A*PA & A*PA2: Up to 20x faster exact global alignment

Ragnar Groot Koerkamp
ragnar.grootkoerkamp@inf.ethz.ch
@curious_coding

Pesho Ivanov
pesho@inf.ethz.ch
@peshotrie

ETH zürich

Ragnar Groot Koerkamp and Pesho Ivanov. **Exact Global Alignment Using A* with Chaining Seed Heuristic and Match Pruning.** *Bioinformatics* (2024).

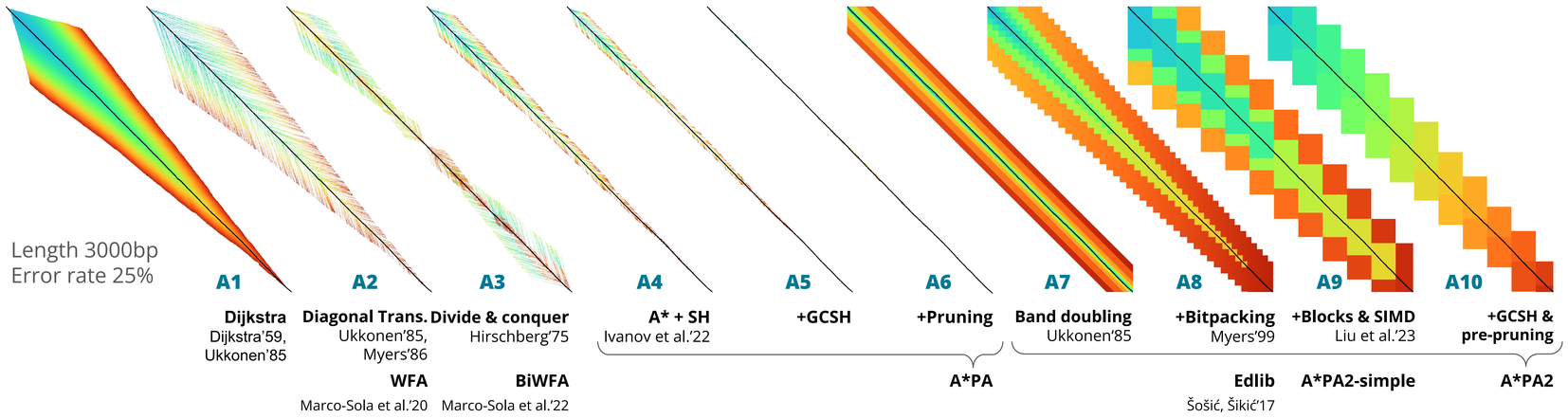
Ragnar Groot Koerkamp. **A*PA2: Up to 20 Times Faster Exact Global Alignment**. *bioRxiv (2024)*

Code, papers, and references:
curiouscoding.nl/notes/astarpa-poster

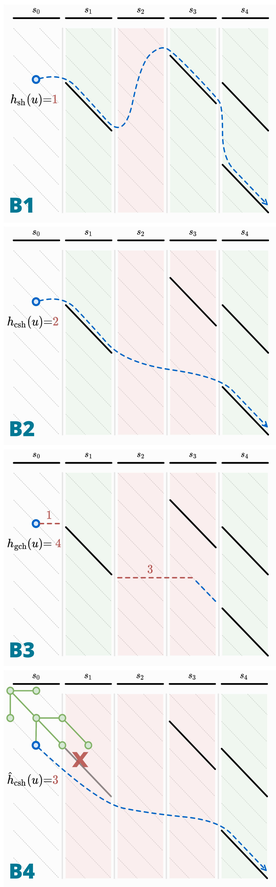
Introduction

Pairwise alignment is a core problem in computational biology. While sequence lengths have increased, exact methods still scale quadratically with length. Our algorithms to compute the edit distance and alignment are empirically fast and provably exact.

- **A*PA** (A* Pairwise Aligner, **A6**) uses A*, and scales near-linearly to very long sequences with up to 12% uniform errors.
- **A*PA2** (**A10**) combines the A* heuristic with DP to be fast even when divergence is high, giving 19x speedup when aligning long reads.

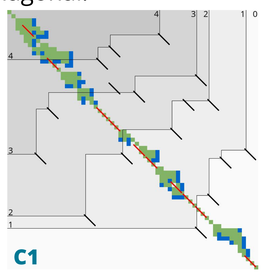


A*PA



Run the **A*** shortest path algorithm on the **edit graph**. A* uses a **heuristic** function *h*, a lower bound on the remaining distance, and visits states in order of *f(u)=g(u)+h(u)*, where *g(u)* is the distance to *u*.

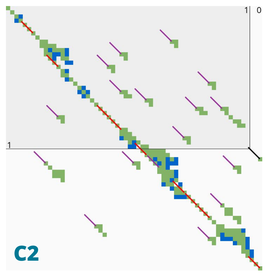
- **Diagonal transition** reduces the number of visited states (**A2**).
- A*ix introduced the **seed heuristic** (**B1**, **A4**) that splits the first sequence into length *k seeds*, and finds all their *matches*. The number of upcoming seeds without matches lower-bounds the future edits.
- The **chaining seed heuristic** (**B2**) additionally requires that the matches form a *chain* going down and right. We compute this using *contours* (**C1**).
- The **gap-chaining seed heuristic** (**B3**, **A5**) further penalizes the joining of matches not on the same diagonal.
- **Pruning** (**B4**, **A6**) removes matches as soon as a shortest path to them has been found (shown in red in **C1**).

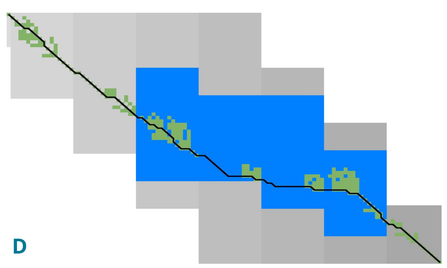


A*PA2

Similar to Dijkstra (**A1**), A*PA (**A6**) is not cache-friendly and requires many writes to update distances and the priority queue of explored states. On the other hand, dynamic programming (**DP**) with bit packing is up to 1000x faster per state because of its simplicity and predictability.

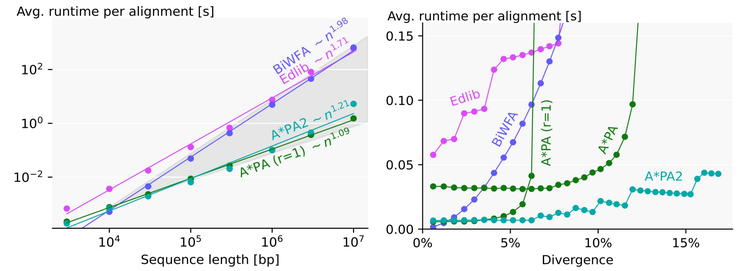
- A*PA2 is based on **band doubling** (**A7**) with **bitpacking**, like Edlib(**A8**).
- Inspired by Block Aligner, A*PA2 uses 256 column wide **blocks** (**A9**).
- A*PA2 reuses the **gap-chaining seed heuristic** with **pruning** (**A10**).
- For the **traceback**, scores are stored at block boundaries. The path is computed backwards. In each block, first a **banded diagonal transition** method is tried (green cells). If that fails, part of the block is recomputed (blue rectangles) (**D**).
- **Incremental doubling** ensures that after doubling, computed values are reused whenever possible.
- **Pre-pruning** removes most off-path matches. All matches are *extended*, and if they run into 2 or more errors over the next seed they are removed (**C2**).





Scaling & runtime

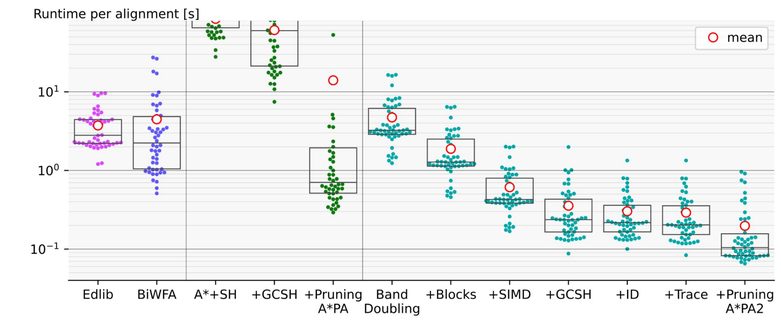
1. Scaling with length and divergence (random seqs with edits)



A*PA and A*PA2 scale near-linear with length (4% divergence)

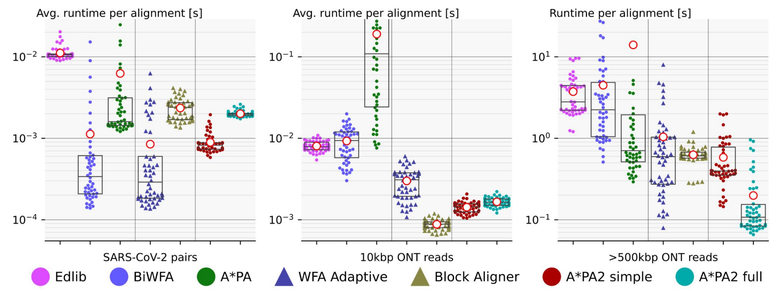
A*PA2 is fast even for high divergence (100kbp length)

2. Incremental effect of features (>500kbp ONT reads)



A*PA is ~2.5x faster in median, but outliers are slow.
A*PA2 is 19x faster than Edlib and BiWFA

3. Comparison on real datasets



A*PA2 is the fastest exact method for long sequences (>10kbp).
A*PA2-simple (without seed heuristic) is faster for short sequences.
*Note that WFA Adaptive and Block Aligner are approximate methods

Conclusion

We developed the A*PA and A*PA2 exact aligners which compete in speed even with the fastest approximate algorithms. Semi-global and affine-cost alignment will be future work.