



To what extent have the techniques for theoretical analysis of edit distance algorithms achieved their goals?

BY PAUL MEDVEDEV

Theoretical Analysis of Edit Distance Algorithms

EDIT DISTANCE—A CLASSICAL problem in computer science—has received ongoing attention from both practitioners and theoreticians. Given two strings A and B , the edit distance is the minimum number of substitutions, insertions, and deletions needed to transform A into B . For example, the edit distance between *apflee* and *rapleet* is 3: $apf \xrightarrow{ins} rapflee \xrightarrow{del} raplee \xrightarrow{sub} rapleet$. The edit distance problem is widely known, as it is often taught as part of the undergraduate curriculum to illustrate two-dimensional dynamic programming. Theoreticians

have studied the problem starting from as early as 1966²⁴ and 1974,⁴³ but it very much remains an active topic of research today (for example, Boroujeni et al.⁸). Simultaneously, bioinformatics practitioners continue to actively develop^{14,15,40} and apply^{9,28,38,39,42,45} fast edit distance solvers ubiquitously. Given its status as a classic problem and its importance to both theoreticians and practitioners, edit distance provides a unique opportunity to study the interaction between theory and practice.

Theoreticians develop abstract algorithms that have superior theoretical performance; practitioners develop implemented algorithms that have superior empirical performance. In an ideal world, practitioners would implement and analyze the empirical performance of the abstract algorithms developed by theoreticians, while theoreticians would analyze the theoretical performance of the implemented algorithms developed by practitioners. In the real world, there is often a wide gap between the practical and theoretical communities; understanding how to close this gap is critical to making theoretical computer science more relevant to applications. The edit distance problem is then an excellent lens through which to understand how the theoretical analysis of algorithms impacts practical implementations.

There are many ways to approach the practice/theory gap in a problem like edit distance. We take one that is systematic and focused on the way theoreticians analyze edit distance algorithms, rather than on the algorithms themselves. From a practical perspective, theoretical analysis has two goals.³⁷ The first goal is to predict the empirical performance of an algorithm, either in an absolute sense or relative to other algorithms. The second goal is to be a yardstick that drives the design of novel algorithms that perform well in practice. In this article, we systematically survey the types of theoretical analysis techniques that have been applied to edit distance and evaluate the extent



to which each one has achieved these two stated goals.

To focus this presentation, we consider only the simplest version of the edit distance problem, where both strings have equal length n and are over a constant sized alphabet; moreover, the algorithm only needs to return the edit distance and not the sequence of edits that achieve it. We also focus only on the runtime analysis as opposed to the memory use. We start by summarizing the state-of-the-art practical implementations then go through the various types of theoretical analysis that has been applied to the edit distance

problem: traditional worst-case analysis, worst-case analysis parametrized by the edit distance, worst-case analysis parametrized by entropy and compressibility, average-case analysis, semi-random models, and advice-based models. For each technique, we evaluate the extent to which it has achieved the prediction and design goals of theoretical analysis on the edit distance computation problem. We will not assume any knowledge of biology or any knowledge of computer science beyond an undergraduate level. We then conclude with a discussion of open problems and their potential solutions.


State-of-the-Art Implementations and Algorithms

Here, we will briefly outline the algorithms that are used in state-of-the-art implementations as well as highlight their empirical performance. The classical algorithm taught in many Algorithms courses is called Needleman-Wunsch.⁴³ It builds a two-dimensional matrix D where the value at $D[i, j]$ is the edit distance between the i -long prefix of A and the j -long prefix of B . This matrix can be computed in the standard dynamic programming manner using the recurrence $D[i, j] = \min(D[i - 1, j] + 1, D[i, j - 1] + 1, D[i, j] + I_{i,j})$, where $I_{i,j}$ is 0


if the i^{th} character of A is equal to the j^{th} character of B and 1 otherwise. The edit distance is then the value at $[n, n]$ (recall that n is the length of the strings). Doubling banded alignment⁴¹ is a modification of Needleman-Wunsch whose main idea is to reduce the number of cells of D that need to be computed. It uses the idea that if one only computes the values of $D[i, j]$ within a diagonal band of width d (that is, when $|i - j| \leq d/2$), then by checking if $D[n, m] \leq d$ one can either determine the edit distance if it is at most d or determine the edit distance is greater than d . Using this idea, it runs the checking algorithm repeatedly by doubling the value of d until the edit distance is found. The Myers' bit-parallel technique³² is a hardware optimization of Needleman-Wunsch that encodes the dynamic programming matrix into bitvectors and then rewrites the recurrences in terms of word-sized bitvector operations. These three algorithms/techniques comprise the core of all state-of-the-art implementations today.

There are at least three broadly used software libraries/tools that implement edit distance computation.^{14,15,40} Edlib⁴⁰ is optimized specifically for the edit distance problem, while SeqAn¹⁵ and Parasail¹⁴ are designed for more general alignment problems, but support edit distance as a special case. Edlib and SeqAn both implement the banded alignment algorithm using Myers' bit-parallel technique.³² Myers' technique does not change the asymptotic runtime but gives a significant constant speedup in practice. Parasail¹⁴ implements the Needleman-Wunsch algorithm using high-performance computing techniques. These include both task-level parallelism (that is, multi-threading) and instruction-level parallelism (that is, SIMD vectorized instructions). Parasail's code is also customized during compilation for the instruction set of the host architecture. Another implementation, BGSA,⁴⁴ implements the Needleman-Wunsch algorithm with the Myers' bit-parallel technique but supports multi-core, task-level, and instruction-level parallelism for batch execution.

There are also approaches to speed-up edit distance by using specialized hardware, such as GPUs, FPGAs, or even custom-designed processors (for references, see Alser et al.³). These re-



In the real world, there is often a wide gap between the practical and theoretical communities; understanding how to close this gap is critical to making theoretical computer science more relevant to applications.



sult in orders-of-magnitude constant-time speedups over their CPU counterparts in practice. However, until there is more widespread availability and integration of such specialized hardware in bioinformatics compute infrastructures, these tools are unlikely to be widely used.

How well do the widely used implementations perform? On two sequences of 1 million nucleotides each, one of the fastest implementations (edlib) takes 1.1 seconds for sequences with edit distance of $0.01n$ and 30 seconds for sequences with edit distance of $0.40n$, on a single core server.⁴⁰ For sequences of 100,000 nucleotides each, the runtimes are 0.01s and 0.40s, respectively. As we will later see, this corresponds to the theoretical prediction that the runtime deteriorates with increasing edit distance. For many applications, these runtimes are good enough. However, edit distance computation remains a bottleneck for applications that use it as a subroutine to make thousands or millions of comparisons (for example, comparing long reads against each other³⁸).

Traditional Worst-Case Analysis

The most common way to analyze running time, taught in undergraduate computer science classes, is traditional worst-case analysis. For example, it says the classical merge sort algorithm runs in $O(n \log n)$ worst-case time, which formally means there exists a constant c such that for any large-enough input of n elements, merge sort takes at most $cn \log n$ time. Has traditional worst-case analysis led to the design of edit distance algorithms that perform well in practice? There are two candidates. The first is the classical Needleman-Wunsch algorithm; it was originally described in Wagner and Fischer,⁴³ which gave the dynamic programming recurrence and proved the runtime is $\Theta(n^2)$. Their algorithm modified an earlier dynamic programming algorithm³⁵ whose run time was $\Theta(n^3)$.^a It seems likely then that traditional worst-case analysis was a

^a As a historical note, the algorithm presented in Needleman and Wunsch³⁵ is not the algorithm we call "Needleman-Wunsch" today. The "Needleman-Wunsch" algorithm was described later.⁴³

driving force behind the creation of the Needleman-Wunsch algorithm. Moreover, the algorithm is a success in practice because many implemented algorithms, including some of the fastest ones, are either modifications of it or use it as a subroutine. This includes both banded alignment and the Myers' bit-parallel technique used by edlib, parasail, and SeqAn. Thus, though Needleman-Wunsch is not the fastest algorithm in practice or in theory, it exemplifies how traditional worst-case analysis achieved the design goal.

The second candidate is the fastest known algorithm under traditional worst-case analysis—the Four-Russians speedup to Needleman-Wunsch.^{6,29} It takes $\Theta(n^2/\log^2 n)$ time (in a unit-cost RAM model). The algorithm was clearly designed to optimize the runtime under traditional worst-case analysis. But how does it perform in practice? Does the $\Theta(\log^2 n)$ speedup outweigh the additional constant factors due to higher algorithm and data structure complexity? To answer this question, there have been implementations and experimental evaluations of this algorithm.^{22,36} The improvement over Needleman-Wunsch was a factor of about 5 for $n = 2^{18}$, and, extrapolating from Rejmon,³⁶ would not exceed 10 even for sequences of a billion characters. Thus, in practice, the Four Russians algorithm is dominated by other algorithms that have $\Theta(n^2)$ run time but have better constant factors (for example, Myers' bit-parallel algorithm).³⁶ Moreover, the fastest implementations of edit distance today^{14,15,40} do not implement the Four Russians algorithm, even as a subroutine, highlighting how exploiting the properties of the CPU (for example, the bit-parallel or SIMD implementations) can bring constant speedups that in practice outperform asymptotic speedups. We therefore conclude that in the case of the Four Russians algorithm, traditional worst-case analysis has led us astray into the design of an algorithm that is not practically useful.^b

^b While the algorithm itself was not practically useful, one could argue it had an impact on practice because some of its ideas were later used by algorithms such as Myers' bit-parallel algorithm. While we may deem an algorithm not practically useful, it may nevertheless have been an important steppingstone on the road to another practically useful algorithm.

Has traditional worst-case analysis been able to accurately predict the empirical performance of algorithms? The analysis of Needleman-Wunsch shows that $\Theta(n^2)$ time is taken for every input instance, not just in the worst-case. The Four-Russians runtime analysis is similar in that it also holds for all inputs, not just worst-case ones. Therefore, the predictions of traditional worst-case analysis accurately reflect these algorithms' runtime on real data.

Can traditional worst-case analysis lead to the design of new algorithms that perform well in practice? A famous recent result states that under the strong exponential time hypothesis, there cannot be an $(n^{2-\delta})$ algorithm, for any $\delta > 0$.⁷ Such an algorithm is called strongly sub-quadratic. There are other “barrier” results of this type which we will not elaborate on.^{1,2,10} These results make it unlikely that better algorithms can be designed using traditional worst-case analysis as a guide. However, they say nothing about the existence of provably and substantially better algorithms, if they are analyzed using a different technique than traditional worst-case analysis.

Worst-Case Analysis Parametrized by the Edit Distance

One step away from traditional worst-case analysis is parametrized worst-case analysis, which is worst-case analysis in terms of properties of the input besides just its size n . In the case of edit distance, the parameter that has proven most useful is the edit distance itself, usually denoted by k (note that $k \leq n$). The most notable algorithm designed using k -parametrized worst-case analysis is doubling banded alignment.⁴¹ Parametrized analysis shows that it computes the edit distance in time $\Theta(kn)$, on all inputs. There are several other algorithms parametrized by k ,^{19,33} they achieve various trade-offs between k and n and have some other differences outside the scope of this article. Doubling banded alignment is the most notable of these because it is very simple to describe and implement (that is, it does not use any complex data structures) and forms a main component of one of the empirically fastest implementations today—edlib.⁴⁰

K -parametrized analysis predicts several ways in which doubling banded

alignment is theoretically an improvement over Needleman-Wunsch. The first is when $k = o(n)$, doubling banded alignment scales sub-quadratically with the input size, while Needleman-Wunsch does not. The second is the closer the sequences are (that is, the smaller the edit distance), the smaller the runtime. Thus, the algorithm captures a natural notion of complexity of the input and takes advantage of input that is less complex, while Needleman-Wunsch does not. These two predictions are reflected empirically as well since the runtime analyses are for all inputs and both algorithms do not hide any significant implementation constants. Third, even when $k = \Theta(n)$, the empirical advantage of doubling banded alignment over Needleman-Wunsch can be significant.⁴⁰

However, when $k = \Theta(n)$, the fact remains the banded algorithm scales quadratically with the input size, both in theory and in practice. Unfortunately, this is the case for the following predominant application of edit distance. Biological sequences evolve from each other via a mutation process, which, for the purposes of this discussion, can be considered as mutating each position with some constant probability, independently for each position. The mutation probability is, for the most part, independent of n and the edit distance is therefore proportional to n with a constant called divergence. For example, the sequence divergence in coding regions of genes between human and other species is 1%–2% for bonobo and about 19% for mouse.¹² Thus, regardless of species, and even for species that are very close, the edit distance is still approximately a constant proportion of the sequence length. This illustrates the importance of using theoretical analysis to design exact algorithms that scale sub-quadratically when $k = \Theta(n)$.

In summary, the k -parameterized worst-case analysis technique has been a tremendous success for edit distance. It led directly to the design of the banded alignment algorithm, which is widely used in practice, and it can predict the empirical improvement of banded alignment over Needleman-Wunsch. However, it still did not produce an algorithm that scales strongly sub-quadratically with the input size

for applications where $k = \Theta(n)$. Moreover, it may not be able to do so in the future due to the barrier results against strongly sub-quadratic algorithms in the worst-case framework.^{1,2,7,10}

Worst-Case Analysis Parametrized by Entropy and Compressibility

Another way to parameterize the analysis of edit distance algorithms is by the entropy of the input, h . Entropy is a value between 0 and 1 which measures the amount of order in the strings.³⁴ Strings containing short repetitive patterns tend to have a lower entropy. As an extreme case, the string of all T s has entropy 0 while a string generated uniformly at random has entropy close to 1. Intuitively, an edit distance algorithm could take advantage of the repetitive patterns to run faster on strings with lower entropy. An algorithm following this intuition was developed by Crochemore et al.¹³ and runs in time $O(hn^2/\log n)$ for most inputs. When the input strings have low entropy, this is theoretically faster than Needleman-Wunsch. Note that this runtime is not comparable with banded alignment, since the edit distance can be low for strings with high entropy, and vice versa.

A related notion are algorithms that compute the edit distance of A and B directly from their compressed representations. The time to compress two strings is in most cases asymptotically negligible compared to the time it takes to compute the edit distance. Hence, one could solve the edit distance computation problem by first compressing A and B and then running an edit distance algorithm on the compressed strings. For example, there is an algorithm to compute edit distance between two run-length encoded³⁴ strings in $O(\ell n)$ time^{5,13,26} where ℓ is the size of the encoded strings. For a more general class of compression algorithms called straight-line programs,²⁰ there is an algorithm¹⁷ that runs in $O(\ell n \sqrt{\log(n/\ell)})$ time. In these cases, the algorithms are designed to optimize the runtime with respect to ℓ -parametrized worst-case analysis.

Unfortunately, these algorithms have not been broadly applied in practice. A major reason is because a long DNA sequence, while exhibiting some

repetitive patterns, still has high entropy (for example, 0.85²⁵) and low compressibility.^c Thus, while worst-case analysis parametrized by entropy or compressibility has led to the design of novel algorithms, it has not achieved the design or prediction goals of theoretical analysis since these algorithms have not been useful in practice.

Average-Case Analysis

One of the pitfalls of traditional and parametric worst-case analysis is that it assumes the inputs are chosen by an adversary who wants to make things as difficult as possible for the algorithm. In practice, however, biological sequences are not chosen this way and may have much nicer properties. On such instances, the optimal path in the alignment matrix stays very close to the diagonal, even much tighter than the band of width d used by the doubling banded alignment algorithm. For example, a heuristic algorithm that stops the doubling banded alignment prematurely and reports the value of $[n, n]$ would still likely report the correct edit distance. Worst-case analysis cannot tell us either the probability of success nor tell us how long to double to guarantee a high probability of success.

One theoretical analysis technique to alleviate the shortcoming of worst-case analysis is average-case analysis (sometimes called distributional analysis). Here, the inputs are assumed to be drawn from some kind of distribution, and what is measured is the expected performance over this distribution. Other alternatives include measuring the performance that can be achieved with high probability and/or measuring the performance in terms of a trade-off with the probability of the algorithm achieving it. Re-analyzing the performance of existing algorithms under this model would not be beneficial for most of the algorithms we have looked at so far (that is, Needleman-Wunsch, Four Russians, and banded alignment) because their runtime holds for all inputs, not just worst-case ones. However, if we have an input distribution in mind, we can

design a new algorithm to work well under that model and validate it empirically. Researchers have aimed to design an algorithm with a strongly sub-quadratic expected runtime.

An obvious first attempt is to assume each string of length n is randomly drawn uniformly and independently from the universe of all strings of length n . For each string, this corresponds to generating each character independently, following an identical categorical distribution for each position. However, this model does not capture essential properties of real data and has not resulted in any useful algorithms or predictive running time analysis. When the real input strings are evolutionary related, then the assumption they are independent of each other is highly inaccurate. Even in the case that the real input strings are not evolutionary related, the uniformity assumption remains unrealistic since biological sequences have evolved to serve a function and thus exhibit non-random behavior.

A more accurate model uses an indel channel.¹⁶ Here, the first string is chosen uniformly at random, while the second string is obtained by randomly mutating each nucleotide, with a probability at most a small constant. The algorithm of Ganesh and Sy¹⁶ runs in $O(n \log n)$ time and, with high probability over this input distribution, returns the correct edit distance. This algorithm's runtime is significantly better than just strongly sub-quadratic. However, we are not aware of any implementation, and, for this algorithm to be practical, it must prove it performs reasonably well even for real input that does not follow the model's distribution.

A successful application of average case analysis is in the analysis of the “furthest reaching” algorithm, which was proposed in Myers³¹ and Ukkonen.⁴¹ This analysis combines average-case with parametrized analysis. It runs in $O(nk)$ time in the worst case,⁴¹ but an average case analysis gives $O(n + k^2)$.³¹ The model used in this analysis is like the one previously mentioned, which describes the model of Ganesh and Sy.¹⁶ While this algorithm does not break the quadratic time barrier and is not used by state-of-the-art edit distance algorithms, it has nevertheless been practically successful. It was the

^c If compressing multiple DNA sequences together, much better compressibility is possible. However, edit distance has not been typically performed against such collections.


basis of an implementation of the “diff” Unix tool³¹ and serves as a foundation for generalizations of the edit distance problem (for example, to affine gap penalties as in Marco-Sola et al.²⁷).

Semi-Random Models


One can view average-case analysis and worst-case analysis as two extremes. The main idea of more sophisticated semirandom models is to achieve a middle ground between an adversary and randomness.^{4,8,23} Here, the performance is measured as worst-case over the choices of the adversary and average case over the random distribution. The semi-random models for edit distance have only led to approximation, rather than exact, algorithms. A c -approximation algorithm is one that returns a value at most a multiplicative factor of c (called the approximation ratio) away from the edit distance. Approximation algorithms relax the requirement of finding the exact solution in exchange for better runtime. For an edit distance approximation algorithm to be relevant in practice, the approximation ratio must be a constant very close to one for example, 1.01). For example, even a 3-approximation algorithm would not be able to always distinguish two random DNA sequences (that is, expected edit distance of approximately $0.53n$ (personal simulations, data not shown)) from a mouse and human sequence pair (that is, edit distance of about $0.19n$).¹² Thus, the usefulness of the models is predicated on their ability to achieve a tiny approximation ratio.

Smoothed analysis is based on the idea that worst-case inputs designed to fool algorithms are not very stable; that is, if bad input is tweaked a little, the algorithm no longer performs poorly.³⁷ In this model, an adversary first picks an input (that is, a worst-case choice), but then some small random noise is added to this input. The algorithm’s performance on a particular adversarial input is defined as the expected performance over the distribution of noisy inputs centered around the adversarial input. The algorithm’s overall performance is then defined as the worst-case performance over all choices of the adversary’s input.

Smoothed analysis was considered for edit distance in Andoni and



Smoothed analysis is based on the idea that worst-case inputs designed to fool algorithms are not very stable; that is, if bad input is tweaked a little, the algorithm no longer performs poorly.



Krauthgamer.⁴ In their model, an adversary chooses the two input strings and a longest common subsequence between them. Then, each position is randomly perturbed with a small probability p (that is, the nucleotide is replaced with another random one). However, the perturbations are constrained so the positions of the longest common subsequence are perturbed identically. This model captures the idea that for two evolutionary related strings, if we view their longest common subsequence as their ancestral sequence, then all the nucleotides outside this common subsequence would have evolved somewhat independently of each other. Allowing the adversary to choose some worst-case values for them gives her too much power; instead, some noise is added to them to make them more independent.

In Andoni and Krauthgamer,⁴ the algorithms’ approximation ratios are not precisely derived (that is, big-Oh notation is used). There has also been another line of work focusing on fast approximation algorithms under traditional worst-case analysis with the best-known approximation constant of 1680.¹¹ In both cases, the algorithms achieve strongly sub-quadratic run time, something that could not be done by exact algorithms using worst-case analysis for $k = \Theta(n)$. However, without a precise derivation of a tiny approximation ratio, or an implementation and validation on real data, it is difficult to predict the applicability of these algorithms.

Asymmetric (p, B) -pseudo-random model. As we mentioned previously, a string chosen at random does not reflect a biologically evolved sequence. More precisely, a uniformly random string has the property that the probability that any two non-overlapping equal-length substrings are identical decreases with their length. In fact, once their length exceeds a certain critical threshold, this probability is, for all practical purposes, zero. This property is in contrast with biological sequences, which are often composed of long similar elements called repeats. It is true that longer repeats tend to be less frequent and less similar, but this decrease does not happen at the same rate as for random strings. This was captured in a more realistic

model of randomness, proposed in Kuszmaul.²³ They say that a string is (p, B) -pseudorandom if the edit distance of any two disjoint B -long substrings is at least a fraction p of their length. This generalizes uniform randomness, that is, a uniformly random string is $(\Omega(1), O(\log n))$ -pseudorandom with high probability.²³ But by choosing p and B appropriately, we can more realistically match the repeat properties of a biological string.

The asymmetric (p, B) -pseudorandom model²³ is to first choose a string at random from all (p, B) -pseudorandom strings and then have an adversary choose the other string and modify some small portion of the pseudorandom string. By allowing the adversary to choose one of the strings, this model allows the two strings to be evolutionarily related. Moreover, the additional power of the adversary to modify the pseudorandom string makes the model even more realistic, because a true biological sequence would usually have some substrings that break the (p, B) , even when the values of p and B are chosen to minimize these cases.

This model is used in Kuszmaul²³ to design several new algorithms. The main algorithm has a runtime $\tilde{O}(nB)$ (the \tilde{O} notation is like O but ignores log factors), which is strongly sub-quadratic if B is strongly sub-linear. However, as in smoothed analysis, the algorithm is only an approximation algorithm, and the exact approximation ratio is not calculated. Thus, whether this model leads to any practical algorithms remains to be seen. However, the fact that it seems to intuitively better capture biological reality while lending itself to theoretical analysis is promising.

The random model of Boroujeni et al. A recently proposed model⁸ has the adversary first choose a seed string s and then constructs A by permuting s uniformly at random. After observing A and the random permutation, the adversary then constructs B . The algorithm given by Boroujeni et al.⁸ is an approximation algorithm with an expected runtime is $O(n^{1.898})$, which is strongly sub-quadratic. The approximation ratio is $1 + o(1)$, which is low enough to be of practical relevance.

This analysis model has led to the design of an algorithm that brakes

through a barrier of previous models, that is, it achieves a low approximation ratio while maintaining sub-quadratic time. However, the actual runtime improvement is only $n^{0.102}$, which is less than 9 for inputs up to a billion nucleotides long. It is unlikely that this improvement would justify the additional overhead of a more complex algorithm. Nevertheless, the model can ultimately be successful if the runtime of the algorithm can be improved, the implementation of the algorithm kept simple, and the usefulness of the model empirically validated.

In summary, none of the semi-random models have yet led to a better understanding of the performance of existing algorithms or to the design of algorithms that perform well in practice. The proposed algorithms, at least in their current form, are not promising: the algorithms of Andoni and Krauthgamer⁴ and Kuszmaul²³ have impractical approximation ratios, while the Boroujeni et al. algorithm⁸ improves the runtime by a factor that is too small to have an effect in practice. However, the models themselves are promising, and can ultimately be successful if the runtime of the algorithms can be improved, the complexity of the algorithms kept low, and the usefulness of the models empirically validated. These ongoing efforts may eventually result in an algorithm that outperforms banded alignment, in practice, on inputs with $k = \Theta(n)$.

Analyzing with Advice

The biological problem is usually more general than the mathematical abstraction created for it. Sometimes the algorithm has access to other information, not included in the problem definition, that can serve as advice. In this case, one can both expand the problem definition and the theoretical analysis to incorporate this advice. An advice-based analysis measures the algorithm runtime with respect to the amount of such advice used. In the case of edit distance, Goldwasser and Holden¹⁸ argue that, for an input instance A and B , it is possible to have access to a collection of correlated instances. Intuitively, a correlated instance is one whose sequence of edits in the shortest edit sequence is, with some high probability, like the

one between A and B . They show that if an algorithm has access to $O(\log n)$ of such instances, it can find the edit distance between A and B in $O(n \log n)$ time, with high probability.

This approach has not been implemented but is promising in the sense the runtime is not just sub-quadratic but nearly linear, and the algorithm is exact and seems easy to implement. Compared to the banded alignment algorithms, the $O(n \log n)$ algorithm is likely to have significant empirical speed improvements for moderately sized inputs even for small values of k . Unfortunately, it is not clear how correlated instances can be obtained in practice and whether they would be captured by the Goldwasser and Holden definition.¹⁸

Conclusion

We have surveyed the various approaches to the theoretical analysis of edit distance algorithms, focusing on whether these approaches have led to the design of algorithms that are fast in practice and to the theoretical prediction of the empirical runtimes of existing algorithms. We showed the track record has been mixed. On one hand, a few algorithms widely used in practice (Needleman-Wunsch, doubled banded alignment, furthest reaching) have been born out of theoretical analysis and their empirical performance is captured well by theoretical predictions. On the other hand, very little of the algorithms developed using theoretical analysis as a yardstick since then have had any practical relevance.

From a practical perspective, a major open problem is to implement an algorithm with linear-like empirical scaling on inputs where the edit distance is linear in n . Theoretical analysis has the potential to lead the way in achieving this goal. Semi-random models are the most promising approach, due to the barrier results for worst-case analysis. A reasonable model which might give a good balance between capturing reality and ease of analysis is one where B is assumed to have evolved from A via a mutation process (like Ganesh and Sy¹⁶). For the theoretical work to have practical relevance, however, it must be implemented and validated; in particular, the algorithm's runtime


and accuracy must be robust to data not drawn from the modeled distribution and the constant-time overhead of the algorithm must not override the asymptotic gains. Unfortunately, current community incentives leave the implementation and validation of an algorithm as an afterthought that is rarely pursued.

To solve this open problem and, more generally, close the gap between theory and practice, implementation and validation cannot be treated as a separate step of the process. We need multi-disciplinary teams that can interleave the theoretical analysis of algorithms with their implementation and validation. Allowing for a back-and-forth between practitioners and theoreticians during the development process can allow iterations over the theoretical model and practical heuristics that would otherwise be impossible. Such teams will be able to use both empirical and theoretical performance as a yardstick and will be better able to develop algorithms whose empirical performance is not only superior but also accurately captured by theoretical analysis.

It is also essential to continue to study the relationship between theoretical analysis and practical implementations. One of the limitations of this study is that it only focuses on a single problem, making it difficult to draw more general conclusions. More studies such as this one could establish patterns and identify clear directions to closing the practice/theory gap. In a recent paper,³⁰ we applied the same lens more broadly (but also more anecdotally) to study the analysis of algorithms in sequencing bioinformatics, attempting to establish patterns and draw conclusions for that domain. Ultimately, the relationship between theory and practice must be understood not only from a technical angle but also from a social science and philosophy of science perspective. In other disciplines, this relationship is studied by philosophers of science (for example, in education policy²¹) and so a similar approach may be fruitful in computer science.

Acknowledgments

This article was inspired by the online videos for the 2014 class “Beyond

Worst-Case Analysis” by Tim Roughgarden. This material is based upon work supported by the National Science Foundation under Grants No. 1439057, 1453527, and 1931531. 

References

1. Abboud, A., Backus, A., and Williams, V.V. Tight hardness results for LCS and other sequence similarity measures. In *Proceedings of the 2015 IEEE 56th Annual Symp. Foundations of Computer Science*, 59–78.
2. Abboud, A., Hansen, T.D., Williams, V.V., and Williams, R. Simulating branching programs with edit distance and friends: Or a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM Symp. Theory of Computing*, (2016), 375–388.
3. Alser, M., Hassan, H., Kumar, A., Mutlu, O., and Alkan, C. Shouji: A fast and efficient pre-alignment filter for sequence alignment. *Bioinformatics* 35, 21 (2019), 4255–4263.
4. Andoni, A. and Krauthgamer, R. The smoothed complexity of edit distance. *ACM Trans. Algorithms* 8, 4 (2012), 1–25.
5. Arbell, O., Landau, G.M., and Mitchell, J.S.B. Edit distance of run-length encoded strings. *Information Processing Letters* 83, 6 (2002), 307–314.
6. Arlazarov, V., Diniz, Y., Kronrod, M., and Faradzhev, I. On economical construction of the transitive closure of an oriented graph. *Doklady Akademii Nauk* 194. Russian Academy of Sciences, (1970), 487–488.
7. Backurs, A. and Indyk, P. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Computing* 47, 3 (2018), 1087–1097.
8. Boroujeni, M., Seddighin, M., and Seddighin, S. Improved algorithms for edit distance and LCS: Beyond worst case. In *Proceedings of the 14th Annual ACM-SIAM Symp. Discrete Algorithms*, (2020), 1601–1620.
9. Brinda, K., Boeva, V., and Kucherov, G. Ocoo: An online consensus caller, (2017); arXiv:1712.01146.
10. Bringmann, K. and Künnemann, M. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of the 2015 IEEE 56th Annual Symp. Foundations of Computer Science*, 79–97.
11. Chakraborty, D., Das, D., Goldenberg, E., Koucky, M., and Saks, M. Approximating edit distance within constant factor in truly sub-quadratic time. In *Proceedings of the 2018 IEEE 59th Annual Symp. Foundations of Computer Science*.
12. Cooper, G.M., Brudno, M., and NISC Comparative Sequencing Program. Quantitative estimates of sequence divergence for comparative analyses of mammalian genomes. E.D. Green, S. Batzoglou, and A. Sidow, (eds). *Genome Res.* 13, 5 (May 2003), 813–820.
13. Crochemore, M., Landau, G.M., and Ziv-Ukelson, M. A subquadratic sequence alignment algorithm for unrestricted scoring matrices. *SIAM J. Computing* 32, 6 (2003), 1654–1673.
14. Daily, J. Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. *BMC Bioinformatics* 17, 1 (2016), 81.
15. Döring, A., Weese, D., Rausch, T., and Reinert, K. Seqan an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics* 9, 1 (2008), 11.
16. Ganesh, A. and Sy, A. Near-linear time edit distance for indel channels. In *Proceedings of the 20th Intern. Workshop on Algorithms in Bioinformatics* 172, Leibniz International Proceedings in Informatics. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, (2020), 17:1–17:18.
17. Gawrychowski, P. Faster algorithm for computing the edit distance between SLP-compressed strings. In *Proceedings of the 2012 Intern. Symp. String Processing and Information Retrieval*. Springer, 229–236.
18. Goldwasser, S. and Holden, D. The complexity of problems in p given correlated instances. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conf. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik*, (2017).
19. Gusfield, D. *Algorithms on Strings, Trees, and Sequences*. (1997).
20. Hermelin, D., Landau, G.M., Landau, S., and weimann, O.A. A unified algorithm for accelerating edit-distance computation via textcompression, (2009); arXiv:0902.2649.
21. Joyce, K.E. and Cartwright, N. Bridging the gap between research and practice: Predicting what will work locally. *Amer. Educ. Research J.* 57, 3 (2020), 1045–1082.
22. Kim, Y., Na, J.C., Park, H., and Sim, J.S. A space-efficient alphabet-independent Four-Russians’ lookup table and a multithreaded Four-Russians’ edit distance algorithm. *Theoretical Computer Science* 656, (2016), 173–179.
23. Kuzmaul, W. Efficiently approximating edit distance between pseudorandom strings. In *Proceedings of the 30th Annual Symp. on Discrete Algorithms*. SIAM, (2019), 1165–1180.
24. Levenshtein, V.I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10, 707–710.
25. Loewenstein, D. and Yanilos, P.N. Significantly lower entropy estimates for natural DNA sequences. *J. Computational Biology* 6, 1 (1999), 125–142.
26. Mäkinen, V., Ukkonen, E., and Navarro, G. Approximate matching of run-length compressed strings. *Algorithmica* 35, 4 (2003), 347–369.
27. Marco-Sola, S., Moure, J.C., Moreto, M., and Espinosa, A. Fast gap-affine pairwise alignment using the wavefront algorithm. *Bioinformatics* 37, 4 (Feb. 2021), 456–463.
28. Maretty, L. et al. Sequencing and de novo assembly of 150 genomes from Denmark as a population reference. *Nature* 548, 7665 (2017), 87–91.
29. Masek, W.J. and Paterson, M.S. A faster algorithm computing string edit distances. *J. Computer and System Sciences* 20, 1 (1980), 18–31.
30. Medvedev, P. The limitations of the theoretical analysis of applied algorithms. (2022); arXiv:2205.01785.
31. Myers, E.W. An $O(nd)$ difference algorithm and its variations. *Algorithmica* 1, 1–4 (1986), 251–266.
32. Myers, G. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *JACM* 46, 3 (1999), 395–415.
33. Navarro, G. A guided tour to approximate string matching. *ACM Computing Surveys* 33, 1 (2001), 31–88.
34. Navarro, G. *Compact Data Structures: A Practical Approach*. Cambridge University Press (2016).
35. Needleman, S.B. and Wunsch, C.D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Molecular Biology* 48, 3 (1970), 443–453.
36. Rejmon, M. Multi-Threaded implementation of four russians edit distance algorithm. *Bachelor’s thesis*. Czech Technical University in Prague, Faculty of Information Technology, (2019).
37. Roughgarden, T. Beyond worst-case analysis. *Commun. ACM* 62, 3 (Mar.2019), 88–96.
38. Sahlin, K. and Medvedev, P. De novo clustering of long-read transcriptome data using a greedy, quality value-based algorithm. *J. Computational Biology* 27, 4 (2020), 472–484.
39. Sarkar, H., Zakeri, M., Malik, L., and Patro, R. Towards selective-alignment: Bridging the accuracy gap between alignment-based and alignment-free transcript quantification. In *Proceedings of the 2018 ACM Intern. Conf. Bioinformatics, Computational Biology, and Health Informatics*, 27–36.
40. Šošić, M. and Šikić, M. Edlib: a C/C++ library for fast, exact sequence alignment using edit distance. *Bioinformatics* 33, 99 (2017), 394–1395.
41. Ukkonen, E. Algorithms for approximate string matching. *Information and Control* 64, 1–3 (1985), 100–118.
42. Vaser, R., Sovic, I., Nagarajan, N., and Sikic, M. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Research* 27, 5 (2017), 737–746.
43. Wagner, R.A. and Fischer, M.J. The String-to-String Correction Problem. *JACM* 21, 1 (1974), 168–173.
44. Zhang, J. et al. BGSA: A bit-parallel global sequence alignment toolkit for multi-core and many-core architectures. *Bioinformatics* 35, 13 (2019), 2306–2308.
45. Zook, J.M. et al. A robust benchmark for germline structural variant detection. *BioRxiv*. (2019), 664623.

Paul Medvedev is a professor in the Department of Computer Science and Engineering and the Department of Biochemistry and Molecular Biology, and the Director of the Center for Computational Biology and Bioinformatics at the Pennsylvania State University, University Park, PA, USA.



This work is licensed under a Creative Commons Attribution-NonDerivs International 4.0 License.