The theoretical analysis of sequencing bioinformatics algorithms (DRAFT)

Paul Medvedev[1,2,3]

[1]Department of Computer Science and Engineering, Pennsylvania State University
[2]Department of Biochemistry and Molecular Biology, Pennsylvania State University
[3]Center for Computational Biology and Bioinformatics, Pennsylvania State University

November 16, 2020

Abstract

The theoretical analysis of algorithm performance has been an important tool in the engineering of algorithms in many application domains. Its goals are to predict the empirical performance of an algorithm and to be a yardstick that drives the design of novel algorithms that perform well in practice. However, when it comes to sequencing bioinformatics, an application area concerned with algorithms for biological sequencing data, theoretically analyzing algorithms has been challenging has had a mixed record in achieving its goals. This survey is, to the best of our knowledge, the first attempt to systematically and critically study the application of theoretical analysis of algorithm techniques to sequencing bioinformatics. We explore some of the techniques that have been applied and the extent to which they have been successful. We use the edit distance computation problem as a case study but also explore the problems of genome assembly, structural variation detection, and compact data structures. We conclude by discussing steps that can be taken to help improve the impact and applicability of the theoretical analysis of algorithms in sequencing bioinformatics.

1 Introduction

The theoretical analysis of algorithm performance has been an important tool in the engineering of algorithms in many applications domains. Techniques for the theoretical analysis of algorithms (which we abbreviate as TAA) include traditional worst-case analysis and more advanced techniques such as parametrized analysis, average-case analysis, or semi-random models. The goals of TAA are twofold [50]. One is to *predict* the empirical performance of an algorithm, either in an absolute sense or relative to other algorithms. The second is to be a yardstick that drives the *design* of novel algorithms that perform well in practice. TAA has been a tremendous success, directly responsible for the design and performance prediction of many algorithms used in practice.

However, the continuing success of TAA should not be taken for granted. Consider *sequencing bioinformatics (which we will abbreviate as SB)*, a inter-disciplinary field that uses algorithms to extract biological meaning from sequencing data. SB has posed several significant challenges to the application of TAA. First, traditional worst-case analysis is in most cases too pessimistic when it comes to real biological data and fails to separate high performing algorithms, which take advantage of the structure of real data, from poorly performing ones that do not. Second, SB algorithms are usually developed, analyzed, and applied under significant time pressure, which limits the applicability of more sophisticated TAA techniques. Finally, because SB researchers require a broad inter-disciplinary skill set, they often lack the technical expertise necessary to apply more sophisticated TAA techniques. None of these challenges are unique to SB, however, their combination is formidable.

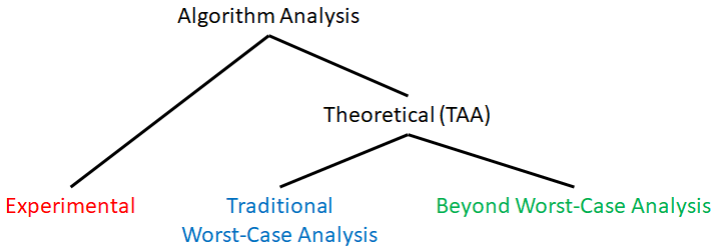

Figure 1: A hierarchy of techniques for analyzing the performance of an algorithm. These may be applied to analyzing the running time, memory usage, or accuracy.

To date, there have not been any attempts to systematically and critically study the application of TAA to SB. What TAA techniques have been applied in SB? To what extent has TAA achieved its two goals in SB? What can be done to improve the applicability of TAA in SB? Is TAA even necessary or can SB rely mostly on experimental evaluation? In this survey, we attempt to answer these questions, which we view as a crucial first step to increasing the impact that TAA has in SB. As we will discuss in Section 6, such improvements would address several current shortcomings of the SB field. Moreover, novel theoretical techniques developed for SB are likely to be applicable to other domains.

We start our survey with a brief background of TAA and SB (Section 2). Because the SB field is too broad to survey in its entirety, we will use the edit distance computation problem as a case study (Section 3). This is arguably the most theoretically studied SB problem and therefore allows us to present various TAA approaches and evaluate their success. This includes traditional worst-case analysis, parameterized analysis, random models, and others. One notable TAA technique whose tremendous success in SB is not well-exemplified by the edit distance computation problem is TAA for compact data structures, which we describe in Section 4. We then exemplify cases when TAA has failed to achieve its goals in SB using two problems: genome assembly (Section 5.1) and structural variation detection (Section 5.2). Finally, we conclude by trying to answer directly whether TAA has been successful in SB, arguing why experimental evaluation is not enough, and suggesting steps that can be taken to help develop novel TAA approaches (Section 6).

2 Background

2.1 Algorithm engineering and analysis

Algorithm engineering is the process of designing, implementing, analyzing, and validating an algorithm [53]. In this paper, we focus on the analysis phase, which measures an algorithm's performance with respect to some behavior. In SB, this behavior is usually runtime, memory usage, or accuracy. Different approaches for analyzing algorithm performance can be related via a hierarchy (Figure 1). In experimental analysis, performance is measured by running the implemented algorithm on a broad range of inputs. In theoretical analysis, a mathematical framework is derived for the performance of the algorithm as a function of the input (for example, saying that merge sort takes $\Theta(n \log n)$ time for inputs of n elements). Experimental and theoretical analysis are complementary [41, 62, 40] and we elaborate more on this in Section 6.

Within theoretical analysis, there are different approaches. The most common one used in SB is the type taught in undergraduate computer science classes. We will refer to this as *traditional worst-case analysis*. For example, it would give that for any input of n elements, for large enough n, merge sort would take at most $cn \log n$ time, for some constant c. Colloquially, we would say that merge sort runs in $\mathcal{O}(n \log n)$ worst-case time. As is commonly the case, we assume the RAM model of computation in this paper.

While traditional worst-case analysis achieves the design and prediction goals in many cases, there are also many failures [50]. There have been many attempts by the theoretical computer science community to overcome the limitations of traditional worst-case analysis, referred to as *beyond worst-case analysis* (Figure 1). Some of these techniques include parameterized analysis, modeling stability of the input instances,

TAA	Theoretical Analysis of Algorithms
SB	Sequencing Bioinformatics
TASBA	Theoretical Analysis of Sequencing Bioinformatic Algorithms
BWCA	Beyond Worst-Case Analysis

Table 1: List of non-standard acronyms used in the paper, in order of appearance.

planted and semi-random models, smoothed analysis, and average-case analysis [55]. We will see some examples in this paper, but for a treatment of these techniques in a context broader than bioinformatics, we refer the reader to the survey [50] and to the excellent course material on which the survey is based [20].

2.2 Sequencing bionformatics

In this section, we briefly describe the nature of sequencing data and three SB computational problems that arise from it. Sequencing technology refers to a type of instrument which can take a fragment of DNA or RNA from a "library" (i.e. a collection of fragments) and read its sequence from the 5' end (i.e. from the "beginning"). Each resulting sequence is called a *read*, and a single experiment produces millions or billions of reads. The instrument typically has a limit on how much sequence it can read from each fragment, anywhere from hundreds to thousands of nucleotides. When sequencing long molecules (e.g. a whole genome), the DNA is first amplified and then fragmented into much shorter fragments. These then form the library which the instrument sequences. The collection of reads from a single or multiple experiments forms a dataset to be computationally analyzed, and we refer to the engineering of algorithms that work on this data as *sequencing bioinformatics*.

There are hundreds of computational problems in SB, but we will focus on the following three in this survey. A classic use of sequencing technology is to sequence reads from a whole genome and then attempt to reconstruct the genome from the reads. This is known as the problem of *de novo genome assembly*, or simply genome assembly [56]. Once a genome is assembled for a species, it forms what is called a reference genome. Follow up studies would then sequence different individuals of the same species. However, such studies would not perform a *de novo* assembly. Instead, they would catalog the variations between the sequenced genome and the reference, under the assumption that the genome is unchanged from the reference in places where there is no alternative evidence. *Structural variation detection* is the problem of detecting variants that affect large regions (e.g. > 500 nucleotides) [39].

A lower level problem is the *edit distance* computation problem, which takes two strings A and B and returns the minimum number of substitutions, insertions, and deletions needed to transform A into B. This is sometimes referred as the unit-distance version of edit distance or the Levenshtein distance [32]. Edit distance computation is a problem older than SB, but it forms a subroutine of numerous widely-used SB tools (e.g. Racon [63]). In SB, edit distance is not necessarily computed between two reads [52]; it can be between substrings of reads [51], between a read and a genome assembly [63] or a transcriptome assembly [54], or between parts of different assemblies [66]. There are many variants of the edit distance computation problem, which itself is a special case of more general alignment problems. To focus our presentation, we consider only the most simple version of the problem. In particular, we assume that both strings have equal length n and are over a constant sized alphabet; moreover, the algorithm only needs to return the number of edits rather than the edits themselves.

In this paper, we attempt to draw a line between TAA's direct application to SB and its indirect influence. TAA can be credited with the development and analysis of methods which have become part of SB's toolbox, e.g. integer linear programming, hidden Markov models, clustering algorithms, suffix trees/arrays, Bloom filters, minhash sketches, machine learning, etc. Such indirect influence has undoubtedly been enormous. However, in this paper we focus on situations where theoretical analysis was applied to problems that are specific to SB, or at least for which SB was a major motivation. We are interested in cases when TAA led to the design or analysis of a SB algorithm, rather than an algorithm that later found an application in SB.

3 Case study in edit distance

In this section we survey the theoretical techniques that have been applied to analyze the runtime of algorithms for the edit distance computation problem. We do not describe the algorithms but rather focus on the analysis techniques themselves. For each technique, we evaluate the extent to which it has achieved the prediction and design goals of TAA on the edit distance computation problem.

3.1 Traditional worst-case analysis

Has traditional worst-case analysis led to the design of algorithms that perform well in practice? There are two candidates. The first is the classical Needleman-Wunsch algorithm, taught in nearly all introductory bioinformatics classes and textbooks. It is a dynamic programming algorithm with a simple recurrence that can be implemented in a few lines of Python code. Needleman-Wunsch is not the fastest algorithm in practice or in theory, however, we judge it to be a success in practice because many other algorithms, including some of the fastest ones, are either modifications of it or use it as a subroutine.

The algorithm was described in [64], who gave the dynamic programming recurrence and proved that the runtime is $\Theta(n^2)$. Their algorithm modified an earlier dynamic programming algorithm [45] whose run time was $\Theta(n^3)$. [1] It seems likely then that traditional worst-case analysis was a driving force behind the creation of the Needleman-Wunsch algorithm.

The second candidate is the fastest known algorithm under traditional worst-case analysis — the Four-Russians speedup to Needleman-Wunsch [8, 38]. 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 [49, 29]. The improvement over Needleman-Wunsch was a factor of about 5 for $n = 2^{18}$ [49], and, extrapolating from [49], 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 (e.g. Myers' bit-parallel algorithm, which we will see later) [49]. Moreover, the fastest implementations of edit distance today [58, 21, 22] do not implement the Four Russians algorithm, even as a subroutine. 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 turned out to not be practically useful.

We now ask if traditional worst-case analysis has allowed us to predict algorithm performance in practice. The analysis of Needleman-Wunsch actually 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.

Looking into the future, can we expect traditional worst-case analysis to lead to the design of new algorithms that perform well in practice? No, since a substantially faster algorithm, in the traditional worst-case analysis framework , is believed to be unlikely. In support of this, a recent result proved that, under the strong exponential time hypothesis, there cannot be an $O(n^{2-\delta})$ algorithm, for any $\delta > 0$ [9]. Such an algorithm is called *strongly sub-quadratic*. There are other "barrier" results of this type which we will not elaborate on [2, 1, 14]. These results are useful only to the extent that they tell us we are unlikely to design better algorithms using traditional worst-case analysis as a guide. However, they tell us nothing about the existence of provably and substantially better algorithms, as long as they are analyzed using a different technique than traditional worst-case analysis.

[1] As a historical note, the algorithm presented in the paper by Needleman and Wunsch [45] is not the algorithm we call "Needleman-Wunsch" today. The "Needleman-Wunsch" algorithm was actually described later in [64].

3.2 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 banded alignment[2] [61]. Banded alignment is a modification of Needleman-Wunsch whose main idea is to reduce the number of cells that need to be computed by the dynamic programming recurrence. Parametrized analysis shows that it computes the edit distance in time $\Theta(kn)$, on all inputs. There are several other algorithms parametrized by k [43, 26]; they achieve various tradeoffs between k and n and have some other differences outside the scope of this article. Banded alignment is the most notable of these because it is very simple to describe and implement (i.e. it does not use any complex data structures) and forms a main component of one of the empirically fastest implementation today (i.e. edlib [58]).

K-parametrized analysis predicts several ways in which banded alignment is theoretically an improvement over Needleman-Wunsch. The first is that when $k = o(n)$, banded alignment scales sub-quadratically with the input size, while Needleman-Wunsch does not. The second is that the closer the sequences are (i.e. 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 banded alignment over Needleman-Wunsch can be significant [58].

However, when $k = \Theta(n)$, the fact remains that 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 thought of 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 [18]. Thus, regardless of species, and even for species that are very close, the edit distance is still roughly a constant proportion of the sequence length. This illustrates the importance of developing algorithms that perform well 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 is able to 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 $k = \Theta(n)$; and, 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.

3.3 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 [44]. Strings containing short repetitive patters tend to have a lower entropy. As an extreme case, the string of all Ts 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 [19] and runs in time $\mathcal{O}(hn^2/\log n)$ for most inputs. When the input strings have low entropy, this is theoretically faster than Needleman-Wunsch. Note that this runtime

[2]There is some confusion in the literature regarding what banded alignment means. The algorithm we refer to here solves the edit distance computation problem as stated, without knowing k in advance. In some papers (e.g. [65]) though it refers to an algorithm for a variant of the edit distance computation problem that takes k' as a parameter and returns the edit distance if it is at most k' and a "fail" otherwise. We will only discuss the former algorithm in this survey, though the two are closely related.

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 encoded [44] strings in $\mathcal{O}(\ell n)$ time [36, 19, 7], where ℓ is the size of the encoded strings. For a more general class of compression algorithms called straight-line programs [27], there is an algorithm [24] that runs in $\mathcal{O}(\ell n\sqrt{\log(n/\ell)})$ time. In these cases, the algorithms are designed to optimize the runtime with respect to ℓ-parametrized worst-case analysis.

Unfortunately, the above algorithms have not been broadly applied in practice. A major reason is that a long DNA sequence, while exhibiting some repetitive patterns, still has fairly high entropy (e.g. 0.85 [34]) and low compressibility. Thus, while worst-case analysis parametrized by entropy or compressibility has led to the design of novel algorithms, it has not achieved the design goal of TAA for edit distance. For the same reason, it has not achieved the prediction goal either.

3.4 Average-case analysis and semi-random models

One of the pitfalls of traditional and parametric worst-case analysis is that it assumes that 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. One TAA technique to alleviate this shortcoming is to use average-case 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 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 (i.e. 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. In particular, researchers have aimed to design an algorithm with a strongly sub-quadratic expected runtime.

An obvious first attempt is to assume that each string of length n is randomly drawn uniformly and independently from the universe of all strings of length n. How well does this model capture real data? When the real input strings are evolutionary related, then the assumption that 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. Therefore, this model is too simple to be useful.

A more sophisticated approach is to model the input using an indel channel [23]. 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 [23] runs in $\mathcal{O}(n \log n)$ 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, in order for this algorithm to be practical, it would need to be shown that it performs reasonably well even for real input which does not follow the model's distribution.

One can view average-case analysis and worst-case analysis as two extremes. The main idea of more sophisticated models, also called semi-random models, is to achieve a middle ground between an adversary and randomness [6, 31, 10]. Here, the performance is measured as worst-case over the choices of the adversary, and average-case over the random distribution. The precise models differ but the key ideas of the edit distance models are replacing the uniform distribution with a pseudorandom one that better reflects biological sequences [31] or forcing random noise onto the adversary's choice [6, 10].

Unfortunately, these semi-random models 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. In order for an edit distance approximation algorithm to be relevant in practice, the approximation ratio has to be a tiny constant. For example, even a 3-approximation algorithm would not be able to always distinguish two random DNA sequences (i.e. expected edit distance of roughly $0.53n$ (personal simulations, data not shown)) from a mouse and human sequence pair (i.e. edit distance of about $0.19n$ [18]). Thus, the usefulness of the models is predicated on their ability to achieve a tiny approximation ratio.

In [6, 31], the algorithms' approximation ratios are not precisely derived (i.e. 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 [16]. In all these 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.

On the other hand, the approximation ratio of [10] is derived to be $1 + o(1)$, which is low enough to be of practical relevance. The algorithm's expected runtime is $\mathcal{O}(n^{1.898})$, which is strongly sub-quadratic. Thus, their semi-random model led to the design of an algorithm that brakes through a barrier of previous models (i.e. achieving 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 the more complex algorithm.

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 [6, 31] have impractical approximation ratios, while the algorithm of [10] 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 $k = \Theta(n)$.

3.5 Analyzing with advice

The biological problem is sometimes more general than the mathematical abstraction created for it. Does the user have access to other information, not included in the problem definition, that can serve as advice to an algorithm? An advice-based analysis measures the algorithm runtime with respect to the amount of such advice used. In the case of edit distance, [25] 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, similar to the one between A and B. They show that if an algorithm has access to $\mathcal{O}(\log n)$ of such instances, it can find the edit distance between A and B in $\mathcal{O}(n \log n)$ time, with high probability.

This approach has not been implemented but is promising in the sense that 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 $\mathcal{O}(n \log n)$ 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 definition of [25].

3.6 What is done in practice

There are at least three broadly used software libraries/tools that implement edit distance computation [58, 22, 21]. Edlib [58] is optimized specifically for the edit distance computation problem, while SeqAn [22] and Parasail [21] 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 [42]. Myers' bit-parallel technique encodes the dynamic programming matrix into bitvectors and then rewrites the recurrences in terms of word-sized bitvector operations. This is a hardware optimization that,

while not changing the $\Theta(kn)$ runtime, gives a significant constant speedup in practice. Parasail [21] implements the Needleman-Wunsch algorithm using high-performance computing techniques. These include both task-level parallelism (i.e. multi-threading) and instruction-level parallelism (i.e. SIMD vectorized instructions). Parasail's code is also customized during compilation for the instruction set of the host architecture. Another implementation, BGSA [65], implements the Needleman-Wunsh algorithm with the Myers' bit-parallel technique but supports multi-core, task-level, and instruction-level parallelism for batch execution These implementation highlight how exploiting the properties of the CPU (e.g. the bit-parallel or SIMD implementations) can bring constant speedups that in practice outperform asymptotic speedups (e.g. the Four-Russians speedup).

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 the introduction in [5]). These result in orders-of-magnitude constant time speedups over their CPU counterparts in practice. However, until there is more wide-spread availability and integration of such specialized hardware in bionformatics 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 $k = 0.01n$ and 30 seconds for $k = 0.40n$, on a single core server [58]. For sequences of 100,000 nucleotides each, the runtimes are 0.01s and 0.40s, respectively. As the theory predicts, the runtime deteriorates with increasing k. 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. (e.g. comparing long reads against each other [51]). These results also underscore the importance of using TAA to design exact algorithms that scale better than banded alignment for high values of k.

4 A success story: compact data structures

One of the biggest successes of TAA in SB is its role in the design of compact data structures. A compact data structure is one that allows efficient query operations, while using as little space as possible [44]. The theoretical analysis of the memory use of compact data structures differs from traditional worst-case analysis in that the higher order terms are often written without asymptotic notation. For example, such analysis led to the design of a compact representation of a popular SB data structure called the de Bruijn graph with space usage $4m+o(m)$ bits, where m is the number of edges [11]. This data structure uses very little memory in practice [48] and forms the core of the widely used MEGAHIT assembler [33]. This type of analysis has directly led to the design of several other data structures that perform well on real data [3, 17, 4, 46, 57]. One particularly successful example is the pufferfish index [4], which forms part of the Salmon [47] software with thousands of users. There are also compact data structures such as the Burrows-Wheeler transform that are even more ubiquitous, but we do not consider them as an application of TAA to SB as they were not designed directly for SB.

This type of TAA satisfies all the goals. We can use it to predict how much memory data structures will use in practice and compare the relative performance of different methods (e.g. quantifying the tradeoffs between memory and query time of various indices). Moreover, using it as a yardstick has led to the design of practical data structures that are included in broadly used software.

5 Case studies of two TAA failures

In this section, we give two examples of SB problems where TAA has not achieved its goals and the SB field has been held back because of it. For these problems, accuracy is (arguably) a more important driver of algorithm design than running time; thus we will focus on accuracy as the performance metric which is analyzed.

5.1 Accuracy of genome assembly algorithms

The challenge of theoretical analysis of assembly algorithms has been to formulate a proper measure to evaluate accuracy. One approach proposed by [13] is to evaluate an algorithm by the conditions under which it can fully reconstruct the original genome sequence. For example, how many reads does it need or what error rate can it tolerate. This particular evaluation framework did lead to the design of a new assembler [13, 28]. However, the problem is that in practice these conditions rarely arise (i.e. the genome usually has too many repetitive sequences to be reconstructed completely and unambiguously). Therefore it is not clear if designing algorithms to optimize this would generally lead to assemblers that perform well in practice. An alternate framework is to measure what percentage of all the substrings that could possibly be inferred to exist in the genome are output by the assembler [60]. However, this framework has proven technically challenging to apply to real data. A common challenge to all analysis attempts is that most assembly algorithms rely heavily on complex heuristics. This limits the usefulness of worst-case analysis and makes applying other techniques technically challenging.

In summary, TAA cannot be credited with the design of any of the widely used assemblers, nor has it led to any theoretical analysis that can predict the performance of an assembler on real data. This is not to downplay the importance of theory in the design of assemblers. In fact, theory has been successfully applied to assembly (e.g. through combinatorial algorithms for optimization problems), just not to the theoretical analysis of assembly algorithms.

These shortcomings of TAA have been felt in practice. The assemblathon2 competition [12] performed an empirical evaluation of short-read assemblers and found it very challenging to rank the assemblers. In particular, the ranking of different assemblers according to their relative accuracy depended on the dataset, on the evaluation metrics being used, and on the parameter choices made in the evaluation scripts. These are exactly the shortcomings of empirical evaluation that TAA is intended to address. Moreover, the assemblers themselves are simply not as good as they could be, or, as one of the reviewers concluded, "on any reasonably challenging genome, and with a reasonable amount of sequencing, assemblers neither perform all that well nor do they perform consistently" [59]. In the last five years, the practical situation has to some extent improved due to newer long-read sequencing technologies that are able to generate higher quality data. Nevertheless, the experience with short-read assembly is instructive to appreciate the need for better TAA in SB.

5.2 Accuracy of structural variation detection algorithms

Algorithms for the problem of structural variation detection started to appear around 2008 (see [39, 35] for surveys) and a recent assessment identified at least 69 usable tools [30]. As with genome assembly, many of these tools are based on combinatorial algorithms for optimization problems [39]. However, the majority of these tools are heuristics with no theoretical analysis of their accuracy. There are some exceptions, with a probabilistic formulation sometimes used to achieve a desired false discovery rate (e.g. [37]). In such cases, TAA can take some credit for the design of the algorithm.

The theoretical analysis of accuracy requires a statistical model for the distribution of variant type, frequency, and location. Because we have a limited understanding of the biological process that generates structural variants, coming up with realistic models is challenging. Therefore, even when accuracy is predicted theoretically, it does not correspond to what is observed in practice because the models are too idealized [37]. The lack of good models also hampers empirical evaluation. A recent assessment warned developers against considering "simulation results representative of real-world performance" [15], and, in fact, accuracy on simulated data is typically much higher for most tools then on real data [30]. Thus, even in the limited cases where TAA has been applied, it has not achieved its prediction goals. The TAA open challenge is to come up with a generative model for structural variants such that algorithms designed to maximize the expected accuracy on this model outperform existing tools in practice with an observed accuracy matching the predicted one.

As with genome assembly, the algorithms used in practice suffer from many of the limitations that TAA is intended to address. There were two recent studies assessing the accuracy of structural variant

callers (i.e. detection algorithms) on short-read whole-genome sequencing data [30, 15]. They found that the tools suffered from low recall and the ranking of the tools according to accuracy varied greatly across different subtypes of variants [30]. The problems described in [15] are the types inherent to empirical-only evaluation: "But with newly published callers invariably reporting favourable performance, it is difficult to discern whether the results of these studies are representative of robust improvements or due to the choice of validation data, the other callers selected for comparison, or over-optimisation to specific benchmarks."

6 Conclusion

In this survey, we have attempted to present TAA techniques that have been applied in SB and evaluate the extent to which they have achieved their goals. Rather than attempting a laundry list of examples, we focused on the two extremes. On one hand, we looked at edit distance, which has arguable received the most attention of any SB problem from the theory community. We saw some examples of successful TAA techniques and some examples of techniques that have not been successful yet. On the other hand, we looked at genome assembly and structural variation, two problems where there has been a lack of TAA and where the shortcomings plaguing the field are the type that could be addressed by better TAA.

What separates the success stories from failures? Since our survey only looked at a few case studies, we cannot hope to answer this question generally. However, a couple of patterns emerge that warrant more comprehensive future study. First, the TAA success stories (edit distance and compact data structures) have a single, broadly accepted metric of performance in practice (while there is some variability in how running time and memory usage is measured, it is fairly minor). The failures do not; in particular, there turn out to be many ways to measure accuracy of an algorithm for genome assembly or a structural variation detection. Which one is most meaningful varies depending on the downstream application. Second, the success stories have a straightforward way to verify the theoretically predicted performance using real data (for example, simply comparing the scaling of the observed running time compared to what is theoretically predicted). In the failures, such verification is challenging and difficult to do on a broad scale. The ground truth for genome assembly or structural variation detection is generally not known except for a handful of benchmark datasets.

Has TAA been successful in SB? The SB field is broad, with many thousands of algorithms, making a comprehensive judgment of whether TAA has achieved its goals in SB challenging. However, anecdotally, the vast majority of SB papers either do not perform TAA or perform a worst-case analysis only to say that the algorithm performs much better in practice than the analysis indicates [37]. Even if there are isolated successes not reflected in this survey, it is fair to say that the success of TAA in SB has been limited at best. Compact data structures, covered in Sec 4, seem to be the only consistent exception to this.

Is TAA even necessary? SB researchers consistently develop algorithms that have an enormous impact on biological studies. These algorithms are designed using performance on empirical data as the yardstick, with benchmark datasets and/or community competitions proving very useful. Thus empirical analysis can often achieve the same design and prediction goals as TAA. Nevertheless, as we saw in Section 5, empirical analysis is often not enough and some of the problems plaguing SB are exactly the type that would be aided by better TAA. Coming up with a generative model for simulation that would be reflective of reality is challenging and algorithms that perform well on simulated data often do not perform well on real data. In many cases, benchmarks and competitions with real data have not been developed, or have been developed years later than many of the algorithms (e.g. in structural variation). Moreover, algorithms that do well on benchmarks and/or contests do not necessarily generalize well to other datasets; in fact, the incentives sometimes favor algorithms that overfit the data (e.g. in genome assembly). On the other hand, algorithms that are designed to do well on a good theoretical yardstick (i.e. matching empirical observation) have the potential to be more generalizable, and the theoretical yardstick has the potential to predict algorithm

performance on different datasets in a way that an empirical benchmark cannot. In summary, the many success of the SB field do not in any way reduce the potential improvements that TAA can bring.

What makes applying TAA to SB so challenging? We touched upon the challenges of applying TAA to SB in the introduction. In other areas of bioinformatics, such as whole-genome analysis and phylogeny, TAA has arguably played a bigger role. What makes SB distinct is that the technology generating the input data is evolving much faster, putting a lot of pressure on quick solutions. The theoretical analysis of SB algorithms not only favors but in fact requires simplicity of the analysis technique. A simpler technique will be more trusted by biologists, more broadly understood and applied by bioinformaticians, more easily taught to students, and more likely to be included in bioinformatics curricula. The challenge is to have a simple technique which at the same time accurately captures empirical performance and is an effective yardstick for the development of empirically better algorithms.

What can be done? The first step to tackling the challenges of TAA in SB is to recognize the Theoretical Analysis of Sequencing Bioinformatic Algorithms (TASBA) as its own research area, distinct from the design of the algorithms themselves. Currently, it is seen as a side note of SB algorithm development. The value of a TASBA contribution is not always appreciated, even at more theoretical algorithmic bioinformatic conferences. While a breakthrough result will likely be appreciated, a paper describing incremental progress on an algorithm is much more likely to be seen favorably than a paper describing substantial progress on an analysis technique. Moreover, there is generally an expectation that a bioinformatics paper, even a theoretical one, delivers a biologically validated algorithm. In most cases, this is a valid expectation, but it is not always appropriate for TASBA papers. These challenges limit the formulation of TASBA subproblems and are in general detrimental to progress in the TASBA field. Recognizing TASBA as a research area will generally help with the formation of a community of like-minded researchers and all the synergies that go along with it.

In terms of research questions, it would be helpful to develop beyond-worst case analysis (BWCA) success stories. In this survey, we presented several successes (i.e. parametrized analysis and TAA of compact data structures) and several other approaches that may prove to be successful in the future. These form a good starting point, but more success stories are needed. These may be retrospective, i.e. to explain performance that is already understood well empirically. When the theoretical computer science community tackled the limits of the competitive ratio for the online paging problem, the empirically best algorithm was already known; the challenge was to find the right BWCA technique to reach the same conclusion [50]. Similarly, a distinct TASBA research program would not be afraid to tackle the question of performance prediction for short-read assembly, even though the best methods have already been established. BWCA success stories can then form a toolbox of TASBA techniques that could be tried for new problems as they arise. With such templates available, bioinformatics researchers will find it easier to apply BWCA to new problems. Some more general goals of a TASBA research program could be 1) to develop TAA techniques that are simple yet predictive of real-world performance, 2) to establish best practices for the application of TAA techniques to bioinformatics problems, and 3) to establish the most effective way to combine empirical and theoretical analysis of algorithm techniques.

Acknowledgments: This manuscript was inspired by the online videos for the 2014 class Beyond Worst-Case Analysis by Tim Roughgarden. I would like to thank Rob Patro, Jens Stoye, Alexandru Tomescu, and Fabio Vandin for reading the manuscript and providing great feedback, Michael Brudno for illuminating discussions on SVs, the twitter respondents at https://twitter.com/pashadag/status/1279232810407124992, and Jouni Siren for helping me understand the role of BOSS in VG. This material is based upon work supported by the National Science Foundation under Grants No. 1439057, 1453527, and 1931531.

References

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for lcs and other sequence similarity measures. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 59–78. IEEE, 2015.

[2] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 375–388, 2016.

[3] Fatemeh Almodaresi, Prashant Pandey, and Rob Patro. Rainbowfish: A succinct colored de Bruijn graph representation. In Russell Schwartz and Knut Reinert, editors, *WABI 2017: Algorithms in Bioinformatics*, volume 88 of *LIPIcs-Leibniz International Proceedings in Informatics*, pages 18:1–18:15. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[4] Fatemeh Almodaresi, Hirak Sarkar, Avi Srivastava, and Rob Patro. A space and time-efficient index for the compacted colored de Bruijn graph. *Bioinformatics*, 34(13):i169–i177, 2018.

[5] Mohammed Alser, Hasan Hassan, Akash Kumar, Onur Mutlu, and Can Alkan. Shouji: a fast and efficient pre-alignment filter for sequence alignment. *Bioinformatics*, 35(21):4255–4263, 2019.

[6] Alexandr Andoni and Robert Krauthgamer. The smoothed complexity of edit distance. *ACM Transactions on Algorithms (TALG)*, 8(4):1–25, 2012.

[7] Ora Arbell, Gad M Landau, and Joseph SB Mitchell. Edit distance of run-length encoded strings. *Information Processing Letters*, 83(6):307–314, 2002.

[8] V Arlazarov, Y Dinitz, M Kronrod, and I Faradzhev. On economical construction of the transitive closure of an oriented graph. In *Doklady Akademii Nauk*, volume 194, pages 487–488. Russian Academy of Sciences, 1970.

[9] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM Journal on Computing*, 47(3):1087–1097, 2018.

[10] Mahdi Boroujeni, Masoud Seddighin, and Saeed Seddighin. Improved algorithms for edit distance and lcs: Beyond worst case. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1601–1620. SIAM, 2020.

[11] Alexander Bowe, Taku Onodera, Kunihiko Sadakane, and Tetsuo Shibuya. Succinct de Bruijn graphs. In *WABI*, volume 7534 of *Lecture Notes in Computer Science*, pages 225–235. Springer, 2012.

[12] Keith R Bradnam, Joseph N Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, Inanç Birol, Sébastien Boisvert, Jarrod A Chapman, Guillaume Chapuis, Rayan Chikhi, et al. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*, 2(1), 2013.

[13] Guy Bresler, Ma'ayan Bresler, and David Tse. Optimal assembly for high throughput shotgun sequencing. *BMC Bioinformatics*, 14(S5), 2013.

[14] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 79–97. IEEE, 2015.

[15] Daniel L Cameron, Leon Di Stefano, and Anthony T Papenfuss. Comprehensive evaluation and characterisation of short read general-purpose structural variant calling software. *Nat. Commun.*, 10(1):1–11, July 2019.

[16] Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucky, and Michael Saks. Approximating edit distance within constant factor in truly Sub-Quadratic time. *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018.

[17] Rayan Chikhi and Guillaume Rizk. Space-efficient and exact de Bruijn graph representation based on a Bloom filter. In Ben Raphael and Jijun Tang, editors, *WABI 2012: Algorithms in Bioinformatics*, volume 7534 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2012.

[18] Gregory M Cooper, Michael Brudno, NISC Comparative Sequencing Program, Eric D Green, Serafim Batzoglou, and Arend Sidow. Quantitative estimates of sequence divergence for comparative analyses of mammalian genomes. *Genome Res.*, 13(5):813–820, May 2003.

[19] Maxime Crochemore, Gad M Landau, and Michal Ziv-Ukelson. A subquadratic sequence alignment algorithm for unrestricted scoring matrices. *SIAM journal on computing*, 32(6):1654–1673, 2003.

[20] CS264: Beyond Worst-Case Analysis. `http://timroughgarden.org/f14/f14.html`. Accessed: July, 2019.

[21] Jeff Daily. Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. *BMC bioinformatics*, 17(1):81, 2016.

[22] Andreas Döring, David Weese, Tobias Rausch, and Knut Reinert. Seqan an efficient, generic c++ library for sequence analysis. *BMC bioinformatics*, 9(1):11, 2008.

[23] Arun Ganesh and Aaron Sy. Near-Linear Time Edit Distance for Indel Channels. In *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*, volume 172 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[24] Paweł Gawrychowski. Faster algorithm for computing the edit distance between slp-compressed strings. In *International Symposium on String Processing and Information Retrieval*, pages 229–236. Springer, 2012.

[25] Shafi Goldwasser and Dhiraj Holden. The complexity of problems in p given correlated instances. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[26] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. 1997.

[27] Danny Hermelin, Gad M Landau, Shir Landau, and Oren Weimann. A unified algorithm for accelerating edit-distance computation via text-compression. *arXiv preprint arXiv:0902.2649*, 2009.

[28] Sreeram Kannan, Joseph Hui, Kayvon Mazooji, Lior Pachter, and David Tse. Shannon: An information-optimal de novo RNA-Seq assembler. *bioRxiv*, page 039230, 2016.

[29] Youngho Kim, Joong Chae Na, Heejin Park, and Jeong Seop Sim. A space-efficient alphabet-independent four-russians' lookup table and a multithreaded four-russians' edit distance algorithm. *Theoretical Computer Science*, 656:173–179, 2016.

[30] Shunichi Kosugi, Yukihide Momozawa, Xiaoxi Liu, Chikashi Terao, Michiaki Kubo, and Yoichiro Kamatani. Comprehensive evaluation of structural variation detection algorithms for whole genome sequencing. *Genome Biol.*, 20(1):1–18, June 2019.

[31] William Kuszmaul. Efficiently approximating edit distance between pseudorandom strings. In *Proceedings of the thirtieth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1165–1180. SIAM, 2019.

[32] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

[33] Dinghua Li, Ruibang Luo, Chi-Man Liu, Chi-Ming Leung, Hing-Fung Ting, Kunihiko Sadakane, Hiroshi Yamashita, and Tak-Wah Lam. Megahit v1. 0: a fast and scalable metagenome assembler driven by advanced methodologies and community practices. *Methods*, 102:3–11, 2016.

[34] David Loewenstern and Peter N Yianilos. Significantly lower entropy estimates for natural DNA sequences. *Journal of computational Biology*, 6(1):125–142, 1999.

[35] Medhat Mahmoud, Nastassia Gobet, Diana Ivette Cruz-Dávalos, Ninon Mounier, Christophe Dessimoz, and Fritz J Sedlazeck. Structural variant calling: the long and the short of it. *Genome Biol.*, 20(1):1–14, November 2019.

[36] Veli Mäkinen. Approximate matching of run-length compressed strings. *Algorithmica*, 35(4):347–369, 2003.

[37] Tobias Marschall, Ivan G Costa, Stefan Canzar, Markus Bauer, Gunnar W Klau, Alexander Schliep, and Alexander Schönhuth. CLEVER: clique-enumerating variant finder. *Bioinformatics*, 28(22):2875–2882, 2012.

[38] William J Masek and Michael S Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 20(1):18–31, 1980.

[39] Paul Medvedev, Monica Stanciu, and Michael Brudno. Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods*, 6(11 Suppl):S13–20, November 2009.

[40] Michael Mitzenmacher. Theory without experiments: have we gone too far? *Communications of the ACM*, 58(9):40–42, 2015.

[41] Bernard ME Moret. Towards a discipline of experimental algorithmics. In *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, volume 59, pages 197–213, 2002.

[42] Gene Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM (JACM)*, 46(3):395–415, 1999.

[43] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[44] Gonzalo Navarro. *Compact data structures: A practical approach*. Cambridge University Press, 2016.

[45] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[46] Prashant Pandey, Michael A Bender, Rob Johnson, and Rob Patro. A general-purpose counting filter: making every bit count. In *SIGMOD '17: Proceedings of the 2017 ACM International Conference on Management of Data*, pages 775–787. Association for computing machinery, 2017.

[47] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature methods*, 14(4):417–419, 2017.

[48] Amatur Rahman and Paul Medvedev. Representation of k-mer sets using spectrum-preserving string sets. In Russell Schwartz, editor, *Proceedings of Research in Computational Molecular Biology - 24th Annual International Conference, RECOMB 2020*, volume 12074 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2020.

[49] Martin Rejmon. Multi-threaded implementation of Four Russians edit distance algorithm. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology.

[50] Tim Roughgarden. Beyond worst-case analysis. *Communications of the ACM*, 62(3):88–96, 2019.

[51] Kristoffer Sahlin and Paul Medvedev. De novo clustering of long-read transcriptome data using a greedy, quality value-based algorithm. *Journal of Computational Biology*, 27(4):472–484, 2020.

[52] Kristoffer Sahlin, Marta Tomaszkiewicz, Kateryna D Makova, and Paul Medvedev. Deciphering highly similar multigene family transcripts from iso-seq data with isocon. *Nature communications*, 9(1):1–12, 2018.

[53] Peter Sanders. Algorithm engineering: An attempt at a definition using sorting as an example. In *Proceedings of the Meeting on Algorithm Engineering & Experiments*, ALENEX 2010, pages 55–61. Society for Industrial and Applied Mathematics, 2010.

[54] Hirak Sarkar, Mohsen Zakeri, Laraib Malik, and Rob Patro. Towards selective-alignment: Bridging the accuracy gap between alignment-based and alignment-free transcript quantification. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 27–36, 2018.

[55] Robert Sedgewick and Philippe Flajolet. *An introduction to the analysis of algorithms*. Pearson Education India, 2013.

[56] Jared T Simpson and Mihai Pop. The theory and practice of genome sequence assembly. *Annual review of genomics and human genetics*, 16:153–172, 2015.

[57] Jouni Sirén. Indexing variation graphs. In *2017 Proceedings of the ninteenth workshop on algorithm engineering and experiments (ALENEX)*, pages 13–27. SIAM, 2017.

[58] Martin Šošić and Mile Šikić. Edlib: a C/C++ library for fast, exact sequence alignment using edit distance. *Bioinformatics*, 33(9):1394–1395, 2017.

[59] C Titus Brown. Thoughts on the assemblathon 2 paper. `http://ivory.idyll.org/blog/thoughts-on-assemblathon-2.html`. Accessed: 2020-1-9.

[60] Alexandru I Tomescu and Paul Medvedev. Safe and complete contig assembly through omnitigs. *Journal of Computational Biology*, 24(6):590–602, 2017.

[61] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1-3):100–118, 1985.

[62] Jeffrey D Ullman. Experiments as research validation: Have we gone too far? *Communications of the ACM*, 58(9):37–39, 2015.

[63] Robert Vaser, Ivan Sović, Niranjan Nagarajan, and Mile Šikić. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome research*, 27(5):737–746, 2017.

[64] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

[65] Jikai Zhang, Haidong Lan, Yuandong Chan, Yuan Shang, Bertil Schmidt, and Weiguo Liu. Bgsa: A bit-parallel global sequence alignment toolkit for multi-core and many-core architectures. *Bioinformatics*, 35(13):2306–2308, 2019.

[66] Justin M Zook, Nancy F Hansen, Nathan D Olson, Lesley M Chapman, James C Mullikin, Chunlin Xiao, Stephen Sherry, Sergey Koren, Adam M Phillippy, Paul C Boutros, et al. A robust benchmark for germline structural variant detection. *BioRxiv*, page 664623, 2019.