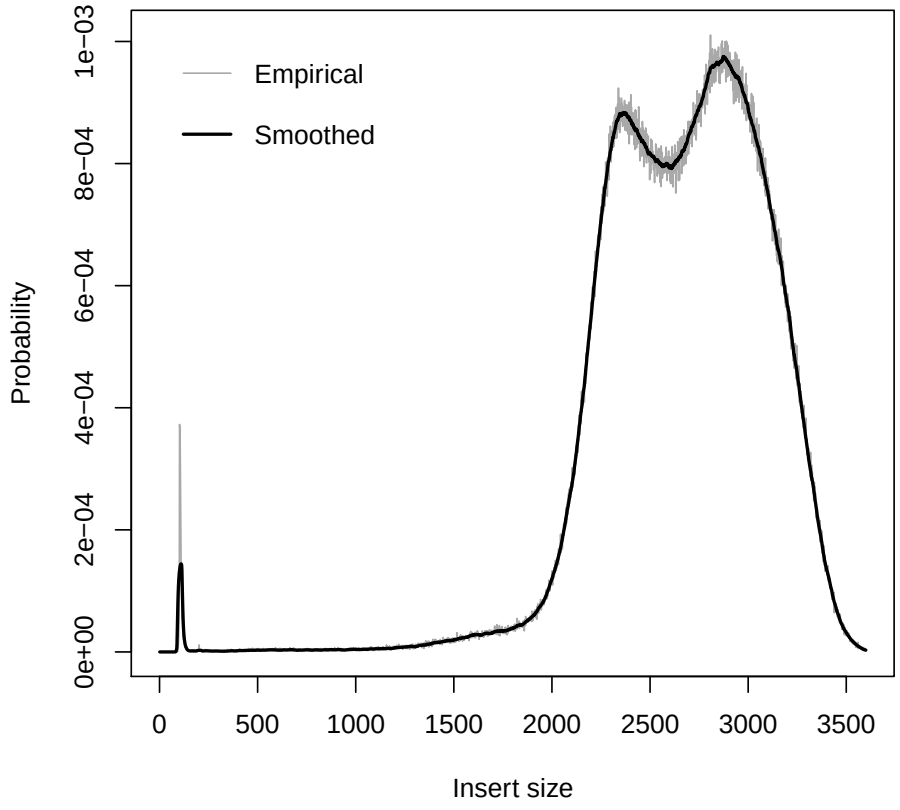
Supplementary information for SWALO: scaffolding with assembly likelihood optimization

Atif Rahman[1] and Lior Pachter[1,2]

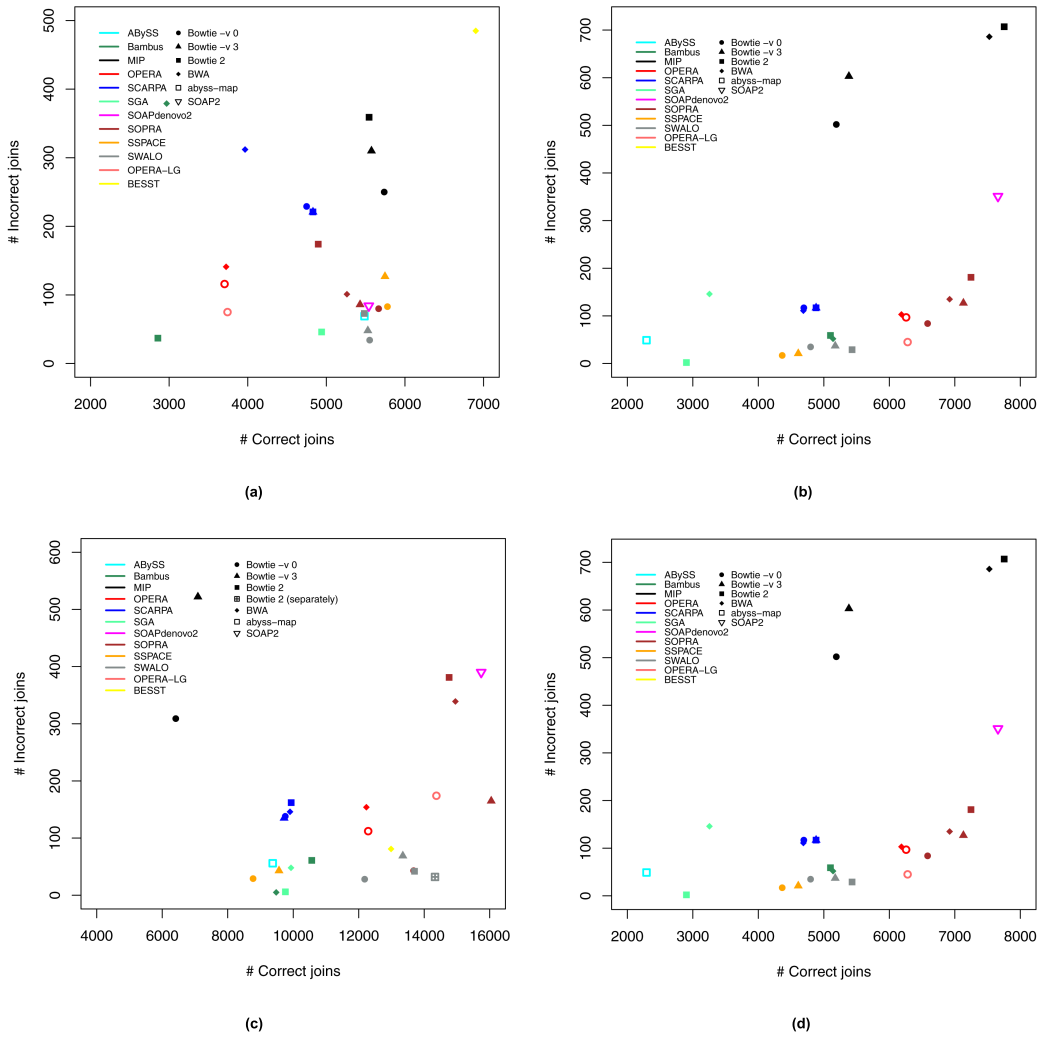
[1]Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, CA, USA
[2]Departments of Mathematics and Molecular & Cell Biology, UC Berkeley, Berkeley, CA, USA

Contents

1 Supplementary Figures



Supplementary Figure 1: **Insert size distribution for human chromosome 14 short jump library from the GAGE project.**



Supplementary Figure 2: **Performance of scaffolders.** Scatter plots showing number of correct joins vs incorrect joins made by different scaffolders on (a) *P. falciparum* short library, (b) *P. falciparum* short jump library, (c) human chromosome 14 short jump library, and (d) human chromosome 14 fosmid library. Values for all scaffolders except SWALO, Opera-LG and BESST are from [1].

2 Supplementary Tables

Supplementary Table 1: **Summary of datasets used to analyze performance of stand-alone scaffolders**

Reference	Size (Mb)	Number of contigs	Type	Number of reads (millions)	Read length	Insert size
S. aureus	2.8	28	Simulated	0.76	76	505
		28	Simulated	0.76	76	2795
		941	Simulated	0.76	76	505
		941	Simulated	0.76	76	2995
S. aureus (GAGE)	2.9	168	Real	3.5	37	3385
R. sphaeroides (GAGE)	4.6	571	Real	2.1	101	3695
P. falciparum	23.3	9303	Real	52.5	76	645
		9303	Real	12.0	75	2705
Human chromosome 14 (GAGE)	88.2	19936	Real	22.7	101	2865
		19936	Real	2.4	57–82	34500

Supplementary Table 2: **Parameters used to run aligners and SWALO** The values for maximum insert size for mapping and scaffolding used by Hunt *et al.* were used as the values for maximum insert size for mapping and minimum contig length for scaffolding respectively. When the mean and the standard deviation of the insert size distribution were needed, same values as the ones used by Hunt *et al.* were used.

Dataset	Mapping parameters (Bowtie 2 / Bowtie)	Max insert size (Bowtie2 / Bowtieconvert)	Min contig length (SWALO)	Others (SWALO)
S. aureus (100kb contigs, short inserts)	-a	650	650	-
S. aureus (100kb contigs, long inserts)	-a	4000	4000	-
S. aureus (3kb contigs, short inserts)	-a	650	650	-
S. aureus (3kb contigs, long inserts)	-a	4000	4000	-d 3000 200 (all)
S. aureus (GAGE)	-a	6000	4400	-
R. sphaeroides (GAGE)	-a	6000	4400	-d 3700 400 (bowtie -v 0, -v 3)
P. falciparum (short inserts)	-k 5	1200	1200	-
P. falciparum (long inserts)	-k 5	6000	6000	-d 3045 750 (bowtie -v 0)
Human chromosome 14 (GAGE short jump)	-k 5	6000	3600	-
Human chromosome 14 (GAGE fosmid)	-k 5	50000	40000	-c -d 35000 2000 (all)

Supplementary Table 3: **Comparison of performance of scaffolders on simulated datasets.** None of the scaffolders made any incorrect joins for 100kb contigs. Values for all scaffolders except SWALO are from [1].

Scaffolder	Aligner	100kb contigs		3kb contigs			
		500b lib	3kb lib	500b lib		3kb lib	
		%correct	%correct	%correct	%wrong	%correct	%wrong
ABySS	abyss-map	100.0	66.7	98.9	0.0	99.5	0.0
Bambus2	Bowtie 2	100.0	66.7	63.7	0.0	95.9	0.0
	BWA	48.2	48.2	59.6	0.0	96.2	0.0
MIP	Bowtie -v 0	100.0	96.3	98.9	0.0	98.4	0.0
	Bowtie -v 3	100.0	96.3	98.4	0.0	97.9	0.0
	Bowtie 2	100.0	29.6	96.3	0.5	98.7	0.5
	BWA	100.0	33.3	98.0	1.1	98.2	0.4
Opera	Bowtie	100.0	92.6	98.4	0.0	1.0	10.0
	BWA	100.0	92.6	99.8	0.2	1.2	80.0
SCARPA	Bowtie -v 0	100.0	96.3	98.9	0.0	95.0	0.0
	Bowtie -v 3	100.0	96.3	98.6	0.0	96.3	0.0
	Bowtie 2	85.2	96.3	96.8	0.0	76.3	0.7
	BWA	85.2	92.6	96.6	0.0	77.9	0.4
SGA	Bowtie 2	100.0	96.3	97.3	0.0	97.6	0.0
	BWA	100.0	92.6	99.0	0.0	96.2	0.0
SOAP2	SOAP2	96.3	96.3	98.6	0.0	99.5	0.0
SOPRA	Bowtie -v 0	100.0	96.3	98.3	0.0	98.2	0.0
	Bowtie -v 3	100.0	96.3	97.2	0.0	97.2	0.0
	Bowtie 2	74.1	100.0	91.5	0.5	85.6	0.5
	BWA	74.1	88.9	92.9	0.2	83.1	0.4
SSPACE	Bowtie -v 0	100.0	92.6	99.1	0.0	99.6	0.0
	Bowtie -v 3	100.0	92.6	98.7	0.0	99.3	0.0
SWALO	Bowtie 2	100.0	100.0	99.0	0.0	99.6	0.0
	Bowtie -v 0	100.0	100.0	99.3	0.0	99.8	0.0
	Bowtie -v 3	100.0	100.0	99.0	0.0	99.6	0.0

Supplementary Table 4: **Comparison of performance of scaffolders on *S. aureus* dataset.** One join made by Swalo (and possibly other scaffolders) labeled incorrect is a join from end to start of a circular sequence and is therefore correct.

Scaffolder	Aligner	Correct	Iincorrect	Lost	Skipped	N50	Corrected N50
ABySS	abyss-map	99	2	0	13	619764	619764
Bambus2	Bowtie 2	95	2	0	25	242814	242650
	BWA	96	0	0	27	242822	242658
MIP	Bowtie -v 0	2	0	0	1	46221	46221
	Bowtie -v 3	0	0	0	0	46221	46221
	Bowtie 2	0	0	0	0	46221	46221
	BWA	0	0	0	0	46221	46221
OPERA	Bowtie	112	11	0	22	1084108	686577
	BWA	108	15	0	21	2465956	924127
SCARPA	Bowtie -v 0	84	8	0	9	143701	113557
	Bowtie -v 3	82	10	1	9	284167	196383
	Bowtie 2	77	16	0	10	112264	112083
	BWA	78	6	0	7	112264	112083
SGA	Bowtie 2	83	1	0	10	309286	309153
	BWA	72	1	0	5	173624	173599
SOAP2	SOAP2	7	12	0	13	643384	621109
SOPRA	Bowtie -v 0	89	2	0	14	492893	230629
	Bowtie -v 3	69	3	0	13	141058	140823
	Bowtie 2	40	2	0	7	112278	112083
	BWA	42	3	0	10	108113	100779
SSPACE	Bowtie -v 0	110	9	0	13	684710	303550
	Bowtie -v 3	105	13	0	13	332784	261710
SWALO	Bowtie 2	143	1	0	10	688093	688093
	Bowtie -v 3	148	2	0	8	1088327	1088327
	Bowtie -v 0	148	1	0	9	688108	687670
OPERA-LG	Bowtie	120	5	0	22	1724129	1088544
BESST	BWA	107	4	0	13	685563	641094

Supplementary Table 5: **Comparison of performance of scaffolders on *R. sphaeroides* dataset.** Up to 3 joins made by SWALO (and possibly other scaffolders) labeled incorrect are joins from end to start of circular sequences and are therefore correct.

Scaffolder	Aligner	Correct	Iincorrect	Lost	Skipped	N50	Corrected N50
ABySS	abyss-map	384	7	0	54	280984	276804
Bambus2	Bowtie 2	329	8	0	44	146002	145952
	BWA	337	3	0	36	104405	109268
MIP	Bowtie -v 0	238	11	2	24	37651	36825
	Bowtie -v 3	299	19	7	31	46621	42898
	Bowtie 2	419	37	4	16	488095	487941
	BWA	336	15	2	31	67417	66969
OPERA	Bowtie	316	1	0	23	108182	108172
	BWA	310	2	0	20	108182	96270
SCARPA	Bowtie -v 0	214	5	0	24	38079	37830
	Bowtie -v 3	212	5	0	23	38079	37830
	Bowtie 2	209	5	0	23	37667	37581
	BWA	208	4	0	23	37667	37581
SGA	Bowtie 2	232	1	0	26	42825	42722
	BWA	199	0	0	22	33911	33886
SOAP2	SOAP2	468	8	0	26	2522483	2522482
SOPRA	Bowtie -v 0	195	2	0	28	32336	32091
	Bowtie -v 3	403	3	0	37	157931	160942
	Bowtie 2	242	15	0	24	32232	30492
	BWA	385	8	0	33	132974	132969
SSPACE	Bowtie -v 0	162	5	0	21	28242	27979
	Bowtie -v 3	357	7	0	49	109776	108410
SWALO	Bowtie 2	477	6	0	34	2530357	2528706
	Bowtie -v 3	411	5	0	33	138350	137079
	Bowtie -v 0	176	4	0	26	26249	26036
OPERA-LG	Bowtie	316	1	0	23	114113	108371
BESST	BWA	365	1	0	15	1005157	1004790

Supplementary Table 6: **Comparison of performance of scaffolders on combined *P. falciparum* datasets.**

Scaffolder	Aligner	Correct	Iincorrect	Lost	Skipped	N50	Corrected N50
ABySS	abyss-map	5724	68	0	101	6828	6529
MIP	Bowtie -v 0	8349	351	49	52	64093	52279
	Bowtie -v 3	8305	415	73	69	71752	53940
	Bowtie 2	8082	513	44	75	56672	38704
	BWA	8002	695	122	167	61960	37227
OPERA	Bowtie	6434	177	0	1171	42450	38409
	BWA	6424	200	0	1166	46821	40815
SCARPA	Bowtie -v 0	7406	368	236	256	38512	24515
	Bowtie -v 3	7514	353	235	245	37453	25372
	Bowtie 2	7336	370	237	251	36945	23951
	BWA	7416	376	226	271	39191	25128
SGA	Bowtie 2	4910	44	0	419	6606	6134
	BWA	4751	261	0	605	11332	10430
SOAP2	SOAP2	5977	228	0	254	12076	10629
SOPRA	Bowtie -v 0	8137	69	0	57	42354	40057
	Bowtie -v 3	7773	66	0	128	32939	31072
	Bowtie 2	7018	60	0	171	16366	15511
	BWA	7584	45	0	118	27483	26075
SSPACE	Bowtie -v 0	5857	82	0	38	6206	5912
	Bowtie -v 3	5889	123	0	76	6383	5982
SWALO	Bowtie 2	7550	49	0	40	20641	19637
	Bowtie -v 3	7186	33	0	34	19220	18267
	Bowtie -v 0	7499	38	0	29	14886	13971
OPERA-LG	Bowtie	6496	121	0	1177	44061	41195
BESST	BWA	7611	730	0	290	121910	35810

Supplementary Table 7: **Comparison of performance of scaffolders on combined human chromosome 14 datasets.**

Scaffolder	Aligner	Correct	Iincorrect	Lost	Skipped	N50	Corrected N50
ABySS	abyss-map	9391	42	0	3076	198501	195474
Bambus2	Bowtie 2	10282	158	0	5705	299753	99505
	BWA	10761	234	0	5623	317802	103564
MIP	Bowtie -v 0	6872	340	107	2467	22741	19172
	Bowtie -v 3	8534	696	187	3213	44372	31148
OPERA	Bowtie	12853	58	0	3409	1692782	1062031
	BWA	12737	137	0	3493	1870068	828143
SCARPA	Bowtie -v 0	10531	152	9	2410	125293	96758
	Bowtie -v 3	10477	152	8	2423	122189	96282
	Bowtie 2	10712	161	11	2376	134364	106654
	BWA	10687	153	10	2367	131726	105258
SGA	Bowtie 2	9764	3	0	3214	134574	133192
	BWA	8200	37	0	2739	82759	80860
SOAP2	SOAP2	15748	382	0	2575	561198	447849
SOPRA	Bowtie -v 0	13172	54	0	3026	167307	155422
	Bowtie -v 3	10418	238	0	3322	112239	75046
SSPACE	Bowtie -v 0	8778	26	0	2835	67970	67344
	Bowtie -v 3	9249	36	0	2677	66271	65222
SWALO	Bowtie 2	14285	61	0	2865	526942	401013
	Bowtie 2 (sep)	14872	51	0	2615	607886	470381
	Bowtie -v 3	12773	46	0	3034	411433	275558
	Bowtie -v 0	13940	94	0	3528	208240	152133
OPERA-LG	Bowtie	14824	326	0	3419	690278	336870
BESST	BWA	13252	90	0	1817	386925	317383

Supplementary Table 8: **Running time and memory usage of SWALO.**

Dataset	Memory (MB)			Runtime (seconds)		
	Bowtie -v 0	Bowtie -v 3	Bowtie 2	Bowtie -v 0	Bowtie -v 3	Bowtie
S. aureus	37	41	40	51	82	87
R. sphaeroides	75	75	75	64	82	153
P. falciparum (combined)	235	276	331	1077	1256	1601
Human chromosome 14 (combined)	234	278	437	2004	2849	4162

3 Supplementary Notes

3.1 Likelihood of an assembly

To compute the likelihood of an assembly we use the model and parameter estimation approach presented in [2]. We briefly describe the model for clarity of presentation.

A generative model for sequencing

Suppose, N paired end reads, $\mathcal{R} = \{r_1, r_2 \dots r_N\}$ are generated from a genome, $\mathcal{G}$. A fragment represented by two paired-end reads $r_i = (r_{i1}, r_{i2})$ is generated according to the following model:

- An insert size, l_i is chosen according to a distribution, F.
- A site for the $5'$ end of the fragment (start site), s_i is chosen according to a distribution, S.
- The ends of the fragment are read as r_{i1} and r_{i2} according to an error model, E where E comprises mismatches as well as indels.

The generative model is illustrated in Figure S1.

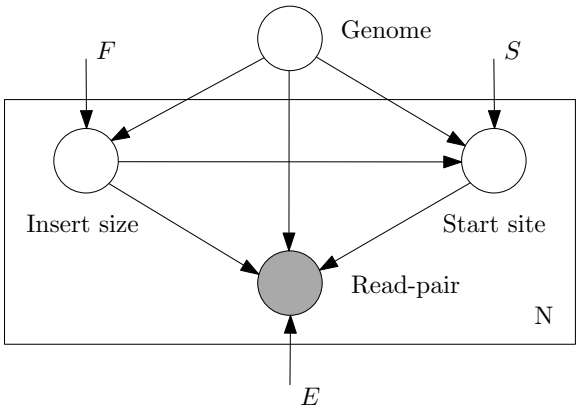


Figure S1: **A generative graphical model for sequencing.** N paired end reads are generated independently from a genome. Here, F denotes the distribution of insert sizes, S is the distribution of start sites of reads and E stands for error parameters.

Computing likelihood

The likelihood of an assembly *i.e.* the probability of the set of reads can be computed with respect to a proposed assembly using the model described previously.

The probability of a sequence of length L generating a paired-end read r_i is

$$p(r_i) = \sum_{l=1}^{L} p_F(l) \sum_{s=1}^{L-l+1} p_S(s) \sum_{e \in \mathcal{E}} p_E(r_i | a_s \dots a_{s+l-1})$$

where $a_s \dots a_{s+l-1}$ is the assembly subsequence starting at s of length l, $\mathcal{E}$ denotes all possible ways of obtaining r_i from $a_s \dots a_{s+l-1}$ and

$$\begin{aligned} p_F(l) &= \text{Probability that the fragment length is } l, \\ p_S(s) &= \text{Probability that the } 5' \text{ end of the fragment is at site } s, \\ p_E(r_i|a_s \dots a_{s+l-1}) &= \text{Probability of obtaining } r_i \text{ from } a_s \dots a_{s+l-1} \\ &\quad \text{with sequencing errors given by } e. \end{aligned}$$

Although in theory a read could have been generated from any site (assuming that every base could have been an error), possible mappings with large number of errors can be ignored and the probability $p(r_i)$ can be approximated by mapping the read to the assembly using a short read aligner. If M_i is the number of such mappings of read r_i, the probability is given by

$$p(r_i) \approx \sum_{j=1}^{M_i} p_F(l_{i,j}) p_S(s_{i,j}) p_E(r_i|a_{i,j})$$

where $l_{i,j}$, $s_{i,j}$, $a_{i,j}$ and $e_{i,j}$ are fragment length, start site, assembly subsequence and errors corresponding to j-th mapping of i-th read respectively. The above equation generalizes to assemblies with more than one contig. Given an assembly $\mathcal{A}$ and a set of reads $\mathcal{R} = \{r_1, r_2 \dots r_N\}$, assuming reads are generated independently the *log likelihood* is given by

$$\begin{aligned} l(\mathcal{A}; \mathcal{R}) &= \log \prod_{i=1}^{N} p(r_i|\mathcal{A}) \\ &\approx \sum_{i=1}^{N} \log \sum_{j=1}^{M_i} p_F(l_{i,j}) p_S(s_{i,j}) p_E(r_i|a_{i,j}). \end{aligned}$$

The model requires that $M_i \geq 1$ for all i. However, the tools available for mapping reads are usually not able to map all the reads. Therefore, a random subset of the unmapped reads are aligned using the Smith-Waterman algorithm and are used to assign probabilities to unmapped reads.

Learning distributions

We use distributions and approaches to learn them similar to [2] with some modifications.

- **Insert size distribution:** We use the empirical distribution as normal distribution is not always a good approximation. The distribution is learned using contigs longer than a minimum length (provided by the user) so that the distribution is not biased. We use a smoothed version of the distribution using a window size of 25 as the number of reads to learn the distribution from may not be very high for some datasets. We also compute the mean, μ as well as the left-sided and right-sided standard deviations, σ_l and σ_r respectively and truncate the distribution at $\mu - 2.5 * \sigma_l$ and $\mu + 2.5 * \sigma_r$ to avoid linking contigs based on chimeric reads.

In situations where the number of read-pairs mapped to contigs long enough to learn distribution is very low (less than 100,000), we recommend switching to normal distribution and require the user to input a mean and a standard deviation.

- **Distribution of fragments:** We ignore sequencing bias such as GC content bias and assume all sites have the same probability. The probability that an insert of length l starts at s is

$$\begin{aligned} p_S(s) &= \frac{1}{\tilde{T}(l)} \\ &= \frac{1}{\sum_{c \in \{contigs\}} (l_c - l + 1)} \end{aligned}$$

  where $\tilde{T}(l)$ is the total effective length *i.e.* number of possible start sites for insert size l and l_c is the length of contig c.

- **Error Model:** The error model described in [2] is used and learned using reads that map uniquely to contigs. The model includes substitutions and insertions-deletions and takes into account differences in error rates across positions in reads.

3.2 The scaffold graph

The next phase of our algorithm is to construct the *scaffold graph*. The *scaffold graph* is a weighted bidirected graph [3] where there is a vertex for each contig. There is an edge between two contigs if joining them would result in an increase in likelihood of the assembly. The edges are bidirected as each contig in the pair may correspond to one of the two DNA strands resulting in four possible orientations of the pair of contigs. There may be more than one edge between two contigs provided the edges are of different types. The edge weights correspond to the increase in likelihood achieved if the contigs were to be joined. We also have a gap estimate between two contigs associated with each edge. Computing the edge weights is done in following two steps.

- Estimate the gaps between pairs of contigs using maximum likelihood. Gaps may be negative if contigs overlap.
- Compute the change in assembly likelihood if contigs are linked with maximum likelihood gap estimates. This constitutes computing probability of linking reads as well as adjusting probabilities of all other reads.

Estimating gaps using maximum likelihood

The gaps between contigs are estimated using a generative model similar to the one used for computing likelihood of assembly. However, we modify the generative model to correct for the issue that we may not observe inserts from the entire distribution due to gaps between contigs [4] and lengths of contigs.

Consider contigs A and B separated by a gap g in Figure S2. If the $5'$ end of an insert is at s, then we will not observe inserts smaller than l_{min} and greater than l_{max} where l_r is the length of the second read of the pair. The probability of a read is then given by

$$p(r) \approx \sum p'_S(s) p'_F(l) p_E(r|a)$$

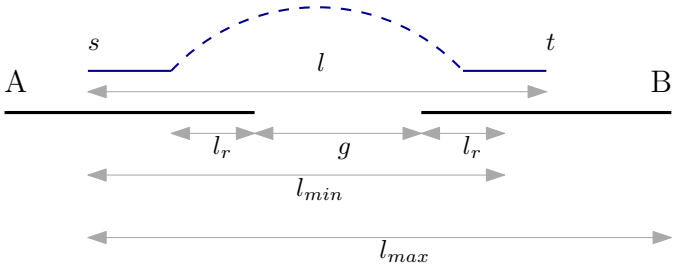


Figure S2: **Gap estimation.** Figure illustrates that inserts smaller than l_{min} and greater than l_{max} will not be observed due to lengths of contigs A and B and the gap g between them.

where p'_F and p'_S are the corrected insert size and start site distributions respectively given by

$$\begin{aligned} p'_F(l) &= \frac{p_F(l)}{\sum_{k=l_{min}}^{l_{max}} p_F(k)}, \\ p'_S(s) &= \frac{\text{p\{fragment starting at s and ending in B\}}}{\text{p\{fragment starting in A and ending in B\}}}. \end{aligned}$$

While p'_F can be efficiently pre-computed, computation of p'_S is time consuming. So we make the following approximation.

$$p'_S(s) \approx \frac{\text{p\{fragment starting at s and ending at t\}}}{\text{p\{fragment starting in A and ending at t\}}}.$$

We then find the gap, g that maximizes likelihood of linking reads, $\max_g \prod_{r\in\{linking\}} p(r)$ for every pair of contigs with read pairs linking them.

Resolving multi-mapped pairs using the EM algorithm

Application of a probabilistic model allows us to resolve read pairs with multiple mappings using the *expectation maximization* (EM) algorithm [5, 6] which are ignored in most other scaffolders. In the initialization step, we probabilistically assign read pairs to contig pairs using only the error probabilities and obtain gap estimates as discussed above. We then iterate the two steps of the EM algorithm.

- *E-step:* Determine the expected assignments of read pairs to contig pairs according to the generative model with current gap estimates.
- *M-step:* Find maximum likelihood estimates of gaps using the current assignments.

In early iterations when the gap estimates are inaccurate, we use relaxed cut-offs for insert size distribution and divide by two in each iteration to converge towards the cut-off points determined by standard deviations. In every iteration, the probabilities of reads beyond cut-off points are reduced by a constant factor.

Computing edge weights

We then approximate the change in assembly likelihood if two contigs are joined with the MLE of gap. The likelihood of linking reads can be computed using the generative model for sequencing (Figure S1). However, we also need to adjust probabilities of all other reads since the effective length changes. Recall that the likelihood of an assembly is given by

$$l(\mathcal{A};\mathcal{R}) \approx \sum_{i=1}^{N} \log \sum_{j=1}^{M_i} p_F(l_{i,j}) \frac{1}{\tilde{T}(l_{i,j})} p_E(r_i|a_{i,j}).$$

Adjusting the probabilities using the above equation would require iterating over of all multi-mapped reads. In order to make this step practical we make the following approximation.

$$\begin{aligned} l(\mathcal{A};\mathcal{R}) &\approx \sum_{i=1}^{N} \log \frac{1}{\tilde{T}(\hat{l_i})} \sum_{j=1}^{M_i} p_F(l_{i,j}) p_E(r_i|a_{i,j}) \\ &= \sum_{i=1}^{N} \log \frac{1}{\tilde{T}(\hat{l_i})} + \sum_{i=1}^{N} \log \sum_{j=1}^{M_i} p_F(l_{i,j}) p_E(r_i|a_{i,j}) \end{aligned}$$

where $\hat{l_i}$ is insert size corresponding to most probable mapping of read i. This allows us to count number of reads corresponding to a particular insert size and efficiently calculate the new likelihood. If $n_{\hat{l}}$ is the number of reads with insert size $\hat{l}$, then

$$l_{new}(\mathcal{A};\mathcal{R}) \approx l_{old}(\mathcal{A};\mathcal{R}) - \sum_{\hat{l}} n_{\hat{l}} \left(\log \frac{1}{\tilde{T}_{old}(\hat{l})} - \log \frac{1}{\tilde{T}_{new}(\hat{l})} \right)$$

where $\tilde{T}_{new}(l) = \tilde{T}_{old}(l) + l - 1 + g$ is the new effective length if both contigs are sufficiently large compared to insert sizes and can be precomputed for various gap sizes. The adjustments for cases where contigs are small can also be precomputed.

The likelihood of linking reads and likelihood adjustments of all other reads are combined to estimate the edge weight. We retain the edge and assign it the computed weight if it is positive and delete the edge otherwise.

3.3 Selecting joins

Once the scaffold graph is constructed, one approach might be to select the set of edges that maximizes likelihood. Analogous problems have been proved to be NP-hard [7] but fixed parameter tractable algorithms are known [8]. However, this may lead to incorrect joins if there are repeats longer than the insert sizes. So, we use the following heuristic instead.

i Make unambiguous joins *i.e.* join contigs connected by an edge such that one has outdegree one and the other has indegree one. Then compute the standard deviation, σ_L of the likelihoods of the joins made.

ii Sort other candidate joins in decreasing order of likelihood.

iii Join contigs if likelihood, l of the join is $\alpha(l) = \max(5e^{\lambda l}, 2)$ times greater or $2.5\sigma_L$ more than all other conflicting joins where $\lambda = -\frac{\ln(2/5)}{5\sigma_L}$.

For the remainder of this section we say two joins are inconsistent or in conflict with each other if none of the contigs with the estimated gap fits into the gap between the other two times a stretch factor of 1.1 and neither has increase in likelihood higher than the other determined by the rule above.

The intuition behind the heuristic is to learn a distribution of likelihoods of unambiguous joins and make a join if there is no inconsistent join within two and a half standard deviations of it. But this leads to missing joins with increase in likelihood less than $2.5\sigma_L$ if there are other conflicting joins. So, we add a factor determined by an exponential decay function, α with $\alpha(0) = 5$ and $\alpha(l) = 2$ for $l > 5\sigma_L$.

Following exceptions are however made based on practical observations.

- Similar to some other scaffolders [9, 8], we compute the mean sequencing depth per base of contigs and the right sided standard deviation and avoid joining any contig with sequencing depth two standard deviations more than the mean and more than 1.5 times the mean.
- Gap estimates based on only one linking read pair are often inaccurate and lead to incorrect joins. So, we do not join contigs linked by a single read pair.

Selecting from multiple consistent joins

In many cases although there are multiple outgoing or incoming edges from or to a contig, one or more contigs fit into the gap between the contigs to be joined. In these cases more than one join can be made and we use the following algorithm.

i Find all contigs with edges from the two contigs currently being joined that fit into interval determined by the estimated gap times a stretch factor.

ii Remove contigs with conflicting edges to other contigs.

iii Select consistent set of contigs that optimizes likelihood using the dynamic programming algorithm for Weighted interval scheduling.

iv Remove selected contigs inconsistent with not selected ones.

In actual implementation there may be two possible positions for each contig within the interval. This requires solving the Discrete weighted interval scheduling problem. However, the problem is known to be NP-hard [10, 11] and we find that the two possible positions almost always overlap. So, we run the Weighted interval scheduling algorithm and remove one in the rare case both were selected.

Edge propagation

When we merge two contigs, if there are edges to contigs with gap estimates large enough so that the contig being joined to along with the gap can fit in there, we propagate the edge from the contigs to the meta-node corresponding to the merged contigs.

3.4 Special cases

Under usual circumstances, SWALO takes as input a single parameter denoting the minimum size of contigs to be used to learn the insert size distribution. But in some cases discussed below, special modes are recommended.

Inadequate number of inserts to learn the distribution:

If the contigs are small compared to insert sizes or if the number of inserts mapping concordantly to contigs is low (less than 100,000), the insert size distribution cannot be learned properly. So, we recommend that the user switches to Gaussian distribution and provide a mean and a standard deviation as inputs.

Insert size standard deviation very high:

If the insert size standard deviation is very high (greater than 1000), then there are large errors in gap estimates. So, we switch to a conservative mode where only uniquely mapped reads are used and multiple consistent joining step is skipped.

Multiple insert size libraries:

There are two approaches for multiple insert size libraries. The first is to build the scaffold graph separately and then combine the graphs before actually merging contigs. The other approach is to do the scaffolding hierarchically. We recommend the first approach unless some insert libraries have large standard deviations in which case we recommend scaffolding using all other libraries using the first approach and then scaffold using libraries with high standard deviation together using the conservative mode. In future, we plan to modify the implementation to process multiple libraries together.

3.5 Implementation

The methods have been implemented in a tool called 'scaffolding with assembly likelihood optimization (SWALO)' using C/C++. The gap estimation phase (the M-step of the EM algorithm) is mutli-threaded for speed-up in computation.

3.6 Software versions

The following software were used for mapping reads and scaffolding:

- Bowtie v1.1.2
- Bowtie2 v2.2.5
- SWALO-0.9.1 - SWALO-0.9.8 (differs only in time and memory usage)
- OPERA-LG_v2.0.4
- BESST V2.2

The results in [1] were generated using following scaffolders:

- AbySS 1.3.6
- Bambus2 3.1.0
- MIP 0.5
- OPERA_v1.2
- SCARPA 0.22
- SGA-0.9.4.3
- SOAPdenovo2 r 223
- SOPRA 1.4.6
- SSPACE 2 (basic)

References

[1] Martin Hunt, Chris Newbold, Matthew Berriman, and Thomas Otto. A comprehensive evaluation of assembly scaffolding tools. *Genome Biology*, 15(3):R42, 2014.

[2] Atif Rahman and Lior Pachter. CGAL: computing genome assembly likelihoods. *Genome Biology*, 14(1):R8, 2013.

[3] Jack Edmonds and EllisL. Johnson. Matching: A well-solved class of integer linear programs. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization — Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 27–30. Springer Berlin Heidelberg, 2003.

[4] Jarrod A. Chapman, Isaac Ho, Sirisha Sunkara, Shujun Luo, Gary P. Schroth, and Daniel S. Rokhsar. Meraculous: *De Novo* genome assembly with short paired-end reads. *PLoS ONE*, 6(8):e23501, 08 2011.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38, 1977.

[6] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J Van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

[7] Daniel H. Huson, Knut Reinert, and Eugene W. Myers. The greedy path-merging algorithm for contig scaffolding. *J. ACM*, 49(5):603–615, September 2002.

[8] S Gao, W-K Sung, and N Nagarajan. Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *J Comput Biol*, 18:1681–1691, 2011.

[9] A Dayarian, TP Michael, and AM Sengupta. SOPRA: scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11:345, 2010.

[10] Katsuto Nakajima, S. Louis Hakimi, and Jan Karel Lenstra. Complexity results for scheduling tasks in fixed intervals on two types of machines. *SIAM Journal on Computing*, 11(3):512–520, 1982.

[11] Antoon WJ Kolen, Jan Karel Lenstra, Christos H Papadimitriou, and Frits CR Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.