LoRDEC
Accurate and efficient long read error correction

Leena Salmela[1] and Eric Rivals[2]

[1]Department of Computer Science and Helsinki Institute for Information Technology HIIT, FI-00014 University of Helsinki, Finland and
[2]LIRMM and Institut de Biologie Computationelle, CNRS and Université Montpellier, 34095 Montpellier Cedex 5, France

Aim

Reads from 3rd generation sequencing have up more that 15% higher error reads, which can complicate analysis especially in less covered regions or with less expressed RNA. Correcting sequencing errors in "long reads" from 3rd generation sequencing is complex and can require long calculation times. LoRDEC is a tool that aims to accurately and more efficiently correct erroneous positions by utilising high accuracy 2nd generation sequencing reads (later "short reads") and an efficient correction algorithm that uses a De Brujin graph.

Methods

The method of correcting sequencing errors in 3rd generation long read (LR) and circumventing long calculation times is by using a hybrid correction method paired with de Brujine graph assembly.
Unlike in self correction, the hybrid method uses a second dataset of 2nd generation short read (SR) that is mapped onto the LRs to correct sequencing mistakes, so obviously both sets have to come from the same library. Although how exactly this mapping is achieved varies between different models and has a big influence on computation time and accuracy. In contrase to previous programs like LSC (Au et al., 2012) and PacBioToCA (Koren et al., 2012), which were alignment based that try to directly map SR to LR, LoRDEC uses a short read assembly based hybrid method that builds a full De Bruijn Graph (DBG) from the short read data and uses this to correct erroneous positions.

LoRDEC

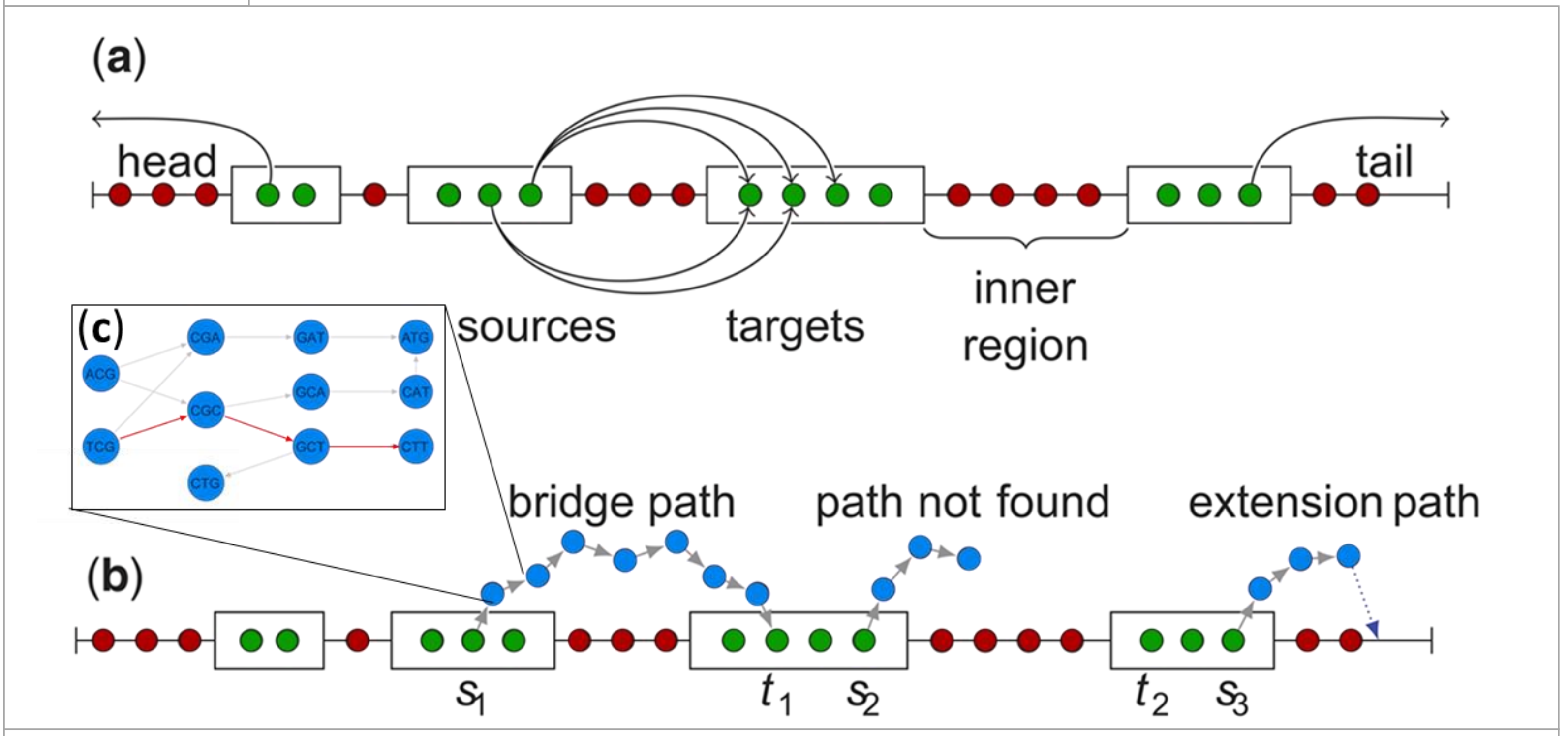


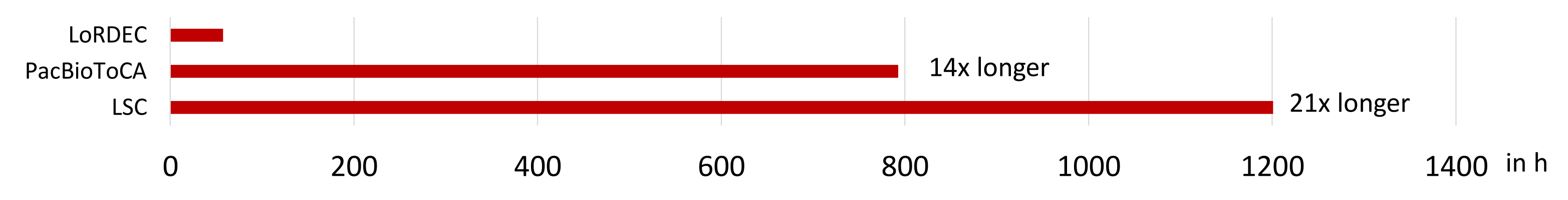
Fig. 1.
LoRDECs search for bridge paths (blue) from solid k-mers (green) of the LR, across weak k-mers (red) to find the corrected sequence within the SR De Bruijn graph. (a) For each solid k-mer the algorithm searches a bridge from this k-mer to a user defined number (default t=5) of target k-mers t_i. All found path form a weighted "path graph" that is used to find the best correction. (b) The algorithm might not be able to find a path in distance t or if weak regions occur at the ends so for this a "best extension" procedure is used instead of the De Bruijn graph (DBG) bridges, to estimate a good fitting correction.
(c) This represents an example DBG in which the bridge is searched. The solid source k-mer represents an entry into the DBG and paths from this node to the target k-mer are searched using depth first search. This exploration is stopped if all extensions would result in exceeding a user defined maximum allowed error rate or till either a dead end or the target k-mer is found. The overall search is stopped if the number of encountered paths exceeds a user defined branching limit.

The Algorithm works by first identifying which k-mers in the LR might contain a sequencing error. It does this by flagging each k-mer that is found in the DBG as solid and each one that is not found is flagged for correction as a weak k-mer. The Algorithm then searches bridges over weak regions (described below Fig.1) by using Dijksta's algorithm on a graph which has the k-mers as nodes connected by edges for each identified bridge which are weighted by the minimum edit distance and thus the least changes/correction that would have to be made to the LR. The shortes path through this graph represents the individual long read corrected by our short read library

Results

In *E. coli*, as well as parrots and yeast LoRDEC has proven a significantly better efficiency than LSC and PacBioToCA while simultaneously improving the sensitivity. With its new approach to correct errors in long reads by using a de Brujin based short read correction it made the multiple alignment expandable. LoRDEC exploits the perks of both 2nd and 3rd generation sequencing. By using this new method it drastically lowers the calculation time and memory. LoRDEC is usable for large data sets and works with Pacific Biosciences SMRT sequencing (SMRT = Single Molecule Real Time) as well as with Oxford Nanopore MINion technology making it quite flexible method. [1, 3]

Elapsed time Yeast



Full comparison

Data	Method	CPU time	Elapsed time	Memory	Disk	FP	TP	FN	Sensitivity	Gain
E.coli 98 Mb	PacBioToCA	45 h 18 min	3 h 12 min	9.91	13.59	NA	NA	NA	NA	NA
	LSC	39 h 48 min	2 h 56 min	8.21	8.51	695773	3149629	7845597	0.2865	0.2232
	LoRDEC	2 h 16 min	10 min	0.96	0.41	102427	9994561	1000665	0.9090	0.8997
Yeast 1.5 Gb	PacBioToCA[a]	792 h 41 min	21 h 57 min	13.88	214	NA	NA	NA	NA	NA
	LSC[b]	1200 h 46 min	130 h 16 min	24.04	517	7766700	38741658	80597251	0.3246	0.2596
	LoRDEC	56 h 8 min	3 h 37 min	0.97	1.63	2784685	100568850	18770059	0.8427	0.8194
Parrot 6.8 Gb	LoRDEC[b]	568 h 48 min	29 h 7 min	4.61	74.85	10591097	226996640	26296446	0.8962	0.8544

Fig. 2.
Comparing LoRDEC with earlier error correction tools. (Top) Shows a plot of the CPU time for the different tools in correcting the sample Yeast genome. (Bottom) Also shows the complete comparison for all open source datasets that where used.

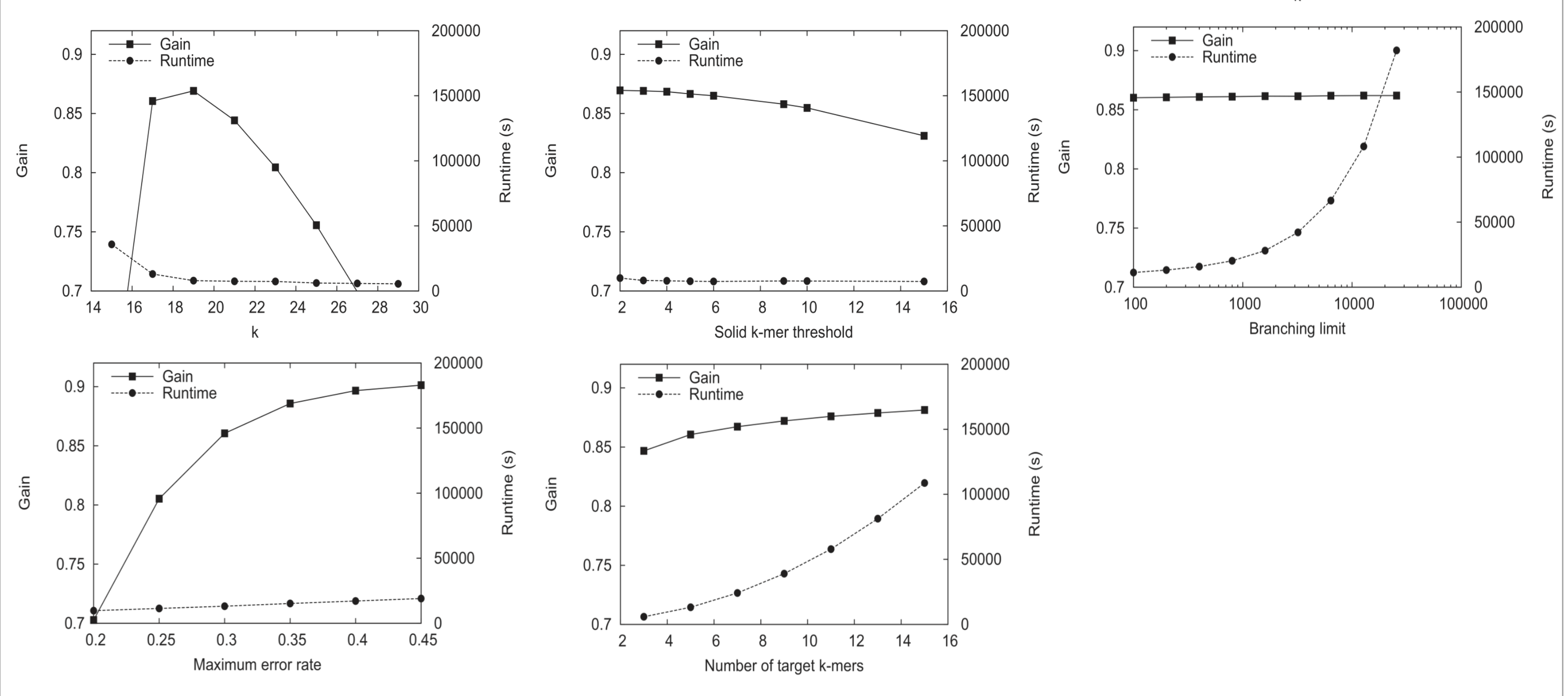


Fig. 3.
Pots for gain (how well LoRDEC corrects errors without introducing new ones) and runtime across varying correction parameters. A user can adjust the 5 parameters: k (kmer size), solid k-mer threshold (above which coverage a k-mer is considered solid), branching limit (how far the DBG search is allowed to branch out), maximum error rate (how different the DBG correction can be from the LR) and the number of target k-mers (for how many target kmer the search is executed for each solid k-mer).
The plots show that the choise of k has the biggest impact on gain, followed by the maximum error rate, while both don't have a big impact on runtime in the region of interest. The other parameters have a smaller influence on the gain but can sometimes come with a considerable increase of runtime. The bigger parrot dataset showed the best results with k=23 instead of 19 so the parameters should be explored for different datasets to optimise gain.

Conclusion/Review

By the time of its release LoRDEC was in a sense revolutionary. It was the first hybrid method to use a De Brujin graph based correction method [1]. Although it was overtaken by e.g. Jabba and FMLRC in calculation time and alignment identity, both Jabba and FMLRC as well as HALC and Salmela et al.`s very own LoRMA are based on the LoRDEC algorithm or use an alike one[2, 4]. With that being said LoRDECs memory usage is still seemingly the best for hybrid correction methods [2]. Overall the state of the art hybrid correction method is FMLRC since it provides an even better calculation time and alignment identity.

Data set	Method	Genome fraction (%)	Alignment identity (%)	CPU time (hh:mm:ss)	Memory usage (GB)
E. coli LR	Original	100.000	86.8763	-	-
	LoRMA	100.000	99.6814	45:24:49	67.2
	Canu	100.000	99.6970	07:47:33	6.0
E. coli LR + SR	FMLRC	100.000	99.6983	03:05:06	9.8
	LoRDEC	100.000	98.4149	15:03:42	2.0
	LSC	-	-	-	-
	PacBioToCA	-	-	-	-

Table 2.
Comparison of LR self correction (LR only) and hybrid correction methods (LR + SR).
Adapted from (Zhang et al. pre-print 2019) a review paper on the comparison of Long Read error correction tools.
LoRDEC outperformed older tools but newer, better and faster tools are now available

Literatur

[1] Leena Salmela and Eric Rivals, "LoRDEC: accurate and efficient long read error correction", *Bioinformatics*, vol. 30, no. 24 2014, pp. pages 3506–3514, 2014
[2] Haowen Zang, Chirag Jain and Srinivas Aluru, "A comprehensive evaluation of long read error correction methods", *Briefings in Bioinformatics*, preprint not released yet
[3] "LoRDEC: hybrid correction of long reads", 05.06.2019, University of Montpellier, date accessed 29.06.2019, URL: http://www.lirmm.fr/~rivals/lordec/
[4] Leena Salmela et al., "Accurate self-correction of errors in long reads using de Bruijn graphs", *Bioinformatics*, vol. 33, 2017, pp. pages 799–806, 2016

UNIVERSITY OF HELSINKI

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nils Nover, Christopher Held; Aktuelle Themen der Sequenzanalyse; SoSe 2019