Bioinformatics, 2016, 1–2
doi: 10.1093/bioinformatics/btw753
Advance Access Publication Date: 31 January 2017
Applications Note

OXFORD

Sequence analysis

Edlib: a C/C++ library for fast, exact sequence alignment using edit distance

Martin Šošić[1] and Mile Šikić[1,2,*]

[1]Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, Zagreb, HR 10000, Croatia, [2]Bioinformatics Institute, A*STAR, #07-01 Matrix, 138671 Singapore, Singapore

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on August 19, 2016; revised on November 3, 2016; editorial decision on November 21, 2016; accepted on December 1, 2016

Abstract

Summary: We present Edlib, an open-source C/C++ library for exact pairwise sequence alignment using edit distance. We compare Edlib to other libraries and show that it is the fastest while not lacking in functionality and can also easily handle very large sequences. Being easy to use, flexible, fast and low on memory usage, we expect it to be easily adopted as a building block for future bioinformatics tools.
Availability and Implementation: Source code, installation instructions and test data are freely available for download at https://github.com/Martinsos/edlib, under the MIT licence. Edlib is implemented in C/C++ and supported on Linux, MS Windows, and Mac OS.
Contact: mile.sikic@fer.hr
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

One of the fundamental operations in bioinformatics is pairwise sequence alignment—a way to measure either the similarity or distance between two sequences. Due to the quadratic time complexity, deterministic algorithms that yield optimal alignment are inefficient for the comparison of long sequences. Therefore, they are used in the very last step when the aligning substrings of the given sequences are roughly determined using heuristic methods.

Deterministic, optimal alignment algorithms are unavoidable for the resequencing of genomes when the exact alignments of reads and reference are necessary for the successful determination of differences—especially, the existence of single nucleotide variants. Owing to that, many aligners use some of the efficient variants of these algorithms for the final phase. For example, SNAP (Zaharia *et al.*, 2011) uses Landau–Vishkin (Landau *et al.*, 1986) as the core component.

The increased need for exact algorithms that could align longer segments has recently emerged as a consequence of the advent of long-read sequencing technologies such as Pacific Biosciences Single Molecule Real-Time (SMRT) sequencing technology and Oxford Nanopore Technologies (ONT), which produce reads over 10 kbp in length.

Deterministic methods can be categorized as local, global or semi-global (overlap) alignment methods, regarding their scoring scheme. The basic global alignment algorithm is the dynamic programming Needleman–Wunsch algorithm (Needleman and Wunsch, 1970), and the basic local alignment algorithm is its variation, the Smith–Waterman algorithm (Smith and Waterman, 1981). Semi-global alignment methods, quite popular for read alignment, are similar to global alignment but they do not penalize gaps at the beginning or/and the end of the sequences. Both Needleman–Wunsch and Smith–Waterman algorithms have quadratic time and space complexity, so there has been a lot of work on trying to improve that. Ukkonen's banded algorithm (Ukkonen, 1985) reduces needed time by cleverly reducing the space of search, while Hirschberg's algorithm (Hirschberg, 1975), adapted by (Myers and Miller, 1988) to accommodate affine gap penalties, trades space for speed, reducing space complexity from quadratic to linear.

An important sub-category of alignment methods is the calculation of Levenshtein distance—the minimum number of single-character edits (insertions, deletions or substitutions) required to change one sequence into the other (also referred to as edit distance). Myers managed to exploit its special properties by developing a bit-vector algorithm (Myers, 1999), reducing the computation time by a constant factor. Although it is one of the fastest deterministic alignment algorithms it is quite complex to implement and does not support global alignment. Hence, it is rarely implemented in practice. In this article, we present Edlib—our implementation of Myers's bit-vector algorithm, extended with additional methods and features that are important for its practical application.

2 Methods

Myers's bit-vector algorithm transforms the dynamic programming matrix which enables us to store multiple cells as a bit-vector into one CPU register and achieve parallelization. Myers additionally applies Ukkonen's banded algorithm to reduce the number of calculated bit-vectors.

The original algorithm was designed only for a semi-global alignment method where gaps at the start and at the end of the query sequence are not penalized (infix method). In Edlib, we extended Myers's algorithm to support the global alignment method and the semi-global alignment method where gaps at the end of the query sequence are not penalized (prefix method). For this, we came up with extended banded algorithm that also supports prefix and global method (Supplementary Methods).

Originally, Myers's algorithm returns no information about the optimal alignment path—the optimal sequence of edit operations that need to be performed on the query sequence to transform it to the target sequence. In Edlib, we further extended Myers's algorithm with the finding of the optimal alignment path for all three supported alignment methods in linear space by combining it with Hirschberg's algorithm. Inspired by the SSW Library (Zhao, 2013), we reduced the problem of finding the path for infix and prefix method to finding the path for global alignment, which both simplifies the implementation and improves the computation speed (Supplementary Figure 1).

Unlike Myers's algorithm, Edlib can work without defined upper limit for edit distance and guarantees to find optimal solution in such case.

We implemented Edlib as both C/C++ library and a stand-alone application.

3 Results

We compared Edlib with SeqAn library (Döring *et al.*, 2008), Parasail library (Daily, 2016) and the original Myers's implementation.

We chose SeqAn to be the center of our comparison, since Döring *et al.* (2008) show it is the library with the fastest implementation of sequence alignment using edit distance (they also combine Myers's bit vector algorithm with Hirschberg's algorithm), and to our knowledge, there are no developments up to now showing otherwise. SeqAn is one of the most advanced sequence alignment libraries and offers many additional methods next to calculation of edit distance.

Next we chose Parasail, which is one of the fastest sequence alignment libraries that support similarity search with custom score matrix, and not only edit distance. We did this comparison to show the difference in speed between Edlib, which is specialized for edit distance, and more general libraries like Parasail.

Comparison was done against SeqAn v2.2.0 and Parasail v1.1.0, which were the latest releases at the moment of writing this article.

The tests were performed on Linux, Intel Core i7-4710HQ 2.5 Ghz with 16GB RAM. As test data, we used real DNA sequences ranging from 10 to 5000 kbp in length and their artificially mutated versions, in order to show how the similarity and length of aligned sequences affect performance.

The run times for finding the global alignment edit distance, with and without alignment path, are displayed in Table 1. Results show that Edlib is 2.5–100 times faster than SeqAn, and 12–1000 times faster than Parasail, the difference being the largest when the sequences are large and similar. Additionally, we ran similar tests (Supplementary Tables 1–3) for the semi-global methods where Edlib also outperformed both SeqAn and Myers's implementation, while Parasail does not support infix and prefix semi-global methods.

Table 1. Run time comparison of finding global alignment edit distance and alignment path for different sequence lengths and similarities

Seq. sizes	Similarity (%)	Edlib (path)	SeqAn (path)	Parasail
$10^6 \times 10^6$	99	1.1s (2.58s)	111.83s (252.71s)	1234.5s
$10^6 \times 10^6$	90	7.16s (17.35s)	111.51s (253.32s)	1212.37s
$10^6 \times 10^6$	80	14.42s (34.6s)	111.7s (252.95s)	1247.44s
$10^6 \times 10^6$	70	33.8s (65s)	112s (253.1s)	1205.16s
$10^6 \times 10^6$	60	30.75s (71.59s)	111.44s (252.61s)	1212s
$10^5 \times 10^5$	99	0.01s (0.06s)	1.01s (2.27s)	4.79s
$10^5 \times 10^5$	90	0.13s (0.24s)	0.98s (2.32s)	4.68s
$10^5 \times 10^5$	80	0.2s (0.45s)	0.98s (2.31s)	4.79s
$10^5 \times 10^5$	70	0.16s (0.49s)	1s (2.29s)	4.76s
$10^5 \times 10^5$	60	0.4s (0.83s)	1s (2.3s)	4.76s

The similarity of two sequences was calculated as $1 - edit\_distance/\min(length_{query}, length_{target})$. Two different DNA sequences were used for these tests. We artificially mutated them to achieve different similarities. Myers's implementation is not included in this comparison as it does not support global alignment. For SeqAn and Edlib, time needed for finding of not only score but also of alignment path is provided in parentheses.

Regarding the alignment path, SeqAn could not complete our tests with semi-global methods because it was allocating too much memory, while Parasail and Myers do not support the finding of alignment path.

As can be seen from the results, Edlib exhibits significant improvement in speed with increase of sequence similarity, in contrast to other libraries. This is due to our implementation of the banded algorithm, which significantly reduces search space for similar sequences.

Acknowledgements

The authors would like to thank Ivan Sović for valuable help with testing Edlib and providing comments on the manuscript.

Funding

This work has been supported in part by Croatian Science Foundation under the project UIP-11-2013-7353 "Algorithms for Genome Sequence Analysis".

Conflict of Interest: none declared.

References

Daily,J. (2016) Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. *BMC Bioinformatics*, **17**, 11.

Döring,A. *et al.* (2008) SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**, 11.

Hirschberg,D.S. (1975) A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, **18**, 341–343.

Landau,G.M. *et al.* (1986) An efficient string matching algorithm with k differences for nucleotide and amino acid sequences. *Nucleic Acids Res.*, **14**, 31–46.

Myers,E.W. and Miller,W. (1988) Optimal alignments in linear space. *Comput. Appl. Biosci.*, **4**, 11–17.

Myers,G. (1999) A fast bit-vector algorithm for approximate string matching based on dynamic programming. *J. ACM*, **46**, 395–415.

Needleman,S.B., and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol*, **48**, 443–453.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Ukkonen,E. (1985) Algorithms for approximate string matching. *Inform. Control*, **64**, 100–118.

Zaharia,M. *et al.* (2011) Faster and more accurate sequence alignment with Snap. *arXiv*, 1111.5572.

Zhao,M. *et al.* (2013) SSW Library: an SIMD Smith–Waterman C/C++ library for use in genomic applications. *PloS One*, **8**, e82138.