GAML: Genome Assembly by Maximum Likelihood

Vladimír Boža, Broňa Brejová, and Tomáš Vinař

Faculty of Mathematics, Physics, and Informatics, Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia

Abstract. The critical part of genome assembly is resolution of repeats and scaffolding of shorter contigs. Modern assemblers usually perform this step by heuristics, often tailored to a particular technology for producing paired reads or long reads. We propose a new framework that allows systematic combination of diverse sequencing datasets into a single assembly. We achieve this by searching for an assembly with maximum likelihood in a probabilistic model capturing error rate, insert lengths, and other characteristics of each sequencing technology.

We have implemented a prototype genome assembler GAML that can use any combination of insert sizes with Illumina or 454 reads, as well as PacBio reads. Our experiments show that we can assemble short genomes with N50 sizes and error rates comparable to ALLPATHS-LG or Cerulean. While ALLPATHS-LG and Cerulean require each a specific combination of datasets, GAML works on any combination.

Data and software is available at http://compbio.fmph.uniba.sk/gaml

1 Introduction

The second and third generation sequencing technologies have dramatically decreased the cost of sequencing. Nowadays, we have a surprising variety of sequencing technologies, each with its own strengths and weaknesses. For example, Illumina platforms are characteristic by low cost and high accuracy, but the reads are short. On the other hand, Pacific Biosciences offer long reads at the cost of quality and coverage. In the meantime, the cost of sequencing was brought down to the point, where it is no longer a sole domain of large sequencing centers; even small labs can experiment with cost-effective genome sequencing. In this setting, it is no longer possible to recommend a single protocol that should be used to sequence genomes of a particular size. In this paper, we propose a framework for genome assembly that allows flexible combination of datasets from different technologies in order to harness their individual strengths.

Modern genome assemblers are usually based either on the overlap–layout–consensus framework (e.g. Celera by Myers et al. (2000), SGA by Simpson and Durbin (2010)), or on de Bruijn graphs (e.g. Velvet by Zerbino and Birney (2008), ALLPATHS-LG by Gnerre et al. (2011)). Both approaches can be seen as special cases of a string graph (Myers, 2005), in which

D. Brown and B. Morgenstern (Eds.): WABI 2014, LNBI 8701, pp. 122–134, 2014.

we represent sequence fragments as vertices, while edges represent possible adjacencies of fragments in the assembly. A genome assembly is simply a set of walks through this graph. The main difference between the two frameworks is how we arrive at a string graph: through detecting long overlaps of reads (overlap–layout–consensus) or through construction of de Bruijn graphs based on k-mers.

However, neither of these frameworks is designed to systematically handle pair-end reads and additional heuristic steps are necessary to build larger scaffolds from assembled contigs. For example, ALLPATHS-LG (Gnerre et al., 2011) uses libraries with different insert lengths for scaffolding of contigs assembled without the use of paired read information, while Cerulean (Deshpande et al., 2013) uses Pacific Biosystems long reads for the same purpose. Recently, the techniques of paired de Bruijn graphs (Medvedev et al., 2011) and pathset graphs (Pham et al., 2013) were developed to address paired reads systematically, however these approaches cannot combine several libraries with different insert sizes.

Combination of sequencing technologies with complementary strengths can help to improve assembly quality. However, it is not feasible to design new algorithms for every possible combination of datasets. Often it is possible to supplement previously developed tools with additional heuristics for new types of data. For example, PBJelly (English et al., 2012) uses Pacific Biosystems reads solely to aid gap filling in draft assemblies. Assemblers like PacbioToCa (Koren et al., 2012) or Cerulean (Deshpande et al., 2013) use short reads to "upgrade" the quality of Pacific Biosystems reads so that they can be used within traditional assemblers. However, such approaches hardly use all information contained within the data sets.

We propose a new framework that allows a systematic combination of diverse datasets into a single assembly, without requiring a particular type of data for specific heuristic steps. Recently, probabilistic models have been used very successfully to evaluate the quality of genome assemblers (Rahman and Pachter, 2013; Clark et al., 2013; Ghodsi et al., 2013). In our work, we use likelihood of a genome assembly as an optimization criterion, with the goal of finding the highest likelihood genome assembly. Even though this may not be always feasible, we demonstrate that optimization based on simulated annealing can be very successful at finding high likelihood genome assemblies.

To evaluate likelihood, we use a relatively complex model adapted from Ghodsi et al. (2013), which can capture characteristics of each dataset, such as sequencing error rate, as well as length distribution and expected orientation of paired reads (Section 2). We can thus transparently combine information from multiple diverse datasets into a single score. Previously, there have been several works in this direction in much simpler models without sequencing errors (Medvedev and Brudno, 2009; Varma et al., 2011). These papers used likelihood to estimate repeat counts, without considering other problems, such as how exactly are repeats integrated within scaffolds.

To test our framework, we have implemented a prototype genome assembler GAML (Genome Assembly by Maximum Likelihood) that can use any combination of insert sizes with Illumina or 454 reads, as well as PacBio reads. The starting

point of the assembly are short contigs derived from Velvet (Zerbino and Birney, 2008) with very conservative settings in order to avoid assembly errors. We then use simulated annealing to combine these short contigs into high likelihood assemblies (Section 3). We compare our assembler to existing tools on benchmark datasets (Section 4), demonstrating that we can assemble genomes of up to 10 MB long with N50 sizes and error rates comparable to ALLPATHS-LG or Cerulean. While ALLPATHS-LG and Cerulean each require a very specific combination of datasets, GAML works on any combination.

2 Probabilistic Model for Sequence Assembly

Recently, several probabilistic models were introduced as a measure of the assembly quality (Rahman and Pachter, 2013; Clark et al., 2013; Ghodsi et al., 2013). All of these authors have shown that the likelihood consistently favours higher quality assemblies. In general, the probabilistic model defines the probability $\Pr(R|A)$ that a set of sequencing reads R is observed assuming that assembly A is the correct assembly of the genome. Since the sequencing itself is a stochastic process, it is very natural to characterize concordance of reads and an assembly by giving a probability of observing a particular read. In our work, instead of evaluating the quality of a single assembly, we use the likelihood as an optimization criterion with the goal of finding high likelihood genome assemblies. We adapt the model of Ghodsi et al. (2013), which we describe in this section.

Basics of the likelihood model. The model assumes that individual reads are independently sampled, and thus the overall likelihood is the product of likelihoods of the reads: $\Pr(R|A) = \prod_{r \in R} \Pr(r|A)$. To make the resulting value independent of the number of reads in set R, we use as the main assembly score the log average probability of a read computed as follows: $\mathrm{LAP}(A|R) = (1/|R|) \sum_{r \in R} \log \Pr(r|A)$. Note that maximizing $\Pr(R|A)$ is equivalent to maximizing $\mathrm{LAP}(A|R)$.

If the reads were error-free and each position in the genome was sequenced equally likely, the probability of observing read r would simply be $\Pr(r|A) = n_r/(2L)$, where n_r is the number of occurrences of the read as a substring of the assembly A, L is the length of A, and thus $2L$ is the length of the two strands combined (Medvedev and Brudno, 2009). Ghodsi et al. (2013) have shown a dynamic programming computation of read probability for more complex models, accounting for sequencing errors. The algorithm marginalizes over all possible alignments of r and A, weighting each by the probability that a certain number of substitution and indel errors would happen during sequencing. In particular, the probability of a single alignment with m matching positions and s errors (substitution and indels) is defined as $R(s,m)/(2L)$, where $R(s,m) = \epsilon^s(1-\epsilon)^m$ and ϵ is the sequencing error rate.

However, full dynamic programming is too time consuming, and in practice only several best alignments contribute significantly to the overall probability. Thus Ghodsi et al. (2013) propose to approximate the probability of observing read r with an estimate based on a set S_r of a few best alignments of r to

genome A, as obtained by a standard fast read alignment tool:

$$\Pr(r|A) \approx \frac{\sum_{j \in S_r} R(s_j, m_j)}{2L}, \tag{1}$$

where m_j is the number of matches in the j-th alignment, and s_j is the number of mismatches and indels implied by this alignment. The formula assumes the simplest possible error model, where insertions, deletions and substitutions have the same probability and ignores GC content bias. Of course, much more comprehensive read models are possible (see e.g. Clark et al. (2013)).

Paired reads. Many technologies provide paired reads produced from the opposite ends of a sequence insert of certain size. We assume that the insert size distribution in a set of reads R can be modeled by the normal distribution with known mean μ and standard deviation σ. The probability of observing paired reads r_1 and r_2 can be estimated from sets of alignments S_{r_1} and S_{r_2} as follows:

$$\Pr(r_1, r_2|A) \approx \frac{1}{2L} \sum_{j_1 \in S_{r_1}} \sum_{j_2 \in S_{r_2}} R(s_{j_1}, m_{j_1}) R(s_{j_2}, m_{j_2}) \Pr(d(j_1, j_2)|\mu, \sigma) \tag{2}$$

As before, m_{j_i} and s_{j_i} are the numbers of matches and sequencing errors in alignment j_i respectively, and $d(j_1, j_2)$ is the distance between the two alignments as observed in the assembly. If alignments j_1 and j_2 are in two different contigs, or on inconsistent strands, $\Pr(d(j_1, j_2)|\mu, \sigma)$ is zero.

Reads that have no good alignment to A. Some reads or read pairs do not align well to A, and as a result, their probability $\Pr(r|A)$ is very low; our approximation by a set of high-scoring alignments can even yield zero probability if set S_r is empty. Such extremely low probabilities then dominate the log likelihood score. Ghodsi et al. (2013) propose a method that assigns such a read a score approximating the situation when the read would be added as a new contig to the assembly. We modify their formulas for variable read length, and use score $e^{c+k\ell}$ for a single read of length ℓ or $e^{c+k(\ell_1+\ell_2)}$ for a pair of reads of lengths ℓ_1 and ℓ_2. Values k and c are scaling constants set similarly as in Ghodsi et al. (2013). These alternative scores are used instead of the read probability $\Pr(r|A)$ whenever the probability is lower than the score.

Multiple read sets. Our work is specifically targeted at a scenario, where we have multiple read sets obtained from different libraries with different insert lengths or even with different sequencing technologies. We use different model parameters for each set and compute the final score as a weighted combination of log average probabilities for individual read sets $R_1, \ldots, R_k$:

$$\mathrm{LAP}(A|R_1, \ldots, R_k) = w_1 \mathrm{LAP}(A|R_1) + \ldots + w_k \mathrm{LAP}(A|R_k) \tag{3}$$

In our experiments we use weight $w_i = 1$ for most datasets, but we lower the weight for Pacific Biosciences reads, because otherwise they dominate the likelihood value due to their longer length. The user could also increase or decrease weights w_i of individual sets based on their reliability.

Penalizing spuriously joined contigs. The model of Ghodsi et al. (2013) does not penalize obvious misassemblies when two contigs are joined together without any evidence in the reads. We have observed that to make the likelihood function applicable as an optimization criterion for the best assembly, we need to introduce a penalty for such spurious connections. We say that a particular base j in the assembly is *connected* with respect to read set R if there is a read which covers base j and starts at least k bases before j, where k is a constant specific to the read set. In this setting, we treat a pair of reads as one long read. If the assembly contains d disconnected bases with respect to R, penalty αd is added to the $\mathrm{LAP}(A|R)$ score (α is a scaling constant).

Properties of different sequencing technologies. Our model can be applied to different sequencing technologies by appropriate settings of model parameters. For example, Illumina technology typically produces reads of length 75-150bp with error rate below 1% (Quail et al., 2012). For smaller genomes, we often have a high coverage of Illumina reads. Using paired reads or mate pair technologies, it is possible to prepare libraries with different insert sizes ranging up to tens of kilobases, which are instrumental in resolving longer repeats (Gnerre et al., 2011). To align these reads to proposed assemblies, we use Bowtie2 (Langmead and Salzberg, 2012). Similarly, we can process reads by the Roche 454 technology, which are characteristic by higher read lengths (hundreds of bases).

Pacific Biosciences technology produces single reads of variable length, with median length reaching several kilobases, but the error rate exceeds 10% (Quail et al., 2012; Deshpande et al., 2013). Their length makes them ideal for resolving ambiguities in alignments, but the high error rate makes their use challenging. To align these reads, we use BLASR (Chaisson and Tesler, 2012). When we calculate the probability $\Pr(r|A)$, we consider not only the best alignments found by BLASR, but for each BLASR alignment, we also add probabilities of similar alignments in its neighborhood. More specifically, we run a banded version of the forward algorithm by Ghodsi et al. (2013), considering all alignments in a band of size 3 around a guide alignment produced by BLASR.

3 Finding a High Likelihood Assembly

Complex probabilistic models, like the one described in Section 2, were previously used to compare the quality of several assemblies (Ghodsi et al., 2013; Rahman and Pachter, 2013; Clark et al., 2013). In our work, we instead attempt to find the highest likelihood assembly directly. Of course, the search space is huge, and the objective function too complex to admit exact methods. Here, we describe an effective optimization routine based on the simulated annealing framework (Eglese, 1990).

Our algorithm for finding the maximum likelihood assembly consists of three main steps: preprocessing, optimization, and postprocessing. In *preprocessing*, we decrease the scale of the problem by creating an assembly graph, where vertices correspond to contigs and edges correspond to possible adjacencies between

contigs supported by reads. In order to make the search viable, we will restrict our search to assemblies that can be represented as a set of walks in this graph. Therefore, the assembly graph should be built in a conservative way, where the goal is not to produce long contigs, but rather to avoid errors inside them. In the *optimization step*, we start with an initial assembly (a set of walks in the assembly graph), and iteratively propose changes in order to optimize the assembly likelihood. Finally, *postprocessing* examines the resulting walks and splits some of them into shorter contigs if there are multiple equally likely possibilities of resolving ambiguities. This happens, for example, when the genome contains long repeats that cannot be resolved by any of the datasets.

In the rest of this section, we discuss individual steps in more detail.

3.1 Optimization by Simulated Annealing

To find a high likelihood assembly, we use an iterative simulated annealing scheme. We start from an initial assembly A_0 in the assembly graph. In each iteration, we randomly choose a *move* that proposes a new assembly A' similar to the current assembly A. The next step depends on the likelihoods of the two assemblies A and A' as follows:

- If $\mathrm{LAP}(A'|R) \geq \mathrm{LAP}(A|R)$, the new assembly A' is accepted and the algorithm continues with the new assembly.
- If $\mathrm{LAP}(A'|R) < \mathrm{LAP}(A|R)$, the new assembly A' is accepted with probability $e^{(\mathrm{LAP}(A'|R)-\mathrm{LAP}(A|R))/T}$; otherwise A' is rejected and the algorithm retains the old assembly A for the next step.

Here, parameter T is called the temperature, and it changes over time. In general, the higher the temperature, the more aggressive moves are permitted. We use a simple cooling schedule, where $T = T_0/\ln(i)$ in the i-th iteration. The computation ends when there is no improvement in the likelihood for a certain amount of time. We select the assembly with the highest LAP score as the result.

To further reduce the complexity of the assembly problem, we classify all contigs as either *long* (more than 500bp) or *short* and concentrate on ordering the long contigs correctly. The short contigs are used to fill the gaps between the long contigs.

Recall that each assembly is a set of walks in the assembly graph. A contig can appear in more than one walk or can be present in a single walk multiple times. In all our experiments, the starting assembly simply contains each long contig as a separate walk. However, other assemblies (such as assemblies from other tools) can easily serve as a starting point as long as they can be mapped to the assembly graph.

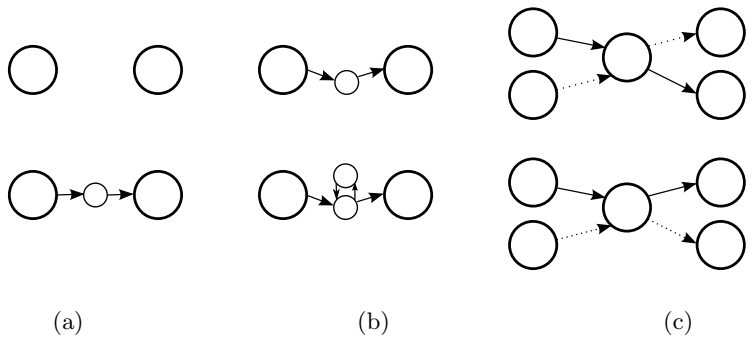

Fig. 1. Examples of proposal moves. (a) Walk extension joining two walks. (b) Local improvement by addition of a new loop. (c) Repeat interchange.

Proposals of new assemblies are created from the current assembly using the following moves:

- *Walk extension.* (Fig.1a) We start from one end of an existing walk and randomly walk through the graph, at every step uniformly choosing one of the edges outgoing from the current node. Each time we encounter the end of another walk, the two walks are considered for joining. We randomly (uniformly) decide whether we join the walks, end the current walk without joining, or continue walking.
- *Local improvement.* (Fig.1b) We optimize the part of some walk connecting two long contigs s and t. We first sample multiple random walks starting from contig s. In each walk, we only consider nodes from which contig t is reachable. Then we evaluate these random walks and choose the one that increases the likelihood the most. If the gap between contigs s and t is too big, we instead use a greedy strategy where in each step we explore multiple random extensions of the walk (of length around 200bp) and pick the one with the highest score.
- *Repeat optimization.* We optimize the copy number of short tandem repeats. We do this by removing or adding a loop to some walk. We precompute the list of all short loops (up to five nodes) in the graph and use it for adding loops.
- *Joining with advice.* We join two walks that are spanned by long reads or paired reads with long inserts. We fist select a starting walk, align all reads to the starting walk and randomly choose a read which has the other end outside the current walk. Then we find to which node this other end belongs to and join appropriate walks. If possible, we fill the gap between the two walks using the same procedure as in the local improvement move. Otherwise we introduce a gap filled with Ns.
- *Disconnecting.* We remove a path through short contigs connecting two long contigs in the same walk, resulting in two shorter walks.

– *Repeat interchange.* (Fig.1c) If a long contig has several incoming and outgoing walks, we optimize the pairing of incoming and outgoing edges. In particular, we evaluate all moves that exchange parts of two walks through this contig. If one of these changes improves the score, we accept it and repeat this step, until the score cannot be improved at this contig.

At the beginning of each annealing step, the type of the move is chosen randomly; each type of move has its own probability. We also choose randomly the contig at which we attempt to apply the move.

Note that some moves (e.g. local improvement) are very general, while other moves (e.g. joining with advice) are targeted at specific types of data. This does not contradict a general nature of our framework; it is possible to add new moves as new types of data emerge, leading to improvement when using specific data sets, while not affecting the performance when such data is unavailable.

3.2 Preprocessing and Postprocessing

To obtain the assembly graph, we use Velvet with basic error correction and unambiguous concatenation of k-mers. These settings will produce very short contigs, but will also give a much lower error rate than a regular Velvet run.

The resulting assembly obtained by the simulated annealing may contain positions with no evidence for a particular configuration of incoming and outgoing edges in the assembly graph (e.g., a repeat that is longer than the span of the longest paired read). Such arbitrary joining of walks may lead to assembly errors, since data give no indication which configuration of edges is correct. In the postprocessing step, we therefore apply the repeat interchange move at every possible location of the assembly. If the likelihood change resulting from such a move is negligible, we break corresponding walks into shorter contigs.

3.3 Fast Likelihood Evaluation

The most time consuming step in our algorithm is evaluation of the assembly likelihood, which we perform in each iteration of simulated annealing. This step involves alignment of a large number of reads to the assembly. However, we can significantly reduce required time by using the fact that only a small part of the assembly is changed in each annealing step.

To achieve this, we split walks into overlapping windows, each window containing several adjacent contigs of a walk. Windows should be as short as possible, but the adjacent windows should overlap by at least $2\ell_r$ bases, where ℓ_r is the length of the longest read. As a result, each alignment is completely contained in at least one window even in the presence of extensive indels.

We determine window boundaries by a simple greedy strategy, which starts at the first contig of a walk, and then extends the window by at least $2\ell_r$ bases beyond the boundary of the first contig. The next window always starts at the latest possible location that ensures a sufficient overlap and extends at least $2\ell_r$ bases beyond the end of the previous window.

For each window, we keep the position and edit distance of all alignments. In each annealing step, we identify which windows of the assembly were modified. We glue together overlapping windows and align reads against these sequences using a read mapping tool. Finally, we use alignments in all windows to calculate the probability of each read and combine them into the score of the whole assembly. This step requires careful implementation to ensure that we count each alignment exactly once.

To speed up read mapping even more, we use a simple prefiltering scheme, where we only align reads which contain some k-mer (usually $k = 13$) from the target sequence. In the current implementation, we store an index of all k-mers from all reads in a simple hash map. In each annealing step, we can therefore iterate over all k-mers in the target portion of the genome and retrieve reads that contain them. We use a slightly different filtering approach for PacBio reads. In particular, we take all reasonably long contigs (at least 100 bases) and align them to PacBio reads. Since BLASR can find alignments where a contig and a read overlap by only around 100 bases, we can use these alignments as a filter.

4 Experimental Evaluation

We have implemented the algorithm proposed in the previous section in a prototype assembler GAML (Genome Assembly by Maximum Likelihood). At this stage, GAML can assemble small genomes (approx. 10 Mbp) in a reasonable amount of time (approximately 4 days on a single CPU and using 50GB of memory). In future, we plan to explore efficient data structures to further speed up likelihood computation and to lower the memory requirements.

To evaluate the quality of our assembler, we have adopted the methodology of Salzberg et al. (2012) used for Genome Assembly Gold-Standard Evaluation, using metrics on scaffolds. We have used the same genomes and libraries as in Salzberg et al. (2012) (the *S. aureus* genome) and in Deshpande et al. (2013) (the *E. coli* genome); the overview of the data sets is shown in Tab.1. An additional dataset EC3 (long insert, low coverage) was simulated using the ART software (Huang et al., 2012). We have evaluated GAML in three different scenarios:

1. combination of fragment and short insert Illumina libraries (SA1, SA2),
2. combination of a fragment Illumina library and a long-read high-error-rate Pacific Biosciences library (EC1, EC2),
3. combination of a fragment Illumina library, a long-read high-error-rate Pacific Biosciences library, and a long jump Illumina library (EC1, EC2, EC3)

In each scenario, we use the short insert Illumina reads (SA1 or EC1) in Velvet with conservative settings to build the initial contigs and assembly graph. In the LAP score, we give Illumina datasets weight 1 and PacBio dataset weight 0.01.

The results are summarized in Tab.2. Note that none of the assemblers considered here can effectively run in all three of these scenarios, except for GAML.

Table 1. Properties of data sets used

ID	Source	Technology	Insert len. (bp)	Read len. (bp)	Coverage	Error rate
Staphylococus aureus (2.87Mbp)						
SA1	Salzberg et al. (2012)	Illumina	180bp	101bp	90	3%
SA2	Salzberg et al. (2012)	Illumina	3500bp	37bp	90	3%
Escherichia coli (4.64Mbp)						
EC1	Deshpande et al. (2013)	Illumina	300bp	151bp	400	0.75%
EC2	Deshpande et al. (2013)	PacBio		4000bp	30	13%
EC3	simulated	Illumina	37,000bp	75bp	0.5	4%

In the first scenario, GAML performance ranks third among zero-error assemblers in N50 length. The best N50 assembly is given by ALLPATHS-LG (Gnerre et al., 2011). A closer inspection of the assemblies indicates that GAML missed several possible joins. One such miss was caused by a 4.5 kbp repeat, while the longest insert size in this dataset is 3.5 kbp. Even though in such cases it is sometimes possible to reconstruct the correct assembly thanks to small differences in the repeated regions, the difference in likelihood between alternative repeat resolutions may be very small. Another missed join was caused by a sequence coverage gap penalized in our scoring function. Perhaps in both of these cases the manually set constants may have caused GAML to be overly conservative. Otherwise, the GAML assembly seems very similar to the one given by ALLPATHS-LG.

In the second scenario, Pacific Biosystems reads were employed instead of jump libraries. These reads pose a significant challenge due to their high error rate, but they are very useful due to their long length. Assemblers such as Cerulean (Deshpande et al., 2013) deploy special algorithms taylored to this technology. GAML, even though not explicitly tuned to handle Pacific Biosystems reads, builds an assembly with N50 size and the number of scaffolds very similar to that of Cerulean. In N50, both programs are outperformed by PacbioToCA (Koren et al., 2012), however, this is again due to a few very long repeats (approx. 5000 bp) in the reference genome which were not resolved by GAML or Cerulean. (Deshpande et al. (2013) also aim to be conservative in repeat resolution.) Note that in this case, simulated annealing failed to give the highest likelihood assembly among those that we examined, so perhaps our results can be improved by tuning the likelihood optimization.

Finally, the third scenario shows that the assembly quality can be hugely improved by including a long jump library, even if the coverage is really small (we used 0.5× coverage in this experiment). This requires a flexible genome assembler; in fact, only Celera (Myers et al., 2000) can process this data, but GAML assembly is clearly superior. We have attempted to also run ALLPATHS-LG, but the program could not process this combination of libraries. Compared to the previous scenario, GAML N50 size increased approximately 7 fold (or approx. 4 fold compared to the best N50 from the second scenario assemblies).

Table 2. Comparison of assembly accuracy in three experiments. For all assemblies, N50 values are based on the actual genome size. All misjoins were considered as errors and error-corrected values of N50 and contig sizes were obtained by breaking each contig at each error (Salzberg et al., 2012). All assemblies except for GAML and conservative Velvet were obtained from Salzberg et al. (2012) in the first experiment, and from Deshpande et al. (2013) in the second experiment.

Assembler	Number of scaffolds	Longest scaffold (kb)	Longest scaffold corr. (kb)	N50 (kb)	Err.	N50 corr. (kb)	LAP
Staphylococcus aureus, read sets SA1, SA2							
GAML	28	1191	1191	514	**0**	514	**−23.45**
Allpaths-LG	**12**	1435	**1435**	1092	**0**	**1092**	−25.02
SOAPdenovo	99	518	518	332	**0**	332	−25.03
Velvet	45	958	532	762	17	126	−25.34
Bambus2	17	1426	1426	1084	**0**	1084	−25.73
MSR-CA	17	**2411**	1343	**2414**	3	1022	−26.26
ABySS	246	125	125	34	1	28	−29.43
Cons. Velvet*	219	95	95	31	**0**	31	−30.82
SGA	456	286	286	208	1	208	−31.80
Escherichia coli, read sets EC1, EC2							
PacbioToCA	55	1533	1533	**957**	**0**	**957**	**−33.86**
GAML	29	1283	1283	653	**0**	653	−33.91
Cerulean	**21**	**1991**	**1991**	694	**0**	694	−34.18
AHA	54	477	477	213	5	194	−34.52
Cons. Velvet*	383	80	80	21	**0**	21	−36.02
Escherichia coli, read sets EC1, EC2, EC3							
GAML	**4**	**4662**	**4661**	**4662**	**3**	**4661**	**−60.38**
Celera	19	4635	2085	4635	19	2085	−61.47
Cons. Velvet*	383	80	80	21	**0**	21	−72.03

*: Velvet with conservative settings used to create the assembly graph in our method.

5 Conclusion

We have presented a new probabilistic approach to genome assembly, maximizing likelihood in a model capturing essential characteristics of individual sequencing technologies. It can be used on any combination of read datasets and can be easily adapted to other technologies arising in the future.

Our work opens several avenues for future research. First, we plan to implement more sophisticated data structures to improve running time and memory and to allow the use of our tool on larger genomes. Second, the simulated annealing procedure could be improved by optimizing probabilities of individual moves or devising new types of moves. The tool could also be easily adapted to improve existing assemblies after converting a given assembly to a set of walks. Finally, it would be interesting to explore even more detailed probabilistic models, featuring coverage biases and various sources of experimental error.

Acknowledgements. This research was funded by VEGA grants 1/1085/12 (BB) and 1/0719/14 (TV). The authors would like to thank Viraj Deshpande for sharing his research data.

References

Chaisson, M.J., Tesler, G.: Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. BMC Bioinformatics 13(1), 238 (2012)

Clark, S.C., Egan, R., Frazier, P.I., Wang, Z.: ALE: a generic assembly likelihood evaluation framework for assessing the accuracy of genome and metagenome assemblies. Bioinformatics 29(4), 435–443 (2013)

Deshpande, V., Fung, E.D.K., Pham, S., Bafna, V.: Cerulean: A hybrid assembly using high throughput short and long reads. In: Darling, A., Stoye, J. (eds.) WABI 2013. LNCS, vol. 8126, pp. 349–363. Springer, Heidelberg (2013)

Eglese, R.: Simulated annealing: a tool for operational research. European Journal of Operational Research 46(3), 271–281 (1990)

English, A.C., Richards, S., et al.: Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology. PLoS One 7(11), e47768 (2012)

Ghodsi, M., Hill, C.M., Astrovskaya, I., Lin, H., Sommer, D.D., Koren, S., Pop, M.: De novo likelihood-based measures for comparing genome assemblies. BMC Research Notes 6(1), 334 (2013)

Gnerre, S., MacCallum, I., et al.: High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Proceedings of the National Academy of Sciences 108(4), 1513–1518 (2011)

Huang, W., Li, L., Myers, J.R., Marth, G.T.: ART: a next-generation sequencing read simulator. Bioinformatics 28(4), 593–594 (2012)

Koren, S., Schatz, M.C., et al.: Hybrid error correction and de novo assembly of single-molecule sequencing reads. Nature Biotechnology 30(7), 693–700 (2012)

Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with Bowtie 2. Nature Methods 9(4), 357–359 (2012)

Medvedev, P., Brudno, M.: Maximum likelihood genome assembly. Journal of Computational Biology 16(8), 1101–1116 (2009)

Medvedev, P., Pham, S., Chaisson, M., Tesler, G., Pevzner, P.: Paired de Bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers. Journal of Computational Biology 18(11), 1625–1634 (2011)

Myers, E.W.: The fragment assembly string graph. Bioinformatics 21(suppl 2), ii79–ii85 (2005)

Myers, E.W., Sutton, G.G., et al.: A whole-genome assembly of Drosophila. Science 287(5461), 2196–2204 (2000)

Pham, S.K., Antipov, D., Sirotkin, A., Tesler, G., Pevzner, P.A., Alekseyev, M.A.: Pathset graphs: a novel approach for comprehensive utilization of paired reads in genome assembly. Journal of Computational Biology 20(4), 359–371 (2013)

Quail, M.A., Smith, M., Coupland, P., Otto, T.D., Harris, S.R., Connor, T.R., Bertoni, A., Swerdlow, H.P., Gu, Y.: A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC Genomics 13(1), 341 (2012)

Rahman, A., Pachter, L.: CGAL: computing genome assembly likelihoods. Genome Biology 14(1), R8 (2013)

Salzberg, S.L., Phillippy, A.M., et al.: GAGE: a critical evaluation of genome assemblies and assembly algorithms. Genome Research 22(3), 557–567 (2012)
Simpson, J.T., Durbin, R.: Efficient construction of an assembly string graph using the FM-index. Bioinformatics 26(12), i367–i373 (2010)
Varma, A., Ranade, A., Aluru, S.: An improved maximum likelihood formulation for accurate genome assembly. In: Computational Advances in Bio and Medical Sciences (ICCABS 2011), pp. 165–170. IEEE (2011)
Zerbino, D.R., Birney, E.: Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Research 18(5), 821–829 (2008)