Hybrid correction of highly noisy long reads using a variable-order de Bruijn graph

Supplementary material

Pierre Morisse[1], Thierry Lecroq[1] and Arnaud Lefebvre[1]

[1]Normandie Univ, UNIROUEN, LITIS, 76000 Rouen, France

```
   Input    : $s$ string, $k$ integer, $b$ boolean, $K$ integer
   Output   : $E$, the edges of the node representing the $k$-mer $s$ in the
              de Brujin graph of order $k$. If $b = 0$, the graph is
              traversed forward, otherwise, it is traversed backward.
   Auxiliary: $occs$ (integer, integer) set, $i$ integer, $id$ integer, $pos$
              integer
 1 begin
 2     $E \leftarrow \emptyset$
 3     $occs \leftarrow \emptyset$
 4     if $b = 0$ then
 5         $occs \leftarrow getOccurrencesPositions(s[1..k-1])$
 6     else
 7         $occs \leftarrow getOccurrencesPositions(s[0..k-2])$
 8     $i \leftarrow 0$
 9     while $i <$ $size(occs)$ and $size(E) < 4$ do
10         $(id, pos) \leftarrow occs[i]$
11         if $b = 0$ and $pos + k \leq K$ then
12             $E \leftarrow E \cup \{(s, \text{getKmer}(id)[pos..pos+k-1])\}$
13         else if $b = 1$ and $pos > 0$ then
14             $E \leftarrow E \cup \{(s, \text{getKmer}(id)[pos-1..pos+k-2])\}$
15         $i \leftarrow i + 1$
       return   : $E$
```

Algorithm S1: Retrieve the edges of a given node. getOccurrencesPositions and getKmer are PgSA functions that allow respectively to retrieve the occurrences positions of the given string in the set of K-mers, and to retrieve the sequence corresponding to the K-mer of identifier *id*. Line 2: Start with an empty set of edges. Lines 3-7: If traversing the graph forward, get the occurrences positions of the suffix of s in the set of K-mers, if traversing it backward, get the occurrences positions of its prefix. Lines 8-15: Process the list of occurrences positions. The processing stops when all the occurrences have been processed or when 4 edges have been found, as we work with the DNA alphabet and cannot find more than 4 edges per node. Lines 11-12: If traversing forward and if the position component does not represent the suffix of length $k-1$ of the K- mer of identifier *id*, add an edge to the k-mer starting at position *pos* in this K-mer. Lines 13-14: If traversing backward and if the position component does not represent the prefix of length $k-1$ of the K-mer of identifier *id*, add an edge to the k-mer starting at position $pos - 1$ in this K-mer.

Input : S (string, integer, integer, integer, string) array, $minOverlap$ integer
Output : The set of seeds, after merging overlapping seeds
Auxiliary: i integer, j integer, $s1$ string, $s2$ string, $suffLen$ integer, $suffSeq$ string, $suffScore$ integer

```
begin
    $sortByPosition(S)$
    $i \leftarrow 0$
    $j \leftarrow 1$
    while $i < size(S) - 1$ and $j < size(S)$ do
        if $S[j].pos \leq S[i].pos + S[i].len$ then
            $s1 \leftarrow S[i].seq[S[j].pos - S[i].pos..S[i].len - 1]$
            $s2 \leftarrow S[j].seq[0..len(s1) - 1]$
            if $S[j].pos + S[j].len > S[i].pos + S[i].len$ and
             $length(s1) \geq minOverlap$ and $s1 = s2$ then
                $suffLen \leftarrow S[j].pos + S[j].len - S[i].pos - S[i].len$
                $suffSeq \leftarrow S[j].seq[S[j].len - suffLen..S[j].len - 1]$
                $suffScore \leftarrow (S[j].score/S[j].len) \times suffLen$
                $S[i].seq \leftarrow S[i].seq + suffSeq$
                $S[i].len \leftarrow S[i].len + suffLen$
                $S[i].score \leftarrow S[i].score + suffScore$
                $delete(S[j])$
            else if $S[i].score < S[j].score$ then
                $delete(S[j])$
            else
                $delete(S[i])$
        else
            $i \leftarrow j$
            $j \leftarrow j + 1$
```

Algorithm S2: Merge seeds with overlapping alignment positions.

```
Input     : $S$ (string, integer, integer, integer, string) array,
            $maxDistance$ integer, $minOverlap$ integer
Output    : The set of seeds, after merging overlapping seeds
Auxiliary: $i$ integer, $j$ integer, $overlap$ integer, $suffLen$ integer
begin
    $sortByPosition(S)$
    $i \leftarrow 0$
    $j \leftarrow 1$
    while $i < size(S) - 1$ and $j < size(S)$ do
        if $S[j].pos - S[i].pos - S[i].len \leq maxDistance$ then
            $overlap \leftarrow overlapLength(S[i].seq, S[j].seq)$
            if $overlap \geq minOverlap$ then
                $suffLen \leftarrow S[j].len - overlap$
                $suffSeq\ S[j].seq[overlap..S[j].len - 1]$
                $suffScore \leftarrow (S[j].score/S[j].len)\times\ suffLen$
                $S[i].seq \leftarrow S[i].seq+\ suffSeq$
                $S[i].len \leftarrow S[i].len+\ suffLen$
                $S[i].score \leftarrow S[i].score+\ suffScore$
                $delete(S[j])$
            else
                $i \leftarrow j$
                $j \leftarrow j + 1$
        else
            $i \leftarrow j$
            $j \leftarrow j + 1$
```

Algorithm S3: Merge consecutive seeds with close alignment positions that overlap.

Description of Algorithm S2

Line 2: Sort the seeds in ascending order of their alignment start position. Lines 3-4: Begin with the two first seeds. Line 5-23: Keep processing while some seeds remain. Line 6-20: The seeds have overlapping alignment positions, attempt to merge them. Lines 7-8: Retrieve the overlapping sequences from the seeds that should coincide. Line 9-16: If the j^{th} seed can extend the i^{th} seed, if the seeds overlap over a sufficient length and if their overlapping sequences do coincide, merge the seeds. Line 10-12: Get the length, sequence and score of the non-overlapping suffix of the j^{th} seed. We define the suffix score as the average score of a base times the length of suffix. Lines 13-15: Actually merge the seeds. Append the non-overlapping sequence of the j^{th} seed to the sequence of the i^{th} seed, and update the alignment length and score of the i^{th} seed accordingly. Line 16: The i^{th} and j^{th} seeds have been merged, remove the j^{th} seed from the array. Lines 17-20: The seeds cannot be merged, only keep the one with the best alignment score. Lines 21-23: The seeds do not have overlapping alignment positions, move on to the next ones.

Description of Algorithm S3

Line 2: Sort the seeds in ascending order of their alignment start position. Lines 3-4: Begin with the two first seeds. Line 5-21: Keep processing while some seeds remain. Line 6-18: The seeds have close alignment positions, attempt to merge them. Line 7: Compute then length of the prefix-suffix overlap between the seeds. Lines 8-15: The overlap between the seeds is long enough, merge them. Line 9-11: Get the length, sequence and score of the non-overlapping suffix of the j^{th} seed. Again, we define the suffix score as the average score of a base times the length of suffix. Lines 12-14: Actually merge the seeds. Append the non-overlapping sequence of the j^{th} seed to the sequence of the i^{th} seed, and update the alignment length and score of the i^{th} seed accordingly. Line 15: The i^{th} and j^{th} seeds have been merged, remove the j^{th} seed from the array. Lines 16-18: The overlap between the seeds is too short, and the seeds cannot be merged. Move on to the next seeds. Lines 19-21: The seeds do not have close alignment positions, move on to the next ones.

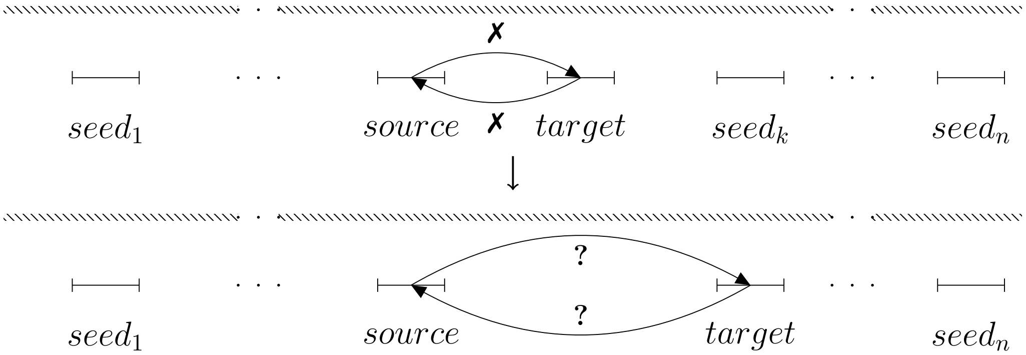

Figure S1: Illustration of the process of skipping a seed. Hatched lines represent the long read and segments represent the seeds. Top: No path allowing to link *source* and *target* together has been found. *target* is thus considered as erroneous and is ignored. Bottom: *target* is redefined as *seed*$_k$, whereas *source* remains unchanged. A new linking iteration is then performed between these two seeds.

Dataset	*A. baylyi*	*E. coli*	*S. cerevisiae*	*C. elegans*
Reference organism				
Strain	ADP1	K-12 substr. MG1655	W303	Bristol N2
Reference sequence	CR543861	NC_000913	scf7180000000{084-13}	GCA_000002985.3
Genome size	3.6 Mbp	4.6 Mbp	12.2 Mbp	100 Mbp
Simulated Pacific Biosciences data				
Number of reads	8,765	11,306	30,132	_
Average length	8,202	8,226	8,204	_
Number of bases	72 Mbp	93 Mbp	247 Mbp	_
Coverage	20x	20x	20x	_
Real Oxford Nanopore data				
Accession number	ERR77685{1-5} Genoscope[1]	Genoscope[2] Sequences from Loman Lab	Genoscope[3] Sequences from Schatz Lab	ERR1802061[4]
Number of reads	89,011	22,270	205,923	363,500
Average length	4,284	5,999	5,698	5,524
Number of bases	381 Mbp	134 Mbp	1,173 Mbp	2,008 Mbp
Coverage	106x	29x	95x	20x
Illumina data				
Accession number	ERR788913[4]	Genoscope[5] Sequences from Loman Lab	Genoscope[6] Sequences from Schatz Lab	ART
Number of reads	900,000	775,500	2,500,000	20,057,100
Read length	250	300	250	250
Number of bases	224 Mbp	232 Mbp	625 Mbp	5,000 Mbp
Coverage	50x	50x	50x	50x

Table S1: Description of the data used in the experiments.

[1] http://www.genoscope.cns.fr/externe/nas/datasets/MinION/acineto/, reads from run6.

[2] http://www.genoscope.cns.fr/externe/nas/datasets/MinION/ecoli/

[3] http://www.genoscope.cns.fr/externe/nas/datasets/MinION/yeast/

[4] Only a subset of the data was used.

[5] http://www.genoscope.cns.fr/externe/nas/datasets/Illumina/ecoli/

[6] http://www.genoscope.cns.fr/externe/nas/datasets/Illumina/yeast/

Method	Original	CoLoRMap	HALC	HG-CoLoR	Jabba	LoRDEC	Nanocorr	NaS	Canu	Daccord	LoRMA
S. cerevisiae											
Error rate (%)	66.6190	8.2290	8.3101	**8.1013**	8.3748	10.5821	8.3165	8.9441	15.3633	8.1599	41.1934
Throughput (Mbp)	247.2	22.2	23.1	23.4	20.1	23.2	23.6	17.8	21.9	**23.8**	1.6
Deletions (%)	3.8998	0.8258	0.8479	0.8130	**0.8094**	1.0236	0.8595	0.8403	2.1142	0.8551	3.4136
Insertions (%)	13.990	4.7652	4.811	**4.7537**	4.8851	6.1702	4.9694	5.5993	10.3363	4.9015	23.6213
Substitutions (%)	57.9724	3.7327	3.7514	3.6307	3.7767	4.6795	3.5894	3.7016	4.1568	**3.5049**	16.3801
Split reads (%)	N/A	13.31	18.20	3.67	13.66	68.53	**0**	**0**	**0**	1.30	60.59
Runtime	N/A	4h42min	1h19min	4h32min	**5min**	28min	39h52min	217h20min	30min	1h15min	20min

Table S2: Statistics of the long reads after correction with the different methods, as reported by LRCStats, on the *S. cerevisiae* daset. The best result for each statistic is highlighted. These results are however erroneous, as LRCStats reported an error rate of 66%, whereas parsing the files generated by SimLord reported an error rate of 18.6%, comparable to that of the other simulated datasets.

Method	Original	Nanocorr	CoLoRMap	LoRDEC	HALC	LoRMA
A. baylyi						
Number of reads	89,011	24,105	17,380	22,288	35,099	17,984
Split reads (%)	N/A	0	43.63	45.08	13.70	89.38
Average length	4,284	7,205	3,883	3,449	4,498	229
Number of bases (Mbp)	381	174	141	175	190	76
Average identity (%)	70.09	91.95	99.32	99.80	96.62	99.58
Genome coverage (%)	100	100	100	100	100	66.52
Runtime	N/A	22h28min	3h41min	16min	47h41min	29min
E. coli						
Number of reads	22,270	21,764	20,161	21,983	22,215	14,569
Split reads (%)	N/A	0	19.28	26.60	7.97	84.78
Average length	5,999	5,899	4,475	4,135	8,409	165
Number of bases (Mbp)	134	128	115	125	131	18
Average identity (%)	79.46	95.80	99.30	99.83	99.36	99.61
Genome coverage (%)	100	100	100	100	100	25.07
Runtime	N/A	5h48min	2h01min	13min	2h14min	12min
S. cerevisiae						
Number of reads	205,923	66,953	39,088	59,075	89,860	14,856
Split reads (%)	N/A	0	45.02	75.03	28.04	55.34
Average length	5,698	3,455	2,294	1,126	1,893	230
Number of bases (Mbp)	1,173	231	165	221	256	11
Average identity (%)	55.49	87.10	99.45	98.45	98.45	95.93
Genome coverage (%)	99.90	99.59	99.09	98.87	99.13	3.80
Runtime	N/A	158h53min	10h44min	1h09min	2h56min	1h36min
C. elegans						
Number of reads	363,500	_	135,544	50,448	_	10,109
Split reads (%)	N/A	_	20,68	62.99	_	66.87
Average length	5,524	_	2,273	1,322	_	270
Number of bases (Mbp)	2,008	_	419	222	_	10
Average identity (%)	71.07	_	98.11	96.63	_	97.76
Genome coverage (%)	99.99	_	96.37	85.20	_	1.71
Runtime	N/A	_	91h17min	1h01min	_	1h13min

Table S3: Statistics of the long reads, before and after correction by the different methods excluded from the main comparison.

Similarly to the experiments on simulated data, LoRMA also performed the worst on real data. Its throughput was the smallest among all the tools, the corrected reads it output displayed a size closer to that of short reads, and covered all the reference genomes very poorly. Nanocorr was the slowest among all the tools, behind NaS, except on the *A. baylyi* dataset, where HALC was the slowest. In addition, Nanocorr did not manage to produce corrected long reads of good quality. Indeed, the lowest error rate it managed to reach was still of more that 4%, on the *E. coli* dataset. On average, the error rate of the output long reads was comparable to, or even worse than what self-correction methods achieved. The long reads, however, covered the reference genomes very well. On the larger *C. elegans* dataset, Nanocorr was not run due to its large runtimes. On all the datasets but *C. elegans*, on which it failed to perform correction because of an internal error of LoRDEC, HALC produced the greatest number of corrected long reads. However, its throughput was not proportional to that number, as, despite correcting more long reads, it actually output less bases than HG-CoLoR, up to two times less on the *S. cerevisiae* dataset. Moreover, its runtimes were quite unpredictable, as it took near to two days to correct the *A. baylyi* dataset, and less than 3 hours to correct the *S. cerevisiae* dataset, despite the former being three times smaller than the latter. As observed during the experiments on simulated data, despite being fast, LoRDEC did once again split an important proportion of long reads, as high a 75% on the *S. cerevisiae* dataset. As a result, even though they aligned with a high identity, except on *C. elegans* to which LoRDEC did not scale, the long reads corrected with LoRDEC displayed the shortest average length among all the other long reads but those corrected with LoRMA, reaching less than 20% of the length of the original long reads on *S. cerevisiae.* Finally, CoLoRMap did also split a lot of long reads, and thus output corrected long reads of much shorter length that the original long reads. The throughput of CoLoRMap was also smaller than that of all the other methods, except for LoRDEC on the *C. elegans* dataset. Compared to HG-CoLoR, which has a bit shorter, but comparable runtimes, CoLoRMap performed worse on every studied statistic.

Method	Nanocorr	CoLoRMap	LoRDEC	HALC	LoRMA
A. baylyi					
Long reads coverage	48x	39x	49x	53x	21x
Number of contigs	1	2	1	1	7
NG50	3,571,959	3,627,107	3,620,390	3,598,721	_
Genome coverage (%)	98.59	100	100	99.92	0.56
Identity (%)	99.98	99.99	99.99	99.98	97.81
E. coli					
Long reads coverage	28x	25x	27x	28x	4x
Number of contigs	13	1	1	2	1
NG50	824,971	4,642,509	4,649,617	4,650,960	_
Genome coverage (%)	98.61	100	100	99.99	0.59
Identity (%)	99.98	99.99	99.98	99.97	97.96
S. cerevisiae					
Long reads coverage	19x	13x	18x	21x	1x
Number of contigs	111	89	398	108	1
NG50	181,605	224,554	14,761	130,894	_
Genome coverage (%)	96.53	97.71	68.85	96.60	1.07
Identity (%)	99.81	99.94	99.86	99.89	96.33
C. elegans					
Long reads coverage	_	4x	2x	_	0x
Number of contigs	_	2,164	832	_	9
NG50	_	16,610	_	_	_
Genome coverage (%)	_	63.54	15.71	_	0.15
Identity (%)	_	99.42	99.81	_	98.00

Table S4: Statistics of the assemblies generated from the long reads corrected with the different methods excluded from the main comparison.

In agreement with the results observed in Supplementary Table S3, the long reads corrected with LoRMA could not be assembled at all. The long reads corrected with Nanocorr, despite covering the reference genomes well, only assembled into a satisfying number of contigs for the *A. baylyi* dataset. The assembly results on the *S. cerevisiae* dataset were however comparable to those of the other methods. Moreover, for all the assemblies generated from long reads corrected with Nanocorr, a few regions of the reference genomes were not resolved, likely because of the relatively high error rate of the long reads. The long reads corrected with CoLoRMap, despite the small throughput of the tool, surprisingly assembled quite well. A single contig was obtained for *E. coli*, and two contigs, including one of the size of the actual reference genome, were obtained for *A. baylyi*. On the *S. cerevisiae* dataset, these corrected long reads assembled into a smaller number of contigs, covering the reference genome better than those corrected with NaS. The genome coverage was also slightly higher than that of the assembly obtained with long reads corrected with HG-CoLoR, but the number of contigs and NG50 size were less satisfying. However, on the *C. elegans* dataset, the obtained assembly was highly unsatisfying, because of the weak coverage of the corrected long reads, underlining the fact that the method does not scale to larger datasets. The long reads corrected with LoRDEC also assembled into a single contig for both the *A. baylyi* and the *E. coli* datasets, despite being highly split. However, on the two larger datasets, the obtained assemblies were composed of a lot of contigs, and failed to resolve large regions of the reference genomes. Indeed, close to a third of *S. cerevisiae* was not resolved, and only a bit more than 15% of *C. elegans* was covered by contigs. Finally, the long reads corrected with HALC assembled quite well for the first two datasets. A single contig was obtained for *A. baylyi*, and two contigs, including one of the size of the actual reference genome, were obtained for *E. coli*. On the *S. cerevisiae* dataset, the corrected long reads yielded an assembly comparable to Nanocorr in terms of number of contigs, and to NaS in terms of NG50 size.

Pre-processing	QuorUM	Karect
A. baylyi		
Number of reads	16,618	16,618
Split reads (%)	4.90	4.86
Average length	10,260	10,260
Number of bases (Mbp)	179	179
Average identity (%)	99.40	99.40
Genome coverage (%)	99.82	99.80
E. coli		
Number of reads	21,005	21,006
Split reads (%)	4.98	4.88
Average length	5,797	5,794
Number of bases (Mbp)	128	128
Average identity (%)	99.81	99.81
Genome coverage (%)	99.43	99.41
S. cerevisiae		
Number of reads	33,484	33,250
Split reads (%)	11.47	10.55
Average length	6,455	6,613
Number of bases (Mbp)	243	244
Average identity (%)	99.54	99.55
Genome coverage (%)	93.32	93.19

Table S5: Comparison of the long reads corrected with Jabba, when correcting the short reads with QuorUM or with Karect.

Method	Original	HG-CoLoR (125 bp SR)	HG-CoLoR (250-300 bp SR)
A. baylyi			
Error rate	0.178534	0.000186	0.000310
Throughput	71,891,604	64,640,676	64,608,112
Deletions	2,797,255	6,464	7,802
Insertions	10,036,447	5,284	11,511
Substitutions	516,638	2,155	3,791
Split reads (%)	N/A	0	0.01
Runtime	N/A	48min	47min
E. coli			
Error rate	0.179267	0.000417	0.000596
Throughput	93,005,258	83,557,763	83,447,846
Deletions	3,635,647	25,359	23,342
Insertions	13,038,057	10,436	28,927
Substitutions	671,040	4,479	5,223
Split reads (%)	N/A	0.04	0.03
Runtime	N/A	59min	45min

Table S6: Statistics of the simulated long reads after correction by HG-CoLoR, with short reads of length 125 bp, as reported by LRCStats. The results on the *S. cerevisiae* dataset are omitted, as LRCStats reported erroneous results.

Method	Original	HG-CoLoR (125 bp SR)	HG-CoLoR (250-300 bp SR)
A. baylyi			
Number of reads	89,011	26,450	25,278
Split reads (%)	N/A	1.60	1.01
Average length	4,284	11,143	11,157
Number of bases (Mbp)	381	299	285
Average identity (%)	70.09	99.90	99.75
Genome coverage (%)	100	99.99	100
Runtime	N/A	1h44min	1h56min
E. coli			
Number of reads	22,270	22,138	21,970
Split reads (%)	N/A	0.29	0.07
Average length	5,999	6,047	6,093
Number of bases (Mbp)	134	134	134
Average identity (%)	79.46	99.94	99.84
Genome coverage (%)	100	99.99	100
Runtime	N/A	1h15min	1h05min
S. cerevisiae			
Number of reads	205,923	73,670	72,228
Split reads (%)	N/A	3.55	5.13
Average length	5,698	7,484	6,724
Number of bases (Mbp)	1,173	572	512
Average identity (%)	55.49	99.50	99.10
Genome coverage (%)	99.90	99.93	99.40
Runtime	N/A	8h50min	8h36min
C. elegans			
Number of reads	363,500	282,425	278,614
Split reads (%)	N/A	8.42	8.85
Average length	5,524	5,324	5,127
Number of bases (Mbp)	2,008	1,641	1,567
Average identity (%)	71.07	98.90	98.93
Genome coverage (%)	99.99	99.98	99.95
Runtime	N/A	95h17min	80h34min

Table S7: Statistics of the real long reads, before and after correction by HG-CoLoR, with short reads of length 125 bp and of length 250-300 bp.

Method	HG-CoLoR (125 bp SR)	HG-CoLoR (250-300 bp SR)
A. baylyi		
Long reads coverage	83x	79x
Number of contigs	2	1
NG50	3,594,329	3,634,461
Genome coverage (%)	99.97	99.99
Identity (%)	99.98	99.94
E. coli		
Long reads coverage	29x	29x
Number of contigs	1	1
NG50	4,640,101	4,659,731
Genome coverage (%)	99.99	100
Identity (%)	99.99	99.99
S. cerevisiae		
Long reads coverage	46x	41x
Number of contigs	47	67
NG50	452,906	297,575
Genome coverage (%)	99.12	97.57
Identity (%)	99.91	99.92
C. elegans		
Long reads coverage	16x	15x
Number of contigs	236	352
NG50	820,836	458,250
Genome coverage (%)	98.95	98.41
Identity (%)	99.86	99.86

Table S8: Statistics of the assemblies generated from the real long reads, after correction by HG-CoLoR, with short reads of length 125 bp and of length 250-300 bp. Reported identities stand for the 1-to-1 alignments.

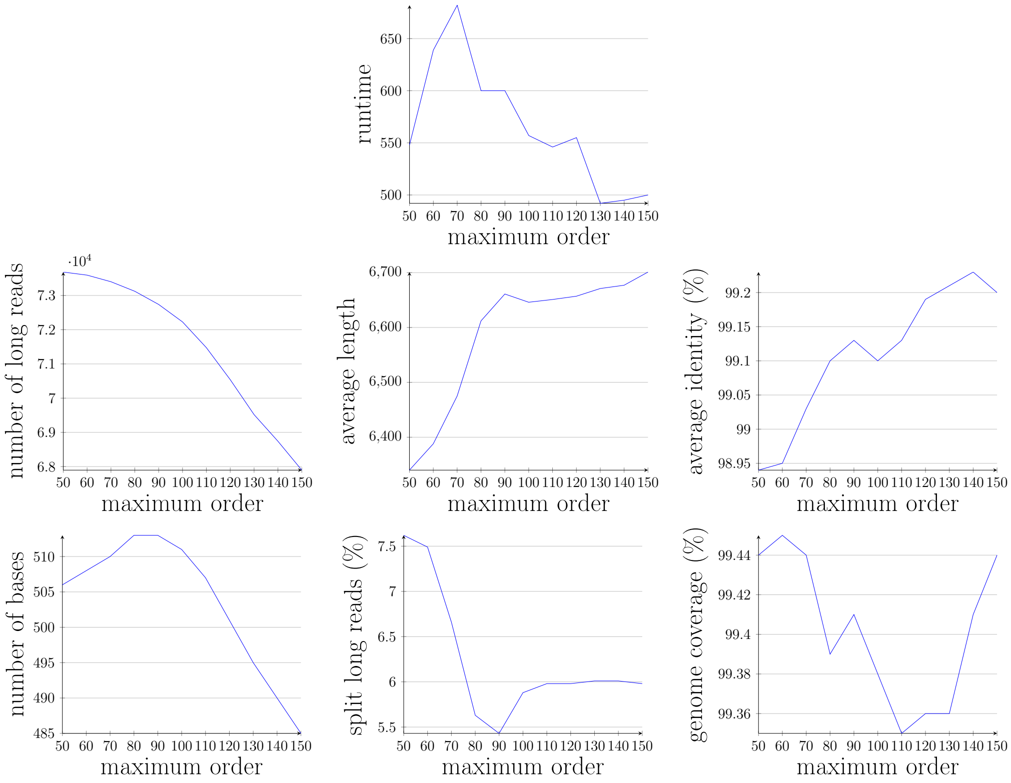

Figure S2: Impact of the maximum order of the graph on the results, when fixing other parameters. To obtain fair comparisons, the minimum order of the graph was set to half of the maximum order for each experiment. Runtimes are reported for the execution of the whole correction pipeline. We acknowledge that a maximum order of 100 yields more split, and thus slightly shorter reads, that display a lower identity than a maximum order of 90. However, it allows the pipeline to run almost an hour faster, and thus provides a satisfying compromise. Compared to these two values, higher orders tend to display a higher identity, but correct less long reads, and thus output less bases, whereas lower orders tend to correct more long reads, that are however more split, and thus shorter, in addition to display larger runtimes.

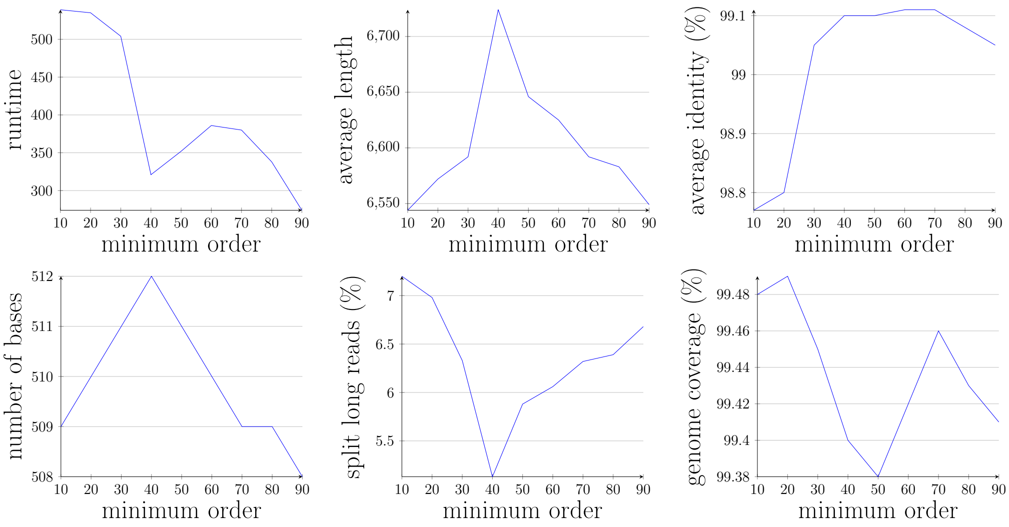

Figure S3: Impact of the minimum order of the graph on the results, when fixing other parameters. Runtimes are reported for the execution of the seeds linking and tips extension steps only. The statistics of the number of corrected long reads are not shown, as all the minimum order values corrected the same number of long reads. Apart from genome coverage and average identity of the long reads, all the other statistics displayed a clear peak with a minimum order value of 40.

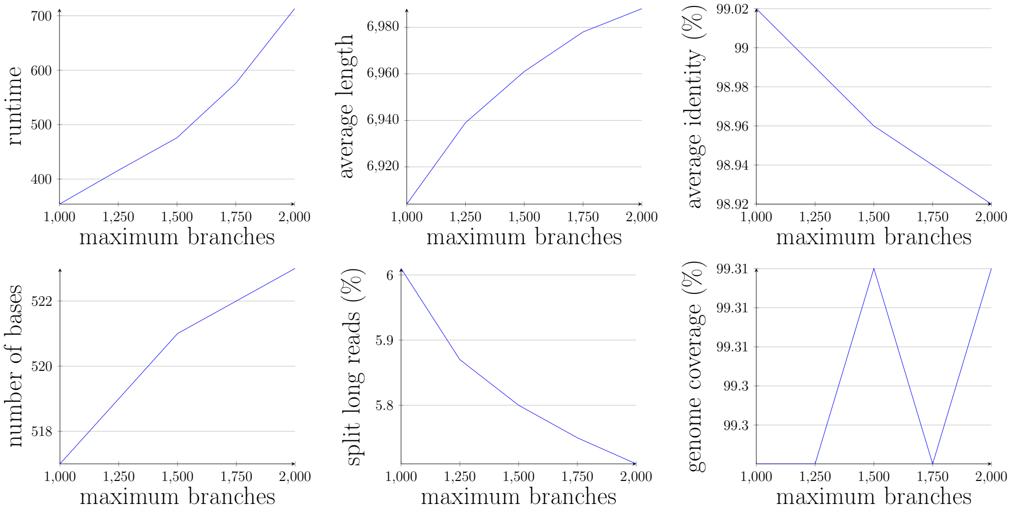

Figure S4: Impact of the maximum number of branches explorations on the results, when fixing other parameters. Runtimes are reported for the execution of the seeds linking and tips extension steps only. The statistics of the number of corrected long reads are not shown, as all the maximum number of branches explorations values corrected the same number of long reads.