PA-Bench: A framework for benchmarking pairwise aligners

Ragnar Groot Koerkamp[1], **Daniel Liu**[2]
[1] ETH Zurich
[2] University of California Los Angeles

Motivation: The problem of pairwise alignment has been studied for over 50 years. Still, alignment accounts for a significant fraction of the time spent in bioinformatics pipelines, and many advancements are still being made, e.g. [Shao and Ruan 2024]. To date, there has not been a scalable and comprehensive benchmark suite for aligners. Many developers build their own ad-hoc benchmarks, but they fail to account for the diversity of downstream use cases and the complexity of benchmarking. For example, it is possible to align reads to genomes, genomes to genomes, reads to other reads, etc. Pairs of sequences can vary widely, from small substitutions or insertions/deletions (indels), to large structural variations driven by biology. This problem is amplified by the rapid increase in sequencing technologies that produce reads with varying error profiles and length characteristics, including short Illumina reads, long PacBio HiFi reads, and ultra-long Oxford Nanopore reads.

Methods: We introduce PA-Bench, a rigorous benchmarking framework for pairwise alignment that 1) makes it easier for algorithm developers to compare their methods with others on a wide variety of data, 2) allows downstream users to make informed choices in choosing the right algorithms for their data, and 3) provides a library of reusable types and a uniform API and command line interface to existing aligners. The main component is a command-line tool to orchestrate a set of jobs for benchmarking different aligners on different datasets. Currently supported are Parasail [Daily 2016], Edlib [Šošić and Šikić 2017], KSW2 [Li 2018, Suzuki and Kasahara 2018], WFA/BiWFA [Marco-Sola et. al. 2021, 2023], Triple Accel [Liu 2022], Block Aligner [Liu and Steinegger 2023], and A*PA/A*PA2 [Groot Koerkamp and Ivanov 2024, Groot Koerkamp 2024], and new aligners can easily be added by implementing a simple interface.

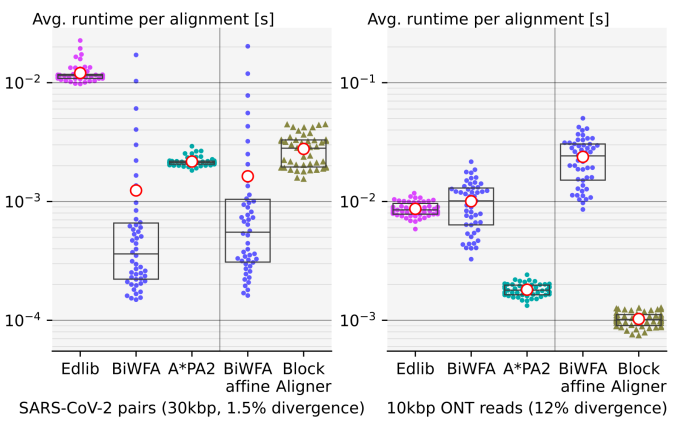
```yaml
- time_limit: 1h
  mem_limit: 32GiB
  datasets:
    - !Generated
      seed: 31415
      error_rates: [0.05]
      error_models: [Uniform]
      lengths: [1000, 10000, 100000, 1000000]
      total_size: 10000000
  traces: [true]
  costs: [{ sub: 1, open: 0, extend: 1 }]
  algos:
    - !Edlib
    - !Wfa
```

Input: PA-Bench's main input is a YAML file that allows the user to specify a number of parameters: 1) the datasets to run on: a file, URL, or generator parameters, 2) whether to compute a traceback, 3) the cost model to use, and 4) the aligners to run. 'Jobs' are created from the cartesian product of these parameters. For example, the configuration shown on the right can be used to compare the scaling of Edlib and WFA on different sequence lengths.

Output: Benchmarking is done by running one job at a time, each pinned to its own thread, optionally with a memory or time limit. Various statistics are collected, such as (wall, system, and user) time usage, peak and incremental memory usage, and the CPU frequency when the job started and finished. PA-Bench verifies alignment results by checking the returned CIGAR for consistency and by comparing the cost of the alignment returned by different algorithms. Results are written to a JSON file that can be parsed and can be plotted by provided python notebooks.

Usage: For convenience, PA-Bench also supports running multiple jobs in parallel and can cache results, so that only new jobs are run when an experiment is modified.

Results: On the right is a figure generated by PA-Bench. It shows the running times of various aligners on two datasets. Included are exact aligners Edlib, BiWFA, and A*PA2. BiWFA-affine and Block Aligner use gap-open cost 1, and Block Aligner is an approximate aligner. This benchmark shows how the relative performance of aligners can vary widely between different types of sequences.



PA-Bench has proven useful for rapidly testing and benchmarking aligners, and it has been used in practice to both run the benchmarks for A*PA and discover bugs in other aligners.

Conclusion: PA-Bench is a convenient tool that makes comparing pairwise aligners easy for

developers and users. Currently, PA-Bench only supports benchmarking global alignment of DNA sequences. Although this does not capture the full complexity of real-world use cases, which include semi-global, local, and e.g. sequence-to-graph alignment, PA-Bench still allows us to better understand the performance characteristics of alignment algorithms.
Availability: github.com/pairwise-alignment/pa-bench

References.

Daily, J. (2016). Parasail: Simd c library for global, semi-global, and local pairwise sequence alignments. BMC Bioinformatics

Groot Koerkamp, R. and Ivanov, P. (2024). Exact global alignment using A* with chaining seed heuristic and match pruning. Bioinformatics

Groot Koerkamp, R. (2024). A*PA2: up to 20 times faster exact global alignment. bioRxiv

Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics

Liu, D. (2022). triple_accel. https://github.com/Daniel-Liu-c0deb0t/triple_accel

Liu, D. and Steinegger, M. (2023). Block Aligner: an adaptive SIMD-accelerated aligner for sequences and position-specific scoring matrices. Bioinformatics

Marco-Sola, S., Moure, J. C., Moreto, M., and Espinosa, A. (2021). Fast gap-affine pairwise alignment using the wavefront algorithm. Bioinformatics

Marco-Sola, S., Eizenga, J. M., Guarracino, A., Paten, B., Garrison, E., and Moreto, M. (2023). Optimal gap-affine alignment in o(s) space. Bioinformatics

Myers, G. (1999). A fast bit-vector algorithm for approximate string matching based on dynamic programming. Journal of the ACM

Shao, H. and Ruan, J. (2024). Bsalign: a library for nucleotide sequence alignment. bioRxiv

Šošić, M, Šikić, M. (2017). Edlib: a C/C++ library for fast, exact sequence alignment using edit distance. Bioinformatics

Suzuki, H. and Kasahara, M. (2018). Introducing difference recurrence relations for faster semi-global alignment of long sequences. BMC Bioinformatics

Ukkonen, E. (1985). Algorithms for approximate string matching. Information and control