}(\mathsf{BWT}, c, s), \mathsf{C}[c] + \mathsf{rank}(\mathsf{BWT}, c, e)]$ is the $c\mathsf{Y}$-interval. This is called *backward search* [12], and a data structure supporting it is called an *FM-index*.

3 Methods

We find alignments in four steps. First, we convert contigs from the sequence domain to the optical map domain through the process of *in silico* digestion. Second, an FM-index is built from the sequence of optical map fragment sizes. Third, we execute a modified version of the FM-index backward search algorithm described in Section 2 that allows inexact matches. As a result of allowing inexact matches, there may be multiple fragments in an optical map that could each be a reasonable match for an *in silico* digested fragment, and in order to include all of these as candidate matches, backtracking becomes necessary in the backward search. For every backward search path that maintains a non-empty interval for the entire query contig, we emit the alignments denoted by the final interval.

3.1 Converting Contigs to the Optical Map Domain

In order to find alignments for contigs relative to the optical map, we must first convert the strings of bases into the domain of optical maps, that is, strings of fragment sizes. We do this by performing an *in silico* digest of each contig, which is performing a linear search over its bases, searching for occurrences of the enzyme recognition sequence and then computing the distances between adjacent restriction sites. These distances are taken to be equivalent to the fragment sizes that would result if the contig's genomic region underwent digestion in a lab. Additionally, the end fragments of the *in silico* digested contig are removed, as the outside ends are most likely not a result of the optical map restriction enzyme digestion, but rather an artifact of the sequencing and assembly process.

3.2 Building an FM-index from Optical Mapping Data

We construct the FM-index for ℓ, the string of fragment sizes. The particular FM-index implementation we use is the SDSL-Lite[2] [14] library's *compressed suffix array with integer wavelet tree* data structure[3].

[2] `https://github.com/simongog/sdsl-lite`.

[3] The exact revision we used was commit ae42592099707bc59cd1e74997e635324b210115.

In preparation for finding alignments, we also keep two auxiliary data structures. The first is the suffix array, SA_F, corresponding to our FM-index, which we use to report the positions in ℓ where alignments of a contig occur. While we could decode the relevant entries of SA on demand with the FM-index in $O(p)$ time, where p is the so-called sample period of the FM-index, storing SA explicitly significantly improves runtime at the cost of a modest increase in memory usage. The second data structure we store is M, which allows us to map from positions in ℓ to positions in the original genome in constant time.

3.3 Alignment of Contigs Using the FM-index

After constructing the FM-index of the optical map, we find alignments between the optical map and the *in silico* digested contigs.

Specifically, we try to find substrings of the optical map fragment sequence ℓ that are similar to the string of each *in silico* digested contig's non-end fragments F satisfying an alignment goodness metric suggested by Nagarajan et al. [22] [4]:

$$\left|\sum_{i=s}^{t} F_i - \sum_{j=u}^{v} \ell_j\right| \leq F_\sigma \sqrt{\sum_{j=u}^{v} \sigma_j^2},$$

where a parameter F_σ will affect the precision/recall tradeoff.

This computation is carried out using a modified FM-index backward search. A simplified, recursive version of our algorithm for finding alignments is shown in Algorithm 1. The original FM-index backward search proceeds by finding a succession of intervals in the suffix array of the original text that progressively match longer and longer suffixes of the query string, starting from the rightmost symbol of the query. Each additional symbol in the query string is matched in a process taking two arguments: 1) a suffix array interval, the Y-interval, corresponding to the suffixes in the text, ℓ, whose prefix matches a suffix of the query string, and 2) an extension symbol c. The process returns a new interval, the $c\mathsf{Y}$-interval, where a prefix of each text suffix corresponding to the new interval is a left extension of the previous query suffix. This process is preserved in TWIN, and is represented by the function *BackwardSearchOneSymbol* in the TWIN algorithm, displayed in Algorithm 1.

Since the optical map fragments include error from the measurement process, it cannot be assumed an *in silico* fragment size will exactly match the optical map fragment size from the same locus in the genome. To accomodate these differences, we determine a set of distinct candidate match fragment sizes, D, each similar in size to the next fragment to be matched in our query. These candidates are drawn from the interval of the BWT currently active in our backward search. We do this by a wavelet tree traversal function provided by SDSL-Lite, which implements the algorithm described in [13] and takes $O(|D| \log(f/\Delta))$ time. This

[4] N.B. Alternative goodness metrics could be substituted. They must satisfy the property that pairs of strings considered to align well are composed of substrings that are also considered to align well would also work.

is represented by the function *RestrictedUniqueRangeValues* in Algorithm 1. We emphasise that, due to the large alphabet of ℓ, the wavelet tree's ability to list unique values in a range efficiently is vital to overall performance. Unlike in other applications where the FM-index is used for approximate pattern matching (e.g. read alignment), we cannot afford a bruteforce enumeration of the alphabet at each step in the backward search.

These candidates are chosen to be within a reasonable noise tolerance, t, based on assumptions about the distribution of optical measurement error around the true fragment length. Since there may be multiple match candidates in the BWT interval of the optical map for a query fragment, we extend the backward search with backtracking so each candidate size computed from the wavelet tree is evaluated. That is, for a given *in silico* fragment size (i.e. symbol) c, every possible candidate fragment size, c', that can be found in the optical map in the range $c - t \ldots c + t$ and in the interval $s \ldots e$ (of the BWT) for some tolerance t is used as a substitute in the backward search. Each of these candidates is then checked to ensure that a left extension would still satify the goodness metric, and then used as the extension symbol in the backward search. So it is actually a set of $c'\mathsf{Y}$-intervals that is computed as the left extension in Twin. Additionally, small DNA fragments may not adhere sufficiently to the glass surface and can be lost in the optical mapping process, so we also branch the backtracking search both with and without small *in silico* fragments to accomodate the uncertainty.

Each time the backward search algorithm successfully progresses throughout the entire query (i.e. it finds some approximate match in the optical map for each fragment in the contig query), we take the contents of the resulting interval in the SA as representing a set of likely alignments.

3.4 Output of Alignments in PSL Format

For each *in silico* digested contig that has an approximate match in the optical map, we emit the alignment, converting positions in the fragment string ℓ to positions in the genome using the M table. We provide a script to convert the human readable output into PSL format.

4 Results

We evaluated the performance of Twin against the best competing methods on *Yersinia kristensenii*, rice and budgerigar. These three genomes were chosen because they have available sequence and optical mapping data and are diverse in size. For each dataset, we compared the runtime, peak memory usage, and the number of contigs for which at least one alignment was found for Twin, SOMA [22], and the software of Valouev et al. [30]. Peak memory was measured as the maximum resident set size as reported by the operating system. Runtime is the user process time, also reported by the operating system. SOMA [22] v2.0 was run with example parameters provided with the tool and the software of Valouev et al. [30] was run with its scoring parameters object constructed with

Algorithm 1. MATCH(s, e, q, h) Provided a suffix array start index s and end index e, query string q, and rightmost unmatched query string index h (initially $s = 1$, $e = m$, $h = |q| - 1$), emit alignments of an *in silico* digested contig to an optical map

```
procedure MATCH($s$,$e$,$q$,$h$)
    if $h = -1$ then
        ▷ Recursion base case. Suffix array indexes $s..e$ denote original query matches.
        $Emit(s, e)$
    else
        ▷ The next symbol to match, $c$, is the last symbol in the query string.
        $c \leftarrow q[h]$
        ▷ Find the approximately matching values in $\mathsf{BWT}[s \ldots e]$, within tolerance $t$.
        $D \leftarrow RestrictedUniqueRangeValues(\text{s, e}, c + t, c - t)$
        ▷ Let $c'$ be one possible substitute for $c$ drawn from $D$
        for all $c' \in D$ do
            ▷ If Equation 1 is still satisified with $c'$ and $c$, ...
            if $\left|\sum_{i=0}^{|q|-h} \mathsf{SA}[s]_i + c' - \sum_{j=h}^{|q|-1} q_j - c\right| \le F_\sigma \sqrt{\sum_{j=0}^{|q|-h} \sigma_j^2}$ then
                ▷ ... determine the suffix array range of the left extension of $c'$.
                $s', e' \leftarrow BackwardSearchOneSymbol(s, e, c')$
                ▷ Recurse to attempt to match the currently unmatched prefix.
                MATCH$(s', e', q, h - 1)$
```

arguments (0.2, 2, 1, 5, 17.43, 0.579, 0.005, 0.999, 3, 1). TWIN was run with $D_\sigma = 4$, $t = 1000$, and $[250 \ldots 1000]$ for the range of small fragments. Gentig [2] and BACop [34] were not available for download so we did not test the data using these approaches.

The sequence data was assembled for *Yersinia kristensenii*, rice and budgerigar by using various assemblers. The relevant assembly statistics are given in Table 1. An important statistic in this table is the number of contigs that have at least two restriction sites, since contigs with fewer than two are unable to be aligned meaningfully by any method, including TWIN. This statistic was computed to reveal cases of ambiguity in placement from lack of information. Indeed, Assemblathon 2 required there to be nine restriction sites present in a contig to align it to the optical mapping data [6]. All experiments were performed on Intel x86-64 workstations with sufficient RAM to avoid paging, running 64-bit Linux.

The experiments for *Yersinia kristensenii*, rice and budgerigar illustrate how each of the programs' running time scale as the size of the genome increases. However, due to the possibility of mis-assemblies in these draft genomes, comparing the actual alignments could possibly lead to erroneous conclusions. Therefore, we will verify the alignments using simulated *E. coli* data. See Subsection 4.4 for this experiment.

Table 1. Assembly and genome statistics for *Yersinia kristensenii*, rice and budgerigar. The assembly statistics were obtained from Quast. [15].

Genome	N50	Genome Size	No. of Contigs with ≥ 2 restriction sties
Y. kristensenii	30,719	4.6 Mb	92
Rice	5,299	430 Mb	3,103
Budgerigar	77,556	1.2 Gb	10,019

4.1 Performance on *Yersinia kristensenii*

The sequence and optical map data for *Yersinia kristensenii* are described by Nagarajan *et al.* [22]. The *Yersinia kristensenii* ATCC 33638 reads were generated using 454 GS 20 sequencing and assembled using SPAdes version 3.0.0 [5] using default parameters. Contigs from this assembly were aligned against an optical map of the bacterial strain generated by OpGen using the AflII restriction enzyme. There are approximately 1.4 million single-end reads for this dataset, and they were obtained from the NCBI Short Read Archive (accession SRX013205). Of the 92 contigs that could be aligned to the optical map, the software of Valouev et al. aligned 91 contigs, SOMA aligned 54 contigs, and Twin aligned 61 contigs. Thus, Twin found more alignments than SOMA, and did so faster. It should be noted that, for this dataset, all three tools had reasonable runtimes. However, while the software of Valouev et al. found more alignments, our validation experiments (below) suggest these results may favor recall over precision, and many of the additional alignments may not be credibled.

4.2 Performance on Rice Genome

The second dataset consists of approximately 134 million 76 bp paired-end reads from *Oryza sativa Japonica* rice, generated by Illumina, Inc. on the Genome Analayzer (GA) IIx platform, as described by Kawahara *et al.* [17]. These reads were obtained from the NCBI Short Read Archive (accession SRX032913) and assembled using SPAdes version 3.0.0 [5] using default parameters. The optical map for rice was constructed by Zhou *et al.* [35] using SwaI as the restriction enzyme. This optical map was assembled from single molecule restriction maps into 14 optical map contigs, labeled as 12 chromosomes, with chromosome labels 6 and 11 both containing two optical map contigs.

Again, Twin found alignments for more contigs than SOMA on the rice genome. SOMA and Twin found alignments for 2,434, and 3,098 contigs, respectively, out of 3,103 contigs that could be aligned to the optical map. However, while SOMA required over 29 minutes to run, Twin required less than one minute. The software of Valouev executed faster than SOMA (taking around 3 minutes), though still several times slower than Twin on this modest sized genome.

4.3 Performance on Budgerigar Genome

The sequence and optical map data for the budgerigar genome were generated for the Assemblathon 2 project of Bradnam *et al.* [6]. Sequence data consists of a combination of Roche 454, Illumina, and Pacific Biosciences reads, providing 16x, 285x, and 10x coverage (respectively) of the genome. All sequence reads are available at the NCBI Short Read Archive (accession ERP002324). For our analysis we consider the assembly generated using Celera [21], which was completed by the CBCB team (Koren and Phillippy) as part of Assemblathon 2 [6]. The optical mapping data was created by Zhou, Goldstein, Place, Schwartz, and Bechner using the SwaI restriction enzyme and consists of 92 separate pieces.

As with the two previous data sets, Twin found alignments for more contigs than SOMA on the budgerigar genome. SOMA and Twin found alignments for 9,668, and 9,826 contigs, respectively, out of 10,019 contigs that could be aligned to the optical map. However, SOMA required over 77 days of CPU time and Twin required 35 minutes. The software of Valouev et al. returned 9,814 alignments and required over an order of magnitude (6.5 hours) of CPU time. Hence, Twin was the only method that efficiently aligned the *in silico* digested budgerigar genome contigs to the optical map. It should be kept in mind that the competing methods were developed for prokaryote genomes and so we are repurposing them at a scale for which they were not designed. Lastly, the amount of memory used by all the methods on all experiments was low enough for them to run on a standard workstation.

We were forced to parallelize SOMA due to the enormous amount of CPU time SOMA required for this dataset. To accomplish this task, the FASTA file containing the contigs was split into 300 different files, and then IPython Parallel library was used to invoke up to two instances of SOMA on each machine from a set of 150 machines. Thus, when using a cluster with up to 300 jobs concurrently, the alignment for the budgerigar genome took about a day of wall clock time. In contrast, we ran the software of Valouev et al. and Twin with a single thread running on a single core. However, it should be noted that the same parallelization could have been accomplished for both these software methods too. Also, even with parallelization of SOMA, Twin is still an order of magnitude faster than it.

4.4 Alignment Verification

We compared the alignments given by Twin against the alignments of the contigs of an *E. coli* assembly to the *E. Coli* (str. K-12 substr. MG1655) reference genome. Our prior experiments involved species for which the reference genome may have regions that are mis-asssembled and therefore, contig alignments to the reference genome may be inaccurate and cannot be used for comparison and verification of the *in silico* digested contig alignment. The *E. coli* reference genome is likely to contain the fewest errors and thus, is the one we used for assembly verification. The sequence data consists of approximately 27 million paired-end 100 bp reads from *E. coli* (str. K-12 substr. MG1655) generated by

Table 2. Comparsion of the alignment results for Twin **and competing method.** The performance of Twin was compared against SOMA [22] and the method of Valouev et al. [30] using the assembly and optical mapping data for *Yersinia Kristensenii*, rice, and budgerigar. Various assemblers were used to assemble the data for these species. The relevant statistics and information concerning these assemblies and genomes can be found in Table 1. The peak memory is given in megabytes (mb). The running time is reported in seconds (s), minutes (m), hours (h), and days.

Genome	Program	Memory	Time	Aligned Contigs
Y. Kristensenii				
	Valouev *et al.*	1.81	.17 s	91
	SOMA	1.71	7.32 s	54
	Twin	18	.06 s	65
Rice				
	Valouev *et al.*	11.25	2 m 57 s	2,676
	SOMA	7.94	29 m 38 s	2,434
	Twin	18.25	50 s	3,098
Budgerigar				
	Valouev *et al.*	390	6.5 h	9,814
	SOMA	380.95	77.2 d	9,668
	Twin	127.112	35 m	9,826

Illumina, Inc. on the Genome Analayzer (GA) IIx platform, and was obtained from the NCBI Short Read Archive (accession ERA000206), and was assembled using SPAdes version 3.0.0 [5] using default parameters. This assembly consists of 160 contigs; 50 of which contain two restriction sites, the minimum required for any possible optical alignment, and complete alignments with minimal (<800 bp) total in/dels relative to the reference genome.

We simulated an optical map using the reference genome for *E. coli* (str. K-12 substr. MG1655) since there is no publicly available one for this genome.

The 50 contigs that contained more than two restriction sites were aligned to the reference genome using BLAT [18]. These same contigs were then *in silico* digested and aligned to the optical map using Twin. The resulting PSL files were then compared. Twin found alignment positions within 10% of those found by BLAT for all 50 contigs, justifying that our method is finding correct alignments. We repeated this verification approach with both SOMA and the software from Valouev. All of SOMA's reported alignments had matching BLAT alignments, while of the 49 alignments the software from Valuoev reported, only 18 could be matched with alignments from BLAT.

5 Discussion and Conclusions

We demonstrated that Twin, an index-based algorithm for aligning *in silico* digested contigs to an optical map, gave over an order of magnitude improvement to runtime without sacrificing alignment quality. Our results show that we are able to handle genomes at least as large as the budgerigar genome directly, whereas SOMA cannot feasibly complete the alignment for this genome in a reasonable amount of time without significant parallelization, and even then is orders of magnitude slower than Twin. Indeed, given its performance on the budgerigar genome, and its $O(m^2n^2)$ time complexity, larger genomes seem beyond SOMA. For example, the loblolly pine tree genome, which is approximately 20 Gb [36], would take SOMA approximately 84 machine years, which, even with parallelization, is prohibitively long.

Lastly, optical mapping is a relatively new technology, and thus, with so few algorithms available for working with this data, we feel there remains good opportunities for developing more efficient and flexible methods. Dynamic programming optical map alignment approaches are still important today, as the assembly of the consensus optical maps from the individually imaged molecules often has to deal with missing or spurious restriction sites in the single molecule maps when enzymes fail to digest a recognition sequence or the molecule breaks. Though coverage is high (e.g. about 1,241 Gb of optical data was collected for the 2.66 Gb goat genome), there may be cases where missing restriction site errors are not resolved by the assembly process. In these rare cases (only 1% of alignments reported by SOMA on parrot contain such errors) they will inhibit Twin's ability to find correct alignments. In essence, Twin is trading a small degree of sensitivity for a huge speed increase, just as other index based aligners have done for sequence data. Sirén et al. [27] recently extended the Burrows-Wheeler transform (BWT) from strings to acyclic directed labeled graphs and to support path queries. In future work, an adaptation of this method for optical map alignment may allow for the efficient handling of missing or spurious restriction sites.

Acknowledgements. The authors would like to thank David C. Schwartz and Shiguo Zhou from the University of Wisconsin-Madison, Mihai Pop and Lee Mendelowitz from the University of Maryland, and Erich Jarvis and Jason Howard from Duke University for providing insightful comments and providing data for this project. The authors would also like to thank the reviewers for their insightful comments. MM and CB were funded by the Colorado Clinical and Translational Sciences Institute which is funded by National Institutes of Health (NIH-NCATS,UL1TR001082, TL1TR001081, KL2TR001080). SJP was supported by the the Helsinki Institute of Information Technology (HIIT) and by Academy of Finland through grants 258308 and 250345 (CoECGR). All authors greatly appreciate the funding provided for this project.

References

1. Alkan, C., Sajjadian, S., Eichler, E.: Limitations of next-generation genome sequence assembly. Nat. Methods 8(1), 61–65 (2010)
2. Anantharaman, T., Mishra, B.: A probabilistic analysis of false positives in optical map alignment and validation. In: Proc. of WABI, pp. 27–40 (2001)

3. Antoniotti, M., Anantharaman, T., Paxia, S., Mishra, B.: Genomics via optical mapping iv: sequence validation via optical map matching. Technical report, New York University (2001)
4. Aston, C., Schwartz, D.: Optical mapping in genomic analysis. John Wiley and Sons, Ltd. (2006)
5. Bankevich, A., et al.: others. SPAdes: a new Genome assembly algorithm and its applications to single-cell sequencing. J. Comp. Biol. 19(5), 455–477 (2012)
6. Bradnam, K.R., et al.: Assemblathon 2: evaluating *de novo* methods of genome assembly in three vertebrate species. GigaScience 2(1), 1–31 (2013)
7. Burrows, M., Wheeler, D.: A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, Palo Alto, California (1994)
8. Chamala, S., et al.: Assembly and validation of the genome of the nonmodel basal angiosperm *amborella*. Science 342(6165), 1516–1517 (2013)
9. Church, D.M., et al.: Lineage-specific biology revealed by a finished genome assembly of the mouse. PLoS Biology 7(5), e1000112+ (2009)
10. Dimalanta, et al.: A microfluidic system for large dna molecule arrays. Anal. Chem. 76(18), 5293–5301 (2004)
11. Dong, Y., et al.: Sequencing and automated whole-genome optical mapping of the genome of a domestic goat (*capra hircus*). Nat. Biotechnol. 31(2), 136–141 (2013)
12. Ferragina, P., Manzini, G.: Indexing compressed text. J. ACM 52(4), 552–581 (2005)
13. Gagie, T., Navarro, G., Puglisi, S.J.: New algorithms on wavelet trees and applications to information retrieval. Theor. Comput Sci. 426-427, 25–41 (2012)
14. Gog, S., Petri, M.: Optimized succinct data structures for massive data. Software Pract. Expr. (to appear)
15. Gurevich, A., Saveliev, V., Vyahhi, N., Tesler, G.: QUAST: quality assessment tool for genome assemblies. Bioinformatics 29(8), 1072–1075 (2013)
16. Howard, J.T., et al.: De Novo high-coverage sequencing and annotated assemblies of the budgerigar genome (2013)
17. Kawahara, Y., et al.: Improvement of the *oryza sativa nipponbare* reference genome using next generation sequence and optical map data. Rice 6(4), 1–10 (2013)
18. Kent, J.: BLAT–The BLAST-Like Alignment Tool. Genome Res. 12(4), 656–664 (2002)
19. Lin, H., et al.: AGORA: Assembly Guided by Optical Restriction Alignment. BMC Bioinformatics 12, 189 (2012)
20. Manber, U., Myers, G.W.: Suffix arrays: A new method for on-line string searches. SIAM J. Sci. Comput. 22(5), 935–948 (1993)
21. Miller, J.R., et al.: Aggressive assembly of pyrosequencing reads with mates. Bioinformatics 24, 2818–2824 (2008)
22. Nagarajan, N., Read, T.D., Pop, M.: Scaffolding and validation of bacterial genome assemblies using optical restriction maps. Bioinformatics 24(10), 1229–1235 (2008)
23. Neely, R.K., Deen, J., Hofkens, J.: Optical mapping of DNA: single-molecule-based methods for mapping genome. Biopolymers 95(5), 298–311 (2011)
24. Reslewic, S., et al.: Whole-genome shotgun optical mapping of *rhodospirillum rubrum*. Appl. Environ. Microbiol. 71(9), 5511–5522 (2005)
25. Ronen, R., Boucher, C., Chitsaz, H., Pevzner, P.: SEQuel: Improving the Accuracy of Genome Assemblies. Bioinformatics 28(12), i188–i196 (2012)
26. Salzberg, S.: Beware of mis-assembled genomes. Bioinformatics 21(24), 4320–4321 (2005)

27. Sirén, J., Välimäki, N., Mäkinen, V.: Indexing graphs for path queries with applications in genome research. IEEE/ACM Trans. Comput. Biol. Bioinform. (to appear, 2014)
28. Teague, B., et al.: High-resolution human genome structure by single-molecule analysis. Proc. Natl. Acad. Sci. 107(24), 10848–10853 (2010)
29. Thorvaldsdòttir, H., Robinson, J.T., Mesirov, J.P.: Integrative Genomics Viewer (IGV): High-performance Genomics Data Visualization and Exploration. Brief. Bioinform. 14(2), 178–192 (2013)
30. Valouev, A., et al.: Alignment of optical maps. J. Comp. Biol. 13(2), 442–462 (2006)
31. VanSteenHouse, H. personal communication (2013)
32. Zhou, S., et al.: A whole-genome shotgun optical map of *yersinia pestis* strain KIM. Appl. Environ. Microbiol. 68(12), 6321–6331 (2002)
33. Zhou, S., et al.: Shotgun optical mapping of the entire *leishmania major* Friedlin genome. Mol. Biochem. Parasitol. 138(1), 97–106 (2004)
34. Zhou, S., et al.: A single molecule scaffold for the maize genome. PLoS Genet. 5(11), e1000711 (2009)
35. Zhou, S., et al.: Validation of rice genome sequence by optical mapping. BMC Genomics 8(1), 278 (2007)
36. Zimin, A., et al.: Sequencing and assembly of the 22-gb loblolly pine genome. Genetics 196(3), 875–890 (2014)