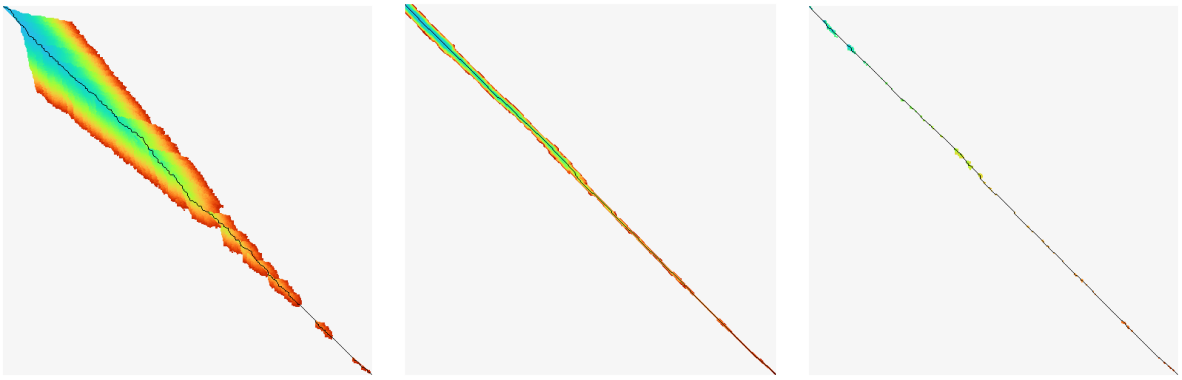
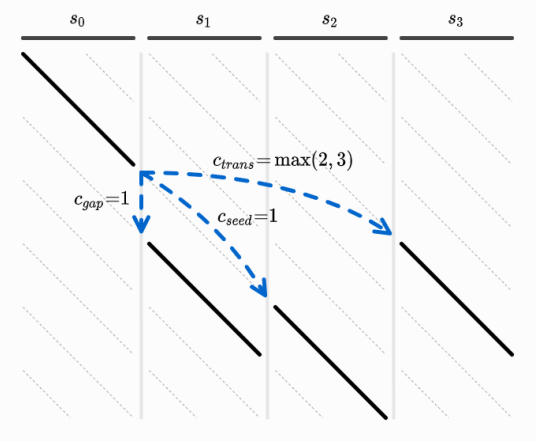
Exact pairwise alignment using A*
with chained-seeds heuristic and match pruning

We present an algorithm that integrates multiple approaches that were used previously for alignment: contours [Hunt'77], seeds [Deorowicz'13], gap-cost [Ukkonen'85], A* shortest path algorithm [Ivanov'22], and multiple-path pruning [Poole'17].

Visual representation of the algorithm behavior. The three figures represent a small example of the explored states by Dijkstra, A*PA (no prune), and A*PA (with pruning) (from left to right). Our algorithm critically lowers the number of explored states. The color corresponds to the order of state explorations (first explored are blue, last are red). Data: A random string of length n=500, with e=20% uniform errors. A*PA parameters: Seeds of length k=9, matched with up to m=1 errors. Resulting average bands (widths) of the explored states: 71, 11, 2.5.



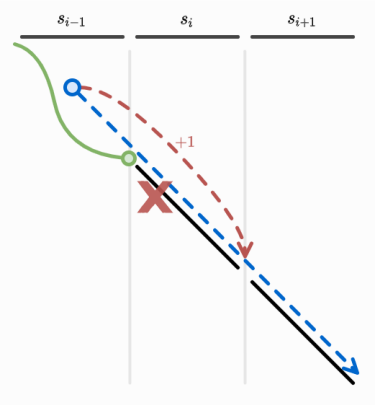
Heuristic function. We partition the first sequence into short seeds of length *k*, and find their matches in the second sequence. The figure on the right demonstrates three transitions from the end of the match of seed s_0 to other seed matches, that are shown as dashed blue arrows: single gap with cost $c_{gap}=1$, mismatching/skipping one seed with cost $c_{seed}=1$, and mismatching/skipping of 2 seeds or making 3 gaps with $c_{trans}=max(2,3)$. The heuristic *h(u)* is the lowest cost among all possible chains from state *u* to the end, each of which is the sum of the transition cost to a next match and the lowest cost h_{match} from there on.



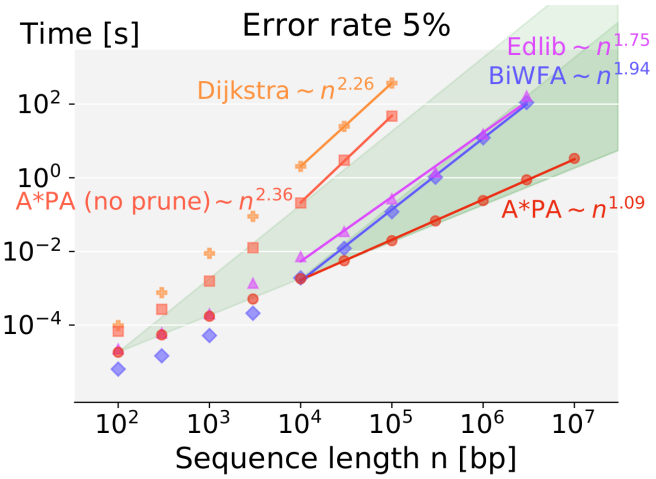
$$h(u) = \min_{\substack{m \in M \\ u \preceq start(m)}} c_{\text{trans}}(u, start(m)) + h_{\text{match}}(m),$$

$$h_{\text{match}}(m) = \begin{cases} Cost(m) + h(end(m)) & \text{if } m \neq \omega, \\ 0 & \text{if } m = \omega. \end{cases}$$

Pruning optimization of the heuristic. The figure on the right demonstrates the pruning of a match (shown as a crossed black segment), as soon as its start is expanded (a green circle). Then the match will no longer be usable for the heuristic from earlier states (like the blue circle). As a result, the cost of the lowest cost chain will increase (red dashed arrow). We prove that once the start of a match is expanded, no shorter path to it exists. Any potential globally-shortest path must either go around the start of the match, or it can reuse the already found shortest path to the match start.



Near-linear runtime scaling. The following plot compares the scaling of our approach (A*PA), partial approaches (Dijkstra and A*PA (no prune) without pruning) and the most performant aligners (Edlib [Šošić'17] and BiWFA [Marco-Sola'22]). One of the input sequences is uniform random. The other is created by introducing 5% of uniform errors, equally split over insertions, deletions, and substitutions. The scaling is estimated empirically by fitting a polynomial $f(n)=an^b$ (using the least squares method) through the points covered in the log-log space. For easier visual comparison, the sides of the two green cones denote the slopes for linear and for quadratic growth.



A*PA is fastest for long reads and low-enough error rates.

We use three dataset to compare the exact aligners: one random sequence of length 10^7 bp with 5% uniform random errors, and two set of 10 simulated ONT reads (10% errors) of length between 750'000 bp and 12'500'000 bp, one from a random sequence and one from a human genome. The seed length *k* and the use of exact or inexact matches are manually chosen.

Input data Aligner	Random sequence n=10^7 uniform errors (e=5%)	Random sequence n=10^6 Simulated ONT (e=10%)	Human genome* n=10^6 Simulated ONT (e=10%)
A*PA	3.0s, 456MB (exact matches, k=13)	3.8s, 307MB (inexact matches, k=12)	11s, 733MB (inexact matches, k=15)
BiWFA	1566s, 206MB	59s, 57MB	51s, 56MB
Edlib	1235s, 103MB	31s, 14MB	28s, 14MB

**** Important note on the human genome:** we had to exclude a simulated read containing a region of low entropy since this creates too many matches, and triggers quadratic runtime.*

Discussion. We demonstrated an algorithm that, to our knowledge, is the first exact algorithm to reach a near-linear runtime on random sequences, and outperforms BiWFA and Edlib by a factor >2.5 on simulated ONT data. The main limitation is that our method is only efficient on sufficiently random inputs, and degenerates to a slow quadratic method for sequences with repeats or non-uniform errors such as long indels. Future work will be to build the A* on top of the diagonal-transition method, and to extend to ends-free alignment modes and other cost models. We would like to prove that a simplified version of our method indeed runs in expected log-linear time when restricted to a suitable domain.

References.

Deorowicz, S. and Grabowski, S. (2014). Efficient algorithms for the longest common subsequence in k-length substrings

Hunt, J. W. and Szymanski, T. G. (1977). A fast algorithm for computing longest common subsequences. Communications of the ACM

Ivanov, P., Bichsel, B., and Vechev, M. (2022). Fast and Optimal Sequence-to-Graph Alignment Guided by Seeds. RECOMB

Marco-Sola, S., Eizenga, J. M., Guarracino, A., Paten, B., Garrison, E., and Moreto, M. (2022). Optimal gap-affine alignment in o(s) space. bioRxiv

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

Poole, D. L. and Mackworth, A. K. (2017). Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press. Second edition

Sellers, P. H. (1974). On the theory and computation of evolutionary distances. SIAM Journal on Applied Mathematics

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