KEZW-BI-ME/00006

August 13, 2020

Alignment Neighborhood EXplorer (ANEX): First attempt to apply *genuine* sequence evolution model with *realistic* insertions/deletions to Multiple Sequence Alignment reconstruction problem.

Kiyoshi Ezawa

Independent (ORCID: 0000-0003-4906-8578)

Current address: 3-1-33 Nakamura-machi, Chichibu 368-0051, JAPAN;

Phone & Fax: +81-494-22-5501; E-mail: kezawa.ezawa3@gmail.com

Abstract

Accurately reconstructing pairwise sequence alignments (PWAs) and/or multiple sequence alignments (MSAs) is a central step in the study of homologous (*i.e.*, ancestor-sharing) biological sequences, such as DNA and protein sequences. It is known, however, that even the state-of-the art methods to reconstruct MSAs mis-align the sequences very frequently, and that a (near) majority of such mis-alignments are caused by the inherent stochastic nature of the sequence evolution. This makes it important to construct a *probability distribution* of alternative MSAs, rather than to reconstruct *only a single* optimum MSA. Although such methods, also known as statistical MSA methods, do exist, all of such methods up to now were *inaccurate* because they were based on *non-evolutionary* probabilistic models of alignments, which cause over- and/or under-estimates of some MSA probabilities.

To address these problems, we have developed a method, called ANEX (Alignment Neighberhood EXplorer), which constructs probability distributions of alternative MSAs in some neighborhoods of an input MSA, under a specified *genuine* sequence evolution model with *realistic* insertions/deletions (indels).

After receiving an MSA, a fixed phylogenetic tree, and a fixed evolution model parameters, as inputs, ANEX first detects and excludes some segments highly likely to include errors that are too complex to be corrected. Then, out of the remaining segments, it prepares possibly overlapping windows each of which contains up to a specified maximum number of gap-blocks (rectangular blocks consisting only of gaps in the MSA). Then, in each window, it explores the neighborhoods of the input sub-MSA, by horizontally moving, merging and/or splitting the gap-blocks or their complementary residue-blocks, while computing the probabilities of the resulting alternative MSAs under the *genuine* sequence evolution model with substitutions and *realistic* indels.

Our manual validation using some simulated MSAs as inputs indicated that, within the segments it examined, the "neighborhoods" explored by ANEX *did indeed* encompass the true MSAs in an overwhelming majority of mis-aligned regions, indicating that this strategy should work at least to some extent.

Although the method is still at a somewhat rudimentary stage, this study represents a significant first step toward the construction of an *accurate* probability distribution of alternative MSAs under *genuine* sequence evolution models with *realistic* indels, which in turn will be indispensable for *precise* and *truthful* evolutionary analyses on homologous biological sequences.

ANEX is available as an open-source package at an FTP repository of Bioinformatics.org (https://www.bioinformatics.org/ftp/pub/anex/); currently , it is available only as an all-Perl version ("ANEX_P"), which runs on UNIX(-like) platforms.

[**Keywords:** sequence alignment, multiple sequence alignment (MSA), probability, evolution, (stochastic) sequence evolution model, insertion/deletion (indel), probability distribution, accurate computation, DNA sequence, biological sequence]

[**Abbreviations:** alignment neighborhood explorer (ANEX), hidden Markov model (HMM), multiple sequence alignment (MSA), preserved ancestral site (PAS), pairwise sequence alignment (PWA), Thorne-Kishino-Felsenstein (TKF)]

Contents

1 Introduction

1.1 Background

[**NOTE:** This study has been conducted based on the results of our previous studies [1, 2, 3, 4]. Thus, for details on its background, or the background of its background, refer to (the background part of) these studies.]

The reconstruction of multiple sequence alignments (MSAs) is *central to* the advanced studies of homologous (*i.e.*, ancestor-sharing) biological sequences, such as DNA, RNA and protein sequences (*e.g.*, [5, 6, 7, 8, 9, 10]). [1] This is because MSAs supplies *essential* inputs to a wide variety of homology-based sequence analyses, such as the inference of phylogenetic relationships among biological sequences [20, 21], the prediction of their 3D structures [22], the prediction and comparison of their functions [23, 24, 25], and the identification of their sites or regions under selection [26]. Because of such central roles played by MSAs, the development of an aligner that accurately and/or quickly reconstructs MSAs has been the subject of vigorous and diligent efforts during the recent decades (e.g., [5, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42]).

As the studies progressed, however, it gradually turned out that this crucial step of MSA reconstruction is *highly* error-prone [43, 37, 44, 45, 3]. For example, Wong et al. [44] pointed out that different aligners produce different MSAs, which often result in conflicting conclusions on the sequence phylogeny or positively selected sites, and they hypothesized that uncertainty in the alignment can lead to such problems. Landan and Graur [45] estimated that, depending on the sequence divergences, about 5-90% of the "homologous" residue

[1] In this study, we deal with *collinear* sequence alignments. An alignment is called "collinear" (*e.g.*, [11]) if it is devoid of genomic rearrangements such as inversions, duplications, and translocations (*e.g.*, [12, 13, 14, 15]). (Possibly non-collinear) alignments of (usually very long) sequences that possibly underwent such genomic rearrangements are called "genome alignments", and they are *not* the subject of this study. Readers who are interested in genome alignments should refer, *e.g.*, to: [11, 16, 17, 18, 19].

pairs in each reconstructed MSA are erroneous. And a previous study of ours [3] showed that, depending on the evolutionary distances among homologous sequences, about 40-99% of gapped segments in MSAs are involved in alignment errors. MSA errors influence the results, and even the conclusions, of the downstream analyses (*e.g.*, [46, 47, 48, 6, 44, 49]), in particular the inference of insertions/deletions [37, 50].

To better understand the nature of such alignment errors in MSAs, we previously conducted a meticulous study [3], in which errors in reconstructed MSAs of simulated DNA sequences were characterized in detail. An important conclusion of that study was that a substantial fraction (about 1/4 - 3/4) of alignment errors are due to **the stochastic nature of evolution processes**, that is, because the true MSA is *not* the optimum *even* with the "complete-likelihood" score [3], which is the logarithm of the occurrence probability of the MSA computed with the very evolution model that was used for the simulation. This result reconfirmed and extended the results of previous studies [51, 52], which revealed how important it is to take account of the stochastic nature of evolution processes when analyzing pairwise alignments (PWAs)(, or, more precisely, analyzing pairs of homologous sequences).

The conclusion in the previous paragraph suggests that finding a *single* optimum MSA will remain considerably error-prone *even if* we get to *exhaustively* search the space of alternative MSAs using the *"golden" score* that perfectly predicts the (log-)probabilities of the MSAs(, which is currently far from feasible, anyway). A way to overcome this problem of stochasticity inherent in sequence alignments will be to construct a **probability distribution of alternative MSAs**, instead of reconstructing a *single* optimum MSA as commonly practiced thus far.

The idea of providing the probability distribution of alignments, often referred to as "*statistical alignment*" [53], is not new. After a couple of ground-breaking works on this subject [54, 55, 56], the results of [55, 56], especially, were later cast into standard Hidden Markov Models (HMMs) (*e.g.,* [53, 57]). Since then, it has become common to address the problem of statistical alignments using (standard) HMMs (or its derivatives such as

transducers) (*e.g.*, [58, 59, 60, 61, 62, 63, 50, 64, 65, 66]), probably because rich mathematical and computational tools are available to handle (standard) HMMs [67].

However, the evolution models handled up to 2003 were dissatisfactory in the sense that their insertion/deletion processes are *not* realistic, in at least two ways: (1) they permit only single-residue indels or multi-residue indels with the geometrically distributed rates, whereas many empirical studies indicate that multi-residue indels occur *quite often*, and with the rates following *power-law* (*e.g.*, [68, 69, 70, 71, 72, 73, 74])[2]; and (2) except [57] (see below), they can*not* take account of nested and/or overlapping indels, which are expected from *natural* evolution processes of biological sequences.

To address these problems, some attempts were made. Knudsen and Miyamoto [57] attempted to incorporate some effects of overlapping indels into standard HMMs. [3] Meanwhile, as an attempt to handle more realistic sequence evolution models, Miklós and Toroczkai [75] proposed a method to compute alignment probabilities under an evolution model that allows any lengths of insertions (with geometrically distributed rates) but *only* single-residue deletions. Then, finally, in a milestone work (in 2004) by Mikós, Lunter, and Holmes [76], the first method based on a *genuine* sequence evolution model [4] was proposed. Their "long indel" model [76] is a space-homogeneous [5] *genuine* sequence evolution model of a biological

[2]Some studies based on standard HMMs alleviated this flaw (*not fully* but to some extent) by using mixed geometric distributions (*e.g.*, [35, 51]).

[3]Recently, there have been a couple of renewed efforts to incorporate the effects of overlapping indels into pair-HMMs [65, 66]. We consider these efforts laudable. At the same time, however, we find it a pity that these studies confined themselves in geometric indel length distributions. We wish that the studies could have extended to more biologically realistic power-law indel length distributions.

[4]A sequence evolution model is regarded as *genuine* only when it follows the "**evolutionary principle**," which is, in short, "long-term evolution of a sequence results from the accumulation of the effects of its short-term evolution." For more detailed "definition" of the "evolutionary principle," see, *e.g.*, the background of [4].

[5]A sequence evolution model is called "space-homogenous" if its rates of evolutionary events are uniform, *i.e.*, independent of the positions, along the sequence.

sequence that can in principle incorporate any indel length distributions, including biologically realistic power-laws. They [76] cast the computation of the probabilities of ancestor-descendant PWAs into a generalized HMM. After this study, some advances had been made on the study using the simulators of *genuine* sequence evolution (*e.g.*, [77, 78, 79]).

Then, over a decade after [76], we generalized their "long indel" model to *genuine* sequence evolution models that allow indel rates to vary across regions, and derived the "sufficient and nearly necessary set of conditions" under which the probability of each ancestor-descendant PWA can be factorized into the product of an overall factor and contributions from gapped segments [1]; and we also went on to extend the results to the factorability conditions of MSA probabilities (*ibid.*). Then, in the subsequent study [2], we developed some (mathematical and computational) tools to facilitate the computation of the probabilities of ancestor-descendant PWAs and MSAs under a given *genuine* sequence evolution model. Especially, we proposed a pair of numerical algorithms to practically exactly compute the finite-time probabilities of isolated gaps in ancestor-descendant PWAs; although this study left it unresolved to (nearly) exactly compute the finite-time probabilities of pairs of mutually adjoining insertion-type and descendant-type gaps, the problem was "solved," in a sense, in a recent study of ours [4]. [6] Regarding the probabilities of MSAs, we devised a "local multi-path downhill algorithm" to enumerate *all* parsimonious sets of ancestral "presence"/"absence" states (at all internal nodes) each of which is consistent with the MSA [2]. And we also devised a method to approximate the MSA probability by summing the contributions from all the indel histories each of which is consistent with the MSA and any of the parsimonious sets of ancestral states (*ibid.*). [7]

Therefore, since [76], the "ingredients" have been *almost* ready to construct the **prob-**

[6] Although in a somewhat different format, this problem was also solved, in a sense, by a recent "simulation-based" approach to statistical PWA [80].

[7] The contribution from each such indel history is computed according to [1], which extended the method developed for HMMs (or transducers) [58, 59] to *genuine* sequence evolution models.

ability distribution of alternative MSAs under a given *genuine* sequence evolution model. There is, however, one problem: unlike with standard HMMs, computing alignment probabilities, especially their indel components, with *genuine* sequence evolution models is *considerably* slow, and even the *exhaustive* computation of PWA probabilities (*via* a dynamic programming [76]) could be prohibitively time-consuming. [8] This means that a simple application or a naïve extension of the previous methods for probabilistic MSAs using HMMs (*e.g.*, [58, 59, 60, 61, 62, 63, 50, 64]) to those using *genuine* sequence evolution models may end up in an *extremely* slow method that could take practically forever to analyze *even* a reasonably-sized MSA.

Fortunately, good news has come from a previous study of ours [3]. The study showed that, as long as the aligned sequences are closely or moderately related, the true MSAs are in the neighborhoods of (*e.g.*, within four block-wise steps from) the reconstructed MSAs in an overwhelming majority [9] of erroneous segments. This suggests a *simple strategy* to construct an ***approximate* MSA probability distribution** by exploring only a **neighborhood** of the input (reconstructed) MSA. Then, thanks to the good properties of the elementary moves of MSAs (explained later in this paper), the MSA probabilities could be computed fairly efficiently, which may enable us to construct good *approximate* MSA probability distributions within a reasonable amount of time, *even* with a *genuine* sequence evolution model,.

We have actually implemented this simple strategy, and named the resulting new program package **ANEX**, which stands for the "**a**lignment **n**eighborhood **ex**plorer". Given an MSA of DNA sequences, a phylogenetic tree and a set of parameters of a *genuine* sequence evolution model, ANEX creates possibly overlapping windows to cover "analyzable portions" of the input MSA, and explores the neighborhoods of the input MSA within each window

[8] This time-consuming nature of Miklós et al.'s algorithm [76] has been mitigated greatly, or at least partially solved, by the recent "simulation-based" approach [80].

[9] The "overwhelming majority" here means about 99.5% for closely related sequences and about 80% for moderately related sequences [3].

while computing the probabilities of MSAs it visits. Our brief validation using a number of reconstructed MSAs of simulated DNA sequences indicated that this simple strategy should work.

1.2 Structure of this paper

This article is structured as follows. Section 2 explains the operations that make up ANEX, as well as the ideas and principles underlying them. The section consists of a number of subsections (and sub-subsections). The subsections describe: (1) the overall workflow (subsection 2.1), (2) scanning the input MSA (subsection 2.2), (3) identifying "likely complex errors" (subsection 2.3), (4) creating windows for analyses (subsection 2.4), (5) exploring neighborhoods (subsection 2.5), (6) computing MSA probabilities (subsection 2.6), (7) outputting the results (subsection 2.7), (8) possible applications (subsection 2.8), and (8) conceivable problems (subsection 2.9, which is just a gateway to appendix L). Then, in section 3, we describe the implementation and "validation" of ANEX. Finally, section 4 is dedicated for discussions.

This paper has lots of appendixes, to which important yet detailed subjects are relegated to, just in order to clarify the main text as much as possible. In particular, appendix A explains, or defines, a number of terms that the readers may not be so familiar with. And appendix D describes "elementary move"s of gap-blocks (or residue-blocks), which are used as important steps in the exploration of MSA neighborhoods.

Before we go on to the main contents, let us briefly note on the evolution model this study deals with. The framework of ANEX, hence this study, assumes that the *genuine* sequence evolution model at hand satisfies the conditions we previously put forth [3]. Specifically: (1) the indel rates (excluding the multiplication factors assigned to the inserted residues) are independent of the residue state and the substitution process before the indel event; (2) the substitution rates at each site are independent of the states and the evolutionary processes

at other sites; and (3) the probability of the residue state of each inserted sub-sequence (conditioned on the insertion) can be factorized into the product of residue probabilities (at the time) over the inserted sites. (Actually, these conditions are satisfied by *most of* the sequence evolution models that have been used in the studies thus far.) As we showed in [3] (by generalizing the proof by Kim and Sinha [81]), under these conditions, the probability of each MSA can be factorized into the ***substitution component***, which depends only on the residue configuration of the MSA, and the ***indel component***, which depends only on the gap-configuration of the MSA. In addition, the indel portion of the evolution model is also assumed to satisfy the three conditions we posed in [1], which are: (i) the rate of each indel event is independent of the sequence state out of the region it affects; (ii) the increment of the exit rate caused by each indel event is independent of the sequence state out of the region it affects; and (iii) the probability of the sequence state at the root is factorized into the product of the probability of a "reference" state and the "multiplicative increments" (from the "reference" probability) contributed independently by the gapped segments. [10] Under these conditions, the indel component of the MSA probability can be factorized into the product of an overall factor and contributions from gapped segments (including "trivial" ones in between adjoining gapless columns).

2 Operations in ANEX and their Underlying Ideas and Principles

[Outline]

This computer program, the "alignment neighborhood explorer" (ANEX), takes an input multiple sequence alignment (MSA), which is typically reconstructed by an aligner, and

[10] See section R8 of [1] for concrete examples of indel evolution models satisfying these conditions. For example, a model whose indel rates are uniform along the sequence satisfies these conditions if one or both end(s) of the sequence allow(s) indels just as the bulk of the sequence does.

explores the MSA's "neighborhoods" in the MSA space. The program explores the neighborhoods via combinations of "elementary moves," such as "shifts" and (horizontal and vertical) "splits" of individual gap-blocks, (horizontal and vertical) "merges", "purges" and "reverse-purge"s (also known as "ex-nihilo"s [45]) of pairs of gap-blocks, and the reversal of "(incomplete-)collapse of independent insertions ((i)CII)". (These "elementary moves" are described in some details in, *e.g.*, [45, 3], and are re-described in appendix D below.) [11]

Then, the program assigns probabilities to the MSAs resulting from such moves. The probabilities are calculated under a *genuine* stochastic evolution model of an entire sequence *via* substitutions and insertions/deletions (indels) with *realistic* indel length distributions, taking advantage of the *quite accurate* multiplication factors pre-computed by our recently developed program package, LASTPIECE [4]. [12]

As a "pre-processing" step, the program also identifies MSA portions with "potentially complex errors," where restoring the true MSA is very likely formidable. In the current version of ANEX, this is done using some properties of gaps in the MSA regions, like the effective indel density and the possible involvement of long effective indels.[13]

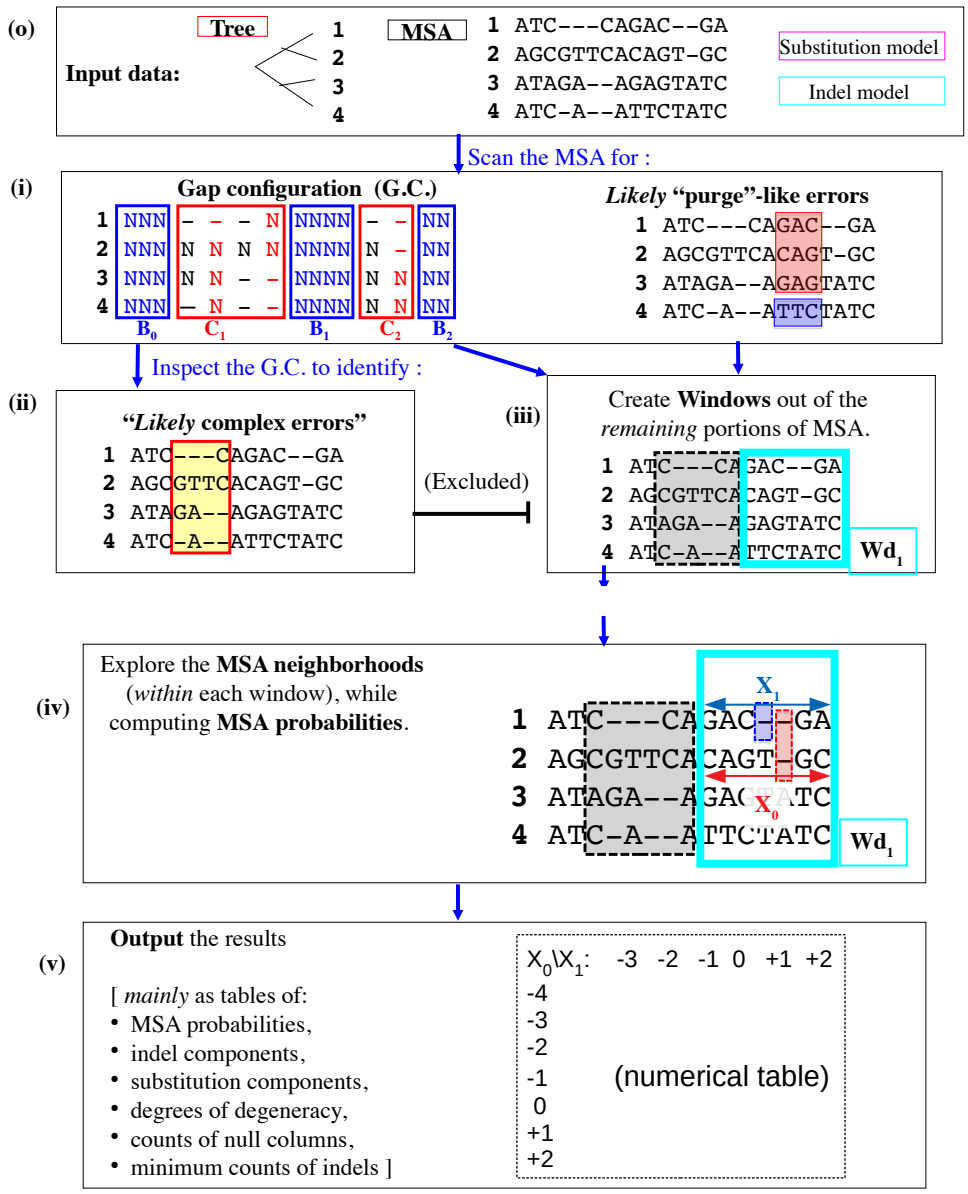


Figure 1: **Flowchart explaining overall workflow of ANEX.** In the gap configuration on the left of **(i)**, the 'N' and '-' represent the presence of a residue and a gap (i.e., the absence of a residue), respectively. And the blue and red rectangles enclose the gapless and gapped segments, respectively. On the right of **(i)**, the alignment of the regions shaded in red and blue indicates a "purge"-like error. In **(ii)**, the yellow-shaded region likely contains a "complex" error. In **(iii)** and **(iv)**, the window ($\mathbf{Wd}_1$) is enclosed by a cyan rectangular box, and the grey-shade indicates that the region has been excluded from further analyses. In **(iv)**, the window ($\mathbf{Wd}_1$) contains two gap-blocks, represented as rectangles shared in semi-transparent red and blue. The coordinates, X_0 (red) and X_1 (blue), spanned by the red and blue both-headed arrows, represent the possible moves ("shifts") of the gap-blocks of the same colors (from the "origin", *i.e.*, the input MSA).

2.1 Overall workflow

The flowchart of the program is shown in Figure 1. After receiving an input MSA and a tree [**(o)**], it first scans the MSA for the **gap configuration** of the entire MSA and also for the portions likely containing **"purge"-like errors** [**(i)**]. During the course of this scan, it also identifies the portions with **"likely complex errors"** [**(ii)**]. Using the results of the scan, it **excludes** such "likely-complex-error" portions from further analyses, and creates (possibly overlapping) **windows** out of the remaining portions of the input MSA [**(iii)**]. Then, in each such window, it explores the *neighborhoods* of the MSA , while calculating the (log-) probabilities of the resulting alignments [**(iv)**]. Finally, the results are **output** [**(v)**].

In the following sub-sections, we will elaborate on how these steps are performed.

2.2 Scanning input MSA

Given an input MSA, the program first scans it *horizontally*, from left to right, and (*conceptually*) chop the MSA into an alternating series of **gapped and gapless segments** (on the left of step **(i)** in Figure 1). This operation is based purely on the gap configuration of the MSA. More precisely, a **gapped segment** is defined as a run of contiguous gapped

[11] Be careful of some changes in the wording. For example, the "merge (opposite types)," "vertical merge (deletions)," and "vertical merge (insertions)" in [3] are referred to as "merge (complementary)" (abbreviated as "c-merge"), plain "vertical merge" (abbreviated as "v-merge") and "complementary vertical merge" (abbreviated as"cv-merge") respectively, in this study.

[12] This strategy of pre-computing and re-using the multiplication factors has already been suggested in Additional File 1 of [2], and has actually been implemented since version 0.6 of LOLIPOG(_P) [2] and version 0.6 of ComplLiMment(_P) [3], both of which were released in 2015. We learned that a recent "simulation-based" approach to statistical PWA [80] also employs this strategy of pre-computing and re-using the probabilities of gapped segments (more precisely, "chop-zones" in their study). We do not know whether they have just borrowed our idea or *independently* come up with this strategy by themselves.

[13] See appendix A for the definition of the "effective indel."

columns delimited by gapless columns (or a gapless column and an MSA end), and a **gapless segment** is defined as a run of contiguous gapless columns [1, 2].

The program also searches for portions where **"purge"-like errors** are likely (on the right of step **(i)** in Figure 1). (For "purge"-like errors, see, *e.g.*, [45, 3].) Briefly, the search is done via a **sliding-window analysis**. In each window, substitutional residue-differences (SRDs) along the branches are estimated from the residue configurations of the columns. If the estimated SRDs are significantly more than expected along a branch in a window, the window is recorded as holding a **"purge"-like error candidate**. If some of such candidates (regarding the same branch) overlap or adjoin, they are merged. (Details on this search process are described elsewhere [82].)

[The current version of ANEX actually searches for "purge"-like error candidates by calling a *satellite script*, "detect_purge_cands.ver0.5.pl."]

2.3 Identifying MSA portions with "likely complex errors"

Quite frequently, MSAs reconstructed by an aligner (even if it is the *state of the art*) contain **"complex" errors**, each of which cannot easily be classified into a definite type or expressed as a combination of definite types [3]. It will also be very difficult to rectify such "complex" errors and to recover the true MSA from the reconstructed one in such an MSA portion. Therefore, it should be wise to exclude such portions from further analyses, and it will also be *honest* (or *truthful*) to tell the users which portions of the input MSA likely contain complex errors. ANEX identifies such portions by examining the **gap configurations** and the **effective indel events** in each gapped segment (or each cluster of gapped segments). (For details on this process, refer to [82].)

[For the actual implementation of the identification of "likely complex errors," see the subroutine, "wrapper_detect_cmplx_error_cand_acls0," in the module, "MyDetect_cmplx_error_cands.pm," of ANEX.]

2.4 Creating windows for MSA neighborhood exploration

After identifying regions likely to contain "complex" errors and excluding them (as well as a specified number of MSA columns flanking them) from further analyses, ANEX creates **windows** in each of which it actually explores the neighborhoods of the input MSA.

Currently, ANEX's creation of windows depends on the positional distribution of **gap-blocks** in the input MSA . [14] Basically, the gap-blocks are determined *via* the Dollo-parsimony-based indel history that can explain the gap-configuration of the input MSA [83, 2]. (See appendix B for more details.)

Currently, ANEX creates two kinds of windows: (a) **ordinary windows** each of which contains up to (and including) N_W gap-blocks , where the N_W is a user-specified number; each ordinary window may include one or two sub-window(s), each of which contains up to (and including) $(N_W - 1)$ gap-blocks; and (b) **"purge-like-error-candidate-containing" windows**, or **PCC windows** for short, each of which is constructed around a single "purge-like-error candidate" (described in subsection 2.2), and each of which can contain up to (and including) $(N_W - 2)$ gap-blocks. As you can see, different maximum numbers of gap-blocks are set for different types of windows; this is in order to keep the number of *simultaneously* "shift"ed gap-blocks less than or equal to N_W (see subsection 2.5 below, especially around Figure 4, for more details).

For more details on how to create the windows, see appendix C.

2.5 Exploring neighborhoods of input MSA

After defining the (possibly overlapping) windows, ANEX explores the neighborhoods of the input MSA *via* a **window analysis** (step **(iv)** in Figure 1). In each window, the program examines *alternative* MSAs obtained from the input MSA *via* an **"elementary move"** [45, 3], or a combination of a number of elementary moves. Basically, each **elementary**

[14] Figure 2 illustrates the gap-blocks. For the definition of the term, "gap-block," see appendix A.

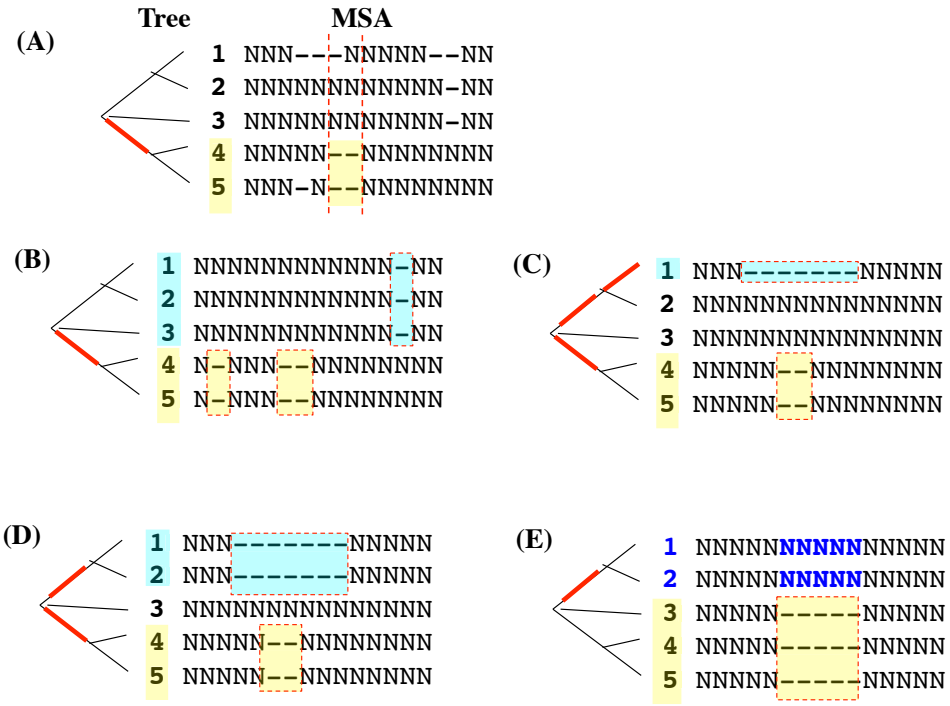


Figure 2: **Gap-block, "isolated" gap-block, and sequence-block.** (See also appendix A for the definitions of the headlined terms.) To focus on the gap-pattern, each residue was represented as 'N' regardless of its identity. **A.** The yellow-shaded portion of the MSA is a gap-block. It is delimited by two inter-column positions (the vertical red dashed lines) and a branch in the tree (the red branch). (The identifiers of the involved (or "supporting") sequences are also shaded in yellow.) **B, C.** "Isolated" gap-blocks (shaded portions enclosed by dashed boxes). In panel **B**, the isolation is obvious because the horizontal positions of the gap-blocks do not overlap. In panel **C**, although the two gap-blocks overlap *horizontally*, they are separated *vertically* with each other by three branches (in red). **D.** A pair of "non-isolated" gap-blocks, which *horizontally* overlap and are *vertically* separated by only two branches (in red). **E.** A sequence-block (in blue), which is *complementary to* a gap-block (shaded in yellow).

move is defined as the move of a "**gap-block**," which is a rectangular block of gaps (Figure 2, panel A), or as the move of a "**residue-block**", which is complementary to the gap-block (Figure 2, panel E) [15]. (In the current version, ANEX performs **multiple-"shift"s**, *i.e.*, simultaneous independent "shift"s of individual gap-blocks, *by default*; if applicable, it also attempts some *non-"shift"* elementary moves, either as single moves or as a pair of moves, followed by the multiple-"shift"s.)

Currently, ANEX attempts the following elementary moves (illustrated in Figure 3): (i) "shift", (ii) "purge", (iii-a) "same-type-merge" (or "s-merge" for short), (iii-b) "complementary-merge" (or "c-merge" for short), (iv-a) "same-type-split" (or "s-split" for short), (iv-b) "complementary-split" (or "c-split" for short), (v) "reverse-purge" (also known as "ex-nihilo" [45]), (vi-a) "vertical-merge" (or "v-merge" for short), (vi-b) "complementary-vertical-merge" (or "cv-merge" for short), (vii-a) "vertical-split" (or "v-split" for short), (vii-b) "complementary-vertical-split" (or "cv-split" for short), and (viii) "reversal of (incomplete-)Collapse of Independent Insertions" (or "reverse-(i)CII" for short). For more details on these elementary moves, see appendix D.

Current version of ANEX employs a simple architecture of combining the elementary moves to explore the neighborhoods of the input MSA. Figure 4 illustrates the architecture.

First [(I)], ANEX constructs ordinary windows based on the gap-configurations, *i.e.*, the set of gapped segments (as in step (iii) in Figure 1). Each ordinary window contains N_{GB} gap-blocks or less, where N_{GB} is a user-specified number. Each ordinary window may (or may not) include one or more sub-window(s) [(II)], each of which, if at all, contains $N_{GB} - 1$ gap-blocks or less. [16]

Then [(II)], in each ordinary window (the "Wd_m," with $m = 1, 2, ...$), ANEX attempts the combinations of elementary moves as follows. (1) By default, it attempts the "multiple-shift"s of *all* gap-blocks, starting from the input MSA. (2) If there are appropriate gap-blocks,

[15] See appendix A for the definitions of the "gap-block" and the "residue-block."

[16] An ordinary window itself is its sub-window, if it contains less than N_{GB} gap-blocks.

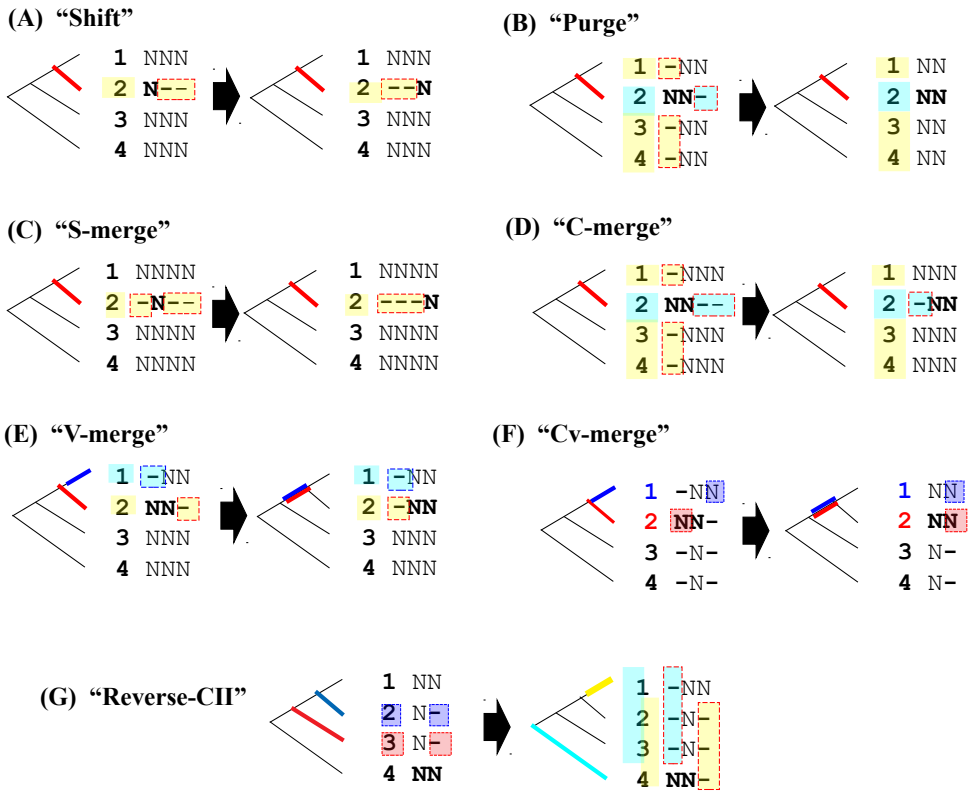


Figure 3: **Elementary moves that ANEX attempts.** The bold 'N's represent the residues that moved. **A.** A "shift" of a single gap-block (shaded portion enclosed by the dashed box). **B.** A "purge" of two gap-blocks supported by the *complementary sets* of sequences. (Its reverse is a "reverse-purge," also known as an "ex-nihilo" [45].) **C.** An "**s-merge**," *i.e.*, a "**same-type-merge**," which merges two gap-blocks supported by the same set of sequences. (Its reverse is an "s-split" of a gap-block.) **D.** A "**c-merge**", *i.e.*, a "**complementary-merge**," which merges gap-blocks supported by the *complementary sets* of sequences. (Its reverse is a "c-split" of a gap-block.) **E.** A "**v-merge**", *i.e.*, a "**vertical-merge**" of gap-blocks. (Its reverse is a "v-split" of a gap-block.) **F.** A "**cv-merge**", *i.e.*, a "**complementary-vertical-merge**" of residue-blocks (each *complementary to* a gap-block). (Its reverse is a "cv-split" of a residue-block.) [**Note:** This panel highlights the residue-blocks, instead of gap-blocks as the other panels do.] **G.** A "**revserse-CII**", *i.e.*, a "**reversal of a collapse of independent insertions**," which *recovers* the independent effective-insertions that *collapsed* when the MSA is reconstructed [3]. [This figure was adapted from Figure 3 of [3], with somewhat modified naming of the moves.]

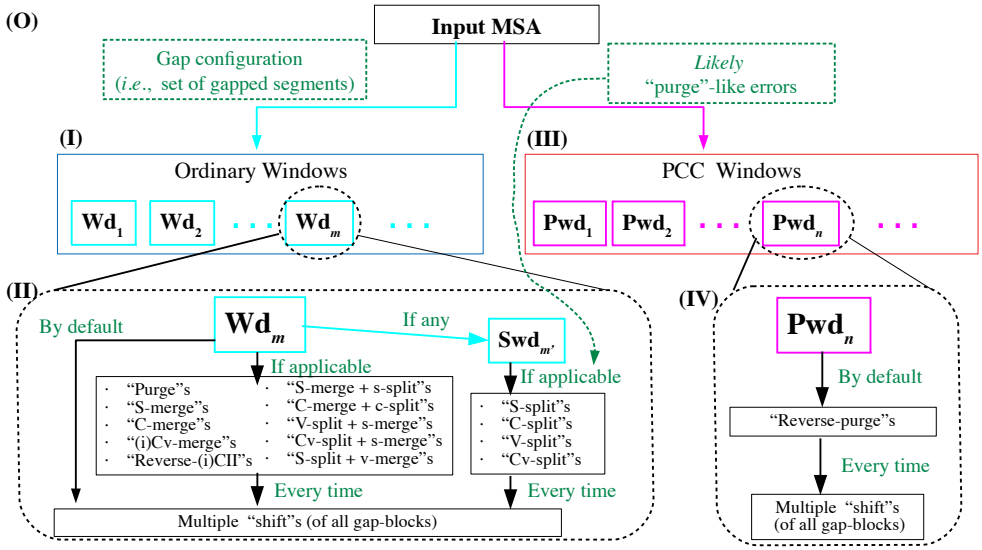


Figure 4: **ANEX's *current* architecture of MSA neighborhood exploration.** In **(III)**, the "PCC" stands for "purge(-like-error)-candidate-containing". In **(I)** and **(II)**, the "Wd_m" ($m = 1,2, ...$) represents the m-th ordinary window. And, in **(III)** and **(IV)**, the "Pwd_n" (n=1,2, ...) represents the n-th PCC window. For clarity, only one ordinary window and only one PCC window are chosen to show how the elementary moves are combined. However, it should be understood that every window attempts the combinations of the elementary moves following the structure shown in **(II)** or **(IV)**. The dark-green dashed curved arrow indicates that *likely* "purge"-like errors help ANEX decide whether to attempt "split"s or not. For more detailed description, see the text.

it attempts some elementary moves, or some *pairs* of elementary moves, that do *not* increase the number of gap-blocks (those listed in the middle-left box in (II)); after attempting each such move, the multiple-"shift"s of all the resulting gap-blocks follow. (3) ANEX attempts "split"s of any of the four types (those listed in the middle-right box in (II)), which *increase* the number of gap-blocks by one, *not directly* in the *full* ordinary window *but* in the sub-window; it attempts a "split" *only if* a likely "purge"-like error suggests that the move may be promising (the dark-green dashed curved arrow); each of such "split"-attempts is followed by the multiple-"shift"s of all the resulting gap-blocks *in the sub-window*. These restrictions on the "split"-attempts are intended to get the computation to finish within a reasonable amount of time.

Independently of the ordinary windows, ANEX also constructs "purge(-like-error)-candidate-containing" (PCC) windows [(III)], each of which contains a merged set of (*horizontally and vertically*) overlapping *likely* "purge"-like errors, as well as less than or equal to $N_{GB} - 2$ gap-blocks. Then [(IV)], in each PCC window (the "Pwd$_n$," with $n = 1, 2, ...$), ANEX first attempts "reverse-purge"s with some selected gap-block sizes; each attempt is followed by the multiple-"shift"s of all new gap-blocks in the PCC window.

[The current version of ANEX performs the neighborhood exploration within each ordinary window by calling a *satellite script*, "anex_for_sgl_wd.ver0.7.pl"; and it performs the neighborhood exploration within each PCC window by calling a second *satellite script*, "anex_rev_purge_for_sgl_wd.ver0.7.pl."]

In the *current version* of ANEX, the architecture of neighborhood exploration is quite simple and naïve. We hope that the architecture will evolve into a *smarter*, more sophisticated one in the future, maybe including appropriate combinations of three or more non-"shift"-type moves depending on the specific configurations of gaps and residues within the window (see [82] for more discussions).

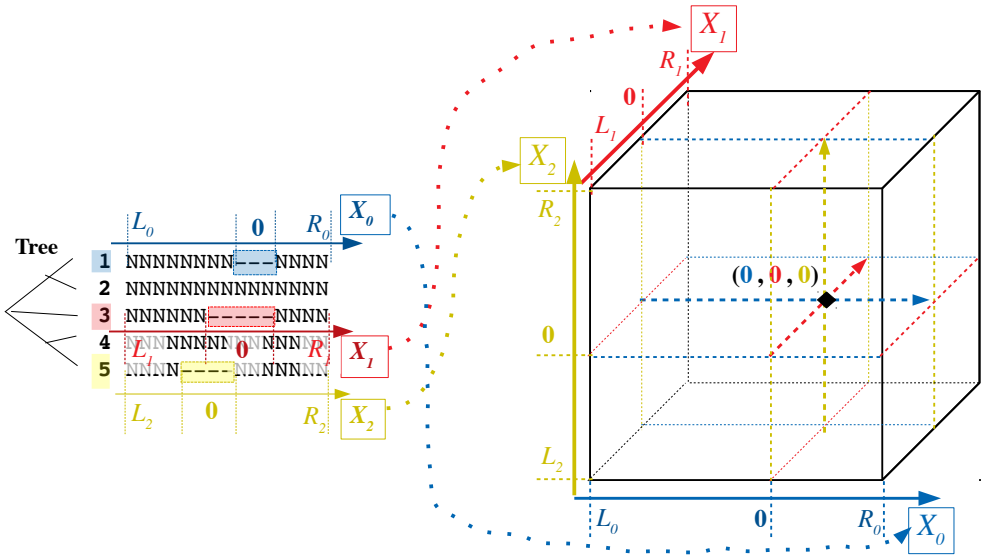


Figure 5: **Mapping multiple-"shift"s onto hyper-rectangular coordinate space.** This example MSA (on the left) contains three non-interfering gap-blocks (blue, red, and yellow), whose ?shift?s are represented by the integer coordinates, X_0, X_1, and X_2, respectively. Hence, each instance of the simultaneous ?shift?s of these gap-blocks corresponds to, *i.e.*, can be mapped onto, a point in a 3-dimensional hyper-rectangular lattice (on the right). (Here, the lattice is represented only with the framework and the origin, as well as colored dashed lines indicating the origins of the individual gap-blocks clarifying the coordinate frame.)

2.5.1 Algorithm to *exhaustively* perform multiple-"shift"s

As Figure 4 indicates, ANEX's neighborhood exploration *heavily depends on* multiple-"shift"s. It is thus crucial to *exhaustively* perform each set of multiple-"shift"s as efficiently as possible. We have developed a quite efficient algorithm to do this within each window of a given MSA, based on stacks with $(B+1)$ layers, where B is the number of gap-blocks in the window. [17]

Basically, we can associate each instance of multiple-"shift"s with a point in a multi-dimensional hyper-rectangular lattice that has integer coordinates, $X_k \in \{ L_k, \ldots, 0, \ldots, R_k \}$, with $k = 0, 1, \ldots B-1$ (Figure 5). Consider that an input MSA corresponds to the origin, $(X_0, X_1, \ldots, X_{B-1}) = (0, 0, \ldots, 0)$. Then, the point, *e.g.*, $(X_0, X_1, \ldots, X_{B-1}) = (k_0, k_1, \ldots, k_{B-1})$, corresponds to the alternative MSA created by "shift"ing the 0th gap-

[17] Actually, we *could* design an algorithm that works in almost the same way, using a recursive-call of a function (or subroutine). *Basically*, the algorithm we provide here was created from such a recursive-calling algorithm, by removing the recursion(s) following the recipe in [84] (chapter 5).

block by k_0, the 1st gap-block by k_1, ..., and the $(B-1)$-th gap-block by k_{B-1}, where k_0, k_1, ..., k_{B-1} are integers each of which may be positive, zero, or negative. [18] As soon as a gap-block is "shift"ed, ANEX computes the properties (including the probabilities) of the resulting MSA and stores the results into a B-dimensional array, whose array of level-$(k+1)$ corresponds to the axis of the coordinate X_k (as in Figure 6).

As briefly mentioned above, the stack used for *exhaustively* performing multiple-"shift"s consists of $(B+1)$ layers. The 0-th layer *constantly* stores information on the input MSA (within the window), whose coordinates are set to be $X_0 = X_1 = ... = X_{B-1} = 0$. And the $(k+1)$-th layer (with $k = 0, 1, ..., B-1$) *temporarily* stores information on the MSA in which the k-th gap-block was "shift"ed by X_k from its "parent MSA" in the k-th layer, which always has X_k $(= ... = X_{B-1}) = 0$. When"shift"ing the k-th gap-block, the algorithm updates the information stored in the $(k+1)$-th layer through the B-th layer. [21] The "shift"s are attempted always from the $(B-1)$-th gap-block down to the 0-th gap-block. Each gap-block (say, the k-th one) is first "shift"ed to the right one by one (as in Figure 6 A); when it is about to step "beyond" the right-boundary of the coordinate, it is returned to the origin (Figure 6 B), and is "shift"ed to the left one by one; when it is about to step "beyond" the left-boundary of the coordinate, the gap-block is returned to the origin again, and the algorithm goes one layer down, and "shift"s the $(k-1)$-th gap-block (Figure 6 C); finally, when the 0-th gap-block is about to step "beyond" the left-boundary, all the multiple-"shift"s end (Figure 6 D).

The elements described above are put together to form an algorithm to *exhaustively* perform multiple-"shift"s, which can be neatly expressed as the following *"pseudo-code"*. [22]

[18] We employ the convention that positive and negative integers correspond to the "shift"s to the right and to the left, respectively. A zero means that the gap-block in question remains un-"shift"ed at the initial position.

[21] The updated information in each of these layers is a copy of the information on the MSA resulting from the "shift" in question.

[22] In the "pseudo-code," some Perl notations are used. For example, a variable beginning with an "$" is

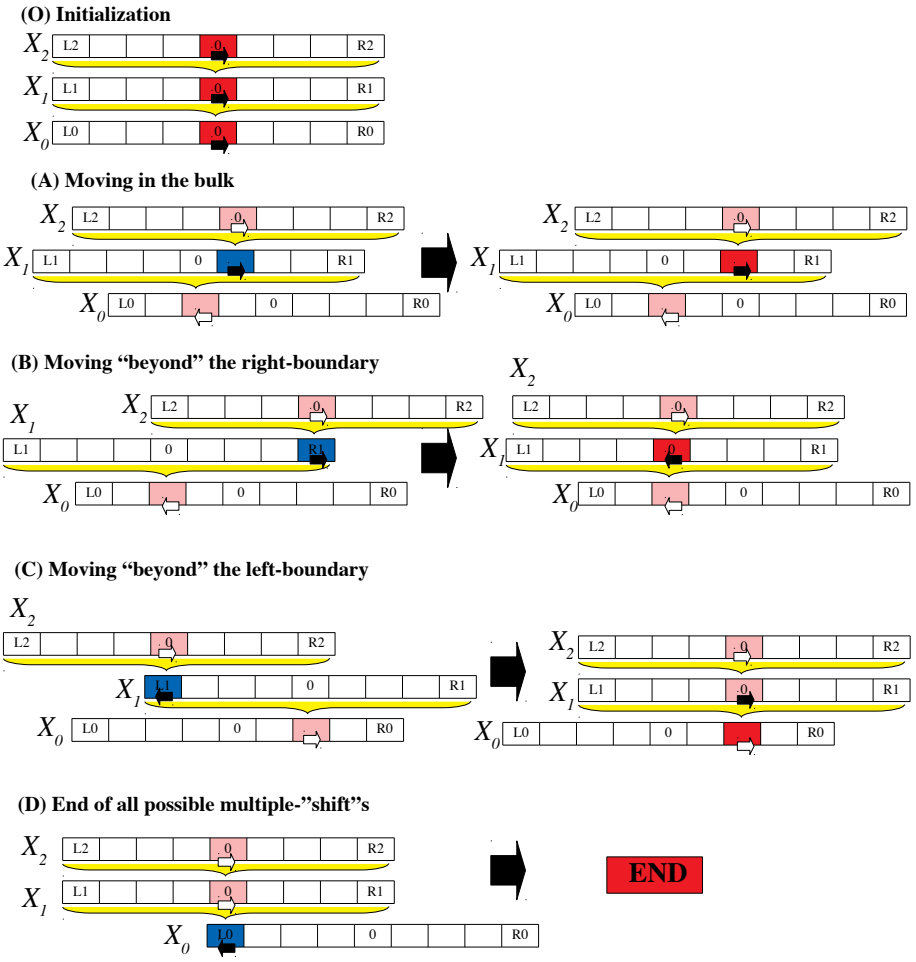


Figure 6: **Algorithm for exhaustively performing multiple-"shift"s of gap-blocks.** This figure shows some typical moves within the multi-dimensional array when ANEX exhaustively performs multiple-"shift"s of gap-blocks in a given window. For maximum clarity, the simplest non-trivial case of three blocks ($B = 3$) is illustrated. As in text, we set the origin to be $(X_0, X_1, X_2) = (0, 0, 0)$. Each horizontal run of boxes represents an array (with the left- and right- boundaries (L_k and R_k) and the origin (0) shown), which corresponds to the coordinate axis of X_k ($k = 0, 1, 2$) in Figure 5. [20] Each downward brace indicates that the array above it is essentially the element of the array below that it points at. In each panel (except (O)), the configurations before and after the move are on the left and right, respectively, of the big solid black arrow. The colored boxes represent the coordinates of the gap-blocks at that time. Especially, the solid blue and red boxes represent the coordinates handled before and after the move, respectively. The red transparent boxes represent the remaining coordinates. The (small) black arrow represents the direction of the "shift" handled at that time, and the white arrows represent the directions of other "shift"s.

[**Algorithm to *exhaustively* perform multiple-"shift"s**]

Input: An MSA within the window, represented as a set of columns

(= @A = (\@c_0, \@c_1, ..., \@$c_{|A|-1}$)), where $|A|$ is the number of columns in the MSA);

The number of gap-blocks (= \$$B$);

Some properties of gap-blocks, that remain *unchanged* during the multiple-"shift"s,

(= @*gb_prp*, with @{\$*gb_prp*[\$$k$]} storing the properties of the \$$k$-th gap-block (\$$k = 0, 1, ..., \$B - 1$));

The *initial* left- and right-ends of the gap-blocks, in terms of the column indexes in @A, (= @*gb_bd*0,

with @{\$*gb_bd*0[\$$k$]} = (left-end, right-end) for the \$$k$-th gap-block (\$$k = 0, 1, ..., \$B - 1$));

The coordinate boundaries of the gap-blocks, (= @L and @R, where \$$L$[\$$k$] and \$$R$[\$$k$]

are the left- and right-boundaries, respectively, of the \$$k$-th gap-block (\$$k = 0, 1, ..., \$B - 1$)).

Output: (The reference to) a \$$B$-dimensional array, @$Out$,

each of whose terminal elements, \$$Out$[\$$i$[0]] $\rightarrow$ [\$$i$[1]]...[\$$i$[\$$B - 1$]],

is the reference to an array storing output properties of the MSA resulting from "shift"ing

the k-th gap-block by \$$X$[\$$k$](= \$$i$[\$$k$] + \$$L$[\$$k$]) sites (with \$$k = 0, 1, ..., \$B - 1$)

from the input MSA.

Algorithm:

Initialize @Out, so that all its terminal elements refer (or point) to respective empty arrays.

Initialize a \$$B$-dimensional array, @$X$(= (\$$X$[0], \$$X$[1], ..., \$$X$[\$$B - 1$])) ,

which specifies the *current* coordinates of the gap-blocks, to be $(0, 0, ..., 0)$.

Compute some properties of the input msa, @A, and store them into an array, @*prpmsa* .

Store the contents of @*prpmsa* into the terminal element of @Out corresponding to the "origin", *i.e.*,

---

a scalar. A variable beginning with an "@" is an array (or a vector), the reference (or pointer) to which is represented by prepending a "\" to it. \$$a$[\$$k$] denotes the \$$k$-th element of an array, @a. When a variable, \$*pa*, refers (or points) to an array, the array itself is denoted as @{\$*pa*}, and \$$p \rightarrow$ [\$$k$] denotes the \$$k$-th element of the array.

$@\{\$Out[0 - \$L[0]] \rightarrow [0 - \$L[1]]...[0 - \$L[B-1]]\} \leftarrow @prpmsa$.

Initialize the stack, $@Stack$, as follows:

Create $@Stack \leftarrow (\ )$.

For $\$n$ from 0 to $\$B$, do the following:

Create $@cp\_A \leftarrow$ (a copy of $@A$);

Create $@gb\_bd \leftarrow$ (a copy of $@gb\_bd0$) ;

Create $@cp\_X \leftarrow$ (a copy of $@X$);

Create $@cp\_prpmsa \leftarrow$ (a copy of $@prpmsa$);

$@\{\$Stack[\$n]\} \leftarrow (\backslash @cp\_A, \backslash @gb\_bd, \backslash @cp\_X, \backslash @cp\_prpmsa)$.

EndFor ($n)

Initialize a B-dimensional array, $@sh$, which specifies the directions of the "shift"s,

to be $(+1, +1, ..., +1)$; (NOTE: the $\$k$-th gap-block (with $\$k = 0, 1, ..., \$B - 1$)

"shift"s to the right/left (by one site) if $\$sh[\$k] = +1/-1$.)

Initialize a scalar, $\$b$, which specifies the gap-block to be "shift"ed, to be $\$B - 1$.

Initialize a scalar, $\$wrk\_layer$, which refers (or points) to the working layer, as follows:

Create $\$wrk\_layer \leftarrow (\text{pop } @Stack)$.

While (1), which means to loop "forever" until interrupted, do the following:

Create $(\$pA,\ \$pgb\_bd,\ \$pX,\ \$pprpmsa) \leftarrow @\{\$wrk\_layer\}$;

Create $\$gb\_coord\_bf \leftarrow (\$pX \rightarrow [\$b])$;

Create $\$gb\_coord\_af \leftarrow \$gb\_coord\_bf + \$sh[\$b]$;

If the $b th gap-block is about to step "beyond" the right-boundary

(*i.e.*, $\$R[\$b] < \$gb\_coord\_af$), then:

Create @*cp_layer* $\leftarrow$ (a copy of @{$*Stack*[$*b*]}) ;

$*wrk_layer* $\leftarrow$ \@*cp_layer* ; (Get the $*b*-th gap-block back to the origin.)

$*sh*[$*b*] $\leftarrow$ -1 ; (Flip the direction of the "shift".)

next ;

ElseIf the $b th gap-block is about to step "beyond" the left-boundary

(*i.e.*, $*gb_coord_af* $< \$L[\$b]$), then:

If $b = 0, then:

last ; (END of all possible multiple-"shift"s.)

EndIf

$*sh*[$*b*] $\leftarrow$ $+1$; (Flip the direction of the "shift".)

$*wrk_layer* $\leftarrow$ (pop @*Stack*) ; (Go one layer down,)

$*b* $\leftarrow$ $*b* $- 1$; (that is, "shift" the ($*b* $- 1$)-th gap-block from next.)

next ;

EndIf

{ *Actually "shift"* the $*b*-th gap-block by $*sh*[$*b*],

and *update* @{$*pA*}, @{$*pgb_bd*} and @{$*pX*}, to reflect the resulting changes };

{ *Compute* the properties of the resulting MSA,

and *update* @{$*pprpmsa*} by *storing* the results into it };

@{$*Out*[$*i*[0]] $\rightarrow$ [$*i*[1]]...[$*i*[$*B* $- 1]]} $\leftarrow$ @{$*pprpmsa*} ,

where $*i*[$*k*] = $*pX* $\rightarrow$ [$*k*] $-$ $*L*[$*k*] for $*k* $= 0, 1, ..., \$B - 1$.

If $*b* $< \$B - 1$, then:

For $*n* from $*b* $+ 1$ to $*B* $- 1$, do the following:

Create @*cp_wrk_layer* $\leftarrow$ (a copy of @{$*wrk_layer*}) ;

$*Stack*[$*n*] $\leftarrow$ \@*cp_wrk_layer*; (Update the higher-layers of the stack.)

EndFor (n)

$b \leftarrow B - 1$; (Update $b, which specifies the subject gap-block.)

EndIf

EndWhile (1)

Return \@*Out* .

EndAlgorithm

[For the actual implementation in Perl, see the subroutine, "nonrec_mlt_shifts_gapblocks", in the module, "MyANEX_Main.pm", in ANEX.] [23]

In retrospect, this algorithm could be simplified to some extent, if all the gap-blocks are shifted to the left-boundaries of their coordinates *before* the (infinite) while-loop; then, in the while-loop, we need only to "shift" the gap-blocks to the right, which makes @*sh* unnecessary.

2.5.2 Performing single "shift" of gap-block: essence

The sub-subsection 2.5.1 above provided an algorithm to *exhaustively* perform multiple-"shift"s. The fundamental building-block of such multiple-"shift"s is the **single "shift"** of each individual gap-block. Actually, this fundamental process can be *concretely* realized in quite a simple manner.

Figure 7 illustrates the simplest case where a single *isolated* gap-block is "shift"ed by one site, to the left and to the right. By carefully comparing the MSAs before and after each

[23] This subroutine may appear quite messy because it also performs several other works while performing multiple-"shift"s, and also because it incorporates several measures to speed up each step (of multiple-"shift"s). Once you know the above "essence" of the algorithm, however, it should not be so hard to "decipher" the actual code.

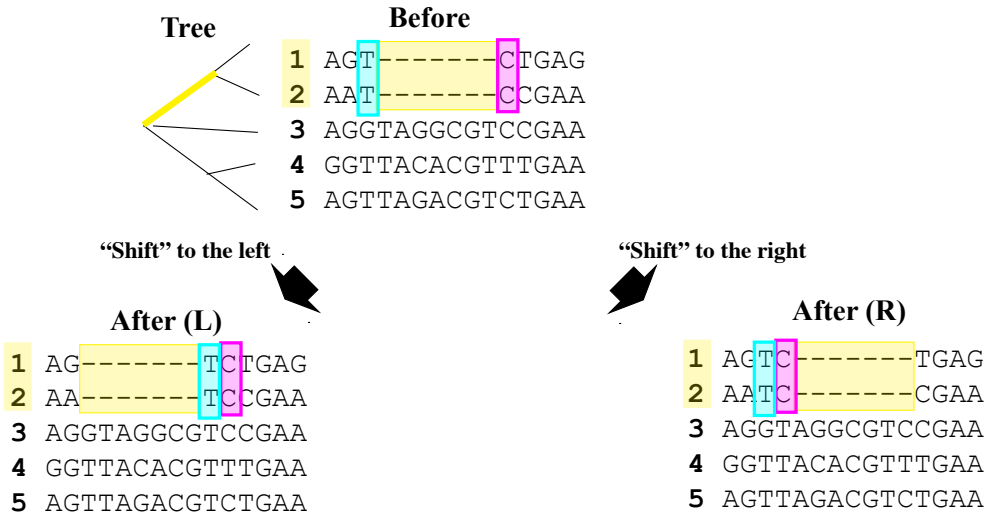


Figure 7: **Changes in MSA caused by single "shift"s of gap-block.** The subject gap-block and the sequences supporting it are shaded in yellow. And the yellow branch phylogenetically delimits the gap-block. Before the "shift"s, the left- and right-flanking sites in the gap-block-supporting sequences are shaded in cyan and magenta, respectively. Carefully compare the positions of these cyan- and magenta-shaded sites before and after the single-"shift"s.

of the single-"shift"s in this figure, we notice that **the *effects* of each single-site "shift"** boils down to the following *"essence"*:

> **In each of the gap-block supporting sequences,**
> ***swap* the site *at the rear-end* of the gap-block**
> **with the one *flanking the front-end* of it.**

This simple nature of the single-"shift" will help *quickly* compute the increment of the substitution component of the log-probability of the MSA in later subsection.

When the gap-block is surrounded by other gap-blocks that *interfere with* it, some complications may arise. In such cases, however, the above "essence" still continues to hold, *ableit* with some additional adjustments. (See appendix E for details.)

Currently, ANEX skips downstream analyses on alternative MSAs resulting from multiple-"shift"s if they have particular gap-block configurations. (See appendix F.) This way, ANEX avoids redundancies in the alternative MSAs resulting from the entire neighborhood exploration. Note that the multiple-"shift"s themselves are continued.

2.5.3 Performing non-"shift"-type moves of gap-blocks

The moves of gap-blocks other than the "shift"s (see Figure 3, appendix D) could be realized in *essentially* the same manner as one or more single-"shift"s we discussed in sub-subsection 2.5.2. You need only to be careful about the "supporting" sequences to be affected by the move. In addition, in some "split"s or "merger"s, we need to add or remove some null-columns before or after the "shift"s. Moreover, in "split"s, you must *conceptually* "split" the subject gap-block, *before* performing necessary "shift"s; in "merge"s, you must *conceptually* "merge" the subject gap-blocks, *after* performing necessary "shift"s; these *conceptual* "split"s/"merge"s re-organize the set of gap-blocks.

The exceptions to the "shift"-based realizations mentioned above are the realizations of the "cv-split", "cv-merge", and "reverse-(i)CII" (*e.g.*, Figure 3 F & G); in these cases , you need to handle *residue-blocks*, instead of gap-blocks. In the next sub-subsection, we discuss "reverse-(i)CII," which is most highly non-trivial.

[For the actual implementation of these non-"shift"-type moves (except "reverse-(i)CII"), see the relevant subroutines(, which you can probably tell by the names,) in the module, "MyANEX_Main.pm", of ANEX.]

2.5.4 Performing "reverse-(i)CII"

Performing "reverse-(i)CII" may be quite hard *if you stick to the gap-blocks*, because a "reverse-(i)CII" usually changes the (effective-)indel history, hence the gap-block structure, *drastically* (Figure 3 G). Instead, if you pay attention to the *residue-blocks*, it may be relatively easy, because "revers"ing a(n) "(i)CII" is, after all, nothing other than *extracting* individual residue-blocks and *separating* them from each other *horizontally* (Figure 3 G).

Broadly speaking, "reverse-(i)CII"s can be performed in two steps: (1) detecting candidate regions that may have suffered "(i)CII"s; and (2) actually performing the "reverse-(i)CII" on each of such "(i)CII"-candidate regions detected. An algorithm for step (1) is

described in appendix G, and that for step (2) is described in appendix I. In step (1), dividing a set of sequences into monophyletic groups plays an important role; an algorithm for this job is provided in appendix H.

Unlike other moves, during which rearranging the set of gap-blocks is relatively easy, the set of gap-blocks must be *constructed from scratch* after the "reverse-(i)CII".

It should also be noted that the current architecture of ANEX (Figure 4) is *not* capable of taking *full* advantage of "reverse-(i)CII"s. This is because they are attempted *after* the creation of (especially ordinary) windows. For example, a(n) "(i)CII" of two residue-blocks (the reverse of Figure 3 G) typically removes two (or one) (effective-)insertion(s) and creates as many (effective-)deletions as the trivalent nodes separating the *true* residue-blocks. Therefore, if the "(i)CII"ed *true* residue-blocks are separated by three or more trivalent nodes, the resulting gapped segment in the reconstructed MSA should have more gap-blocks than the original gapped segment(s) in the *true* MSA. In consequence, the "(i)CII"ed gapped segment will often be excluded from the (ordinary) window-analysis *from the beginning*, especially when the number of aligned sequences is large. One way to avoid this problem should be to perform the "reverse-(i)CII"s *before* creating (ordinary) windows and, after that, to create windows containing "reverse-(i)CII"ed segments in parallel with the windows only containing original input segments. This way, the "reverse-(i)CII"s should be fully exploited, although it may become somewhat bothersome to manage and record the resulting windows and the moves attempted within each window.

2.6 Computing logarithmic probabilities of alternative MSAs

As already mentioned in 1.2, ANEX uses *genuine* sequence evolution models that satisfy the conditions posed in [3]. [24] Under such evolution models, the probability of each MSA can be factorized into the ***substitution component***, which depends only on the residue configura-

[24] Actually, these conditions are satisfied by *most of* the sequence evolution models that have been used in the studies thus far.

tion of the MSA, and the ***indel component***, which depends only on the gap-configuration of the MSA. Thus, given an alternative MSA (within each window), ANEX *separately* computes the substitution and indel components of its probability, *under* a *given* phylogenetic tree and a *given* sequence evolution model.

According to ANEX's current architecture of the MSA neighborhood exploration (Figure 4), ANEX first performs zero, one or two non-"shift"-type moves of gap-blocks (sub-subsection 2.5.3), then, using the resulting MSA as a new starting point (or "origin"), it further performs multiple-"shift"s *exhaustively* within a hyper-rectangular coordinate space (sub-subsection 2.5.1). Taking account of this architecture, ANEX employs an *asymmetrical* strategy for computing the probabilities of MSAs resulting from non-"shift"-type moves and those resulting from multiple-"shift"s. In short, *full* computations of both components are performed every time when a non-"shift"-type move is finished, whereas time-saving measures are taken when computing the components of an MSA resulting from a single-"shift". In the following sub-subsections, we give more details on these computations.

2.6.1 Logarithmic probabilities of MSAs resulting from non-"shift"-type moves

As explained above, ANEX *fully* compute the substitution and indel components of the logarithmic probability of the resulting MSA, every time when a non-"shift"-type move (or a pair of such moves) is finished. Regarding the substitution component, this means that ANEX *fully* performs the pruning algorithm [20, 21]. [25] As most of the programs and methods, ANEX uses substitution models in which each site of the sequence evolves *independently* of the other sites. [26] Therefore, the probability of each MSA (say $\mathcal{A}$), under a tree ($= \mathcal{T}$) and a

[25] But ANEX does *not* resort to the pulley principle, because ANEX allows for *non-equilibrium* evolution processes, which have often been observed in genome data analyses (*e.g.*, [85]).

[26] Actually, the current version of ANEX uses only space-homogeneous models, in which the substitution rate and matrix are uniform across the sites in the sequence. However, it should be possible to extend the program so that it can also use models in which the sites are distributed into discrete classes of substitution rates [86, 87, 26].

substitution model ($= \Theta_S$) can be factorized into the product of contributions from all sites (or columns):

$$P\left[\mathcal{A} \mid \mathcal{T}, \Theta_S\right] \;=\; \prod_{k=1}^{N_c(\mathcal{A})} P\left[c_k \mid \mathcal{T}, \Theta_S\right] \;, \tag{1}$$

where the c_k's (with $k = 1, ..., N_c(\mathcal{A})$) denote the columns in $\mathcal{A}$. Thus the logarithmic probability of $\mathcal{A}$ is expressed as the summation over all sites (or columns):

$$\log\left\{P\left[\mathcal{A} \mid \mathcal{T}, \Theta_S\right]\right\} = \sum_{k=1}^{N_c(\mathcal{A})} \log\left\{P\left[c_k \mid \mathcal{T}, \Theta_S\right]\right\} \;. \tag{2}$$

This fact, as well as the pruning algorithm [20, 21], enables a fairly fast computation of the logarithmic probability of each MSA. Thus, the *full* computation of the substitution component after each non-"shift"-type move will *not* become the rate-limiting step.

The current version of ANEX uses Tamura and Nei's model (TN93) of base substitutions [88]. This TN93 model, which has 6 degrees of freedom, is quite flexible, and includes, as its special cases, Jukes and Cantor's model (JC69) [89], Kimura's 2-parameter model (K80) [90], Felsenstein's 81 model (F81) [20], and Hasegawa, Kishino and Yano's model (HKY85) [91]. [27]

Regarding the indel component, ANEX *basically* employs the computational method we proposed previously to compute the MSA probability under a given *genuine* sequence evolution model with *realistic* insertions/deletions (denoted as Θ_{ID}) [1, 2], but with a couple of *important* modifications. To briefly review our previous method [2] [28] : (1) *horizontally chop*

[27] We are aware of some methods to *numerically* compute the finite-time transition probabilities under the general time-reversible model [92, 93, 94, 95] (see also section 2.6 of [87]), which has 9 degrees of freedom and more flexible than TN93. However, we have not had time to incorporate it into the current version of ANEX. This is left as a future task.

[28] To apply this method, the *genuine* sequence evolution model is assumed to satisfy the "sufficient and nearly necessary set of conditions" for factorable MSA probabilities [1]. Briefly, the conditions are: (i) the rate of each indel is independent of the sequence states in regions outside of that affected by the indel; (ii) the increment of the exit rate due to each indel is independent of the sequence states in regions outside of that

the input MSA ($\mathcal{A}$) [29] into gapped segments (denoted as C_K^0 with $K = 1, ..., N_{GS}$, where N_{GS} is the number of gapped segments in the MSA [30]) and gapless segments (denoted as B_J^0 with $J = 1, ..., N_{GLS}$, where N_{GLS} is the number of gapless segments in the MSA); (2) for each gapped segment (C_K^0), *enumerate* the parsimonious indel histories that can create the segment, using our **"local multi-path downhill search algorithm"** [2]; (3) compute the *1st-approximate* multiplication factor, $\tilde{\tilde{\mathrm{M}}}_P^{\langle 1st \rangle}[\mathcal{A}; s_0^{Root}; C_K^0 \mid \mathcal{T}, \Theta_{ID}]$, contributed from each gapped segment (C_K^0), by *summing* the contributions from *all the* parsimonious indel histories; and (4) compute the *1st-approximate* indel component of the MSA probability as the product of an overall factor and contributions from all the gapped segments, more specifically:

$$P^{\langle 1st \rangle}[\mathcal{A} \mid \mathcal{T}, \Theta_{ID}] = P[(s_0^{Root}, n^{Root})] \times \exp\{-((\lambda_I + \lambda_D) \times N_{GLC} + \Delta[\Theta_{ID}]) \times |\mathcal{T}|\} \times \times \prod_{K=1}^{N_{GS}} \tilde{\tilde{\mathrm{M}}}_P^{\langle 1st \rangle}[\mathcal{A}; s_0^{Root}; C_K^0 \mid \mathcal{T}, \Theta_{ID}] \ . \quad (3)$$

Here, the s_0^{Root} denotes the "presence"/"absence" state [31] of the "reference" ancestral sequence at the root (n^{Root}) [32] ; the $P[(s_0^{Root}, n^{Root})]$ is the probability that, at the root, the ancestral sequence state is s_0^{Root} [33] ; the λ_I and λ_D denote the per-site rate of insertion and deletion, respectively; the N_{GLC} denotes the number of gapless columns in the MSA(, which equals the number of sites in s_0^{Root}); the $|\mathcal{T}|$ denotes the "total tree length", which is the summation of all branch lengths in $\mathcal{T}$; the exponential factor on the right of the $P[(s_0^{Root}, n^{Root})]$ is the probability that, given the "presence"/"absence" state s_0^{Root} at the root, the state is

affected by the indel; and (iii) the probability of the ancestral sequence state at the root is factorable into the product of an overall factor(, which is the probability of a "reference" sequence state) and incremental factors from gapped segments.

[29] In [2], the MSA (= $\mathcal{A}$) was denoted as $\alpha[s_1, ..., s_{N^X}]$, where s_k is the k-th sequence in the alignment, and N^X is the number of external nodes (*i.e.*, sequences).

[30] In [2], the number of gapped segments (= N_{GS}) was denoted as $\mathrm{K}_{\max}^0$.

[31] The "presence" means that the site (or a cell in the MSA column allotted to the sequence) is occupied by a residue (or base), and the "absence" means that the site is occupied by a gap.

[32] As s_0^{Root}, we use the concatenation of all the gapless segments, $B_1^0, ..., B_{N_{GLS}}^0$, as in [2].

[33] In most of practical situations, we can safely assume that $P[(s_0^{Root}, n^{Root})] \propto 1$, as argued in [2].

conserved *all through the tree* ($\mathcal{T}$); in the exponent, the expression multiplied by the $|\mathcal{T}|$ is the "exit rate" from the state s_0^{Root} (to any other states) in the model (Θ_{ID}); especially, the $\Delta[\Theta_{ID}]$ denotes the (constant) "boundary effect" on the exit rate, which depends very much on the model (Θ_{ID}). [34]

Because the "1st-approximation", Eq. 3, is basically the summation of the contributions from *parsimonious* indel histories *alone*, its accuracy is limited, especially when long indels are involved [2]. To overcome this problem, we here *enhance* the "1st-approximation", to *define* the "*1st*-**plus** *approximation*," by summing *all the* indel histories (including the *non-parsimonious* ones) that result in any of the sets of parsimonious ancestral states at all internal nodes.

To do this, we need to remember the method for computing the probability of an MSA *with a set of fixed ancestral states at all internal nodes* (also aligned with extant sequences) [58, 59]; although the method was originally devised in the context of hidden Markov models (or transducers), we now know that it is valid also with *genuine* sequence evolution models [1]. In short, the method first regards the MSA with a set of ancestral states as a "stack" of ancestor-descendant PWAs, one at each branch of the tree. Then, the method computes the probability of the MSA with fixed ancestral states as the product and the (conditional) probabilities of ancestor-descendant PWAs over all branches, as well as of the probability of the root sequence state. Regarding the (conditional) probability of each ancestor-descendant PWA under a *genuine* sequence evolution model, it can be computed using the method proposed in [76] under a space-homogeneous model, which we extended later to more general cases [1]. Briefly, the PWA is *horizontally* chopped into "preserved ancestral sites" (PASs), in each of which an ancestral residue is aligned with a descendant residue, and "gapped segments" in between the PASs next to each other. [35] Then, the probability of the PWA is

[34] As $\Delta[\Theta_I D]$, ANEX uses Dawg's boundary effect [77], $\Delta^{Dawg}[\lambda_I, \lambda_D, f_D(.), L_D^{CO}] = \lambda_I + \lambda_D \times \{\sum_{l=1}^{L_D^{CO}} (l-1) f_D(l)\}$, where the L_D^{CO} is the cutoff-length for the deletion and the $f_D(l)$ (with $l = 1, ..., L_D^{CO}$) is the relative frequency of the deletion of length l.

[35] We *formally* regard that a "gapped segment" exists *even* in between a pair of *adjoining* PASs. Such a

computed as the product of an overall factor, which is the probability that the ancestral sequence state is preserved all along the branch, and the multiplication factors from all gapped segments [1]; the multiplication factor from each gapped segment, in turn, is the summation of the contributions from *all* indel histories (including *non-parsimonious* ones) that can result in the gapped segment. Our recently developed program package, **LASTPIECE(_P)**, can compute the multiplication factors from these gapped segments *quite accurately*, taking account of contributions from *non-parsimonious* indel histories as well [4]. [36] Therefore, this package can be used for the problem at hand.

Taking advantage of these previous achievements, we compute the multiplication factor in the "*1st*-**plus** *approximation*," $\overset{\sim}{\mathrm{M}}{}_P^{\langle 1st+\rangle}[\mathcal{A}; s_0^{Root}; C_K^0 \mid \mathcal{T}, \Theta_{ID}]$, as follows: (i) for each gapped segment (C_K^0), *enumerate* all sets of parsimonious ancestral "presence"/"absence" states at internal nodes; (ii) for each set of ancestral states, construct a "stack" of ancestor-descendant

"gapped segment," which consists of *no* ancestral or descendant sites, is categorized as "case-(i)."

[36] More precisely, the gapped segments in ancestor-descendant PWAs are categorized into four cases [2]: case-(i) consists of no sites in between the PASs; case-(ii) consists of ancestral residues aligned with gaps in the descendant; case-(iii) consists of descendant residues aligned with gaps in the ancestor; and case-(iv) consists of ancestral residues aligned with gaps in the descendant and descendant residues aligned with gaps in the ancestor, with no homology between the ancestral and descendant residues. In [2], we devised a pair of algorithms to compute "practically exact" multiplication factors of case-(i), (ii) and (iii) gapped segments, but an accurate computation of case-(iv) multiplication factors was unresolved *at that time*. Very recently [4], we have invented **a new "perturbation method"** that enables the *systematic* computation of case-(iv) multiplication factors *to a desired level of accuracy* (in principle). And our validation analyses demonstrated that the case-(iv) multiplication factors computed with this new method were *surprisingly accurate* even up to (and including) the 3rd-order "perturbation" level [4]. [**NOTE:** Unlike the perturbative formulation in [1, 2], the "perturbation level" in this new method is *not* equal to the number of *all* indels in each history. In fact, even the 1st-order "perturbation" level includes indel histories with any numbers of indels.] This new "perturbation method" (up to 3rd order), which computes the case-(iv) multiplication factors quite accurately, and a pair of algorithms that compute case-(i), (ii), (iii) multiplication factors much faster than, yet as accurately as, the algorithms proposed in [2], have been implemented in our new program package, **LASTPIECE(_P)** [4].

PWAs out of the input MSA (in each C_K^0); (iii) using the outputs of LASTPIECE(_P) [4], which provides *quite accurate* multiplication factors of PWA gapped segments, compute the (conditional) probability of the ancestor-descendant PWA along each branch *quite accurately*, according to [76, 1] [37] ; (iv) *quite accurately* compute the probability of the MSA (in each C_K^0) with each set of parsimonious ancestral states as the product of the probabilities of ancestor-descendant PWAs and the probability of the root sequence state, according to [58, 59, 1]; and (v) sum the results of (iv) over all the sets of parsimonious ancestral states, to provide the multiplication factor, $\overset{\sim}{\mathrm{M}}_P^{\langle 1st+\rangle}[\mathcal{A}; s_0^{Root}; C_K^0 \mid \mathcal{T}, \Theta_{ID}]$.

When *defining* the "*1st*-**plus** *approximation*" version of Eq.3, another factor must be taken into account. Each gapless segment (B_J^0 with $J = 1, ..., N_{GLS}$), which consists only of gapless columns, must now be regarded as containing a case-(i) gapped segment in between each pair of contiguous PASs along each branch. Thus, let N_{iGLCp} be the number of "inter-gapless-column-positions", each of which is the position in between a pair of contiguous (*i.e.*, adjoining) gapless columns, and let N_{bGLC} be the number of "boundary-gapless-columns," each of which is a gapless column on a "sequence boundary" (*i.e.*, MSA-end). Then, the "*1st*-**plus** *approximation*" version of Eq.3 is *defined* as:

$$\begin{aligned} P^{\langle 1st+\rangle}[\mathcal{A} \mid \mathcal{T}, \Theta_{ID}] &= P[(s_0^{Root}, n^{Root})] \times \exp\{-((\lambda_I + \lambda_D) \times N_{GLC} + \Delta[\Theta_{ID}]) \times |\mathcal{T}|\} \times \\ &\times \left(\overset{\sim}{\mathrm{M}}_P^{\langle 1st+\rangle}[\mathrm{iGLCp} \mid \mathcal{T}, \Theta_{ID}]\right)^{N_{iGLCp}} \times \left(\overset{\sim}{\mathrm{M}}_P^{\langle 1st+\rangle}[\mathrm{bGLC} \mid \mathcal{T}, \Theta_{ID}]\right)^{N_{bGLC}} \times \\ &\times \prod_{K=1}^{N_{GS}} \overset{\sim}{\mathrm{M}}_P^{\langle 1st+\rangle}[\mathcal{A}; s_0^{Root}; C_K^0 \mid \mathcal{T}, \Theta_{ID}] \ . \end{aligned} \tag{4}$$

Here, the $\overset{\sim}{\mathrm{M}}_P^{\langle 1st+\rangle}[\mathrm{iGLCp} \mid \mathcal{T}, \Theta_{ID}]$ is the multiplication factor (for an MSA) contributed from an "inter-gapless-column-position"; it is defined as:

$$\overset{\sim}{\mathrm{M}}_P^{\langle 1st+\rangle}[\mathrm{iGLCp} \mid \mathcal{T}, \Theta_{ID}] \overset{\mathrm{def}}{=} \prod_{b \ \in \ \{\text{all branches in } \mathcal{T}\ \}} \breve{\mu}_P[\ \text{case-(i), (bulk)}\ \mid b, \Theta_{ID}] \ . \tag{5}$$

[37] LASTPIECE(_P) outputs multiplication factors (for ancestor-descendant PWAs) at some fixed (typically equal-spaced) time-lapses. ANEX(_P) takes in such outputs as inputs and *interpolates* the multiplication factors in order to well-approximate the factors along each branch of the input tree.

where the $\breve{\mu}_P[$ case-(i), (bulk) $\mid b, \Theta_{ID}]$ is the case-(i) multiplication factor that is defined in the bulk, *i.e.*, off the boundary, of the sequence along the branch, b . And the $\tilde{\breve{\mathrm{M}}}_P^{\langle 1st+\rangle}[\mathrm{bGLC} \mid \mathcal{T}, \Theta_{ID}]$ is the multiplication factor (for an MSA) contributed from a "boundary-gapless-column"; it is defined with an equation similar to Eq. 5, but with each factor defined on the sequence boundary, instead of in the bulk. This Eq. 4(, actually, its logarithm,) is what ANEX aims to compute for each MSA within each window (including the input one), using (the interpolation of) the case-(i), (ii), (iii) and (iv) multiplication factors output by LASTPIECE [4].

[For the actual implementation of the indel component computation in the "*1st*-**plus** *approximation*," see the subroutine, "apprx1_wlp_tot_lnlk_gpattern_msa2_hs," in the module, "MyTreeMap_indels_ML_hs_hs_wLP.pm," in ANEX.]

2.6.2 Substitution components of log-probabilities of MSAs resulting from single-"shift"s

In contrast to the *full* computation immediately after each non-"shift"-type move, ANEX *more efficiently* computes the substitution components of the alternative MSAs resulting from single-"shift"s. This is because it needs to compute the log-probabilities of thousands, or sometimes even millions, of alternative MSAs in a single run of the algorithm for multiple-"shift"s; with this situation, even the (fairly efficient) pruning algorithm could pose a rate-limiting step.

The *efficient* computation is enabled by two technical tricks. The first trick comes from the *"essence"* of the single-"shift" of a gap-block (in sub-subsection 2.5.2) as well as the *column-wise* factorability of the substitution component (Eq. 2). As the "essence" dictates, the single-"shift" of a gap-block affects *only two* columns, that at the rear-end of the gap and that flanking the front-end of the gap (*before* the "shift") (Figure 7). Meanwhile, Eq. 2 indicates that the log-probabilities of the remaining columns are left unchanged, and thus ignored when you compute the *increment*, $\Delta_{\text{sgl-shift }(gb)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}$, due to the single-

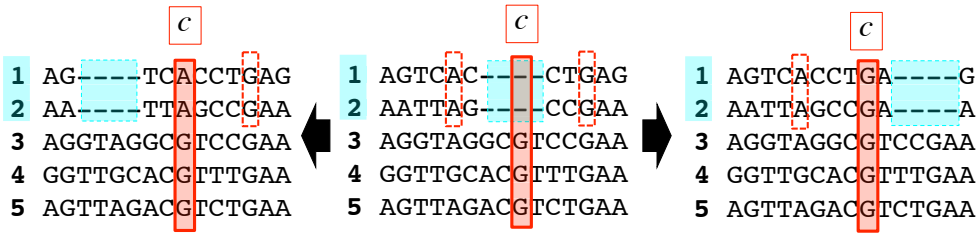


Figure 8: **Possible patterns at each column-position caused by "shift"s of single gap-block.** The "shift"s of a single gap-block (cyan-shaded) give rise to at most 3 possible (residue & gap) patterns at each column-position (a representative is red-shaded). The red dotted rectangles enclose the original "ingredients" of the columns (at the representative position) that resulted from the "shift"s.

"shift" of a gap-block (denoted as "gb"). Symbolically, we have:

$$\begin{aligned}
&\Delta_{\text{sgl-shift }(gb)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\} \\
&= \Big[\log\{P[c_{k_1} \mid \mathcal{T}, \Theta_S]\} + \log\{P[c_{k_2} \mid \mathcal{T}, \Theta_S]\}\Big]\big|_{\textbf{after the "shift" of } gb} \\
&\quad - \Big[\log\{P[c_{k_1} \mid \mathcal{T}, \Theta_S\,]\} + \log\{P[c_{k_2} \mid \mathcal{T}, \Theta_S\,]\}\Big]\big|_{\textbf{before the "shift" of } gb}\ , \qquad (6)
\end{aligned}$$

where the c_{k_1} and c_{k_2} denote the two columns affected by the single-"shift." Thus, each time when a single-"shift" is performed, we need *only* to perform the (column-wise) pruning algorithm *at most* four times. This is much faster than the full computation of the substitution component if the MSA contains dozens of columns.

The second trick comes from the fact that, in general, the number of possible alternative columns is *much smaller* than the number of columns in the input MSA times the number of multiple-"shift"s attempted. This fact could be easily understood if we first consider the effect of the "shift"s of a single gap-block (Figure 8). It should be noted that the "shift"s of a gap-block are "rigid" moves, which do *neither* split *nor* resize the gap-block. Thus, the "shift"s of a single gap-block results in *at most* three (residue & gap) patterns of the column at each specific position in an MSA (Figure 8). When considering the simultaneous "shift"s of two *non-interfering* gap-blocks, the maximum number of possible resulting patterns at each specific column position becomes 9 ($= 3 \times 3$) (Figure 9). When two gap-blocks *interfere* with each other, the number *never* exceeds 9; rather, it could be less. [38] Generalizing this

[38] The number is less than 9 if the two gap-blocks vertically overlap each other.

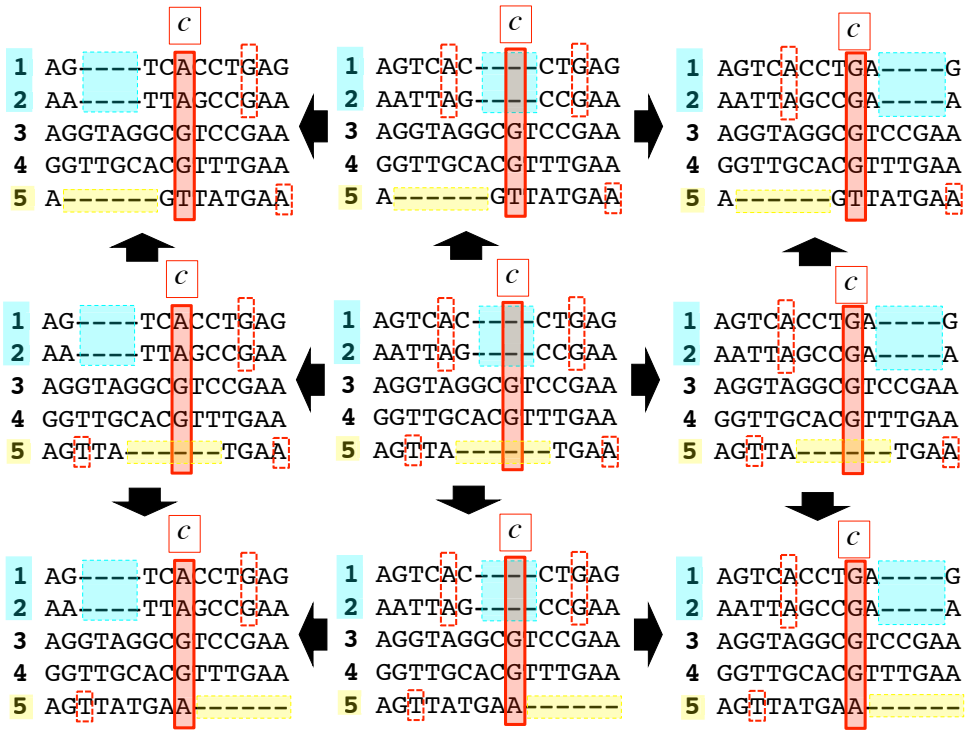


Figure 9: **Possible patterns at each column-position caused by simultaneous "shift"s of two gap-blocks.** The simultaneous "shift"s of two gap-blocks (cyan- and yellow-shaded) give rise to at most 9 possible (residue & gap) patterns at each column-position (a representative is red-shaded). The red dotted rectangles enclose the original "ingredients" of the columns (at the representative position) that resulted from the "shift"s.

argument leads to the following conclusion:

$$\left\{ \begin{array}{c} \text{The number of possible patterns at each specific column position} \\ \textit{via} \text{ multiple-"shift"s of } B \text{ gap-blocks} \end{array} \right\} \leq 3^B \ . \qquad (7)$$

Thus, if the input MSA (denoted as $\mathcal{A}$) has $N_c(\mathcal{A})$ columns, and if you perform multiple-"shift"s of B gap-blocks on the MSA, the number of *resulting* column patterns should be *at most* $N_c(\mathcal{A}) \times 3^B$. This should usually be much less than the total number of all columns in the alternative MSAs, which is $N_c(\mathcal{A}) \times 20^B$ if each gap-block is "shift"ed by up to 10 sites to the left and to the right (from the "origin"). This suggests **the second trick**:

By *storing* the results of column-wise pruning algorithms into a hash-table (or something), and *reusing* the results when encountering the patterns whose probabilities have already been computed, we can save a tremendous amount of time!! [39]

With these two tricks, computing the substitution components *itself* will no longer be a rate-limiting step, even with quite a large number of gap-blocks in a window (e.g., $B = 7$ or 8), although storing (or outputting) the components of the resulting MSAs may pose some problems.

Someone might consider that *factorizing* the effects of the "shift"s of distinct gap-blocks will make the computation even more efficient. Actually, this is what we attempted at first; then, however, we noticed that the factorization is *not* possible when the "shift"s of the gap-blocks *affect* the same column(s), *even if* they are *non-interfering*. We have given the proof of this fact in appendix J, so that the readers can avoid wandering into the impasse we were once trapped in. [40]

[For the actual implementation of the computation of substitution components, see the

[39] This idea of *storing* and *reusing* the *already-computed* column-wise probabilities was first suggested by Dr. Giddy Landan, a former superviser of mine (see the Acknowledgments). The author truly appreciates his suggestion. Later, the author himself came up with the argument on the numbers of possible column-patterns (including Figures 8 & 9).

[40] It should be noted, however, what we have proven is the impossibility of *exact* factorization of the effects of gap-blocks. *Approximate* factorization may still be possible, especially in forms similar to a Markov chain,

subroutine, "shift_bl_and_compt_prob_incr," in the module, "MyANEX_Main.pm," of ANEX.
]

2.6.3 Indel components of log-probabilities of MSAs resulting from single-"shift"s

At least in principle, the effect of a single-"shift" on the indel component of the MSA probability could also be computed easily if we pay attention to the positional change of the subject gap-block. To do this, we *must* store the *site-level* horizontal positional relationships between interfering gap-blocks, as well as the association of such relationships with the sets of ancestral states, which are also specified *to the site-level*. In addition, each set of ancestral states should be associated with their own *current* contribution to the indel component, which should be updated everytime a relevant gap is "shift"ed. *Theoretically*, implementing this should *not* be so hard, because gap-blocks are generally related to a(n) (effective-)parsimonious indel history, and because other indel histories should also be obtained by cutting and joining the gap-blocks, at least as far as (effective-)parsimonious (and also next-to-parsiminious) indel histories are concerned. Once this is implemented, the downstream analyses should also be quite easy, because we can easily retrieve ancestral states and their probabilities.

In the current version of ANEX, however, we were *not* able to implement the aforementioned method to compute the effect of single-"shift"s on indel components, mainly due to the time-shortage. Instead, we have implemented a *much simpler* method, taking advantage of the ***good property*** possessed by the indel component of the (logarithmic) probability of the resulting MSA. [41] Here, the **good property** means the following fact: (provided that the

---

where the joint probability of two *non-overlapping* regions, say R_1 and R_2, mediated by a third region, say R_3, is given as: $P[R_1, R_2, R_3] = \{P[R_1, R_3] \cdot P[R_2, R_3]\}/P[R_3] = P[R_1, R_3] \cdot P[R_2 \mid R_3] = P[R_2, R_3] \cdot P[R_1 \mid R_3]$. This may be somewhat similar to a method to *phylogenetically "partition"* an MSA of many sequences [96]. Pursuing this direction, however, is beyond the scope of the current study.

[41] As explained in sub-subsection 2.6.1, the current version of ANEX computes the indel component of each MSA in the "1st-**plus** approximation", which sums contributions from *all* the indel histories (*including*

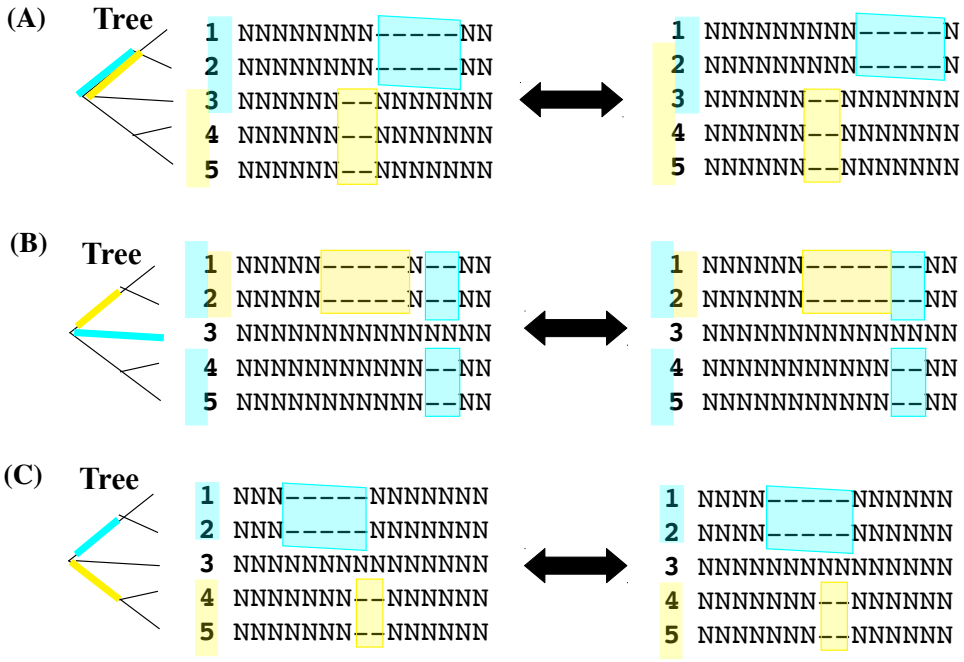


Figure 10: **Topological changes in positional relationships between gap-blocks caused by single-"shift"s. A.** between vertically complementary gap-blocks. **B.** between effective-parent & child gap-blocks. **C.** between effective-sibling gap-blocks.

indel rates are space-homogeneous (*i.e.*, uniform along each sequence) *at least* within each window,) the indel component remains (nearly) unchanged *if* the ***topology*** of the (horizontal) positional relationships does *not* change between the *interfering* gap-blocks *and if* the bulk/ boundary- status of all gap-blocks does *not* change. [42] (See appendix K for a proof (or demonstration) of this good property.) It takes much *less* time to monitor the change

non-parsimonious ones) with *parsimonious* sets of ancestral "presence"/"absence" states.

[42]The following are the cases of topological changes (concerning parsimonious sets of ancestral states) (Figure 10): (i) when a pair of vertically complementary gap-blocks *horizontally* join or get separated (panel A); (ii) when a pair of effective-parent&child gap-blocks *horizontally* join or get separated (panel B); (iii) when a pair of effective-sibling gap-blocks either start or cease to nest each other (panel C). ANEX also monitors similar changes in the relationships between a single gap-block and a composite gap-block, or between two composite gap-blocks. Although the topology also changes when a pair of vertically identical gap-blocks *horizontally* join or get separated, ANEX does not regard this case as a topological change; instead, it simply excludes the resulting MSA from the downstream analyses if vertically identical gap-blocks adjoin or horizontally nest each other.

in the topology of positional relationships and in the bulk/boundary status *than* to compute the full indel component of the MSA probability. Thus, the current version of ANEX employs the **strategy** of ***computing*** the indel components of the log-probabilities of the MSAs resulting from single-"shift"s ***only when*** it *detects* change(s) *in* ***either*** the topology of (horizontal) positional relationships between any pair of *interfering* gap-blocks ***or*** the bulk/boundary -status of any gap-block.

In addition, ANEX employs another time-saving measure, which is similar to the "2nd trick" for the fast computation of substitution components (sub-subsection 2.6.2). Specifically, when ANEX detects a change in the topological relationships or the bulk/boundary-status, it encodes the combination of them into a single code. [43] Then, it searches a hash table to see whether the indel component has already been computed for that code; if so, ANEX reuses the already computed indel component; otherwise, it computes the full indel component of the current MSA, and stores its association with the code of topology and bulk/boundary-status into the hash table.

[For the actual implementation of this strategy, see the subroutine, "nonrec_mlt_shifts_gapblocks," in the module, "MyANEX_Main.pm," of ANEX. For how the topological changes are monitored, see the subroutines, "chk_chng_gptrn_topology_psm_R" and "chk_chng_gptrn_topology_psm_L," in the module, "MyGapPatternTopology.pm," of ANEX. For how the topologies are encoded, see the subroutine, "encode_gptrn_topology_psm," in the module, "MyGapPatternTopology.pm," of ANEX.]

2.7 Outputting the results

Currently, ANEX outputs the results as a *structured* system of text files under a user-specified top output directory. Most of the files are just for diagnostic purposes, whose outputs could

[43] The code of each set of topological relationships is a collection of *all* binary horizontal relationships, each between a pair of gap-blocks (or a gap-block and a composite gap-block) that are vertically identical, vertically complementary, effective-siblings, or effective-parent&child, to each other .

be suppressed to some extent by specifying the global variable, "$TEST_LEVEL_MAIN," in the main script, "anex.ver0.7.pl." However, there are definitely *important* output files, as a matter of course. In this subsection, we will focus on these important files.

2.7.1 Overall structure of output

With the user-specified directory (say, "OUT/," for example) at the top, the output file system is structured as follows.

"**OUT/Interim_Data/**" stores some intermediate results. Under this directory, especially important ones are those under the sub-directories:

- "Init_psm_cands_parsimonized/," which stores information on the gapped segments and their representative (nearly) parsimonious indel histories;
- "Cmplx_error_cands/," which stores information on the regions with likely "complex'-errors;
- "Sliding_Windows_MaxN/," which stores information on the ordinary windows with {maximum number of gap-blocks} = N;
- "PCC_Sliding_Windows/," which stores information on the PCC windows.

"**OUT/Output_Data/Sliding_Window_Analysis/Window{window-ID}/**" stores the results of the analyses on the ordinary window with the ID, "{window-ID}," which are referred to as the "main window" in the following list of contents. Under this directory, especially important files are:

- "**Inputs/msa_in_wd.txt**," which records the input MSA within the window (in a CLUSTAL-like format [29], but without the header);
- "Inputs/set_columns0.txt," which records the set of columns in terms of the horizontal position (or the column index) each residue is from; in this particular file, the horizontal

position is given in terms of the *whole* input MSA; gaps remain gaps; note that each column is displayed *horizontally*, instead of normally as usual;

- "**synopsis_results_mshifts.txt**," which shows the overall results of the MSA neighborhood exploration, especially of the computed MSA probabilities, at a glance [44] ;

- "**Outputs/Initial_Status/**," which stores information on the "initial state," *i.e.*, the input MSA within the main window; under this subdirectory, the following files are important:

  - "**init_features_gblocks.txt**," which characterizes the gap-blocks at the "initial state"; the horizontal positions are also in terms of the input MSA *within the main window*;

  - "**init_set_columns.txt**," which records the set of columns in almost the same format as that for the "Inputs/set_columns0.txt" above, but the horizontal positions assigned to the cells are in terms of the input MSA *within the main window*;

- "**Outputs/I_1.Pure_MltShifts/Out/**," which stores the results of multiple-"shift"s with the input MSA (within the main window) as the "origin"; under this subdirectory, the following files are important:

  - "**coord_frame.txt**," which records the range of the coordinate of each gap-block (with 0 (zero) corresponding to the "origin");

  - "**degeneracy.txt.gz**," which records the degrees of degeneracies of the alternative MSAs resulting from the multiple-"shift"s, in a tabular form;

[44] In the file, each line gives information on a particular non-"shift"-type move and the subsequent multiple-"shift"s. Especially, it gives the log-probability of the MSA resulting from the non-"shift"-type move, and the logarithms of the total and the maximum of the probabilities of all relevant MSAs resulting from the multiple-"shift"s that follow.

- "**diff_ln_prb_{msa, indel, sbst}.txt.gz**," which records the differences when the {log-probability, indel component, substitution component} of the "origin" is subtracted from those of the resulting alternative MSAs, in a tabular form;
- "ln_prb_plus_msa_at_origin.txt," which records the quantities, such as the degree of degeneracy, log-probability, indel component, and substitution component, of the MSA at the "origin";

- "Outputs/Target_Candidates/," which stores the lists of the gap-blocks, or gap-block pairs, that are the candidates of the "targets" (or, more precisely, "subjects") of the non-"shift"-type moves;

- "**Outputs/{non-"shift"-move-identifier}/Out/Origin/**," which stores information on the new "origin," which is the MSA resulting from the move specified by the {non-"shift"-move-identifier} [45] ; under this subdirectory, especially important files are "**init_features_gblocks.txt**" and "**init_set_columns.txt**" [46] ;

- "**Outputs/{non-"shift"-move-identifier}/Out/MShifts/**," which stores the results of multiple-"shift"s starting at the new "origin"; the contents follow the same formats as those for the "Outputs/I_1.Pure_MltShifts/Out/" above [47] ;

- "**Sbwd{sub-window-ID}/**" stores the results of the analyses on the sub-window with the ID, "{sub-window-ID}"; this directory has quite a similar structure as that of its parent directory (*i.e.*, the "OUT/Output_Data/Sliding_Window_Analysis/Window{window-ID}/"); so, we will not detail its contents here.

[45] In general, a {non-"shift"-move-identifier} is a slash ("/")-mediated concatenation of: the type of the move (*e.g.*, "I_3a.SMerge_plus_MltShifts"), the target index (*e.g.*, "Pair_0"), and the size information (if applicable), etc.

[46] The horizontal positions in the former are in terms of the new "origin", but those assigned to the cells in the latter are in terms of the old "origin."

[47] The differences here are the results of subtracting the values of the new "origin." And the "ln_prb_plus_msa_at_origin.txt" records the quantities of the new "origin."

"**OUT/Output_Data/PCC_Sliding_Window_Analysis/PCC_Window{PCC-window-ID}/**" stores the results of the analyses on the PCC window with the ID, "{PCC-window-ID}." This directory has quite a similar structure as that of the "OUT/Output_Data/Sliding_Window_Analysis/Window{window-ID}/". So, we will not repeat the explanation of most of its contents. But one particular file ("list_rlv_pcands.tx") takes on a particular importance here. Besides, it should be noted that some files and directories are named differently:

- "**Inputs/list_rlv_pcands.tx**" records information on the "purge-like-error-candidate," which plays a major role here;
- "**Inputs/msa_in_pcc_wd.txt**" records the input MSA within the PCC window;
- "**Outputs/III_0.Initial_MSA/In/**" stores information on the "initial state," *i.e.*, the input MSA within the PCC window; especially, it contains "**init_features_gblocks.txt**" and "**init_set_columns.txt**";
- "**Outputs/III_0.Initial_MSA/Out/ln_prb_plus_msa.txt**" records the quantities (including the log-probability) of the MSA at the "initial state."

2.7.2 About alternative MSAs resulting from multiple-"shift"s

To avoid wasting the storage space (and memory), ANEX does *not* output, or store for a long time, the alternative MSAs that result from the multiple-"shift"s, either the default one or following a particular non-"shift"-type move.

Instead, we provide a supplementary Perl script, "coordinate_point_lcl_msa.ver0.7.pl," to help understand the alternative MSA at a particular coordinate point (in the multiple-"shift" space).

The script ("coordinate_point_lcl_msa.ver0.7.pl") takes the following inputs:

1. (The path to) a top working directory, which the user must specify;

2. (The path to) a Dawg control file, which should be the same as that fed into the ANEX main script; (This script uses only the tree);

3. (The path to) an input MSA within a main (ordinary) window (or a PCC window);

4. (The path to) the "**init_set_columns.txt**" made from an MSA used as the "origin" of multiple-"shift"s;

5. (The path to) the "**init_features_gblocks.txt**" made from the same MSA at the "origin";

6. Coordinates the user wants, in the format: "$X_0, X_1, X_2, ..., X_{B-1}$" ($B$ is the number of gap-blocks in the relevant window).

Then, it outputs the main outputs to the files in the subdirectory, "Outputs/," which is under the user-specified top working directory. The output files are:

- "**lcl_msa_dstn.txt**," which records the "destination" MSA, which resulted from the multiple-"shift"s specified by the input coordinates;

- "**features_gblocks_dstn.txt**," which characterizes the gap-blocks in the "destination" MSA;

- "**set_columns_dstn.txt**," which records the set of columns in the "destination" MSA; the horizontal positions assigned to the cells are in terms of the input MSA fed into this script.

This script should be useful particularly when performing manual analyses, examining *e.g.*, MSAs with the maximum (or near-maximum) probabilities, etc. It should also be helpful when the user needs to examine, or debug, the behavior of ANEX, or when the user wants to learn about particular ("shift" or non-"shift") moves of the gap-blocks.

2.7.3 About tables of differences in log-probabilities, indel components, substitution components

Among the main outputs of ANEX, the tables, "**degeneracy.txt.gz**" and "**diff_ln_prb_{msa, indel, sbst}.txt.gz**," are probably the most important. [48]

In each of these tables, each row is specified by the coordinates, X_0, X_1, ..., X_{B-2} (B is the number of gap-blocks in the window), and each column is specified by the coordinate, X_{B-1}. When a coordinate point was not analyzed for some reason, the corresponding cell should be occupied by an indicator, such as 'undef' or 'N/A.'

It should not be so hard to write programs or scripts to examine or process these tables, as the user desires. Nevertheless, it may sometimes be useful to "restore" the data structure that actually stored the values of these quantities while ANEX were performing the multiple-"shift"s. For such a purpose, we wrote two subroutines, "readin_coord_frame" and "readin_sgl_qnt_on_multidim_space," both in the module, "MyANEX_IO2.pm." This subroutine, as well as the subroutines, "printout_multidim_storage" (for outputting the values of a quantity stored in the data structure into the same table as the original) in the module, "MyANEX_Main.pm," "find_maxvalue_in_multidim_storage" (for finding the maximum value) in the module, "MyANEX_IO2.pm," and "cmpt_log_total_qnt_in_multidim_storage" (for computing the logarithm of the total of exponentiated values) also in the module "MyANEX_IO2.pm," may help you write some programs (or scripts) that perform a wide variety of analyses or processing of these tables. We also included a Perl script, "test_cmpt_log_total_qnt_in_multidim_storage.pl" in ANEX; it will show you how the aforementioned subroutines are actually used in a script.

As a matter of fact, we were planning to write **a script (or a subroutine) to retrieve**

[48] To remind the readers, the "**degeneracy.txt.gz**" records the degrees of degeneracies of the alternative MSAs resulting from the multiple-"shift"s, and the "**diff_ln_prb_{msa, indel, sbst}.txt.gz**" records the differences when the {log-probability, indel component, substitution component} of the "origin" is subtracted from those of the resulting alternative MSAs.

all the coordinate points in which the log-probabilities are greater than (or equal to) a specified value. Such a program will enable us to draw graphs similar to the "energy vs. entropy plot" or the "energy vs. free-energy plot" in statistical physics (or thermodynamics). [49] Such graphs (especially the latter), in turn, will give us a perspective on the "probability landscape" on the space of multiple-"shift"s (or even on the space of all attempted moves). ***Importantly*, such graphs will indicate whether, in each window, it should be enough to only consider a single most probable MSA, or we need to take account of a small number of highly probable alternative MSAs, quite a large number yet a very small percentage of alternative MSAs (with the probabilities larger than a given value), or *all* alternative MSAs.**

Such a script (or subroutine) should be created by combining the features of the aforementioned subroutines, "find_maxvalue_in_multidim_storage," "cmpt_log_total_qnt_in_multidim_storage" and "printout_multidim_storage." Unfortunately, however, ***we ran out of time, and couldn't make it.*** We hope that, in the (hopefully near) future, someone will write such a program, script, subroutine, library, or whatever, to help improve our understanding of MSA errors from the viewpoint of the number of alternative MSAs we have to take into account.

2.8 Possible applications of results/outputs

The results or outputs obtained will be applied to a wide variety of analyses on molecular evolution, such as the predictions of functions, positive selection, 2ndary and tertiary structures, etc. The output of the program is the *approximate* probability distribution of possible

[49] Briefly, in statistical physics, the energy of a state is (a negative number times) the logarithm of its (un-normalized) probability; the entropy is the logarithm of the number of states; the free-energy is the logarithm of the total probability (or, actually, the normalization factor). Therefore, in an "energy vs. entropy plot" analog, the X-coordinate represents the (log-)probability and the Y-coordinate represents the number of alternative MSAs with the probability; in an "energy vs free-energy plot" analog, the X-coordinate is as above, and the Y-coordinate represents the sub-total of the probabilities of alternative MSAs whose (log-)probabilities are larger than or equal to X.

MSAs *computed under* a *genuine* sequence evolution model with *realistic* indels. Therefore, we can estimate the **expected value (average weighted with probability), variance, confidence intervals, distribution itself, etc.**, for each property/event (or combination of properties/events) we want to analyze. Of course, we expect that the expected value thus obtained will be *more* accurate than the "point estimation" of the property obtained from a single optimum MSA (output by a commonly used aligner). (Of course, this expectation needs be confirmed by extensive analyses.) In our view, however, **ANEX's *most important* feature** is that its output will indicate ***possible* fluctuations** on the results, and thus will ***honestly*** tell **how accurately** we can learn about the subject of our interest ***from*** the **input data** at hand, which will greatly help us ***avoid* overconfidence** on our predictions. As a "by-product," ANEX also indicates regions that likely contain **"complex" errors**. This function itself may also be useful, because such regions are expected to be the hotbeds of **erroneous predictions**, especially on some peculiar evolutionary behaviors of sequences, including excessive positive selection, excessively frequent unconventional events, etc. [50]

2.9 Conceivable problems

Similarly to many brand-new methods, ANEX is *not* free from problems. In appendix L, we will discuss three major problems, namely, those of (1) how to provide the program with accurate model parameters (including the tree), (2) autocorrelation, and (3) overfitting, as well as possible solutions to them.

[50] On the other hand, the complex errors themselves may be the consequences, and thus the indicators, of such peculiar behaviors that indeed occurred in the evolutionary history. In any case, it should be dangerous to *blindly* believing in the results of *naïvely* applying a(n) (especially general-use) program to such *dubious* regions; careful re-analyses should be in order.

3 Implementing and "Validating" ANEX

3.1 Implementation

The method proposed here was implemented into a package of prototype Perl scripts and Perl packages, named "ANEX(_P)," which abbreviates "**A**lignment **N**eighborhood **EX**plorer(, **P**erl-version)." Currently, it is version 0.7. In short, the current version of the package can do the following:

1. Given an input MSA, ANEX creates a set of (generally overlapping) ordinary windows, as well as a set of (also generally overlapping) "purge-like-error-candidate containing" (PCC) windows; (each ordinary window may contain one or two sub-windows); in each ordinary window, ANEX "explores" the neighborhoods of the input MSA *via* pure multiple-"shifts," as well as *via* non-"shift"-type moves (except "reverse-purge"s) followed by multiple-"shift"s, of gap-blocks; in each PCC window, ANEX "explores" the neighborhoods of the input MSA *via* "reverse-purge"s followed by multiple-"shift"s of gap-blocks; (see subsections 2.4 & 2.5 for details);

2. While "exploring" the neighborhoods of the input MSA within each window, ANEX computes the probabilities of the alternative MSAs it "visited" under a (locally) space-homogeneous *genuine* stochastic sequence evolution model, and outputs the computed probabilities in a tabular form per each bunch of multiple-"shift"s; ANEX also outputs a file, "synopsis_results_mshifts.txt," which briefly summarizes the results of all neighborhood explorations within each window; (see subsections 2.6 & 2.7 for details);

3. By default, its main master script, "anex.ver0.7.pl," takes the following inputs:

   (a) (The path to) an input MSA file;

   (b) (The path to) a control file of Dawg, the *genuine* sequence evolution simulator [77]; the control file is used to specify a phylogenetic tree, a substitution model

(including the rate matrix and the base frequencies) [51] , and the total rates of insertions and deletions;

(c) (The path to) the directory storing the tables of multiplication factors for the indel components of ancestor-descendant PWAs with a series of time-lapses; ANEX can make use of the multiplication factors for case-(i), (ii), (iii), and (iv) gapped segments, with case-(i) optional and the others mandatory; especially, the multiplication factors computed *via* the recently developed **LASTPIECE** [4] can be used; if the branch lengths don't match any of the time-lapses in the table, ANEX *linearly* interpolates the multiplication factors;

(d) (The path to) the top output directory, which the user must specify;

(e) The path to the slave script, "anex_for_sgl_wd.ver0.7.pl," which explores the MSA neighborhood within an ordinary window;

(f) The path to the slave script, "detect_purge_cands.ver0.5.pl," which identifies "purge-like-error-candidates" in the input MSA;

4. The "anex.ver0.7.pl" can also take some options; especially important ones are:

(a) "–PATH_REV_PURGE_SCT={string}," which specifies the path to the slave script, "anex_rev_purge_for_sgl_wd.ver0.7.pl," which explores the MSA neighborhood (via "reverse-purge"s followed by multiple-"shift"s) within a PCC window [52] ;

(b) "–NUM_MAX={integer(=N_W)}," which specifies the maximum number of gap-blocks that each ordinary window can contain;

(c) "–HF_WDTH_SHIFT={integer(=W_M)}," which specifies the maximum number of sites by which each gap-block can "shift" in each direction;

[51] As already mentioned in sub-subsection 2.6.1, the current version of ANEX can take up to Tamura and Nei's model [88] but cannot take the general time-reversible model [92, 93, 94, 95].

[52] By default, the current version of ANEX does *not* perform any analyses within PCC windows.

5. the package also contains some supplementary scripts that enable simple analyses on the input MSA and/or some outputs of ANEX; among them, the scripts, "coordinate_point_lcl_msa.ver0.7.pl" and "test_cmpt_log_total_qnt_in_multidim_storage.pl," have been discussed briefly in sub-subsections 2.7.2 & 2.7.3 above; some other scripts are discussed elsewhere ([82]).

This package (ANEX(_P)) is available as an open-source package at an FTP repository of Bioinformatics.org (https://www.bioinformatics.org/ftp/pub/anex/). (Currently, the package runs on the Terminal of Mac OS X; it should run also on some other UNIX platforms, including Linux, although we have not yet confirmed that it does.)

In the current version (ver. 0.7), the main master script ("anex.ver0.7.pl") performs all the computational steps *serially*; it thus uses only a single CPU (or core) and is considerably slow. However, because each window analysis is independent of the others, these window analyses can be easily cast into parallel or distributed computing. Moreover, within each window, the MSA neighborhood exploration contains multiple bunches of multiple-"shift"s; once the inputs (including the MSA at its starting point (or "origin")) are provided, each bunch of multiple-"shift"s is also independent of the others; thus, these bunches of multiple-"shift"s could also be cast into parallel or distributed computing (, albeit requiring some efforts).

We expect that, if the main scripts are also translated into C, for example, and if only the window analyses are cast into parallel or distributed computing, the computation could be much more than 100 times faster.

3.2 "Validation" using simulated MSAs

In order to see whether the entire method proposed here, *i.e.*, ANEX, really works or not, we applied ANEX to several artificial MSAs created by a *genuine* sequence evolution simulator, Dawg [77]. The simulated MSAs are actually among those used in a previous study of

ours [3]. In that study, MSAs were simulated along three different phylogenetic trees: (1) the tree of 12 primates, (2) the tree of 15 mammals, and (3) the tree of 9 fast-evolving mammals. Among those simulated MSAs, those simulated the evolution of sequences from 9 fast-evolving mammals turned out to cause severe errors in the MSAs reconstructed using the state-of-the art aligners of the single-optimum-search type [53] ; those errors seemed too complex to rectify in a straightforward manner. Therefore, in the present study, we only used several MSAs simulated along the trees of 12 primates and 15 mammals. [54]

Regarding details on the parameters used for the simulations, refer to [3]. Briefly, Jukes and Cantor's model (JC69) [89] was used for nucleotide substitutions; the total insertion and deletion rates were both set at $1/16 = 0.0625$ (indels/substitution); the frequency of both insertions and deletions with length l was made proportional to $l^{-1.6}$ (Zipf power-law); insertions and deletions exceeding 100 bases were cut off; and each simulation started at the root of the tree, with a random ancestral sequence that is 1000 bases long.

When applying ANEX to each of these simulated MSAs, we inputted the same Dawg control file as used for simulations, as well as the indel multiplication factors computed *via* LASTPIECE [4] using the same Zipf power-law indel length distributions and the same indel cut-off lengths as the simulations. Moreover, the maximum number of gap-blocks in each ordinary window was set to be $N_W = 4$, and the maximum number of sites by which each gap-block can "shift" in each direction was set to be $W_M = 20$.

ANEX was run on our Mac Pro (Late 2013) desktop computer (with OS version 10.11.6, with one 3.5 GHz 6-core Intel Xeon E5 Processor and 16 GB physical memory). Although ANEX's window analyses can easily be cast into parallel or distributed computing, current

[53] In the study (*i.e.*, [3]), we used MAFFT [31, 32, 33] and Prank [36, 37], as representatives of "similarity-based" and "evolution-based" aligners, respectively, as classified by [97].

[54] We chose either those MSAs that seem to contain more non-"shift"-type errors than average, or those that revealed some problems in the development stages of our previous program packages "LOLIPOG" [2] and "ComplLiMent" [3]. (Incidentally, the revealed problems of the packages were rectified (*i.e.*, debugged) after that.)

version performs the window analyses *serially*, using only one CPU (or core).

Tables 1 & 2 summarize the results. [55] Each of these tables tells us about at least two important aspects regarding **the "accuracy" and "truthfulness" of ANEX**: (1) the accuracy of the "prediction" of "complex" errors, and (2) its ability to recover the true MSA when *at least* a window encompasses an erroneous segment [56] *completely*.

First, we discuss "complex" errors. For this purpose, it should be appropriate to focus on reconstructed MSAs of 15 simulated mammalian sequences (Table 1), because reconstructed MSAs of 12 simulated primate sequences (Table 2) contain very few "complex" errors(, and very few "false-positive" "complex" errors). As table 1 indicates, "complex" errors were *quite* successfully detected and excluded from the main analyses (of neighborhood exploration and probability computation); although a small fraction (about 20% on average) of "complex" errors were (in a sense) "erroneously" overlooked and thus included in the main analyses, the true MSAs were *somehow* recovered by the downstream neighborhood exploration in about 1/2 of such cases. [57] An apparent downside is that a considerable fraction of non-"complex" errors (about 36% on average) were also excluded from the main analyses. Actually, this reflects our stance regarding the detection of "complex" errors; that is, we *deliberately* erred on the cautious side, and aimed for a high sensitivity (*i.e.*, a low

[55] The detailed results (as well as inputs) are available as a tar-gzipped archive accompanying the package of ANEX(_P).

[56] In [3], each reconstructed MSA (or, more precisely, each pair of reconstructed and true MSAs) was chopped into "correct segments", where the reconstructed MSA and the true MSA are equal (or *at least* equivalent) to each other, and "erroneous segments," where the reconstructed MSA differs from the true MSA.

[57] This seemingly strange behavior of ANEX is actually due to the *incomplete* nature of the classification of "complex" errors; although we used the classifications in our previous work [3] as they are, the classification method itself was quite rudimentary, depending on frequently incorrect demarcation of "position-shift-blocks." Thus, once a more improved classification method is developed, a considerable fraction of these "erroneously" overlooked "complex" errors are likely to be classified as non-"complex." We will revisit this issue elsewhere [82].

Table 1: Results of applying ANEX to MSAs created by simulated evolution of sequences from 15 mammals

MSA ID		A	B	C	D
Reconstruction method		Prank (Best-fit)	Prank (Best-fit)	MAFFT (E-INS-i)	MAFFT (E-INS-1)
All erroneous segments		42	24	25	24
Complex Errors	In NO windows *	12	5	11	11
	In some window(s) †	2	2	2	4
	Recovered ‡	0	1	1	3
Non-complex Errors ♭	In NO windows *	10	6	2	6
	In some window(s) †	18	11	10	3
	Recovered ‡	16	10	8	3
Running time (min.)		131.9	216.8	219.4	47.9

(The numbers in each cell (except in the bottom row) is the frequency of erroneous segments in each class (row) in each MSA (column).)

* The erroneous segment was *not completely* included in any windows.

† The erroneous segment was *completely* included in at least one window.

‡ The true MSA was recovered by at least one of the (composite) moves attempted.

(NOTE: this category is included in the "In some window(s)" above.)

♭ The erroneous segment contained no complex errors.

In other words, the error was explained by a(n) (combination of) elementary move(s).

Table 2: Results of applying ANEX to MSAs created by simulated evolution of sequences from 12 primates

MSA ID		E	F
Reconstruction method		Prank (Best-fit)	MAFFT (E-INS-i)
All erroneous segments		17	16
Complex Errors	In NO windows *	0	1
	In some window(s) †	0	0
	Recovered ‡	0	0
Non-complex Errors ♭	In NO windows *	1	2
	In some window(s) †	16	13
	Recovered ‡	15	13
Running time (min.)		650.6	263.5

*, †, ‡, ♭ the same notes as in Table 1 apply.

false-negative rate) of "complex" error detection, while sacrificing the high specificity (*i.e.*, low false-positive rate). In any case, it should be remembered that ANEX currently uses a very simple method to detect "complex" errors, taking advantage only on the **gap configurations** of the input (reconstructed) MSA. Taking this simpleness of the detection method into consideration, the "prediction" of "complex" errors could be regarded as "quite good," and we can definitely expect that the "prediction" of "complex" errors will become much more accurate if the **residue configurations**, or the distribution of inferred substitutions, of the MSA are also incorporated into the "prediction" method. (See also [82].)

Another thing we should note is that a substantial fraction (and eventually (almost) all) of "complex" errors can actually be unraveled into a series (or combination) of a number of elementary moves, as a couple of examples were shown in [3]. We can easily expect that, as the number of "constituent" elementary moves increases, a larger fraction of "complex" errors can be unraveled. This means that the true MSAs can be recovered from a higher fraction of MSA errors if ANEX gets to handle the combinations of a larger number of elementary moves. (We will discuss this issue again, in section 4.)

Next, we discuss the recovery of the true MSA. As both Tables 1 & 2 indicate, out of the non-"complex" errors that were *completely* included in *at least* a window, a very high fraction of them (on average, 88% for 15 mammals and 97% for 12 primates) had their true MSA counterparts recovered by the *current* neighborhood exploration method. Considering that the current architecture of the neighborhood exploration (Figure 4) is quite simple (and could *even* be considered as "naïve"), this high recovery rate is quite amazing, and gives us a hope that a more sophisticated, and *smart*, exploration architecture should recover the true MSAs from much more erroneous segments (including those currently classified as "complex").

Regarding the running time, it took about 50 minutes (for 15 mammals) to about 650 minutes (for 12 primates) to perform ANEX. The running time seemed to be longer for 12 primates than for 15 mammals, despite the more gaps, and more erroneous segments, for the

latter. Two factors may contribute to this tendency. One is the higher rate of complex errors for 15 mammals, leaving only a smaller fraction of MSA regions to be analyzed. And the other is the higher average density of gap-blocks for 15 mammals, making a larger percentage of gap-blocks "shift"able by significantly less than W_M (= 20) sites in each direction. Although the 650 minutes may sound quite a long time, it should be remembered that current ANEX performs window analyses *serially*, and that it is currently written in Perl. Once parallel (or distributed) computing is introduced, and once it is written, *e.g.*, in C, it should be more than 100 times faster. Then, the 650 minutes should be reduced to 6 minutes or less, which should not be so stressful.

To summarize, despite the fact that the current version of ANEX employs a quite simple method to detect "complex" errors and a fairly simple architecture to explore the MSA neighborhood, ANEX detected quite a high percentage (80% on average) of "complex" errors and excluded them from the main analyses, and it also recovered the true MSAs from quite a high percentage (92% on average) of non-"complex" errors each of which was included in at least a window. These results suggest that ANEX should be fairly useful already in the current form, and should become further useful if the "complex" error detection method and the neighborhood exploration architecture get more sophisticated and refined.

4 Discussions

The reconstruction of multiple sequence alignments (MSAs) is central to the advanced studies of homologous (*i.e.*, ancestor-sharing) biological sequences (*e.g.*, [5, 6, 7, 8, 9, 10]). At the same time, it also turned out to be highly error-prone [43, 37, 44, 45, 3]. Furthermore, it was revealed that, in a (near) majority of these errors in MSAs, the true MSAs does *not* optimize *even* the "golden score" that perfectly predicts the (log-)probability of the MSA [3]. Thus, to aim for more accurate and *truthful* sequence analyses, it is essential to construct a **probability distribution of alternative MSAs**, instead of reconstructing a

single optimum MSA.

Although the idea of providing the probability distribution of alignments, often referred to as "*statistical alignment*" [53], has been around for decades [54, 55, 56, 53, 57, 75], the past studies and methods on statistical MSAs (*e.g.*, [58, 59, 60, 61, 62, 63, 50, 64]) did *not* use *genuine* sequence evolution models with *realistic* indels, such as those following power-law length distributions (*e.g.*, [68, 69, 70, 71, 72, 73, 74]) *and* allowing the overlap and/or nesting with other indels (*e.g.*, [76, 77, 78, 79]).

To the best of our knowledge, this study represents the first attempt to apply *genuine* sequence evolution models (with *realistic* indels) to the problem of statistical MSA. One of the major obstacles to applying *genuine* sequence evolution models is the fact that the probability computation cannot be completely factorized into site-wise contributions, which makes the computational burden quite large, and hampers the (near) exhaustive search of the space of alternative MSAs. We circumvented this obstacle by taking advantage of at least two features. One feature is our previous result [3] that, as long as the phylogenetic relationships are not so distant, reconstructed MSAs are in the neighborhood of the true MSAs in an overwhelming majority of erroneous segments. Thus, we chose to explore *only* the neighborhoods of the reconstructed MSA. This choice enabled us to devise the smart strategy to efficiently compute the *increment of* the substitution components of the MSA probabilities (see sub-subsection 2.6.2 for details), as well as the strategy to compute the indel components *only when* the *topology* of the gap-block configuration changes. A fringe benefit of this strategy of exploring neighborhoods is that the results are intuitively understandable, thus potentially making the rather esoteric subject of statistical alignment accessible to a *wider* audience. And the other feature is the reuse of the multiplication factors in ancestor-descendant PWAs that were quite accurately computed beforehand *via* another new program package of ours, LASTPIECE [98]; this also should have saved an enormous amount of computational time. [58]

[58] This strategy of pre-computing and re-using the multiplication factors is very similar to the strategy

We developed a new program package, the "alignment neighborhood explorer" (ANEX), to implement our method, and validated it by applying it to a number of reconstructed MSAs of simulated sequences (that had been prepared before [3]). The result was a moderate success. As intended, ANEX detected about 80% of "complex" errors and excluded them from the downstream analyses. And, in about 88-97% of non-"complex" errors that were completely included in at least a window, the true MSAs were in the neighborhoods explored by ANEX. In fact, this was rather surprising, considering the fact that ANEX currently employs a quite simple architecture of MSA neighborhood exploration (Figure 4).

This gives us a hope that the true MSAs will be recovered from a much higher percentage of MSA errors if we employ a more sophisticated and refined architecture of neighborhood exploration. Such a sophisticated MSA neighborhood exploration should involve three or more non-"shift"-type moves (and the subsequent multiple-"shift"s). As illustrated by a couple of examples in [3], "complex" errors can actually be unraveled into a series (or combination) of a number of elementary moves. [59] Therefore, by increasing the number of non-"shift"-type moves to be combined before the multiple-"shift"s, we should be able to increase the percentage of "complex" errors in each of which the explored neighbor includes the true MSA.

However, we can*not* increase the number of non-"shift"-type moves *unlimitedly*, because at least two problems should arise. One is the limitation in the number of gap-blocks that each window can contain. If a window contains B gap-blocks, and if each gap-block are shifted by W_M sites in each direction in the multiple-"shift"s, the memory space required to store the results of multiple-"shift"s is proportional to $(2W_M)^B$. If we use $W_M = 20$ as in the

employed by a recent "simulation-based" approach to statistical PWA [80]. We are sure, however, that we have devised the strategy by ourselves. See footnote 12 for more details.

[59] After all, the category of "complex" errors means, "the current classification system cannot *automatically* unravel these errors into a series (or combination) of elementary moves." In other words, the errors classified as "complex" in [3] represent the limitation of the error classification program used in that study; as the program improves, the percentage of "complex" errors will decrease.

validation in this paper, the space requirement increases 40-fold if a gap-block is added to the window. Therefore, if we *naïvely* grow the hyper-rectangular coordinate space this way, each window can only contain at most 5 or 6 gap-blocks, even if you can use a quite large memory (of 10 GB or more). The key to overcome this limitation would be a "*smart*" exploration: if we *somehow* get to know "coarse-grained" moves that take us to the regions of the coordinate space where highly probable alternative MSAs are densely distributed, the necessary W_M may be reduced to, say, 5. [60] Then, we may be able to analyze a window containing up to, say, 9 gap-blocks, substantially widening the percentages of "complex" errors that can be analyzed.

The other limitation comes from the enormous number of possible combinations of non-"shift"-type moves, which leads to the problem of "combinatorial explosion." Again, the key to overcome this limitation would also be a "*smart*" exploration. It's true that, if we naïvely attempt every possible combination of non-"shift"-type moves, the number of combinations will explode exponentially. In many cases, however, we can predict, *to some extent*, the types of (combinations) of likely non-"shift"-type moves, by examining the residue configuration, especially the distribution of expected substitutions inferred from it, as well as the gap-block configuration. In fact, the current version of ANEX does this in (probably) the most straightforward manner: it restricts the types of attempted non-"shift"-type moves to only those that are possible from the gap-blocks in the initial window; especially, it allows a "split" of a gap-block *only when* there is a nearby "purge-like error candidate" to support the "split." Besides, the detection of "complex-error-candidates" by using gap configurations may also be regarded as a sort of "straightforward measure." And the "validation" in this study suggested that these straightforward measures are working fairly well, although there

[60] One potentially useful measure to narrow down the coordinate space would be to examine the residue configuration and the gap configuration of the input MSA, as briefly discussed in the next few paragraphs. Another potentially useful measure would be to set or reset the boundary of "shift"s in each direction by monitoring the change in the MSA probability.

were a few cases requiring some improvements.

We believe that, in order to turn these straightforward measures into more sophisticated and "smart" measures to chose the MSA neighborhoods to explore, we need to examine the *combination of* the residue configuration and the gap-block configuration. This should be possible either by conducting studies more meticulous than [3] (and [82]), or by resorting to *machine learning* techniques (*e.g.*, [99, 100, 101]), including the *deep learning via* artificial intelligence (*e.g.*, [102, 103]). Once these "smart" exploration is realized and implemented, the resulting **"smart" version of ANEX** will be more powerful and useful. enabling us to construct very accurate (yet reduced) probability distributions of alternative MSAs that cover most of MSA errors (including those currently classified as "complex").

Although we've discussed rather positive aspects of ANEX thus far, we are also aware that the current version of ANEX also has some downsides and limitations. (Some of them were already discussed in subsection 2.9 and in appendix L.) First, the current version of ANEX excludes a small yet considerable fraction (about 36% on average) of non-"complex" errors from the downstream analyses. This is partly because we deliberately erred on the cautious side of excluding as many "complex" errors as possible, and partly because the current method to detect "complex" error candidates is too simple, using gap-configurations *alone*. We strongly believe that this problem should also be solved, or at least substantially mitigated, by employing a "*smarter*" method to detect "complex" errors that takes advantage of residue configurations as well.

Second, we are aware that, especially when the phylogenetic relationships are moderately remote, the true MSAs are far away from the reconstructed ones in a small yet non-negligible fraction (about 20%) of errors. Definitely, ANEX should be inept at handling such true MSAs that are *not* in the neighborhood (of reconstructed ones). Still, we hope that a considerable percentage of such cases may be successfully handled once the aforementioned "*smart*" exploration method is introduced. Regarding those errors that are intractable *even* to such "smart" exploration, one way to handle them may be to reconstruct the MSA *from*

scratch, by using a totally different method than that reconstructed the input MSA. [61]

Third, ANEX currently takes a phylogenetic tree and some parameters of the evolution model as integral inputs. In general, these inputs should be unknown at first, and they also must be estimated from the sequence data. Because current package of ANEX does not contain programs to estimate these parameters, users must depend on programs outside. In particular, the indel rates and the indel length distributions can be estimated *via*, *e.g.*, the programs provided in [77, 81, 104]. If time permits, it may be better to use several point estimations, rather than relying solely on a single point estimation of the parameters. From this viewpoint, a recent program, SpartaABC [104], provides an approximate distribution of indel model parameters, and thus may be useful.

Although it may be desirable to estimate the joint distribution of the sets of a tree, evolution parameters, and an MSA (as in [58, 105, 60, 61] , all of which used HMMs), this remains a formidable goal if you use a *genuine* sequence evolution model.

Fourth, regarding the limitation on the number of aligned sequences, ANEX's space- and time-complexities depend *at most* linearly on the number of sequences. Therefore, we guess that ANEX should be able to handle at least 100 sequences, although we have not tested yet. But there is an important *caveat*. As the total tree branch length increases, ANEX's performance (especially accuracy) deteriorates rapidly (probably more than exponentially), *as the accuracies of almost all state-of-the-art aligners do* [3]. Thus, you should take care so that the total tree branch length does *not* exceed an upper limit, which probably is less than 2 (expected substitutions/site); if possible, it should be preferable to keep it less than 1.5. Regarding the number of aligned sequences, there has been a wide-spread myth out there that an MSA gets more and more accurate as the number of aligned sequences increases(, probably because the average branch length gets smaller). However, some studies (*e.g.*, [37]) reached the opposite conclusion, demonstrating that the myth is indeed a myth. This may

[61] Or, an even better way may be to use a more accurate aligner (maybe based on some statistical alignment methods) to produce a less error-prone input MSA.

be partly because of the aforementioned severe dependence on the total tree branch length. In any case, the lessons from studies like these are: you should *not* increase the number of aligned sequences *blindly*; instead, you should *carefully* select the sequences to be aligned, taking full account of the purpose of your study; besides, *avoid* including sequences that cause long branches, whenever possible. [62]

Fifth, by design, ANEX re-uses the multiplication factors of gapped segments that have been pre-computed by LASTPIECE(_P), which computes and tabulates the factors (at various time-lapses) under a given *genuine* sequence evolution model [4]. Although the current version of LASTPIECE(_P) implements only power-law and geometric distributions, its underlying theory (fully described in [4]) can accommodate *any* total rates and *any* length distributions of insertions and deletions, *as long as* the rates and distributions are uniform *at least* within each gapped segment(, and *as long as* the entire sequence evolution model satisfies the factorability conditions [1]). Thus, only with minor modifications, LASTPIECE(_P) could get much more versatile, and it may get to incorporate some "genome-specific" or "region-specific" behaviors of indels, for example, insertions of transposable elements *e.g.*, [106, 107]) and the evolution of tandem repeat arrays (*e.g.*, [108, 109, 110]), although we are not completely sure about incorporating the latter yet. (See [4] for more specific discussions.)

Finally, the current version of ANEX does not handle mutations other than substitutions/insertions/deletions, such as genome rearrangements (including inversions, duplications and translocations) (*e.g.*, [12, 13, 14, 15]), homologous recombinations (*e.g.*, [111]), gene conversions (*e.g.*, [112, 113, 114]), and homogenization between the arms of inverted repeats (*e.g.*, [115, 116, 117]). At least formally, these types of mutations can also be handled by *an extended version* of our theoretical formulation of *genuine* sequence evolution models [118]. Therefore, it may be possible to extend ANEX to incorporate these types of mutations

[62] Because ANEX can take *almost full* account of the effects of multiple indels along each branch [4], the final lesson may sound unnecessary. However, ANEX depends essentially on the inputs from other aligners, which generally do *not* take account of such effects. The users should take this into consideration.

as well. Especially, genome rearrangements may be handled in a parsimony-based manner, as insertions/deletions are handled in [2], because they also are expected to be rare in general. If this is indeed the case, we may utilize some theoretical results on genome rearrangements (*e.g.* [119, 120, 121, 122, 123, 124, 125, 126]). For the moment, however, before performing ANEX on an input MSA, it may be better to confirm that the aligned sequences did not undergo these types of mutations, for example by performing local alignments among them and on themselves.

In summary, even though we admit that there are some downsizes and limitations, our brief validation concluded that the current version of ANEX is a moderately successful and very promising approach. Thus, it should be worthwhile to develop a "smarter" version of ANEX, by making the architecture of neighborhood exploration more sophisticated and refined, and also by introducing a "smarter" method to identify candidates of "complex" errors more accurately, both possibly with the aid of the machine learning (*e.g.*, [99, 100, 101]), including the deep learning *via* artificial intelligence (*e.g.*, [102, 103]).

Last but not least, there may have been some people who, either openly or (more likely) behind our backs, expressed doubts about the practical usefulness of our theoretical formulation of *genuine* sequence evolution models [1, 2]. The successes of this study and another recent study of ours [4], both of which are essentially based on that very theoretical formulation, have completely dispelled such doubts. (After all, those who do not *properly* understand a theory should *not* be able to judge or predict, with any certainty, whether the theory is useful or not.) From now on, you can rest assured and feel free to use our theoretical formulation [1, 2], augmented by the results of [4] and appendix K of this paper, to calculate the indel components of MSA probabilities under *genuine* sequence evolution models. We sincerely and earnestly hope that these studies of ours will pave the way for a new era of the homology sequence study, where more *accurate* and *truthful* sequence analyses are commonplace.

4.1 Final Note

Some of our comments in this paper or other papers may sound like harsh criticisms on other researchers or their works. We strongly urge the readers to understand that such comments are our candid expressions of our sincere and pure hope for the advance of the science *in the right direction*, and that we have no intension to attack, harm, or hurt anybody or anybody's works. It should be kept in mind that we, all *hard-working* researchers in the world, are *not* enemies to each other but actually *comrades* to each other, who are fighting against the *common enemies*, *i.e.*, insufficient understanding of the Mother Nature and the lack of tools potent enough to uncover the essence of natural phenomena, as well as being complacent of the status quo like that. We truly hope for the future where we, all researchers, go hand-in-hand with each other to improve our understanding of the Mother Nature, by bringing together ones' own strengths under the common cause *instead of* competing against each other or even sabotaging each other's studies , and by sharing all information with each other*instead of* keeping crucial information to oneself. Then, our understanding of the Mother Nature should surely improve much faster than we've ever experienced. (If, however, there are, by any chance, *corrupt* researchers who are indulging in the complacency and/or who attempt to deform the scientific truths to their own interests, we *wlll* resolutely fight against them.)

5 Acknowledgments

The author (K.E.) greatly thanks Prof. Tetsushi Yada at Kyushu Institute of Techonology, Japan for the logistic support and encouragements during the middle third of this project, which includes this study and some others [1, 2, 3, 118, 4, 82, 127], and which was conducted first in the author's home in Yokosuka, Kanagawa, Japan, second in Kyushu Institute of

Technology, Japan, and last in the author's home in Chichibu, Saitama, Japan. He is also grateful to Prof. Dan Graur at University of Houston, TX, US and Dr. Giddy Landan at Christian-Albrechts-University of Kiel, Germany for letting me participate in their project, "Error Correction in Multiple Sequence Alignments", which was funded by US National Library of Medicine (grand number: LM010009-01 to Dan Graur and Giddy Landan, then at the University of Houston), from September 2009 till June 2011; partly inspired by their project, the author came up with this project. The author appreciates the inspiring discussions with Dr. Reed A Cartwright at Arizona State University, US and with Dr. Ian Holmes at University of California, Berkeley, US. He is also grateful to Prof. Naruya Saitou at National Institute of Genetics (NIG), Japan, and Dr. Kirill Kryukov at Tokai University, Japan for helping his interest in sequence alignment methods originate and grow, while he was studying with them at NIG. Last but not least, the author appreciates all of his family members, relatives, mentors, (ex-)friends, (ex-)supervisors, and (ex-)colleagues, for their support since his infancy, which enabled him to tread (or wander?) the scientific path and to manage to "finish" this project through all those difficulties and tough times.

The project including this study was in part supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan (grant numbers: KAKENHI Grant numbers 221S0002, 15H01358, both to Tetsushi Yada).

5.1 NOTE on the relationship between this study and a predecessor project

As described in the Acknowledgments above, the project including this study was born partly inspired by the project of Drs. Dan Graur and Giddy Landan, entitled:
"Error Correction in Multiple Sequence Alignments." Because of this fact, someone may wonder about the relationship between their project and this study, because, among other studies in our current project, this study appears the most similar to their project. We

sincerely declare that **this study is totally distinct from their project**. The reasons are the following:

- The only goal shared by the two studies are "to improve MSAs"; more precisely, this is the goal of their project, and the goal of ours is "to improve the results of analyses depending on MSAs."

- Whereas their project aims to construct a *single* MSA that should be more accurate than the input reconstructed MSA, this study aims to construct a **probability distribution of MSAs**.

- Their project first horizontally chops the input MSA into an alternating array of highly reliable regions and "likely erroneous" regions, then, attempts to improve the latter; in contrast, this study deals with *all* regions of the input MSA, at least in principle; although, for convenience, this study first detects regions where "complex" errors are highly likely, such regions are then *excluded* from the main analyses, whereas their project *selectively* analyze (or process) the identified "likely erroneous" regions.

- The methods of providing alternative MSAs (within each window) may appear particularly similar; actually, however, they are also distinct; their project poses alternative MSAs, which are basically independent of the input MSA, by trying every possible horizontal positioning of (the minimum number of) gap-masses so that each sequence has the same length as in the input MSA; in contrast, this study poses alternative MSAs from the *neighborhood* of the input MSA; although the multiple-"shift"s of gap-blocks that this study performs may somewhat similar to the trial of every possible positioning of gap-masses in their project, the construction of gap-blocks in this study depends totally on the input MSA, whereas the construction of gap-masses in their project does not; moreover, the definitions of the gap-blocks in this study depend on a phylogenetic tree, whereas the definitions of the gap masses in their project do not; therefore, the superficial similarity of these two methods is just a coincidence.

- Besides, the author has never seen, or been told of, the *concrete* algorithms, or *concrete* implementations of the methods, that they developed; therefore, he has *never* been able to steal their *concrete* ideas, even if he wanted to(, and he actually has *never* want to do such an unethical thing); all the algorithms and (implementations of) the methods in this study, except the two mentioned below and the already published ones, are the author's own inventions.

- There are only two *concrete*, *albeit* rather secondary, ideas (regarding programming) that the author was suggested by one of the then supervisors, Dr. Giddy Landan; one is the re-use of the column-wise probabilities already computed (via the pruning algorithm); and the other is the use of global variables to control the amount of the diagnostic outputs; even today, the author benefits from these ideas; the author truly appreciates Dr. Landan for this, in addition to the gratitude toward him & Dr. Graur the author already expressed in the Acknowledgments above.

For these reasons, it should be obvious that the two studies are *distinct*.

Appendixes

A Terminology Used in This Paper

Here, we explain, or define, some terms in this paper that the readers may not be so familiar with.

A.1 Terms also used by other researchers

First, we briefly explain terms also used by other researchers. (See the references cited for more detailed descriptions or definitions.)

homology structure A **homology structure** [128, 105] is a(n) (*horizontal*) order structure defined *among* the residues in a set of homologous sequences. Briefly, it is defined by two elements: (i) the residue-level homology relationships, which causes the mutually homologous residues to be *vertically* aligned, and (ii) the (*horizontal*) spatial order relationships among the residues in each individual sequence, which causes the residues in each sequence ordered in exactly the same way as in the sequence. Although this structure *uniquely* determines the order among residues in each sequence, it cannot specify the order between some residues that are *not* homologous to each other. [63] In this sense, a homology structure may be considered an *equivalence class* of the alignments of the same set of homologous sequences. Some researchers (including us, as well as those defined the term) consider that homology structures are more important than PWAs or MSAs (*e.g.*, in the matrix format). Some of them impose some rules

[63] For example, consider a PWA of sequences, say, A and B. And assume that you found a portion of the PWA where a run of gaps in A immediately follows a run of gaps in B. In this portion, the homology structure *leaves* the order *unspecified* between any residue in A and any residue in B. Therefore, in terms of the homology structure, the PWA in which the runs of gaps are swapped, and even a PWA in which the run of gaps in A interrupts that in B, are equivalent to the PWA considered first.

to specify the order among *non-homologous residues* as explained above, so that each homology structure will be represented *uniquely* by a PWA or an MSA (*e.g.*, [3]).

indel process, indel history First, along a time-interval, an **indel process** is defined as a series of insertions/deletions that occurred *at specific points of time*. Then, an **indel history** (*e.g.*, [76]) is defined as a series of insertions/deletions that occurred *in a specific order*, *regardless of* exactly when they occurred. Thus, an indel history can be considered as an *equivalence class* of indel processes, consisting of all those indel processes with the same set of indels that occurred various timings (while keeping the same order). In a stochastic model of sequence evolution, the indel processes belonging to each indel history provide a probability density, with an integral element associated with each time-point. Then, the probability of the indel history is given by the (multiple) time-integration of such a probability density. The concept of indel history can be extended to a phylogenetic tree (*e.g.,* [1]). An indel history along a phylogenetic tree is defined as a set of indel histories, each of which is an indel history along the time-interval corresponding to each branch of the tree.

A.2 Terms already used in our previous papers

Next are terms we already used in our previous papers (*e.g.*, [1, 2, 3]).

gapped column, gapless column A **gapped column** is a column in an alignment that contains at least one gap. A **gapless column** is an alignment column that contains no gaps.

preserved ancestral site (PAS) A **preserved ancestral site (PAS)** is a site that is preserved throughout the time-interval (for a PWA) or throughout the phylogenetic tree (for an MSA). The site needs to keep accommodating a residue throughout the history, but the residue may be replaced with other one(s). In other words, a PAS

is a site that has *not* been hit by any indels throughout the history. Consequently, it always results in a gapless column. (Thus, a PAS may be identified with a gapless column in most of practical purposes. PAs can be used when you want to stress the aspect of "preservation" (i.e., no affecting indels).)

alignment segment, PWA/MSA segment An **alignment segment** is defined as a subset of an alignment which consists of some *contiguous* alignment columns. It is thus delimited by two columns, or a column and an alignment-end(, or two alignment-ends). It is also referred to as "**PWA/MSA segment**" if the alignment is a PWA/MSA.

gapped segment, gapless segment A **gapped segment** is a *maximally spanning* alignment segment consisting *exclusively* of gapped columns, *never* interrupted by a gapless column, but *always* flanked by two gapless columns or a gapless column and an alignment-end(, or two alignment-ends). On the other hand, a **gapless segment** is a (*maximally spanning*) alignment segment consisting *exclusively* of gapless columns. [64]

gap-pattern block A **gap-pattern block** is a (*maximally spanning*) alignment segment consisting *exclusively* of alignment columns that show the same "presence"/"absence" (*i.e.*, residue/gap) pattern across aligned sequences. It provides an intermediate-level building block of a gapped segment. (NOTE: It should *not* be confused with the "gap-block" below.)

"presence"/"absence" states When a site of a sequence state, or, nearly equivalently, a cell in an alignment specified by a column and a sequence (in a row), is occupied with a residue, we say that the site (or a cell) is in a **"presence" state**; if it is occupied with a gap, it is said to be in an **"absence" state**.

ancestry index In our quite general model of *genuine* sequence evolution [1, 2, 118], each

[64] When *accurately* computing alignment probabilities, each gapless segment is *technically* regarded as a set of gapless columns *intercalated with* case-(i) "gapped segment"s [2, 4].

site of a sequence is assigned its own *unique* **ancestry index**, to keep track of the evolutionary course of the site. Mutually non-homologous sites must have distinct ancestry indexes, and homologous (and thus vertically aligned) sites must have the same ancestry index. And when some sites are inserted, they must be assigned new ancestry indexes unique to them. As long as the they satisfy these properties, the indexes may be anything (*e.g.*, integers, alphabetical characters or their combinations, etc.). Although the indexes have been introduced *primarily* to keep track of the evolutionary course of each site, they have a fringe benefit of enabling indel rates to vary across regions (or sites) *beyond* the *mere* dependence on the residue state of the (sub-)sequence [1].

A.3 Terms first used by this paper

Finally, we explain, or *define*, terms that this paper is (probably) the first to use.

swappable columns, non-swappable columns Two contiguous alignment **columns** are regarded as (and indeed are) "**non-swappable**" if *any* sequences have residues in *both* columns(, especially when the alignment is merely representing a homology structure). Otherwise, (*i.e.*, if *no* sequences have residues in *both* columns), they are regarded as (and indeed are) "**swappable**"(, especially when considering a homology structure).

gap-block A **gap-block** is a *maximally spanning* rectangular block of gaps in an alignment (Figure 2, panel A); its *vertical* support is a set of sequences (referred hereafter as "*(gap-block-)supporting sequences*") that are *delimited*, *i.e.*, separated from the others, by a branch of the phylogenetic tree; and its *horizontal* support is a set of columns (or sites) (referred hereafter as "*(gap-block-)supporting columns (or sites)*") *not* interrupted by *any* non-swappable columns that contain residues in any of the supporting sequences. (Typically, the supporting columns make up a contiguous set of columns, but this is not always the case; they may be interrupted by swappable columns and/or by other

gap-blocks whose supporting sequences include the supporting sequences of the block we are considering.) (NOTE: The "gap-block" defined here should *not* be confused with the "gap-pattern-block" above.)

residue-block A **residue-block** is an object *complementary to* a gap-block (Figure 2, panel E). More precisely, it is a rectangular *framework* that encompasses a mass of residues; its *vertical* support is a set of sequences *complementary to* the set of sequences supporting the gap-block(, thus *delimited* by the same tree-branch as the gap-block); and its *horizontal* support is identical to that of the gap-block. Unlike each gap-block, which consists *solely* of gaps, each residue-block does *not necessarily* consist solely of residues; depending on the indel history, each residue-block may *accommodate* a number of gap-blocks.

isolated gap-block A **gap-block** is *defined as* "***isolated***" either if it does *not* overlap any other gap-blocks *horizontally*, or if it is *vertically* separated from each *horizontally* overlapping one by *at least* three branches in the phylogenetic tree (panels B, C and D of Figure 2).

swappable blocks, non-swappable blocks The "**swappable/non-swappable**" concept between two contiguous columns (defined above) can be extended also to two contiguous **gap-pattern blocks**. The concept further extends to two contiguous **gap-blocks** and to two **residue-blocks** as well. First, two contiguous residue-blocks are swappable if the sets of sequences supporting them share no sequences with each other. Then, at the same time, the two gap-blocks complementary to such swappable residue-blocks are also swappable.

effective insertion/deletion (indel) An **effective insertion/deletion (indel)** is a single insertion/deletion that can create a gap-block (and its complementary residue-block). Typically, it belongs to a parsimonious indel history that can explain a gapped

segment. [65] It should be noted that such effective indels are *not necessarily* the indels in the *true* evolutionary history that created the alignment, because (the indel component of) such a true history is *not necessarily* parsimonious.

effective-parent/child/sibling Consider a phylogenetic tree of aligned sequences. Then, consider a tri-valent node (*i.e.*, a node connecting three branches) in the tree. Removing this node splits the set of aligned sequences into three sub-sets. Any two of the three sub-sets are *defined as* the "**effective-siblings**" to each other. Next, consider a union of a pair of effective-sibling sub-sets(, which actually equals the *complement* of the remaining sub-set). This union is *defied as* the "**effective-parent**" of the effective-sibling sub-sets; conversely, the effective-sibling sub-sets are *defined as* the "**effective-children**" of the union. These "effective-parent/child/sibling" relationships *translate into* the relationships *between* gap-blocks and those *between* residue-blocks, each supported by the respective sub-set of sequences, as well. Furthermore, the relationships *translates into* the relationships *between* the branches separating these sub-sets of sequences. These "effective-"relationships become *actual* (parent/child/sibling) relationships if the root is placed on the side of the effective-parent branch opposite to the effective-child branches.

effective-complementary-sibling Sometimes, two gap-blocks are said to be **effective-complementary-siblings** to each other, if their *complementary* residue-blocks are *effective-siblings* to each other.

horizontally equivalent, horizontally identical Two gap-blocks are said to be **horizontally equivalent**, or **horizontally identical**, (to each other), if the set of sites (or columns) supporting one is identical to that supporting the other. This concept

[65] If there are multiple parsimonious indel history, we choose the one closest to the indel history constructed by concatenating Dollo parsimonious histories [83] each of which realizes each column's "presence/absence" pattern [2].

applies also to two residue-blocks.

horizontally include A gap-block, say B_1, is said to **horizontally include** another one, say B_2, if the set of sites (or columns) supporting B_1 includes that supporting B_2. Conversely, B_2 in this case is said to be **horizontally included** in B_1. These concepts apply also to two residue-blocks(, although rarely used).

horizontally overlap Two gap-blocks are said to **horizontally overlap** (or to be **horizontally overlapping**) each other if the set of sites (or columns) supporting one overlaps (or is overlapping) that supporting the other. This concept applies also to two residue-blocks.

vertically equivalent, vertically identical Two gap-blocks are said to be **vertically equivalent**, or **vertically identical**, (to each other), if the set of sequences supporting one is identical to that supporting the other. This concept applies also to two residue-blocks.

vertically complementary Two gap-blocks are said to be **vertically complementary** (to each other) if the set of sequences supporting one is complementary to that supporting the other. This concept applies also to two residue-blocks.

vertically include A gap-block, say B_1, is said to **vertically include** another one, say B_2, if the set of sequences supporting B_1 includes that supporting B_2. Conversely, B_2 in this case is said to be **vertically included** in B_1. As special cases, a gap-block *always* vertically includes its effective-child gap-blocks, and is *always* vertically included in its effective-parent gap-block. These concepts apply also to residue-blocks.

vertically overlap Two gap-blocks are said to **vertically overlap** each other if the set of sequences supporting one overlaps that supporting the other. This concept applies also to two residue-blocks.

(vertically) overlapping yet non-nesting, ONCS, ONN Two *vertically overlapping* gap-blocks are said to be **(vertically) overlapping yet non-nesting** if one *neither* vertically include *nor* is vertically included in the other(, *i.e.*, if their sets of supporting sequences do not nest). Such cases are sub-classified into two categories: if the gap-blocks are *effective-complementary-siblings* to each other (see above), they are sub-classified as "**overlapping yet non-nesting but effective-complementary-siblings**" (or "**ONCS**" for short); otherwise, they are sub-classified as "**overlapping yet non-nesting *nor* effective-complementary-siblings**" (or "**ONN**" for short).

interfering/non-interfering gap-blocks Two gap-blocks are said to be **interfering** if they are vertically overlapping, vertically complementary to, or effective-siblings to, each other. Otherwise, the two blocks are said to be **non-interfering**. Another way of defining these terms is to pay attention to the number of branches separating the gap-blocks: if three or more branches *vertically* separate the gap-blocks, they are **non-interfering**; otherwise, they are **interfering**. [66]

topology In this paper, the "**topology**" means a *discrete* classification of positional relationships among gap-blocks from the viewpoint of the sets of parsimonious ancestral "presence"/"absence" states; the **topology** is regarded as "conserved" if the sets of parsimonious ancestor sets of two compared MSAs are obtained from each other *via* continuous (*i.e.*, site-by-site) moves *alone*; the **topology** is regarded as "changed" if the transformation between the two sets of parsimonious ancestor sets *inevitably* involve some *discrete* changes, such as the changes in: the number of ancestor sets, the number of (effective-)parsimonious indel histories that can result in the MSA, the sizes of the effective-parsimonious indels and/or the branches they occur, etc.

[66] This definition assumes that all internal nodes are trivalent. When the root is bifurcated, the two branches that it connects is *collectively* regarded as a single branch. When the gap-blocks are mediated by a node connecting four or more branches, they are considered as *non-interfering* even if only two branches separate them.

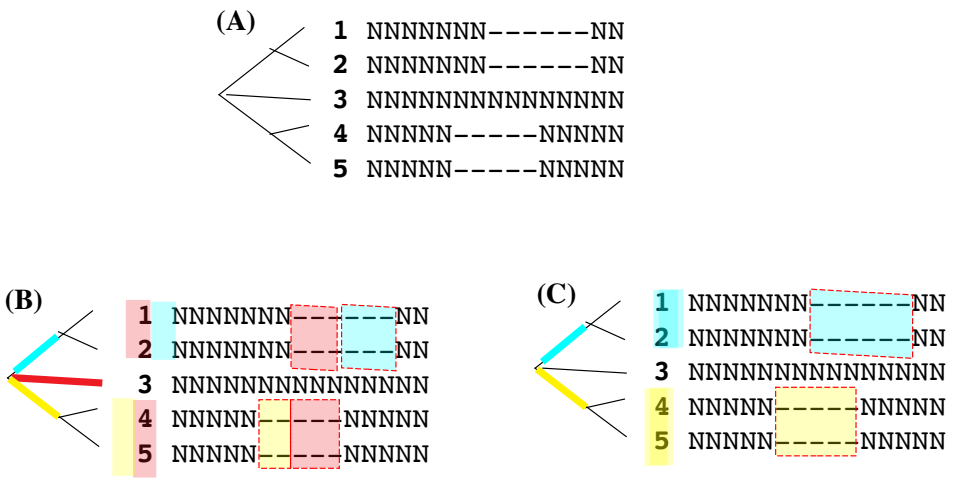


Figure 11: **Example case where Dollo-parsimony-based indel history is *not* truly parsimonious.** **A.** An example gap-configuration accompanied by the phylogenetic tree of the aligned sequences. **B.** The Dollo-parsimony-based indel history demarcates three gap-blocks (yellow, red, and cyan). The red one is the effective-parent of the yellow and cyan ones, which are effective-siblings to each other. **C.** A *truely* parsimonious indel history demarcates two gap-blocks (yelow and cyan).

B Determining Gap-Blocks

In some procedures of ANEX, especially the creation of windows (subsection 2.4 and appendix C) and the exploration of MSA neighborhoods (subsection 2.5), **gap-blocks** play crucial roles.

As defined in appendix A above, a "gap-block" is a *maximally spanning* rectangular block of gaps in an alignment; it is *vertically* supported by a clade of sequences, and *horizontally* supported by a set of (usually (but not always) contiguous) columns.

Basically, the gap-blocks are determined, or demarcated, based on the indel history that was created by horizontally concatenating the Dollo parsimonious histories [83] of "presence"/"absence" states in individual columns of the input MSA [2]. Such an indel history is one of the simplest ones that can explain the gap-configuration of the input MSA.

However, in some cases, such Dollo-parsimony-based indel histories does *not* provide a truly parsimonious history. Especially the case illustrated in Figure 11 A may be quite frequent; in such a case, a pair of effective-sibling gap-blocks are mediated by their common effective-parent gap-block (Figure 11 B). If left as they are, the gap-blocks determined by

these indel histories may *frequently* fail to locate the *true* MSA when the neighborhoods of the input MSA are explored.

The current version of ANEX addresses this problem by performing a **"branch-and-merge" operation** (see, *e.g.*, SM-5.2 of [2]) on each such Dollo-parsimony-based indel history (11 B); this operation "convert"s the Dollo-parsimony-based history into a *truly* parsimonious indel history (or *at least* an indel history closer to the *truly* parsimonious one), in which a pair of effective-sibling gap-blocks horizontally overlap (but *not* nest inside) each other (Figure 11 C).

[Construction of Dollo-parsimony-based indel histories is implemented in the subroutine, "cal_init_psm_cands_for_sgl_msa," in the module, "MyTreeMap_indels_spt_odr2.pm," of ANEX. Conversion of Dollo-based indel histories to histories closer to truly parsimonious ones is implemented in the subroutine, "bch_br_parsimonize_dollo_indel_histories," in the same module. Determination of gap-blocks *via* an indel history (within a gapped segment) is implemented in the subroutine, "br_list_gblocks," in the same module.]

C Creating Windows for MSA Neighborhood Exploration: Details

Subsection 2.4 briefly outlined the processes to create windows. This appendix describes more details on how the two types of windows, ordinary windows and "purge-like-error-candidate containing" (PCC) windows, are created.

There are two key parameters that user can specify to *indirectly* control the sizes of windows: N_W, which is the maximum number of gap-blocks each ordinary window can contain; and W_M, which is the maximum number of sites that each gap-block can "shift" in each direction (to the left or to the right) from the "origin" (*i.e.*, the input MSA).

First, we describe how to create ordinary windows. In short, starting from a gapped

segment as a "seed," ANEX serially incorporates the right-neighboring gapped segment if it is within $2W_M$ columns from the current right-end of the window, until the total number of gap-blocks reaches N_W. After that, ANEX always attempts to extend the window so that each gap-block can "shift" by W_M sites in both directions, but it stops the extension if it is blocked by the flanking gapped segment before finishing the extension, leaving at least one gapless column in between the window and the flanking gapped segment. This is realized by the following set of rules.

1. Each window contains the greatest possible number (but *not* exceeding N_W) of gap-blocks that come in a horizontal series in the input MSA.

2. Neighboring windows overlap in general, with at least one extra gap-block in each window.

3. No window extends to, or across, any region with "likely complex errors"; in other words, the "likely complex errors" define the "prohibited zones," which are "off-limits" to the (both ordinary and PCC) windows.

4. The left-end of each window is whichever is closer to the window center between the following: (a) the column that is two columns to the right of the right-end of the left-neighboring gap-block, and (b) the column that is W_M columns to the left of the left-end of the leftmost gap-block in the window.

5. The right-end of the window is defined similarly to (4), with the "left" and "right" swapped.

6. In any case, each end of each window must always be flanked by either a gapless column or an MSA-end.

7. Whenever the number of (gapless) columns separating a pair of neighboring gap-blocks is more than $2W_M$, these columns (excluding the outer W_M on both sides) separate

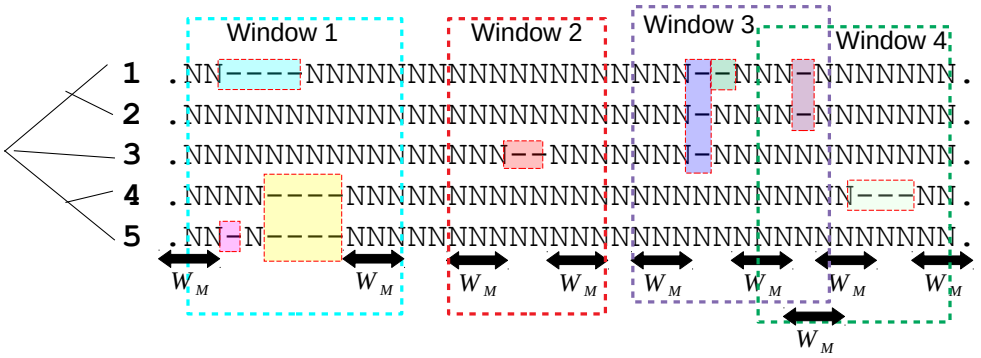


Figure 12: **Rules to create ordinary windows.** The parameters used here are $N_W = 3$ and $W_M = 3$. Each window is horizontally delimited by a colored dashed rectangle. Below the MSA, each both-headed arrow represents W_M columns from each end of each gap-block, which is either the leftmost or rightmost one in each window. **1.** Window 1 has exactly 3 ($= N_W$) gap-blocks. **2.** Gap-blocks in windows 1 and 2 are separated by 8 ($> 2W_M$) gapless columns. Thus they do not overlap. **3.** Gap-blocks in windows 2 and 3 are separated by 7 ($> 2W_M$). **4.** Thus, window 2 consists only of 1 ($< N_W$) gap-block. 4. Window 3 consists of 3 gap-blocks. **5.** Because the 1st and 2nd leftmost gap-blocks in window 3 adjoin each other, they are inseparable. Therefore, windows 3 and 4 overlap each other only by 1 gap-block. **6.** Because window 4 reaches the right-end of the input MSA before incorporating 3 ($= N_W$) gap-blocks, it consists only of 2 ($< N_W$) gap-blocks.

two clusters of overlapping windows.

8. For (1) and (6) to be consistent, we must discard every gapped segment containing more than N_W gap-blocks.

As an illustration, Figure 12 gives an example of applying these rules to a small input MSA.

Depending on its gapped segment contents, each ordinary window may contain one or two **sub-window**(s). Sub-windows are created in *almost* the same way as the full ordinary windows are, with the N_W replaced with $(N_W - 1)$. Currently, ANEX puts each sub-window *under* ***only one*** full ordinary window, in order to avoid redundant analyses.

Next, we describe how to create PCC windows. The creation of each PCC window starts with a "purge-like-error-candidate" (subsection 2.2) as a "seed." Then, the window is extended in both directions, by incorporating the closest gapped segment if it is within W_M columns *from* the "seed" and if the resulting total number of gap-blocks does *not* exceed $N_W - 2$. More precisely, the following rules dictate the PCC window creation.

1. If the "seed" is *not* contained in a window-frame completely (*i.e.*, if it overlaps any "prohibited zones"), dismiss the "seed."

2. Search for gapped segments overlapping the "seed" and the "neighboring" segments within W_M columns from the "seed."

3. If the total number of gap-blocks in the overlapping gapped segments exceeds $(N_W - 2)$, dismiss the "seed."

4. Otherwise, incorporate the overlapping gapped segments into the PCC window.

5. As long as the total number of gap-blocks is less than $N_W - 2$, attempt to expand the window by incorporating the then nearest-neighboring gapped segment into the window.

6. If the total number is expected to exceed $N_W - 2$, stop expanding the window *before* incorporating the gapped segment, *or*, if the "neighboring" segments run out, stop expanding the window *even* if the total number is less than $N_W - 2$.

7. Determine the boundaries of the window so that all constituent gapped segments & the "purge-like-error-candidate" (*i.e.*, "seed") will be contained, but so that *no* other gapped segments will be contained. (The boundaries *must* always be within W_M columns from the outermost ends among those of the "seed" and the incorporated gapped segments.)

8. In any case, each end of each window must always be flanked by either a gapless column or an MSA-end.

[For the actual implementation of how ordinary windows are created, see the subroutine, "prepare_sliding_windows," in the module, "MyANEX_Supple.pm," of ANEX. For the actual implementation of how PCC windows are created, see the subroutine, "mk_pcc_sliding_windows," in the same module.]

C.1 Sorting gap-blocks in each window

Once each window is created, the gap-blocks in it are sorted in an *unambiguous* manner, as follows: (1) sort them in descending order of the number of sequences supporting each of them; (2) when some gap-blocks tie in (1), sort them in the order that their supporting sequences appear in the rows of the MSA (from top to bottom); (3) when some gap-blocks tie even in (2), (*i.e.*, when some blocks are vertically identical to one another,) sort them according to their horizontal positions in the MSA (from left to right).

The coordinates (for "shift"s) are assigned to the gap-blocks according to this order. And multiple-"shift"s are performed using the structure based on this order. (See subsection 2.5.)

D Elementary Moves Used for MSA Neighborhood Exploration

As explained in section 2.5, ANEX explores the neighborhoods of the input MSAs *via* an "elementary move" [45, 3] or a combination of a number of elementary moves. Each elementary move is realized as the move of a gap-block (Figure 2, panel A) or its complementary residue-block (Figure 2, panel E). [67]

Here, we list and describe the elementary moves that current version of ANEX attempts.

(i) A "**shift**" of a gap-block. This move just *horizontally* re-positions a gap-block without affecting others nearby (if at all) (Figure 3 A). This can be done as long as the gap-block is "**isolated**" from others (panels B, C and D of Figure 2). [68]

(ii) A "**purge**" of two equal-sized gap-blocks that affect the *complementary sets* of sequences (, which are separated by a single branch) (Figure 3 B). (This move is the

[67] See appendix A for the definitions of the "gap-block" and the "residue-block."

[68] See appendix A for the definition of the "isolated gap-block."

reverse of the "ex-nihilo" of a pair of (spurious) gap-blocks.) (This move is defined *utterly uniquely*, and therefore can be processed very quickly.)

(iii-a) An "**s-merge**," *i.e.*, a "**same-type-merge**," which merges two gap-blocks that are supported by the *same set* of sequences and that are separated by a number of columns (Figure 3 C). (This move is the reverse of a(n) (erroneous) "s-split" of a gap-block.) (This move is defined *uniquely*, *modulo* a shift of the resulting single gap-block. Thus, it can be processed as quickly as shifts.)

(iii-b) A "**c-merge**", *i.e.*, a "**complementary-merge**," which merges two *non-equal-sized* gap-blocks that are supported by the *complementary sets* of sequences (Figure 3 D). (This move, too, is defined *uniquely*, *modulo* a shift of the resulting single gap-block. Thus, it also can be processed as quickly as shifts.)

(iv-a) An "**s-split**", *i.e.*, a "**same-type-split**," which splits a single gap-block into two gap-blocks both of which are supported by the *same set* of sequences (the reverse of Figure 3 C). There are {the block-length minus one} ways of splitting the gap-block, resulting in pairs of gap-blocks of different length-combinations. (And each of the two resulting gap-blocks may shift by some sites. Thus, it can take some time to process this move.)

(iv-b) A "**c-split**", *i.e.*, a "**complementary-split**," which splits a single gap-block into two gap-blocks, each of which is supported by each of the *complementary sets* of sequences (the reverse of Figure 3 D). (Although there are (theoretically) infinite ways of such moves, which result in different combinations of block-lengths, ANEX *restricts* the moves to attempt by setting an upper bound on the shorter block-length. As in the previous case, each of the two resulting gap-blocks may shift by some sites. Because of these two factors, processing this move can consume quite an amount of time.)

(v) A "**reverse-purge**" (also known as an "**ex-nihilo**" [45]) of a pair of gap-blocks, each of which is supported by each of the *complementary sets* of sequences (the reverse of Figure

3 B). (There are many ways of performing this move, with different sets of original columns affected, and with different branches separating the complementary sets of sequences. Currently, ANEX attempts this move on only those candidate combinations of regions and branches which passed two statistical tests; see [82] for more details.)

(vi-a) A "**v-merge**", *i.e.*, a "**vertical-merge**" of a pair of *equal-sized* "effective-sibling" [69] gap-blocks that are supported by two ("effective-sibling") sequence sets separated by two branches (Figure 3 E). [70] (This is the reverse of a (erroneous) vertical-split of a gap-block.) (This move is defined *uniquely*, *modulo* a shift of the resulting gap-block. Thus, it can be processed as quickly as a shift.)

(vi-b) A "**cv-merge**", *i.e.*, a "**complementary-vertical-merge**," which merges a pair of *equal-sized* "effective-sibling" residue-blocks (each *complementary to* a gap-block) , which iare supported by wo ("effective-sibling") sequence sets separated by two branches (Figure 3 F). [71] (This is the reverse of a(n) (erroneous) vertical-split of a sequence-block.) (This move, too, is defined *uniquely*, *modulo* a shift of the resulting gap-block (complementary to the resulting residue-block). Thus, it can be processed as quickly as a shift.)

(vii-a) A "**v-split**", *i.e.*, a "**vertical-split**," of a single gap-block into its two "effective-child" [72] gap-blocks, which are supported by two ("effective-child") sequence sets separated by two branches (the reverse of Figure 3 E). *As long as* the original gap-block affects a single clade (or the complement of a single clade), we can define a *unique* vertical-split (at the top node of the clade). [73] (In such a vertical-split, each of the two resulting

[69] See appendix A for the definition of the "effective-sibling."

[70] When a pair of *non-equal-sized* gap-blocks are involved, the move is referred to as an "**iv-merge**", *i.e.*, an "**incomplete-vertical-merge**."

[71] When a pair of *non-equal-sized* residue-blocks are involved, the move is referred to as an "**icv-merge**", *i.e.*, an "**incomplete-complementary-vertical-merge**."

[72] See appendix A for the definition of the "effective-child."

[73] Other "vertical splits" actually do exist. However, such moves increase the number of indels by two or

gap-blocks may shift by some sites. Thus, the time-consumption should be on the same order as that for double-shifts.)

(vii-b) A "**cv-split**", *i.e.*, a "**complementary-vertical-split**," which splits a single residue-block into its two "effective-child" residue-blocks (*complementary to* the resulting two gap-blocks), which are supported by two ("effective-child") sequence sets separated by two branches (the reverse of Figure 3 F). In this case, too, *as long as* the original residue-block is supported by a single clade (or the complement of a single clade), we can define a *unique* complementary-vertical-split (at the top node of the clade). [74] (In such a complementary-vertical split, each resulting residue-block may shift. Thus, the time-consumption should be on the same order as that for double-shifts.)

(viii) A "revserse-(i)CII", *i.e.*, a "**reversal of a(n) (incomplete-)collapse of independent insertions**," which *recovers* the independent effective-insertions [75] that *collapsed* when the input MSA is reconstructed [3] (Figure ...). Usually, this move *drastically* changes the set of gap-blocks (and thus the set of effective-indels). Thus, a *reliable* way to realize this move is to create a new MSA by actually recovering the independent effective-insertions, and to re-compute the gap-blocks from scratch. (This is indeed what ANEX does.)

more each, as if they were CIIs (Figure S1A). Such true MSAs seem *very unlikely*, though the probability is not zero.

[74] Other "vertical-split"s do exist. However, such moves are actually "ex-nihilo"s aligned with a gap (Figure S1B). The true MSAs with such patterns seem *very unlikely*, because the number of indels increases by two or more each.

[75] See appendix A for the definition of the "effective-insertion."

E Performing single "shift"s of gap-block *interfering* with other gap-block(s)

In sub-subsection 2.5.2, single-"shift"s of a gap-block was discussed in the simplest situation where the gap-block is *isolated*. When the gap-block to be "shift"ed *interferes* with some gap-blocks around it, however, some complications may arise. Here, we consider what will happen in such cases, classifying the situations. [76]

(1) Interfering with a *vertically identical* gap-block. In this case, ANEX *tentatively* applies the rules for the "shift"s of *vertically including/included* gap-blocks (see (3) and (4) below); the gap-block that is above the other in the list of gap-blocks is *regarded as "vertically including"* the other. Once these gap-blocks *horizontally* overlap or adjoin (*i.e.*, touch), the resulting MSAs are destined to be highly redundant. Therefore, ANEX assigns the "degree of degeneracy = 0" to these MSAs, and will *never* use them in the downstream analyses, including the computation of MSA probabilities (except the substitution component). In any case, such MSAs will be *properly* treated when these gap-blocks are "s-merge"d.

(2) Interfering with a *vertically complementary* gap-block. In this case, the current version of ANEX performs the single-"shift"s of the subject gap-block regardless of the *vertically complementary* one. However, when these gap-blocks *horizontally* overlap (but *excluding* the cases they *adjoin*), they are effectively equivalent to the "c-merge" (or "c-merge + c-split") of the gap-blocks, hence causing some redundancies. Thus, ANEX assigns the "degree of degeneracy = 0" to the resulting MSAs in these cases, and will *never* use them in the downstream analyses; such MSAs will be *properly* treated when these gap-blocks are "c-merge"d (or "c-merge + c-split"ed). Meanwhile, each MSA in which these gap-blocks *adjoin* are very likely to occur *once again*, with the configuration in which the gap-blocks are *swapped* horizontally. When this is indeed the case, ANEX assigns the "degree of degeneracy

[76] This section frequently uses terms like "horizontally xxx" and "vertically yyy," as well as "effective-zzz." See appendix A for the definitions of these terms.

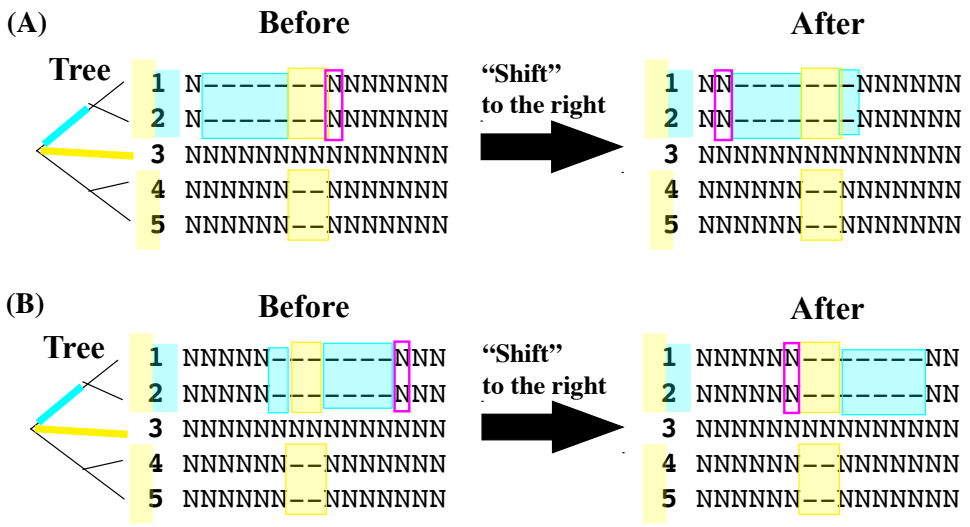


Figure 13: **Single-"shift"s of gap-block interfering with one that *vertically includes* it. A.** Single-"shift" immediately after the two gap-blocks adjoin each other. **B.** Single-"shift" that gets the rear-end of the subject to reach the rear-end of the vertically-including block. After the "shift," the rear-end of the subject is retracted as shown. In each panel, the subject gap-block is shaded in cyan, the vertically-including gap-block is shaded in yellow, and the sites flanking the front-end of the subject (before the "shift") are enclosed in a magenta rectangle.

= 2" to *both* of such MSAs; when computing MSA probabilities, 1/2 will be multiplied to each of their contributions,

(3) Interfering with a gap-block that *vertically includes* the subject. In this case, the single-"shift"s of the subject gap-block proceed as those of an *isolated* gap-block, *except* the two moments (Figure 13), namely, (i) *immediately* after the gap-blocks adjoin each other (panel A), and (ii) when the *rear-end* of the subject reaches the *rear-end* of the vertically-including block (panel B). When (i) the subject gap-block "shift"s *even after* it adjoins the vertically-including block (panel A), we must *regard* the *front-end-flanking* site of the vertically-including block *as* that of the subject as well. Then, the *"essence"* given in sub-subsection 2.5.2 applies as it is. When (ii) the *rear-end* of the subject is about to reach the *rear-end* of the vertically-including block (panel B), the "shift" *itself* can be done as in sub-subsection 2.5.2; however, *as post-processing*, ANEX *retracts* the *rear-end* of the subject to the front-end-flanking site of the vertically-including block.

(4) Interfering with a gap-block that is *vertically included* in the subject. In this case,

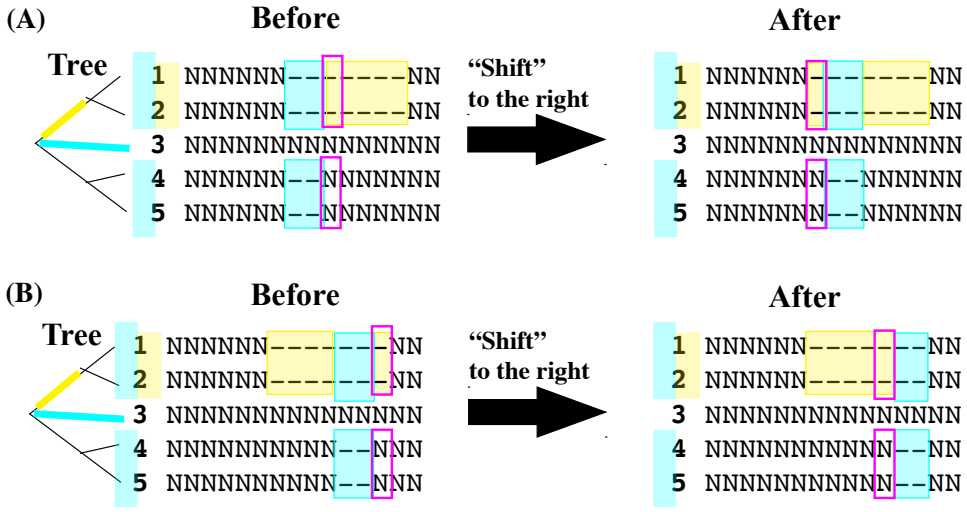


Figure 14: **Single-"shift"s of gap-block interfering with one that is *vertically included* in it. A.** Single-"shift" immediately after the two gap-blocks adjoin each other. **B.** Single-"shift" that gets the front-end of the subject to reach the front-end of the vertically-included block. After the "shift," the front-end of the latter block is retracted as shown. In each panel, the subject gap-block is shaded in cyan, the vertically-including gap-block is shaded in yellow, and the sites flanking the front-end of the subject (before the "shift") are enclosed in a magenta rectangle.

the single-"shift"s of the subject gap-block *itself* completely follows the "essence" given in sub-subsection 2.5.2. However, care must be taken of the *vertically included* gap-block in two moments (Figure 14), namely, (i) *immediately* after the gap-blocks adjoin each other, and (ii) when the *front-end* of the subject reaches the *front-end* of the vertically-included block. When (i) the subject gap-block "shift"s *even after* it adjoins the vertically-included block, the *rear-end* of the latter *extends* to the *rear-end-flanking* site of the subject (*after* the "shift") *naturally*, as a *"side-effect"* of the "shift." When (ii) the *front-end* of the subject reaches the *front-end* of the vertically-included block, the *front-end* of the latter *retracts* to the *rear-end-flanking* site of the subject (*after* the "shift") *naturally*, again, as a *"side-effect"* of the "shift."

(5) Interfering with a gap-block that is *(vertically) overlapping yet non-nesting*(, *i.e.*, in "ONCS" or "ONN" relation) with the subject. [77] In this case, care must be taken only *immediately* after the gap-blocks adjoin each other (Figure 15). When the subject "shift"s

[77] See appendix A for the definitions of "*(vertically) overlapping yet non-nesting*," "ONCS" and "ONN."

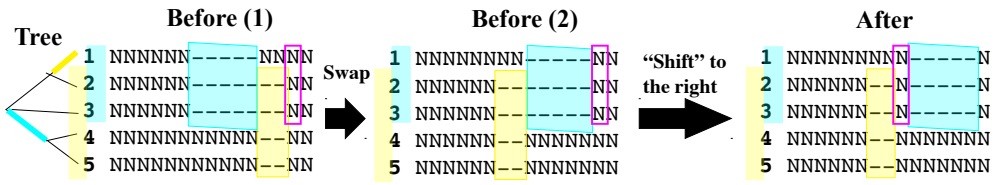


Figure 15: **Single-"shift" of gap-block interfering with one that is *(vertically) overlapping yet non-nesting*(, *i.e.*, in "ONCS" or "ONN" relation) with it.** Single-"shift" immediately after the two gap-blocks adjoin each other. Before the "shift", the two gap-blocks are horizontally swapped. In the figure, the subject gap-block is shaded in cyan, the vertically-including gap-block is shaded in yellow, and the sites flanking the front-end of the subject (before the "shift") are enclosed in a magenta rectangle.

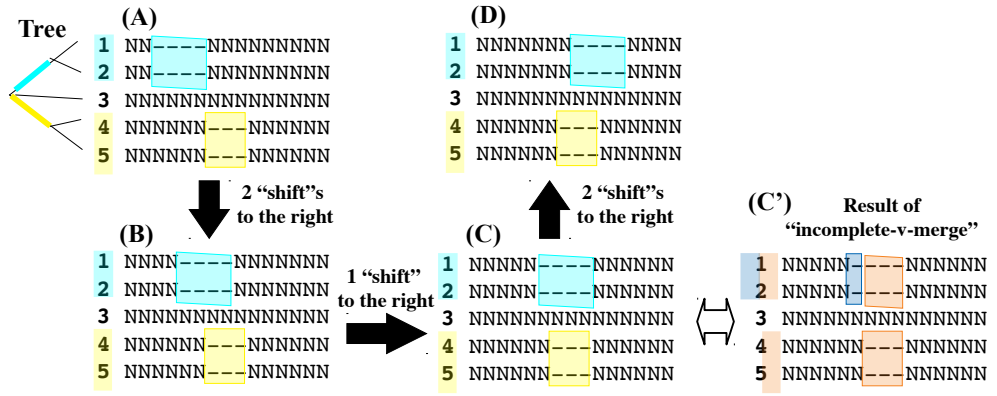


Figure 16: **Single-"shift" of gap-block interfering with its effective-sibling.** The "shift"s of the subject gap-block (the cyan rectangle) are performed in the order: **A** $\rightarrow$ **B** $\rightarrow$ **C** $\rightarrow$ **D**. The yellow rectangle represents an effective-sibling gap-block of the subject. The MSA in panel **C'**, which resulted from an "incomplete-v-merge" of the effective-sibling gap-blocks, is identical to the MSA in panel **C**.

even after it adjoins an ONCS or ONN gap-block, ANEX *horizontally swaps* the two gap-blocks first; then, it "shift"s the subject *exactly* conforming to the *"essence"* in sub-subsection 2.5.2.

(6) Interfering with an effective-sibling gap-block. In this case, the current version of ANEX performs the "shift"s of the subject gap-block regardless of its effective-sibling (Figure 16). When the two gap-blocks are *not* nesting horizontally (panels **A, B, D**), there is no problem. When, however, they *do* nest horizontally (panel **C**), the resulting MSA can also result from a(n) "(incomplete-)v-merge"(panel **C'**); hence, some *redundancies* can occur. The current version of ANEX avoids this "redundancy problem" as follows: first, the "(incomplete-)v-merge"s resulting from the "shift"s of the effective-sibling gap-blocks are

kept and used, *as they are*, for the downstream analyses; second, ANEX does *not* perform the *single* action of "(incomplete-)v-merge"s of effective-sibling gap-blocks; and, third, *if applicable*, ANEX *does* perform "s-split + v-merge"s of effective-sibling gap-blocks, and, in the subsequent multiple-"shift"s, it *excludes* (from the downstream analyses) those MSAs in which the "s-split" gap-blocks are *re-"merge"d*(, because they should also result from the multiple-"shift"s of the input MSA). [78]

(7) Interfering with a *composite*-gap-block. Some gap-blocks do *not* interfere with the subject as single gap-blocks; but they may form a *composite*-gap-block when *horizontally overlapping* each other, and *interfere* with the subject, especially as an *effective-sibling* or as an *effective-child*, or rarely as *vertically complementary* ones. In some cases, a composite gap-block including the subject may *interfere* with another composite gap-block. Currently, ANEX deals with some of these cases, although not necessarily exhaustively.

F Cases where downstream analyses are skipped

The current version of ANEX skips downstream analyses on alternative MSAs it visited during multiple-"shift"s, if they have particular gap-block configurations. This behavior is intended to avoid redundancies in the alternative MSAs resulting from the entire neighborhood exploration. (Note that the multiple-"shift"s themselves are continued.) Here, we list the types of gap-block configurations that make ANEX skip the downstream analyses.

1. When there is at least one null (*i.e.*, gap-only) column; this indicates a "(partial) c-merge," which should be *solidly* performed as a non-"shift"-type move elsewhere.

2. When a pair of vertically identical gap-blocks adjoin or nest; this effectively results in

[78] Similarly, when ANEX performs an "v-split + s-merge" on a pair of effective-parent/child gap-blocks, then, in the subsequent multiple-"shift"s, it *excludes* (from the downstream analyses) those MSAs in which the resulting new gap-blocks *horizontally* nest, because, again, they should also result from the multiple-"shift"s of the input MSA.

an "s-merge," which should be *solidly* performed as a non-"shift"-type move elsewhere.

3. When a pair of vertically identical *and* equal-sized gap-blocks are in the horizontal order opposite from the input one; this results in a gap-configuration totally indistinguishable from that with the same horizontal order as the input, causing redundancy; the simplest way to remove the redundancy is to *ignore* the MSA with the opposite horizontal order.

4. When, as specified by an input variable of the subroutine to perform the multiple-"shift"s, a pair of effective-sibling gap-blocks nest horizontally; this case can occur during the multiple-"shift"s subsequent to the "v-split" of a gap-block followed by the "s-merge" of one of the resulting gap-blocks with a nearby vertically identical one(, which produces the effective-sibling pair); the specified gap-block configuration should result in the same MSA as one of those resulting from multiple-"shift"s of the input MSA, causing the redundancy; to remove the redundancy, the specified gap-block configuration are *ignore*d.

5. When, as specified by an input variable of the subroutine to perform the multiple-"shift"s, an effective-parent&child pair adjoin or nest horizontally; this case can occur during the multiple-"shift"s subsequent to the "s-split" of a gap-block followed by the "v-merge" of one of the resulting gap-blocks with a nearby effective-sibling(, which produces the effective-parent&child pair); then, the same 3rd and 4th sentences as case 4 apply also here.

6. When a gap-block adjoins a vertically complementary gap-block *on each end*; this gap-block configuration is actually equivalent to the configuration where the two (vertically complementary) gap-blocks are "s-merge"d; the latter configuration should be *solidly* dealt with in the session of "s-merge" followed by multiple-"shift"s.

To each alternative MSA that belongs to at least one of the above categories, the degree of degeneracy is infinity, which means that the weight is zero when computing the total

probability. [79]

G Algorithm to detect candidates of CII-containing regions in MSA

As briefly described in sub-subsection 2.5.4, this algorithm pays attention to the residue-blocks in the input MSA, especially the number of monophyletic residue-blocks that vertically overlap one another. This is done efficiently if the input MSA is *horizontally* divided into a set of gap-pattern-blocks. [80] More precisely, a region is *regarded* as a candidate of a "CII-containing region" if it contains more than one monophyletic residue-block; however, if it contains *too many* monophyletic residue-blocks, the region is *tentatively* skipped as it may have resulted in two or more non-interfering (effective-)deletions. The following gives the *"essence"* of the algorithm(, leaving out some details).

[**Algorithm to detect CII-candidates**]

Input: A set of gap-pattern-blocks in the input MSA (within the window)

(= @Set_gpbs = (\@gpb_0, \@gpb_1, ..., \@$gpb_{\$N\_gpb-1}$)),

where $\$N\_gpb$ is the number of gap-pattern-blocks in the MSA);

The individual @gpb_k follows the format:

@gpb_k = ($\$lend\_gpb$, $\$rend\_gpb$, \@$set\_ngap\_seqs$) , where

$\$lend\_gpb$ and $\$rend\_gpb$ are the left- and right-end coordinates, respectively,

of the gap-pattern-block,

@set_ngap_seqs lists (the indexes of) the sequences *not* occupied by gaps in the gap-pattern-block ;

{ Input variables necessary for dividing a set of sequences into monophyletic groups.

(See appendix H.) } ;

[79] In the actual implementation of ANEX, the degree of degeneracy = 0 (zero) is assigned, simply because the infinity cannot be specified as a number.

[80] See appendix A for the definition of the term, "gap-pattern-block."

The upper-bound of the permissible count of monophyletic groups (= $\$MAXCT\_MPG$).

Output: (The reference to) the list of detected CII-candidates, $@list\_CII\_cands$, where
$@\{\$list\_CII\_cands[\$cc]\} = (\$lend, \$rend, \backslash@set\_mpgs, \backslash@set\_br\_up\_or\_lw, \backslash@set\_ngap\_seqs\_core)$
gives information on the $\$cc$-th candidate region, where
$\$lend$ and $\$rend$ are the left- and right-end coordinates, respectively, of the region,
$@set\_mpgs$ is the set of monophyletic groups characterizing the candidate-region,
$@set\_br\_up\_or\_lw$ indicates which branch separates each monophyletic group and
on which end (, upper or lower,) of the branch the monophyletic group is,
$@set\_ngap\_seqs\_core$ is $@set\_ngap\_seqs$ in the "core" gap-pattern-block for this region.

Algorithm:

Create $@list\_raw\_ciicands \leftarrow ( )$;

Create $\$ct\_rawcands \leftarrow 0$;

For $\$k$ from 0 to $\$N\_gpb - 1$, do the following:

Create $(\$lend\_gpb, \$rend\_gpb, \backslash@set\_ngap\_seqs) \leftarrow @\{\$Set\_gpbs[\$k]\}$;

Create $(\backslash@set\_mpgs, \backslash@set\_br\_up\_or\_lw) \leftarrow$

{ Outputs of the algorithm in appendix H with $@set\_ngap\_seqs$ as an input } ;

Create $\$ct\_mpgs =$ { size of $@set\_mpgs$ } ;

If ($\$ct_mpgs < 2$) or ($\$ct_mpgs > \$MAXCT\_MPG$), then:

next ; (Skip the gap-pattern blocks that have too few or too many monophyletic groups.)

EndIf

Search $@list\_raw\_ciicands$ backward for a CII-candidate having

the same $@set\_ngap\_seqs$ as the current gap-pattern-block ;

If (NO such candidates exist in @*list_raw_ciicands*), then:

@{$*list_raw_ciicands*[$*ct_rawcands*]} ←

($*lend_gpb*, $*rend_gpb*, \@*set_mpgs*, \@*set_br_up_or_lw*, \@*set_ngap_seqs*) ;

(*i.e.*, Create a new CII-candidate.)

$*ct_rawcands* ← $*ct_rawcands* + 1 ;

next ;

EndIf

Create $*indx_last* ← { the index of the last such candidate in @*list_raw_ciicands* } ;

Create $*rend_last* ← { the right-end of the last such candidate in @*list_raw_ciicands* } ;

If (*all* the gap-pattern-blocks between $*rend_last* and $*lend_gpb* have

their @*set_ngap_seqs*'s included in the @*set_ngap_seqs* of the present CII-candidate), then:

$*list_raw_ciicands*[$*indx_last*] → [1] ← $*rend_gpb* ;

(*i.e.*, *Merge* the last such candidate and the current gap-pattern-block.)

Else:

@{$*list_raw_ciicands*[$*ct_rawcands*]} ←

($*lend_gpb*, $*rend_gpb*, \@*set_mpgs*, \@*set_br_up_or_lw*, \@*set_ngap_seqs*) ;

(*i.e.*, *Create* a new CII-candidate.)

$*ct_rawcands* ← $*ct_rawcands* + 1 ;

EndIf

EndFor ($*k*)

Create @*list_ext_ciicands* ← ( ) ;

Create $*ct_extcands* ← 0 ;

For $\$i$ from 0 to $\$ct\_rawcands - 1$, do the following:

Create ($\$lend, \$rend, \backslash@set\_mpgs, \backslash@set\_br\_up\_or\_lw, \backslash@set\_ngap\_seqs\_core$)

$\leftarrow$ @{$\$list\_raw\_ciicands[\$i]$} ;

Attempt to extend the CII-candidate region to the left and to the right,

by comparing the $@set\_ngap\_seqs\_core$ and

the $@set\_ngap\_seqs$ of the then flanking gap-pattern-block, if at all ;

If the former includes the latter (in the previous two lines), then:

Extend the CII-candidate by *incorporating* the gap-pattern-block into it ;

EndIf

If the CII-candidate has been extended, then:

Let ($\$lend\_ext, \$rend\_ext$) *be* the left- and right-ends of the *extended* candidate ;

@{$\$list\_ext\_ciicands[\$ct\_extcands]$} $\leftarrow$

($\$lend\_ext, \$rend\_ext, \backslash@set\_mpgs, \backslash@set\_br\_up\_or\_lw, \backslash@set\_ngap\_seqs$) ;

(*i.e.*, *Create* an *extended* CII-candidate.)

$\$ct\_extcands \leftarrow \$ct\_extcands + 1$;

Endif

EndFor ($\$i$)

Create $@list\_CII\_cands \leftarrow$ { Union of $@list\_raw\_ciicands$ and $@list\_ext\_ciicands$,

sorted in spatial order (from left to right) } ;

Return $\backslash@list\_CII\_cands$.

EndAlgorithm

[For the actual implementation in Perl, see the subroutine, "list_cii_candidates", in the module, "MyANEX_MainX.pm", of ANEX.]

H Algorithm to divide set of sequences into monophyletic groups

This algorithm is an integral part of the algorithm given in appendix G. *Verbally*, the algorithm is simple: (1) Pick out a sequence from the input set of sequences; (2) using the picked-out sequence as a seed, grow a monophyletic group of sequences while traversing the tree, first from bottom to top, then from top to bottom; (3) stop growing the monophyletic group *immediately before* it fails to be included in the input set of sequences; (4) put the monophyletic group into the output set, and remove all sequences in it from the input set of sequences; (5) repeat (1) — (4) until the input set is emptied.

This verbal description can be translated into the following "pseudo-code."

[**Algorithm to divide sequence set into monophyletic groups**]

Input: A set of sequences (at external nodes) in a phylogenetic tree (= @*Sequences*) ;

A (rooted) phylogenetic tree (= T) ;

Output: (The references to) @*set_mpgs* and @*set_br_up_or_lw*, where

@*set_mpgs* is the set of monophyletic groups ;

@*set_br_up_or_lw* indicates which branch separates each monophyletic group and on which end (, upper or lower,) of the branch the monophyletic group is.

Algorithm:

Create @*set_mpgs* ← ( ) ;

Create @*set_br_up_or_lw* ← ( ) ;

Create $*ct_mpgs* ← 0 ;

Create @*remaining_seqs* ← (a copy of @*Sequences*) ;

While (0 < { size of @*remaining_seqs* }), do the following: (Outer While-loop)

Create $*seed_seq* ← $*remaining_seqs*[0] ;

Create @*curr_mpg* ← ($*seed_seq*) ;

Create $*curr_br* ← { the branch that separates $*seed_seq* from all other sequences } ;

Create $*curr_up_or_lw* ←

{ the side (upper ('U') or lower ('L')) of $*curr_br* that the $*seed_seq* is on } ;

While ($*curr_up_or_lw* ='L'), do the following: (1st Inner While-loop)

If (the upper-bound of $*curr_br* is the root), then:

If (the root is bifurcated), then:

$*curr_br* ← { the sibling of $*curr_br* } ;

$*curr_up_or_lw* ← 'U' ;

last; (Leave the 1st Inner While-loop.)

Else: (If the root has three or more child nodes.)

Create @*siblings* ← { the set of sibling branches of $*curr_br* } ;

Create $*ct_siblings* ← { the size of @*siblings* } ;

For $*i* from 0 to ($*ct_siblings* − 1), do the following:

Create @*cmpl_mpg_sbl* ←

{ the complementary set of the monophyletic group under $*siblings*[$*i*] } ;

If (@*cmpl_mpg_sbl* is equal to or included in @*remaining_seqs*), then:

$*curr_br* ← $*siblings*[$*i*] ;

$*curr_up_or_lw* ← 'U' ;

@*curr_mpg* ← @*cmpl_mpg_sbl* ;

last; (Leave the For-loop.)

EndIf

EndFor ($*i*)

last; (Leave the 1st Inner While-loop.)

EndIf

EndIf

Create $*curr_pa* ← { the parent branch of $*curr_br* } ;

Create @*mpg_pa* ← { the monophyletic group under $*curr_pa* } ;

If (@*mpg_pa* is equal to or included in @*remaining_seqs*), then:

$*curr_br* ← $*curr_pa* ;

@*curr_mpg* ← @*mpg_pa* ;

Else:

last; (Leave the 1st Inner While-loop.)

EndIf

EndWhile (1st Inner)

While ($*curr_up_or_lw* = 'U'), do the following: (2nd Inner While-loop)

If ($*curr_br* is the external branch), then:

last; (Leave the 2nd Inner While-loop.)

EndIf

Create @*children* ← { the set of child-branches of $*curr_br* } ;

Create $*ct_children* ← { the size of @*children* } ;

If ($*ct_children* = 1), then:

$*curr_br* ← $*children*[0] ;

next ;

EndIf

Create $*if_changed* ← 0 ;

For $*j* from 0 to ($*ct_children* − 1), do the following:

Create @*cmpl_mpg_ch* ←

{ the complementary set of the monophyletic group under $*children*[$*i*] } ;

If (@*cmpl_mpg_ch* is equal to or included in @*remaining_seqs*), then:

$*curr_br* ← $*children*[$*i*] ;

@*curr_mpg* ← @*cmpl_mpg_ch* ;

$*if_changed* ← 1 ;

last; (Leave the For-loop.)

EndIf

EndFor ($*j*)

If ($*if_changed* = 0), then:

last; (Leave the 2nd Inner While-loop)

EndIf

EndWhile (2nd Inner)

@{$*set_mpgs*[$*ct_mpgs*]} ← @*curr_mpg* ;

@{$*set_br_up_or_lw*[$*ct_mpgs*]} ← ($*curr_br*, $*curr_up_or_lw*) ;

$*ct_mpgs* ← $*ct_mpgs* + 1 ;

Remove all sequences in @*curr_mpg from* @*remaining_seqs* ;

EndWhile (Outer)

Return (\@*set_mpgs*, \@*set_br_up_or_lw*) .

EndAlgorithm

[For the actual implementation in Perl, see the subroutine, "partition_rowset_into_monophyl_sets", in the module, "MyANEX_MainX.pm", of ANEX.]

I Algorithm to perform "reverse-(i)CII"

This algorithm, which has already been described briefly in 2.5.4, is also simple *verbally*: (1) Cut the input MSA into three sub-MSAs, namely, the sub-MSA on the left, that in the CII-candidate region, and that on the right; (2) from the sub-MSA in the CII-candidate region, create as many sub-sub-MSAs as the monophyletic groups characterizing the CII-candidate, with each sub-sub-MSA consisting only of sequences in one of the monophyletic groups; (3) vertically extend each of the sub-sub-MSAs into the whole set of sequences by padding the remaining sequences with gaps alone; (4) horizontally concatenate the sub-MSA on the left, all the extended sub-sub-MSAs, and the sub-MSA on the right, to create an output MSA.

This verbal description is translated into the following "pseudo-code."

[**Algorithm for "reverse-(i)CII"**]

Input: An MSA within the window, represented as a set of columns

(= @$In\_A = (\@c_0, \@c_1, ..., \@c_{\$N\_clms-1})$,

where $\$N\_clms$ is the number of columns in the MSA);

The left- and right-end coordinates (within the window) of the CII-candidate region

$(= (\$lend\_cii, \$rend\_cii))$;

The set of monophyletic groups characterizing the CII-candidate region $(= @set\_mpgs)$;

A null-column, $@null\_clm$, which contains as many gap-characters as the aligned sequences ;

Output: (The reference to) an output MSA, $@Out\_A$.

Algorithm:

Create $@subA\_left \leftarrow ( )$;

For $\$i1$ from 0 to $(\$lend\_cii - 1)$, do the following:

$@\{\$subA\_left[\$i1]\} \leftarrow$ (a copy of $@\{\$In\_A[\$i1]\}$) ;

EndFor ($\$i1$)

Create $@subA\_in\_cii \leftarrow ( )$;

For $\$i2$ from $\$lend\_cii$ to $\$rend\_cii$, do the following:

$@\{\$subA\_in\_cii[\$i2 - \$lend\_cii]\} \leftarrow$ (a copy of $@\{\$In\_A[\$i2]\}$) ;

EndFor ($\$i2$)

Create $@subA\_right \leftarrow ( )$;

For $\$i3$ from $(\$rend\_cii + 1)$ to $(\$N\_clms - 1)$, do the following:

$@\{\$subA\_right[\$i3 - \$rend\_cii - 1]\} \leftarrow$ (a copy of $@\{\$In\_A[\$i3]\}$) ;

EndFor ($\$i3$)

Create $\$ct\_mpgs \leftarrow \{$ size of $@set\_mpgs \}$;

Create $@cnct\_ext\_subsubA\_in\_cii \leftarrow ( )$; (Concatenated exetnded-subsub-alignments.)

Create $\$ct\_new\_clms \leftarrow 0$;

For $\$g$ from 0 to ($\$ct\_mpgs - 1$), do the following:

Create $@mpg \leftarrow @\{\$set\_mpgs[\$g]\}$;

Create $\$size\_mpg \leftarrow$ { size of $@mpg$ } ;

For $\$i2$ from $\$lend\_cii$ to $\$rend\_cii$, do the following:

Create $@old\_clm \leftarrow @\{\$subA\_in\_cii[\$i2 - \$lend\_cii]\}$;

Create $@new\_clm \leftarrow$ (a copy of $@null\_clm$) ;

For $\$s$ from 0 to ($\$size\_mpg - 1$), do the following:

$\$indx\_seq \leftarrow \$mpg[\$s]$; (Index of the sequence in MSA.)

$\$new\_clm[\$indx\_seq] \leftarrow \$old\_clm[\$indx\_seq]$;

EndFor ($\$s$)

If ($@new\_clm$ is *not* equal to $@null\_clm$), then:

$@\{\$cnct\_ext\_subsubA\_in\_cii[\$ct\_new\_clms]\} \leftarrow @new\_clm$;

$\$ct\_new\_clms \leftarrow \$ct\_new\_clms + 1$;

EndIf

EndFor ($\$i2$)

EndFor ($\$g$)

Create $@Out\_A \leftarrow (@subA\_left, @cnct\_ext\_subsubA\_in\_cii, @subA\_right)$;

Return $\backslash @Out\_A$.

EndAlgorithm

The alignment output by this algorithm will go through some post-processing.
[For the actual implementation in Perl, see the subroutines, "reverse_cii_on_clmset" and "reverse_spec_cii_in_wd", in the module, "MyANEX_MainX.pm", of ANEX.]

J Non-factorability of effects of "shift"s of multiple gap-blocks on substitution component of MSA probability

Here we attempt to factorize the effects of "shift"s of multiple gap-blocks on the substitution component of the MSA probability, and clarify when the effects are factorable and when they are not. Let $\Delta_{\text{sgl-shift }(gb_k)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}$ be the increment of the logarithm of the substitution component of the probability of the MSA (denoted as $\mathcal{A}$) by a single-"shift" of a gap-block (gb_k, with $k = 1$ or 2). And let $\Delta_{\text{dbl-shifts }(gb_1,gb_2)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}$ be the increment by simultaneous "shift"s of the gap-blocks, gb_1 and gb_2. *Symbolically*, what we want to see is expressed as:

$$\begin{aligned} & \Delta_{\text{dbl-shifts }(gb_1,gb_2)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\} \\ \stackrel{?}{=} \; & \Delta_{\text{sgl-shift }(gb_1)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\} + \Delta_{\text{sgl-shift }(gb_2)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\} \ . \end{aligned} \tag{8}$$

To discuss further, we consider three typical cases involving two *non-interfering* gap-blocks (panels A, B and C of Figure 17). First, when their "shift"s only affect different columns independently (as in Figure 17 A), it is *almost trivial* to show that the change in the log substitution probability by such two simultaneous shifts is exactly the summation of two changes, each of which resulted from a single individual "shift"; thus, in this case,

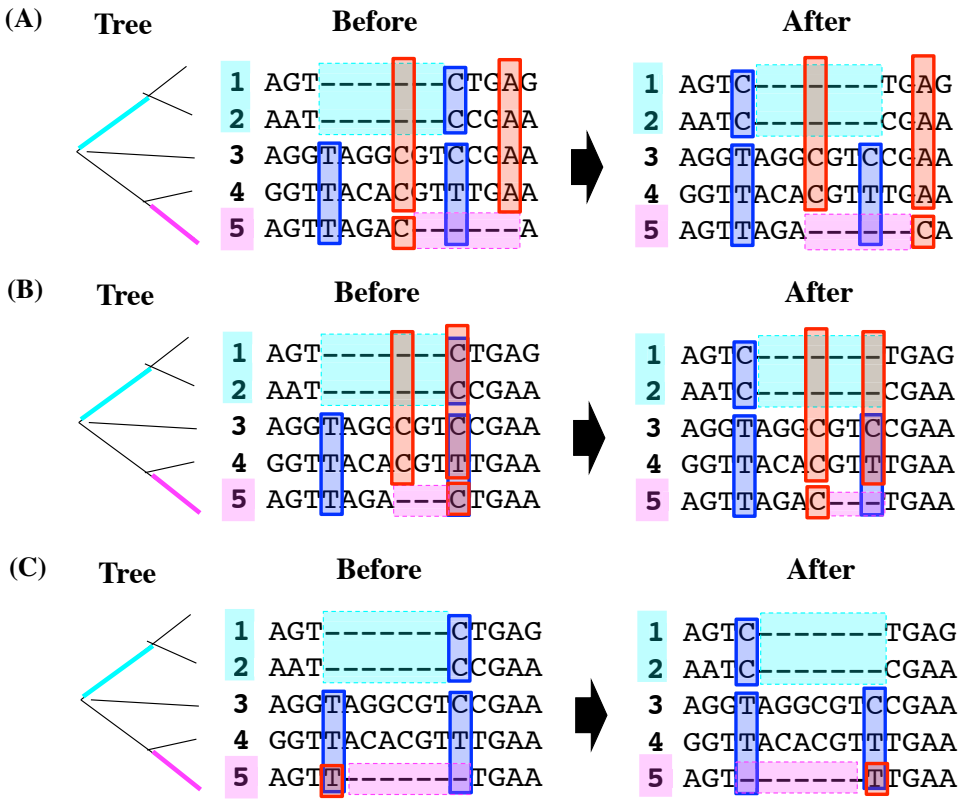


Figure 17: **Effects of simultaneous "shift"s of two non-interfering gap-blocks on substitution component of MSA probability. A.** The "shift"s affect different columns independently. **B.** The "shift"s affect the same single column simultaneously. **C.** The "shift"s affect the same *pair* of columns simultaneously. In each column, "shift"s of the gap-blocks shaded in cyan and magenta affect the portions of the columns shaded in blue and red, respectively. The colored branches in the tree phylogenetically delimit the gap-blocks shaded in the same color.

Eq. 17 holds *exactly*. (Actually, the gap-blocks need *not* necessarily be *non-interfering* in situations like this.)

Thus, we can *focus on* cases where the same column (or the same pair of columns) is affected *simultaneously* by the "shift"s of two *non-interfering* gap-blocks (as in Figure 17, panels B and C). By carefully examining the cases as in panels B and C of Figure 17, we find that the *effects* of the "shift"s of two *non-interfering* gap-blocks can be regarded as *independent* of each other (i.e., Eq. 17 is satisfied) **if the following equation holds** :

$$\begin{aligned} & \log\{P[(c', c''', c'') \mid \mathcal{T}, \Theta_S]\} + \log\{P[(-, c''', -) \mid \mathcal{T}, \Theta_S]\} \\ \stackrel{?}{=} \; & \log\{P[(c', c''', -) \mid \mathcal{T}, \Theta_S]\} + \log\{P[(-, c''', c'') \mid \mathcal{T}, \Theta_S]\} \; . \end{aligned} \tag{9}$$

(The proof is given in J.1 below.) Here, the c' and c'' are the portions of MSA columns flanking (the front-ends of) the two gap-blocks in question, and the c''' is the remaining portion of the column; the (c', c''', c'') denotes the MSA column created by aligning these portions; the $(c', c''', -)$ denotes the column made from the (c', c''', c'') by replacing the residues in c'' with gaps; other similar symbols can be interpreted accordingly. Unfortunately, our further examination revealed that **Eq. 9 should *fail* to hold *in general***; however, it may *approximately* hold if *at least one* of the involved branches is so long that the transition probabilities nearly saturate. (For details, see J.2 below.)

J.1 Proof of necessary and sufficient condition, Eq. 9, for independent effects of overlapping "shift"s of *non-interfering* gaps

In this sub-appendix, we prove that Eq. 9 is indeed necessary and sufficient for the independent effects of the "shift"s of *non-interfering* gap-blocks. For this purpose, we here introduce a symbol, $\langle ... \rangle$, which is a shorthand notation of $\log\{P[(...) \mid \mathcal{T}, \Theta_S]\}$. (For example, $\langle c', c''', c'' \rangle$ denotes $\log\{P[(c', c''', c'') \mid \mathcal{T}, \Theta_S]\}$.)

We first consider the situation illustrated in Figure 17 B. We here use the following symbols for the portions of the columns:

- The c' denotes the shifted portion corresponding to the 'CC' (vertical) occupying sequences 1 and 2;
- The c'' denotes the shifted portion corresponding to the 'C' occupying sequence 5;
- The c'''_1 denotes the unmoved portion corresponding to the 'TTT' (vertical) occupying sequences 3, 4 and 5;
- The c'''_2 denotes the unmoved portion corresponding to the '- -CC' (vertical) occupying sequences 1, 2, 3 and 4; and
- The c'''_3 denotes the unmoved portion corresponding to the 'CT' (vertical) occupying sequences 3 and 4.

We *assign* the gap-block numbers so that c' and c'' flank gb_1 and gb_2, respectively.

Remember the column-wise factorability of the substitution component (Eq. 2). Thanks to this, when comparing the substitution component of the MSA after the simultaneous "shift"s with that before the simultaneous "shift"s, it is sufficient to compare the contributions from the columns affected by the "shift"s, *i.e.*, the columns shaded in blue and/or red in Figure 17. Then, the difference of the substitution component after the double-"shift"s from that before the double-"shift"s is expressed as:

$$\begin{aligned} & \Delta_{\text{dbl-shifts }(gb_1,gb_2)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ B}} \\ = \; & \{\langle c', c'''_1\rangle + \langle c'''_2, c''\rangle + \langle -, c'''_3, -\rangle\} - \{\langle -, c'''_1\rangle + \langle c'''_2, -\rangle + \langle c', c'''_3, c''\rangle\} \; . \end{aligned} \tag{10}$$

On the other hand, when only each of the two "shift"ed portions (i.e., c' and c'') is "shift"ed from the MSA before the double-"shift"s, we have two possible differences:

$$\Delta_{\text{sgl-shift }(gb_1)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ B}} = \{\langle c', c'''_1\rangle + \langle -, c'''_3, c''\rangle\} - \{\langle -, c'''_1\rangle + \langle c', c'''_3, c''\rangle\} \tag{11}$$

$$\Delta_{\text{sgl-shift }(gb_2)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ B}} = \{\langle c'''_2, c''\rangle + \langle c', c'''_3, -\rangle\} - \{\langle c'''_2, -\rangle + \langle c', c'''_3, c''\rangle\} \tag{12}$$

The independence of the effects of the "shift"s means that Eq. 10 is equal to the summation of Eq. 11 and Eq. 12. Thus, we must have:

$$\begin{aligned} 0 &= \Delta_{\text{dbl-shifts } (gb_1,gb_2)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ B}} \\ &\quad -\Big\{\Delta_{\text{sgl-shift } (gb_1)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ B}} + \Delta_{\text{sgl-shift } (gb_2)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ B}}\Big\} \\ &= \langle -, c_3''', - \rangle + \langle c', c_3''', c'' \rangle - \langle -, c_3''', c'' \rangle - \langle c', c_3''', - \rangle \ . \end{aligned} \tag{13}$$

This is nothing other than (the affirmation part of) Eq. 9, if we *re-name* c_3''' as c'''.

Next we consider the situation illustrated in Figure 17 C. To represent the portions of the columns, we use the same symbols as above, but associate them with slightly different meanings:

- The c' denotes the shifted portion corresponding to the 'CC' (vertical) occupying sequences 1 and 2;
- The c'' denotes the shifted portion corresponding to the 'T' occupying sequence 5;
- The c_1''' denotes the unmoved portion corresponding to the 'TT' (vertical) occupying sequences 3 and 4;
- The c_2''' denotes the unmoved portion corresponding to the 'CT' (vertical) occupying sequences 3 and 4.

In this case, the difference of the substitution component after the double-"shift"s from that before the double-"shift"s is expressed as:

$$\begin{aligned} &\Delta_{\text{dbl-shifts } (gb_1,gb_2)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ C}} \\ = \ &\{\langle c', c_1''', - \rangle + \langle -, c_3''', c'' \rangle\} - \{\langle -, c_1''', c'' \rangle + \langle c', c_3''', - \rangle\} \ . \end{aligned} \tag{14}$$

The effects of the single-"shift"s are calculated as:

$$\Delta_{\text{sgl-shift } (gb_1)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ C}} = \{\langle c', c_1''', c'' \rangle + \langle -, c_3''', - \rangle\} - \{\langle -, c_1''', c'' \rangle + \langle c', c_3''', - \rangle\} \tag{15}$$

$$\Delta_{\text{sgl-shift } (gb_2)} \log\{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\}|_{17\text{ C}} = \{\langle -, c_1''', - \rangle + \langle c', c_3''', c'' \rangle\} - \{\langle -, c_1''', c'' \rangle + \langle c', c_3''', - \rangle\} \tag{16}$$

Using Eqs. 14, 15, and 16, the independence of the effects of the shifts can be expressed as:

$$\begin{aligned} 0 &= \Delta_{\text{dbl-shifts } (gb_1, gb_2)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\} \mid_{17\text{ C}} \\ &\quad - \Big\{ \Delta_{\text{sgl-shift } (gb_1)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\} \mid_{17\text{ C}} + \Delta_{\text{sgl-shift } (gb_2)} \log \{P[\mathcal{A} \mid \mathcal{T}, \Theta_S]\} \mid_{17\text{ C}} \Big\} \\ &= \Big\{ \langle c', c_1''', - \rangle + \langle -, c_1''', c'' \rangle - \langle c', c_1''', c'' \rangle - \langle -, c_1''', - \rangle \Big\} \\ &\quad + \Big\{ \langle -, c_3''', c'' \rangle + \langle c', c_3''', - \rangle - \langle -, c_3''', - \rangle - \langle c', c_3''', c'' \rangle \Big\} \ . \end{aligned} \tag{17}$$

The expression in each pair of braces on the right-hand-side of Eq. 17 vanishes if the affirmative part of Eq. 9 holds. Thus, in conjunction with Eq. 13, Eq. 17 demonstrates that the effects of horizontally overlapping "shift"s of two *non-interfering* gap-blocks are independent of each other if Eq. 9 is affirmed. And the arguments in this subsection also indicates that the affirmation of Eq. 9 is actually the necessary and sufficient condition for the independence of the effects of horizontally overlapping "shift"s.

J.2 Non-factorability of change in substitution component into contributions from "shift"s of *non-interfering* gap-blocks

In J.1 above, we proved that (the affirmative part of) Eq. 9 is the necessary and sufficient condition for the independence of the effects of horizontally overlapping "shift"s. In order to examine under what conditions Eq. 9 holds, let us consider a situation illustrated in Figure 18.

To do this, we need to prepare some setting on the probabilities of residue patterns in the portions of a column (denoted here as c) (in Figure 18). First, let Ω be an "alphabet," or a set of possible residue (or base) states, and ω, ω' and ω_k ($k = 1, ..., 8$) denote some specific residue states in Ω. Next, c' and c'' denote the portions of the column (c) delimited (from above) by branches b' and b'', respectively. And c''' denotes the remaining portion of c, and branches b_i''' ($i = 1, 2, 3$) play important roles in defining c''' here. Then, the $P(\omega \mapsto \omega'; b_1''')$ denotes the conditional probability that, given ω ($\in \Omega$) at the upper-end of branch b_1''', we

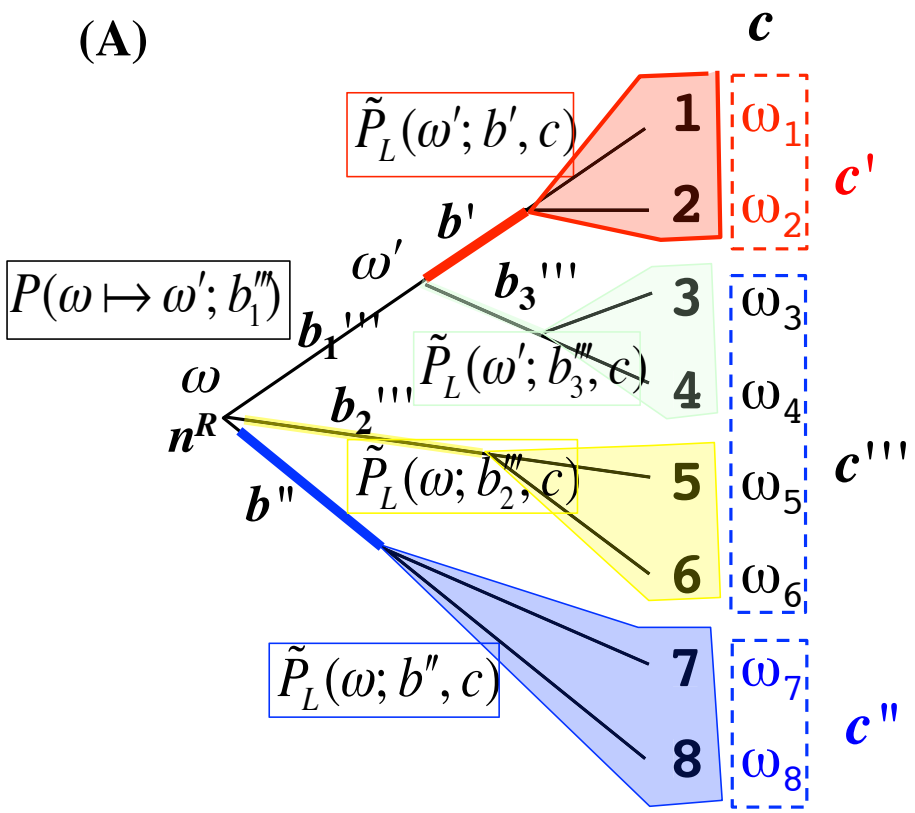


Figure 18: **Model situation to examine whether effects of "shift"s of two *non-interfering* gap-blocks are (nearly) independent or not.** Each portion of the tree (shaded in respective color) is assigned a building-block probability (enclosed by a rectangle of the same color). The column (c) was vertically divided into three parts: c', c'' and c'''. Branches b' and b'' delimit c' and c'', respectively. And branches b_i''' ($i = 1, 2, 3$) are important factors determining c'''. The ω and ω' at both ends of branch b_1''' are the residue states over which the probabilities are summed. For the explanations of $P(...)$ and $\tilde{P}(...)$, see text.

have ω' ($\in \Omega$) at its lower-end. And the $\tilde{P}_L(\omega; b'', c)$, for example, denotes the conditional probability that, given ω ($\in \Omega$) at the upper-end of branch b'', we observe the residues that are in all the extant sequences under b'' and in column c. [81]

Now that the setting is prepared, we introduce a short-hand notation, $P(\omega, \omega'; c''')$, denoting the joint probability that we have ω and ω' at the upper-ends of the branches delimiting c'' and c', respectively, and that we also have the residue configuration in c'''. Under the situation in Figure 18, it is expressed as:

$$P(\omega, \omega'; c''') = P(\omega; n^R)\ P(\omega \mapsto \omega'; b'''_1)\ \tilde{P}_L(\omega; b'''_2, c)\ \tilde{P}_L(\omega'; b'''_3, c)\ , \tag{18}$$

where $P(\omega; n^R)$ is the probability that the residue state at the root (n^R) is ω. Using this probability, we can calculate the probability $P[(c', c''', c'') \mid \mathcal{T}, \Theta_S]$ as:

$$P[(c', c''', c'') \mid \mathcal{T}, \Theta_S] = \sum_{\omega\ \in\Omega} \sum_{\omega'\ \in\Omega} \Big\{ P(\omega, \omega'; c''')\ \tilde{P}_L(\omega; b'', c)\ \tilde{P}_L(\omega'; b', c) \Big\}\ . \tag{19}$$

If, for example, c'' is occupied solely with gaps, we have $\tilde{P}_L(\omega\ ; b'', c) = 1$ for every ω ($\in \Omega$). This and similar facts yield the following:

$$P[(-, c''', c'') \mid \mathcal{T}, \Theta_S] \ = \ \sum_{\omega\ \in\Omega} \Big\{ P(\omega,\ \cdot\ ; c''')\ \tilde{P}_L(\omega; b'', c) \Big\}\ , \tag{20}$$

$$P[(c', c''', -) \mid \mathcal{T}, \Theta_S] \ = \ \sum_{\omega'\ \in\Omega} \Big\{ P(\ \cdot\ , \omega'; c''')\ \tilde{P}_L(\omega'; b', c) \Big\}\ , \tag{21}$$

$$P[(-, c''', -) \mid \mathcal{T}, \Theta_S] \ = \ P(\ \cdot\ ,\ \cdot\ ; c''')\ . \tag{22}$$

Here, we introduced the notations:

$$P(\omega,\ \cdot\ ; c''') \ \stackrel{\text{def}}{=}\ \sum_{\omega'\ \in\Omega} P(\omega, \omega'; c''')\ , \tag{23}$$

$$P(\ \cdot\ , \omega'; c''') \ \stackrel{\text{def}}{=}\ \sum_{\omega\ \in\Omega} P(\omega, \omega'; c''')\ , \tag{24}$$

$$P(\ \cdot\ ,\ \cdot\ ; c''') \ \stackrel{\text{def}}{=}\ \sum_{\omega\ \in\Omega} \sum_{\omega'\ \in\Omega} P(\omega, \omega'; c''')\ . \tag{25}$$

[81] More detailed definitions and descriptions of these probabilities, as well as a pair of fast algorithms to compute them, are given in [82].

Thus, **for (the affirmative part of) Eq. 9 to hold as an *exact* equation for any residue configurations of c' and c'',** (and thus for any probability vectors $\{\tilde{P}_L(\omega; b'', c)\}_{\omega \in \Omega}$ and $\{\tilde{P}_L(\omega'; b', c)\}_{\omega' \in \Omega}$,) **the probabilities $\{P(\omega, \omega'; c''')\}_{\omega,\omega' \in \Omega}$ *must* satisfy the following equation:**

$$P(\omega, \omega'; c''') = \frac{P(\omega, \ \cdot \ ; c''') \ P(\ \cdot \ , \omega'; c''')}{P(\ \cdot \ , \ \cdot \ ; c''')} \quad for \quad {}^{\forall}(\omega, \ \omega') \ \in \Omega^2 \ . \tag{26}$$

In other words, their ω-dependence and ω'-dependence *must* decouple. In the situation at hand, where $\{P(\omega, \omega'; c''')\}_{\omega,\omega' \in \Omega}$ is given by Eq. 18, the condition *approximately* holds when branch b'''_1 is so long that $P(\omega \mapsto \omega'; b'''_1)$ nearly saturates. Or, *even* if Eq.26 does *not* hold even approximately, Eq. 9 could still be satisfied *provided that* either b' or b'' is quite long (and thus either $\{\tilde{P}_L(\omega'; b', c)\}_{\omega' \in \Omega}$ or $\{\tilde{P}_L(\omega; b'', c)\}_{\omega \in \Omega}$ is nearly saturated).

If, however, none of the branches b', b'' and b''' are quite long, it is unlikely that 9 should hold even approximately. **Thus, in general, it would be safe *not* to *assume* (the affirmative part of) the equation, Eq. 9, *even approximately.***

K *Nearly discrete* dependence of indel components of MSA probability on gap-block configurations

In sub-subsection 2.6.3, we took advantage of the "**good property**" exhibited by the indel component of the MSA probability, which is: (provided that the indel rates are space-homogeneous (*i.e.*, uniform along each sequence) *at least* within each window,) the indel component remains (nearly) unchanged *if* the ***topology*** of the (horizontal) positional relationships does *not* change between the *interfering* gap-blocks *and if* the bulk/ boundary-status of all gap-blocks does *not* change.

In the above statement, the dependence on the bulk/boundary-status is somewhat trivial. Hence, in this section, we will demonstrate, or *prove*, the good property regarding the topology of positional relationships between the (*interfering*) gap-blocks.

If the indel evolution model satisfies the "sufficient and nearly necessary set of conditions," the indel component is factorable into a product of the overall contribution (determined by the "absence"/"presence" state of the root sequence) and multiplication factors contributed by gapped segments (see [1], which is a generalization/modification of [76]). Therefore, we can focus on the change in the multiplication factors *from* the gapped segments *affected by* the move in question (and possibly also the change in the overall factor). Especially, remember that the indel rates are now *assumed* to be space-homogeneous (i.e., uniform along the sequence) at least within the MSA portion in question. Then, **(1)** *as long as* the *affected* gap-block is *isolated* from all others, the multiplication factor will remain *virtually* unchanged under its shift. (See **K.1** for a proof.) Besides, **(2)** under a parameter setting where the "first-**plus** approximation" (by *all* indel histories with *parsimonious* ancestral states) is *quite good*, the detailed (site-level) positional relationships between the *non-isolated* gap-blocks do *not* substantially influence the multiplication factor, as long as their **topologies** (and the sizes of the parsimonious indels) remain unchanged. (See **K.2** for a demonstration using some typical examples.)

These facts yield the aforementioned "good property." Thus, proving (or demonstrating) the "good property" boils down to proving (or demonstrating) the "facts" **(1)** and **(2)** above, which will be done in K.1 and K.2 below.

K.1 Proof that indel multiplication factor(s) of is(are) *virtually* independent of relative positions between *isolated* gap-blocks

In a previous work of ours [1], we proved that the alignment probabilities are factorable into the overall factor and the contributions from gapped segments (*i.e.*, local MSAs), *provided* that the *genuine* evolution model satisfies the "sufficient and nearly necessary set of conditions." ANEX's computation of indel components is based on an evolution model satisfying such conditions. Thus it is obvious that the indel component is independent of the rela-

tive horizontal positions *as long as* the gap-blocks in question are mutually separated by at least a gapless column (as in Figure 2 B). [82] We can therefore *focus on* the cases where the *isolated* gap-blocks overlap *horizontally* (as in Figure 2 C and Figure 24 A (below)). It should be recalled here that, *by definition*, horizontally overlapping yet *isolated* gap-blocks (or, more precisely, the clades (*i.e.* set of phylogenetically clustered sequences) supporting them) must be separated from each other by *at least* three branches (see appendix A). This means that there are *at least* two intervening nodes, each of which has at least one descendant sequence with residues aligned with the gap-blocks (*e.g.*, Figure 24 A). Then, thanks to the **phylogenetic correctness condition** that the ancestral sequence states must satisfy [129, 130], it follows that, in each column, the two gap-blocks must always be separated *phylogenetically* by at least two nodes whose gap-aligned positions are stuffed with residues (e.g., Figure 24 B). This in turn means that the indel history that *resulted in* **one gap-block** *must always* be *confined* in a subtree separated from that *confining* **the other gap-block**, and they must *never* interact with each other to form a larger indel history (Figure 24 C), [83] *as long as no* null columns (*i.e.*, gap-only columns) are brought in.

Incidentally, *if* some null columns are brought in, there could be some non-parsimonious indel histories that connect the isolated gap-blocks (Figure 25 (below)), However, such indel histories always require *at least* two more indels that are *exquisitely* coordinated in terms of both the horizontal positions and the indel sizes. Such histories are quite similar (in the exquisite coordination) to the non-parsimonious indel histories that can yield case (i) gapped segments in PWAs [2], and thus, *in general*, their contributions are expected to be

[82] This is true *even if* the gap-blocks are *vertically* interfering with (*i.e.*, non-isolated from) each other.

[83] In fact, these facts originally motivated the aforementioned definition of the isolated gap-blocks.

negligible as well. [84]

Getting back to the main topic, the multiplication factor contributed from each gapped segment (*i.e.*, local MSA) is a summation of terms over the sets of ancestral gap-configurations (Eq. 28 in K.3.1). And each term is a contribution from indel histories with a fixed set of ancestral gap-configurations, which could be expressed as a product of factors over the branches (Eq. 31 in K.3.1). Now, *because*, in *dominantly* contributing ancestral state sets, each of the (at least two) nodes separating the *isolated* gap-blocks has a *virtually* fixed gap-configuration (with all sites occupied by residues), the space of sets of ancestral gap-configurations could be *approximately* expressed as a direct product of *at least* two spaces of *partial* state sets, one containing the clade for one gap-block and another containing the clade for the other gap-block (Eq. 35 in K.3.1; Figure 24 C). **Therefore, in this case, the multiplication factor can be further factorized into the product of contributions: one from the portion of the tree with (*virtually*) "fixed" ancestral states, and the others from the isolated gap-blocks.** (See Eqs. 37 & 38 in K.3.1, and also Figure 24 D.) The independent horizontal re-locations of these gap-blocks do *not* change these factors (as long as they interfere with no neighboring gap-block(s)). Thus, *any* change in the relative (horizontal) position between the *isolated* gap-blocks has *virtually* **no impact** on

[84] Let gb_1 and gb_2 be the gap-blocks in question, and let b_3 and b_4 be the branches immediately descended from the ancestral nodes separating gb_1 and gb_2 (see, *e.g.*, Figure 24 A). Then, the ratio of total contributions from such (connecting) histories to the contributions from separated histories is *roughly* estimated as:

$$R(C/S) = \frac{\{ \text{ total contribution (connected) } \}}{\{ \text{ total contribution (separate) } \}} < L(gb_1 \cap_h gb_2) \cdot |b_3| \cdot |b_4| \cdot (\lambda_D)^2,$$

where the $L(gb_1 \cap_h gb_2)$ is the number of (non-null) columns in the horizontal overlap between gb_1 and gb_2, the $|b_i|$ (with $i = 3, 4$) is the length of branch b_i (measured in the expected number of substitutions per site), and the λ_D is the total deletion rate. In normal sequence studies, we usually have $|b_i| < 1$ and $\lambda_D \leq 0.1$. Thus, we have $R(C/S) < 0.01 \times L(gb_1 \cap_h gb_2)$. This means that the contributions from the connected histories are negligible as long as the horizontal overlap between the *isolated* gap-blocks is substantially less than 100 sites(, say, a few dozens sites or less).

the segment-wise multiplication factor for the indel component of the MSA probability.

See **K.3.1** (below) for the mathematical details.

K.2 Demonstration that indel multiplication factor depends *only weakly* on detailed (site-level, topology-preserving) positional relationships between *non-isolated* gap-blocks

We think it too premature to prove it *generally*, because the related concepts have not adequately been developed yet. Instead, we here consider a few *typical* example cases and demonstrate what the section title claims.

We already proved in K.1 that the indel multiplication factor is independent of the relative positional relationships between *isolated* (*i.e.*, *non-interfering*) gap-blocks. Thus, in this section, we will focus only on the relative relationships between *non-isolated* (*i.e.*, *interfering*) gap-blocks. For this purpose, it should be sufficient to consider the gap states of (ancestral) sequences connected with a 3-OTU [85] tree (as in Figure 19). We consider three typical patterns: **(A)** two horizontally nested runs of gaps (Figure 19 A); **(B)** two horizontally overlapping yet non-nested runs of gaps (Figure 19 B); and **(C)** two vertically complementary gap-blocks that horizontally adjoin each other (Figure 19 C). As will be explained below, the patterns A and B are topologically different [86]; pattern A is topologically equivalent to the pattern in Figure 19 D, but pattern B is not. And pattern C is topologically different from the pattern in Figure 19 E, where the gap-blocks are separated via a gapless segment. (In the following couple of paragraphs, we will see specifically what being "topologically different/equivalent" means.)

Pattern A can be created by two parsimonious indel histories (panels A and B of Figure 20), as well as by numerous non-parsimonious ones including some next-to-parsimonious

[85] The "OTU" is short for "operational taxonomical unit."

[86] For the definition of the term, "topology", used here, see appendix A.

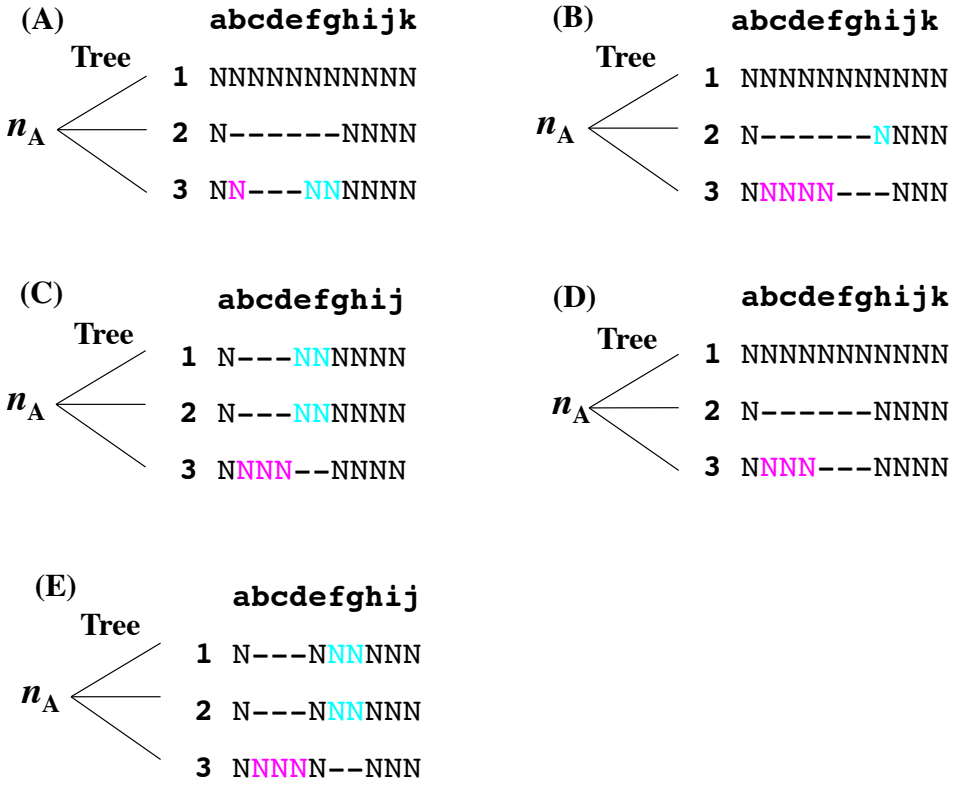


Figure 19: **Typical examples of gap-configurations of 3 sequences connected via 3-OTU tree. A.** A short gap in the 3rd OTU (labeled "3") is horizontally nested in a long gap in the 2nd OTU ("2"). **B.** Gaps in two OTUs horizontally overlap each other in a non-nested manner. **C.** Two *vertically* complementary gap-blocks that are horizontally adjoining each other. **D.** This pattern is considered as topologically the same as in panel A but not as in panel B, because of the (parsimonious) indel histories that can create the patterns. **E.** This pattern is considered as topologically different from that in panel C. In each panel, the "n_A" denotes the "most recent common ancestor (MRCA)" of the three OTUs, and the lower-case letters above the alignment represent the ancestry indexes of the sites. In each panel, some gap-aligned residues are colored cyan or magenta, to clarify the positional relationships between gap-blocks (or runs of gaps) in each MSA.

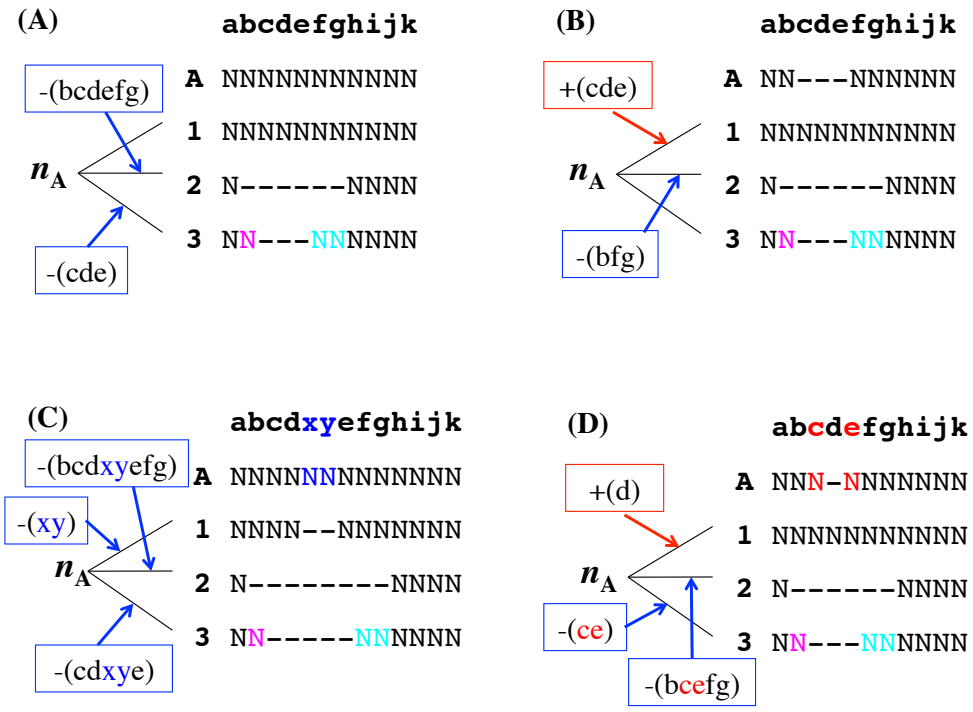


Figure 20: **Indel histories that can create pattern A in Figure 19.** Panels **A** and **B** show the parsimonious histories. Panels **C** and **D** show some next-to-parsimonious histories. The 'A' on the top-left corner of the alignment indicates the ancestral sequence (at node n_A). The "-(abc)" in a blue rectangle indicates that the sites a, b and c were deleted along the branch it points; the "+(def)" in a red rectangle indicates that the sites d, e and f were inserted. Some residues and ancestries are colored in order to facilitate the comparisons among the histories. [**NOTE:** Here we omitted *all* next-to-parsimonious histories in each of which two or more indels occur along a single branch, because they are naturally incorporated in the "1st-**plus** approximation."]

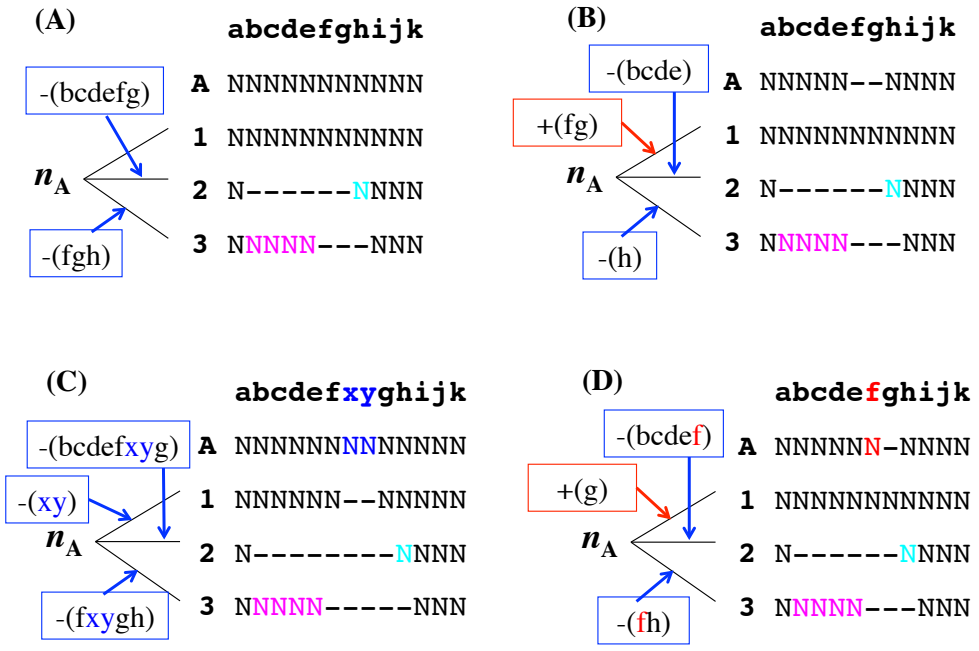


Figure 21: **Indel histories that can create pattern B in Figure 19.** Panel **A** shows the parsimonious indel history. Panels **B**, **C** and **D** show some next-to-parsimonious histories. Notations are the same as those for Figure 20. [**NOTE:** Here, again, we omitted the next-to-parsimonious histories in each of which two indels occur along a single branch.] [**NOTE2:** Panel B is actually the history created by concatenating the column-wise Dollo parsimonious indel histories [83, 2].]

ones (*e.g.*, panels C and D of Figure 20). In contrast, pattern B can be created by only one parsimonious indel history (panel A of Figure 21), as well as by numerous non-parsimonious ones including some next-to-parsimonious ones (*e.g.*, panels B-D of Figure 21). The indel histories that can create the pattern in panel D are nearly the same as those in Figure 20. The histories show the *same* trajectories of the **sequence lengths** and also the *same* series of **insertions** and **deletions** with ***particular* sizes**; they *differ only* in the detailed (*i.e.*, site-level) *horizontal* positions of indels. As can be seen from [1], *when the indel rates are space-homogeneous*, the portion of the multiplication factor *contributed* by each indel history is determined *solely* by the trajectory of the **sequence lengths** and the series (actually the set) of **insertions and deletions with particular sizes**, but it is *independent* of *detailed* horizontal positions of indels. (See, *e.g.*, Eq.(SM-2.7) and Eq.(R8-1.4) in [1].) Therefore, *as long as the parsimonious indel histories (or, more precisely, **all** the indel histories with parsimonious ancestral state sets) contribute predominantly to the multiplication factor*, the multiplication factor from pattern A is nearly equal to that from the pattern in Figure 19 D, but the contributions from patterns A and B will differ considerably from each other.

Patten C can be created by *three types of* parsimonious histories (panels A, B and C of Figure 22), as well as by some non-parsimonious histories (*e.g.*, panel D of Figure 22). In contrast, the pattern in Figure 19 E can be created by two parsimonious histories (panels A and B of Figure 23), as well as by some non-parsimonious histories (e.g., panel C of Figure 23). Therefore, pattern C should be *substantially* more likely to occur than the pattern in Figure 19 E (*provided that* the number of intervening gapless columns is fixed). On the other hand, the occurrence probability should remain unchanged *even if* a different (nonzero) number of gapless columns separate the two gap-blocks in Figure 19 E (*provided that* the entire region encompassing the two gap-blocks has a *fixed* length).

These examples demonstrate that, *at least as long as the contributions by parsimonious indel histories (or, more precisely, by **all** indel histories with parsimonious ancestral state*

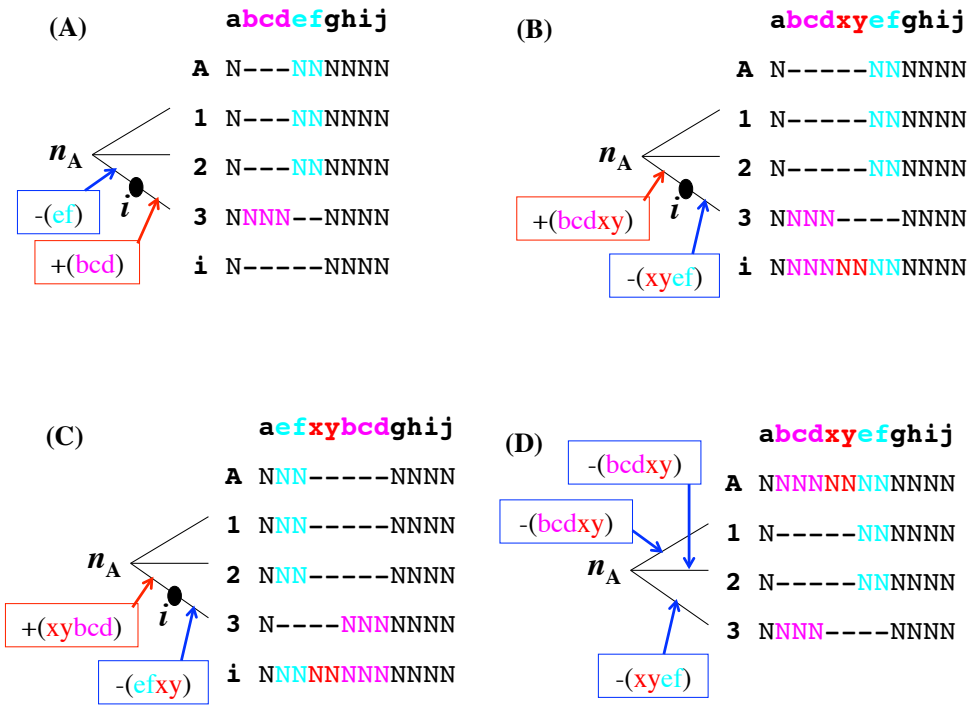


Figure 22: **Indel histories that can create pattern C in Figure 19.** Panels **A**, **B**, and **C** show the three types of parsimonious indel histories. Panel **D** shows a next-to-parsimonious indel history, which was derived from panel B via a "branching" operation [2]. (Note that a different history could be derived from panel C in a similar way.) The notations are basically the same as in Figure 20. In addition, the sequence labelled "i" at the bottom of each MSA (in panels A, B and C) is the "intermediate" state at the point marked with a solid circle. [**NOTE:** The MSA of the extant sequences in panel C is equivalent to that in panel A (and in panel B).] [**NOTE2:** We omitted next-to-parsimonious histories in each of which 3 indels occur along a single branch.]

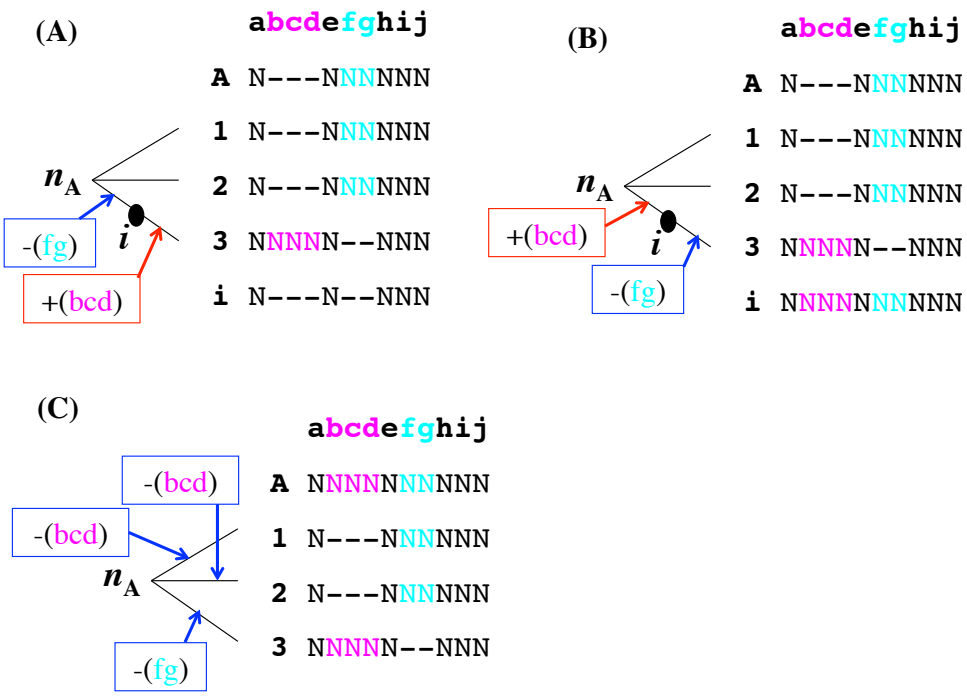


Figure 23: **Indel histories that can create pattern in Figure 19 E.** Panels **A** and **B** show the two parsimonious histories. Panel **C** shows a next-to-parsimonious history. (It was obtained by applying a "branching" operation [2] to panel B.) The notations are basically the same as in Figure 20. **[NOTE:** We omitted those next-to-parsimonious histories each of which requires more than 2 indels along a single branch.]

sets) predominate, **the multiplication factor is determined *in most part* by the topologies of (horizontal) positional relationships between gap-blocks**, but depends *only weakly* on the fine-grained (*i.e.*, site-level, topology-preserving) positional relationships between them.

K.3 Some mathematical details on vertical (*i.e.*, phylogenetic) "factorization" of indel multiplication factors

[**NOTE:** The following contents are quite mathematical *yet* indispensable for completing the theoretical aspects of ANEX. Although we are aware of many frowning faces out there, we nevertheless provide these crucial contents here, hoping that some diligent, serious and sincere researchers will further develop the theories, and methods *beyond* ANEX, in the (hopefully near) future.]

K.3.1 Practical factorability of indel multiplication factor into contributions from *isolated* gap masses

For this purpose, it would be more convenient to use the ancestral-state-based equations derived in section SM-4 of [1], rather than to use the indel-history-based equations derived in subsection 4.2 of [131]. Thus, we begin by recalling the key results of the former.

Provided that the indel model satisfies the conditions (i), (ii) and (iii) (see R6 and R7 of [1]), the probability, $P[\alpha[s_1, s_2, ..., s_{N^X}] \mid \mathcal{T}]$, that a given MSA $(\alpha[s_1, s_2, ..., s_{N^X}])$ (of sequences, $s_1, s_2, ..., s_{N^X}$) result from an indel process along a given tree ($\mathcal{T}$) is factorized as in Eq.(SM-4.20) of [1]:

$$P[\alpha[s_1, s_2, ..., s_{N^X}] \mid \mathcal{T}] = P_0[s_0^{Root} \mid \mathcal{T}] \prod_{K=1}^{K_{\max}} \tilde{\breve{\mathrm{M}}}_P \Big[\alpha[s_1, s_2, ..., s_{N^X}];\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \ . \quad (27)$$

Here, the s_0^{Root} denotes a "reference" root sequence state that is consistent with the MSA, and the $P_0[s_0^{Root} \mid \mathcal{T}]$ is the probability that the sequence state remained s_0^{Root} all across

the tree (see Eq.(SM-4.21) of [1] for details). And the $\tilde{\breve{\mathrm{M}}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ s_0^{Root};\ C_K \mid \mathcal{T}\Big]$ is the multiplication factor contributed from a local region (C_K). It's specific expression is given by Eq.(SM-4.22) of [1]:

$$\begin{aligned}
&\tilde{\breve{\mathrm{M}}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \\
=& \sum_{\substack{\Big\{s(n)-s_0^{Root}\Big\}_{\mathrm{N}^{IN}}[C_K] \\ \in \Delta_\Sigma\Big[C_K;\ s_0^{Root};\ \alpha[s_1,s_2,...,s_{N^X}];\ \{n\in\mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}\Big]}} \breve{\mathrm{M}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ C_K \mid \mathcal{T}\Big]\ .
\end{aligned} \tag{28}$$

Here, the $\Delta_\Sigma[C_K;\ s_0^{Root};\ \alpha[s_1,s_2,...,s_{N^X}];\ \{n\in\mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}]$ is the space of the sets (each denoted as $\{s(n)-s_0^{Root}\}_{\mathrm{N}^{IN}}[C_K]$ in the range of the summation), each over all internal nodes ($n\in\mathrm{N}^{IN}$), of deviations of ancestral states ($s(n)$) from the "reference" (s_0^{Root}) within C_K. And the summand is given by Eq.(SM-4.18) of [1]:

$$\begin{aligned}
&\breve{\mathrm{M}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \\
\overset{\text{def}}{=}\ & \mathrm{M}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ C_K \mid \mathcal{T}\Big] \times \mu_P[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ C_K] \times \\
& \times \exp\Big\{-\sum_{b\in\{b\}_\mathcal{T}} \int_{t(n^A(b))}^{t(n^D(b))} d\tau\, \delta R_X^{ID}(s^A(b),\ s_0^{Root},\ \tau)[C_K]\Big\}\ ,
\end{aligned} \tag{29}$$

with Eq.(SM-4.13) of [1]:

$$\begin{aligned}
&\mathrm{M}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ C_K \mid \mathcal{T}\Big] \\
\overset{\text{def}}{=}\ & \prod_{b\in\{b\}_\mathcal{T}} \Big\{ \prod_{\gamma_{\kappa_b}(b)\subseteq C_K} \tilde{\mu}_P\Big[(\tilde{\Lambda}^{ID}[\gamma_{\kappa_b}(b);\ \alpha(s^A(b),\ s^D(b))],\ b)\Big|(s^A(b),n^A(b))\Big]\Big\}\ .
\end{aligned} \tag{30}$$

Here, the $\tilde{\mu}_P\Big[(\tilde{\Lambda}^{ID}[\gamma_{\kappa_b}(b);\ \alpha(s^A(b),\ s^D(b))],\ b)\Big|(s^A(b),n^A(b))\Big]$ is the multiplication factor contributed from the portion of the PWA ($\alpha(s^A(b),\ s^D(b))$) between the ancestral state ($s^A(b)$) and the descendant state ($s^D(b)$) along the branch (b), confined in a region ($\gamma_{\kappa_b}(b)$, which is within C_K). (See Eq.(SM-4.11), Eq.(R6.8). and Eq.(SM-2.14) of [1].) The $\mu_P[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ C_K]$ is the multiplicative "difference" of the probability of state $s(n^{Root})$ at the root (n^{Root}) from that of the "reference" (s_0^{Root}) originated from C_K (see

Eq.(SM-4.16) of [1]). And the $\delta R_X^{ID}(s^A(b),\ s_0^{Root},\ \tau)[C_K]$ is the increment of the exit rate of the ancestral state ($s^A(b)$) compared to that of the reference state (s_0^{Root}), coming from the state difference confined in C_K (see Eq.(SM-4.14) of [1]).

For the current purpose, it would be more convenient to reorganize Eq.29 accompanied by Eq. 30 as follows:

$$\begin{aligned} & \breve{\mathrm{M}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \\ \stackrel{\text{def}}{=}\ & \mu_P[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ C_K] \prod_{b\in\{b\}_{\mathcal{T}}} \breve{\tilde{\mu}}_P\Big[\alpha(s^A(b),\ s^D(b));\ C_K\Big|(s^A(b),\ n^A(b))\Big] \end{aligned} \quad (31)$$

with

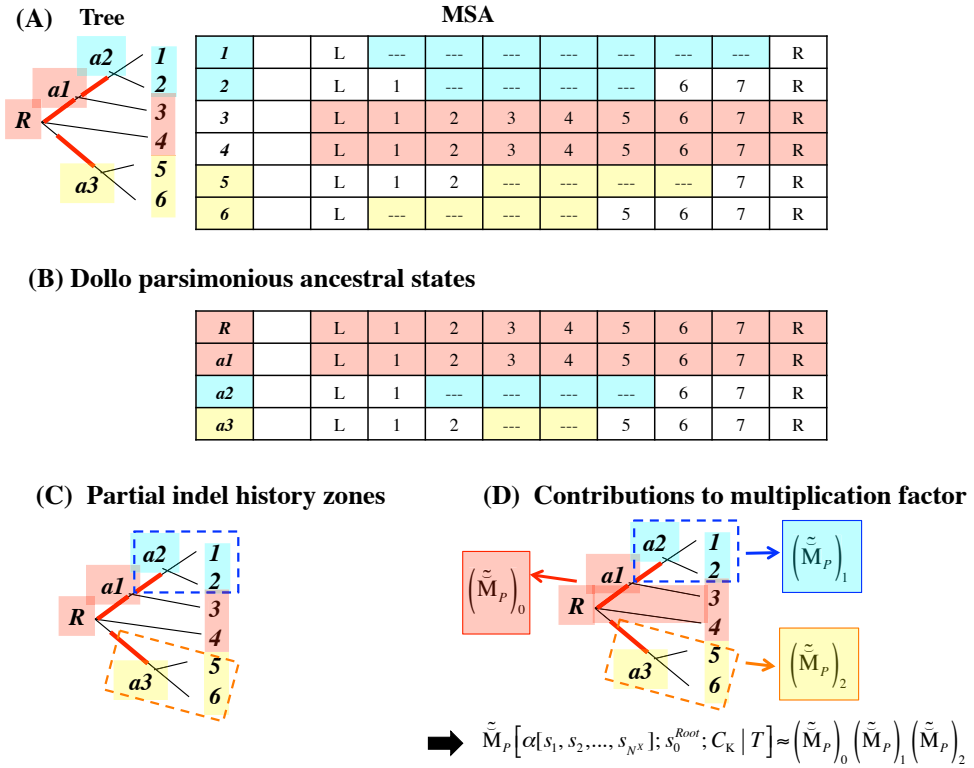
$$\begin{aligned} & \breve{\tilde{\mu}}_P\Big[\alpha(s^A(b),\ s^D(b));\ C_K\Big|(s^A(b),\ n^A(b))\Big] \\ \stackrel{\text{def}}{=}\ & \exp\Big\{-\int_{t(n^A(b))}^{t(n^D(b))} d\tau \delta R_X^{ID}(s^A(b),\ s_0^{Root},\ \tau)[C_K]\Big\} \times \\ & \times \prod_{\gamma_{\kappa_b}(b)\subseteq C_K} \tilde{\mu}_P\Big[(\tilde{\Lambda}^{ID}[\gamma_{\kappa_b}(b);\ \alpha(s^A(b),\ s^D(b))],\ b)\Big|(s^A(b), n^A(b))\Big]\ . \end{aligned} \quad (32)$$

Now, we are ready to discuss the problem at hand, that is, calculating the multiplication factor, Eq. 28, when the local MSA contains at least two *isolated* gap-masses, as in Figure 2 C or Figure 24 A. (We will use the latter for illustration.) In such a case, some of the ancestral nodes ("R" and "a1" in the current example) have virtually fixed gap states (Figure 24 B). [87] Such ancestral nodes could be used to "partition" the set of all internal nodes, N^{IN}, into three sub-sets:

$$\mathrm{N}^{IN} = \mathrm{N}_0^{IN} \cup \mathrm{N}_1^{IN} \cup \mathrm{N}_2^{IN}\ . \quad (33)$$

Here, the N_0^{IN} is the subset consisting of the "partitioning" nodes with *virtually* fixed ancestral states (like the red-shaded nodes in Figure 24 C); each of the N_1^{IN} and N_2^{IN} is the subset

[87] More precisely, as noted also in K.1, the states at these nodes in some non-parsimonious indel histories have *extra* sites that are destined to vanish *completely* from the extant sequences, leaving null columns in the MSA (as in Figure 25). Each of such indel histories, however, requires *at least* two additional indels that are coordinated *exquisitely*. Thus, *in general*, their contributions are negligible. (For a rough estimation of the effect of such indel histories, see footnote 84.)

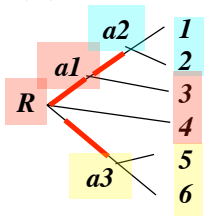


1		L	---	---	---	---	---	---	---	R
2		L	1	---	---	---	---	6	7	R
3		L	1	2	3	4	5	6	7	R
4		L	1	2	3	4	5	6	7	R
5		L	1	2	---	---	---	---	7	R
6		L	---	---	---	---	5	6	7	R

R		L	1	2	3	4	5	6	7	R
a1		L	1	2	3	4	5	6	7	R
a2		L	1	---	---	---	---	6	7	R
a3		L	1	2	---	---	5	6	7	R

Figure 24: **Factorizing effects of horizontally overlapping *non-interfering* gap-blocks. A.** An example local MSA. It is more complex than Figure 2 A, though the two figures are the same in essence. Each cell in the MSA is assigned either an ancestry index (L, R, or an Arabic numeral) or a gap (a lump of triple-dashes). The italicized Arabic numeral on the left indicates the sequence on the external node labeled with the same numeral in the tree. The masses of gaps isolated from each other are shaded in cyan and yellow. **B.** The Dollo parsimonious states obtained from the MSA in panel **A**. Any ancestral states consistent with the MSA must keep the occupied sites in the Dollo parsimonious states. Thus, the states 'R' and 'a1' here (red-shaded) indicate that the two isolated gap masses must have been created independently from each other(, as long as no null columns are brought in, as opposed to Figure 25). **C.** As a result, the indel history (or the ancestral states) yielding the cyan gap mass (enclosed in blue dashed box) is *virtually* independent of the history (or the ancestral states) yielding the yellow gap mass (enclosed in orange dashed box), as the two gap masses are separated by the red-shaded sequence states. (See, *e.g.*, Eq. 35.) **D.** The resulting multiplication factor, $\tilde{\tilde{\mathrm{M}}}_P\left[\alpha[s_1, s_2, ..., s_{N^X}];\ s_0^{Root};\ C_K \mid \mathcal{T}\right]$, is approximately the product of the contributions from different parts of the phylogenetic tree: one from the part with "fixed" ancestral states ($\left(\tilde{\tilde{\mathrm{M}}}_P\right)_0$), and the others from the isolated gap-masses ($\left(\tilde{\tilde{\mathrm{M}}}_P\right)_1$ and $\left(\tilde{\tilde{\mathrm{M}}}_P\right)_2$). (See Eq. 37.)

(A) Tree



MSA (w/ null column)

1		L	---	---	---	---	---	---	---	---	R
2		L	1	---	---	---	---	---	6	7	R
3		L	1	2	3	---	4	5	6	7	R
4		L	1	2	3	---	4	5	6	7	R
5		L	1	2	---	---	---	---	---	7	R
6		L	---	---	---	---	---	5	6	7	R

(B) Ancestral states

R		L	1	2	3	x1	4	5	6	7	R
a1		L	1	2	3	x1	4	5	6	7	R
a2		L	1	---	---	---	---	---	6	7	R
a3		L	1	2	---	---	---	5	6	7	R

Figure 25: **Non-parsimonious history that "connect" *non-interfering* gap-masses.** We will use the example MSA (and tree) given in Figure 24. In this case, an extra site (grey cell with ancestry ?x1?) in the ancestral states (R and a1) was deleted in *all* extant sequences (1-6), resulting in a null column (grey). (In this case, in addition to the four deletions needed for creating MSA in Figure 24, two more deletions, along branches a1-3 and R-4, are necessary in order to completely delete the site with ancestry ?x1?.) Such null columns will not usually be predicted via single-optimum-search aligners.

consisting of internal nodes involved in one of the *isolated* gap-masses (like the nodes enclosed in dashed boxes in Figure 24 C). Likewise, we can also decompose *each* set of ancestral states at all internal nodes, $\left\{s(n)\right\}_{n\in \mathrm{N}^{IN}}$, as:

$$\left\{s(n)\right\}_{n\in \mathrm{N}^{IN}} = \left\{s(n)\right\}_{n\in \mathrm{N}_0^{IN}} \cup \left\{s(n)\right\}_{n\in \mathrm{N}_1^{IN}} \cup \left\{s(n)\right\}_{n\in \mathrm{N}_2^{IN}} . \quad (34)$$

Here, the $\left\{s(n)\right\}_{n\in \mathrm{N}_0^{IN}}$ is *virtually* fixed. And the $\left\{s(n)\right\}_{n\in \mathrm{N}_1^{IN}}$ and $\left\{s(n)\right\}_{n\in \mathrm{N}_2^{IN}}$ are *virtuallly* independent of each other. Hence, $\Delta_\Sigma[C_K;\ s_0^{Root};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}]$ can be approximately expressed as a direct product:

$$\Delta_\Sigma[C_K;\ s_0^{Root};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}] \approx \left(\Delta_\Sigma\right)_0 \times \left(\Delta_\Sigma\right)_1 \times \left(\Delta_\Sigma\right)_1 . \quad (35)$$

Here, the $\left(\Delta_\Sigma\right)_k$ $(k = 0, 1, 2)$ is a shorthand notation of a "component" of $\Delta_\Sigma[C_K;\ s_0^{Root};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}]$, which consists of the sets of local-MSA-consistent ancestral states at nodes in N_k^{IN} (more precisely, their differences from s_0^{Root}). (Note that $\left(\Delta_\Sigma\right)_0 = \left\{\left\{s(n) - s_0^{Root}\right\}_{\mathrm{N}_0^{IN}}\right\}$.) Another essential element is the decomposition

of the set of branches:

$$\{b\}_{\mathcal{T}} = \{b\}^0_{\mathcal{T}} \cup \{b\}^1_{\mathcal{T}} \cup \{b\}^2_{\mathcal{T}} \ . \tag{36}$$

Here, the $\{b\}^k_{\mathcal{T}}$ ($k = 1$ or 2) is the set of branches that are directly connected with at least one node in N^{IN}_k (like the branches in the dashed boxes in Figure 24 C); the $\{b\}^0_{\mathcal{T}}$ is the set of remaining branches (i.e., those connected *solely* with nodes in N^{IN}_0).

Now, substituting Eqs. 35 & 36 into Eq. 28 accompanied by Eqs. 31 & 32, we get an approximate equation:

$$\tilde{\breve{\mathrm{M}}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \approx \Big(\tilde{\breve{\mathrm{M}}}_P\Big)_0 \Big(\tilde{\breve{\mathrm{M}}}_P\Big)_1 \Big(\tilde{\breve{\mathrm{M}}}_P\Big)_2 \ . \tag{37}$$

Here, the

$$\Big(\tilde{\breve{\mathrm{M}}}_P\Big)_k \overset{\text{def}}{=} \sum_{\{s(n)-s_0^{Root}\}} \sum_{n\in \mathrm{N}^{IN}_k [C_K]\in (\Delta_\Sigma)_k} \left[\begin{array}{l} \theta\Big(\mu_P[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ C_K];\ n^{Root};\ \mathrm{N}^{IN}_k\Big)\times \\ \times \prod_{b\in\{b\}_{\mathcal{T}}} \breve{\mu}_P\Big[\alpha(s^A(b),\ s^D(b));\ C_K \Big| (s^A(b),\ n^A(b))\Big] \end{array} \right] \tag{38}$$

is the collection of contributions from the sub-histories of indels along the branches in $\{b\}^k_{\mathcal{T}}$ ($k = 0, 1 or 2$). In this equation, it is tacitly agreed that the sequence states at nodes in $\Big\{s(n)\Big\}_{n\in \mathrm{N}^{IN}_0}$ are fixed as mentioned above. And we also defined the following function:

$$\theta(x;\ n;\ \mathrm{N}') \overset{\text{def}}{=} \begin{cases} x & if \quad n \in \mathrm{N}' \ , \\ 1 & otherwise \ . \end{cases} \tag{39}$$

When there are $k_{\max}$ (≤ 2) isolated gap-masses, Eq. 37 can be generalized as:

$$\tilde{\breve{\mathrm{M}}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \approx \Big(\tilde{\breve{\mathrm{M}}}_P\Big)_0 \prod_{k=1}^{k_{\max}} \Big(\tilde{\breve{\mathrm{M}}}_P\Big)_k \ , \tag{40}$$

where the $\Big(\tilde{\breve{\mathrm{M}}}_P\Big)_k$ is the total contribution from the k-th gap mass (like Eq. 38), and the $\Big(\tilde{\breve{\mathrm{M}}}_P\Big)_0$ is the contribution from the remaining, "fixed," part of the indel histories.

[**NOTE:** The vertical (or phylogenetic) partition (or factorization) given above is *good enough* for justifying the strategy that the current version of ANEX employs to compute the indel

components of MSA probabilities. If you attempt to grade up ANEX by *aggressively* utilizing the vertical factorization, however, the above partition may *not* be adequate. For such a purpose, the method provided in K.3.2 below should be more useful, because it covers almost all conceivable cases (including those with insertions).]

K.3.2 More useful *phylogenetic* factorization of indel multiplication factor

The vertical factorization formulas, Eqs. 37 & 38, in K.3.1 are *good enough* for justifying ANEX's current strategy to compute the indel components of MSA probabilities. When attempting to grade up ANEX by *aggressively* utilizing the vertical factorization, however, the formulas may *not* be so useful, for the following three reasons:

1. They are not applicable to the gapped segments containing insertion-type gaps, as in Figure 26 A;

2. They are not so useful when *actually* calculating the contributions to a *given* MSA (of extant sequences), instead of to a given set of states at all nodes (including both extant and ancestral sequences);

3. They are not so useful, either, when attempting to calculate the *increment(s) of* the contributions (or the entire MSA probability) caused by a move in the gap-configuration(s) of an MSA.

Thus, we hereby attempt to give *practically* more useful formulas for the *phylogenetic* (*i.e.*, *vertical*) factorization of the contributions.

First, we specify a "reference" ancestral sequence state, $s_0(n)$, at each internal node ($n \in \mathrm{N}^{IN}$). (Thus far, only one "reference" sequence state (s_0^{Root}) was specified in each gapped segment, only at the root node (n^{Root}).) A simplest candidate of a set of such reference states would be the ancestral gap states uniquely (and fairly quickly) given by the Dollo parsimony principle [83]. (See Figure 26 B for an example; indeed, our program

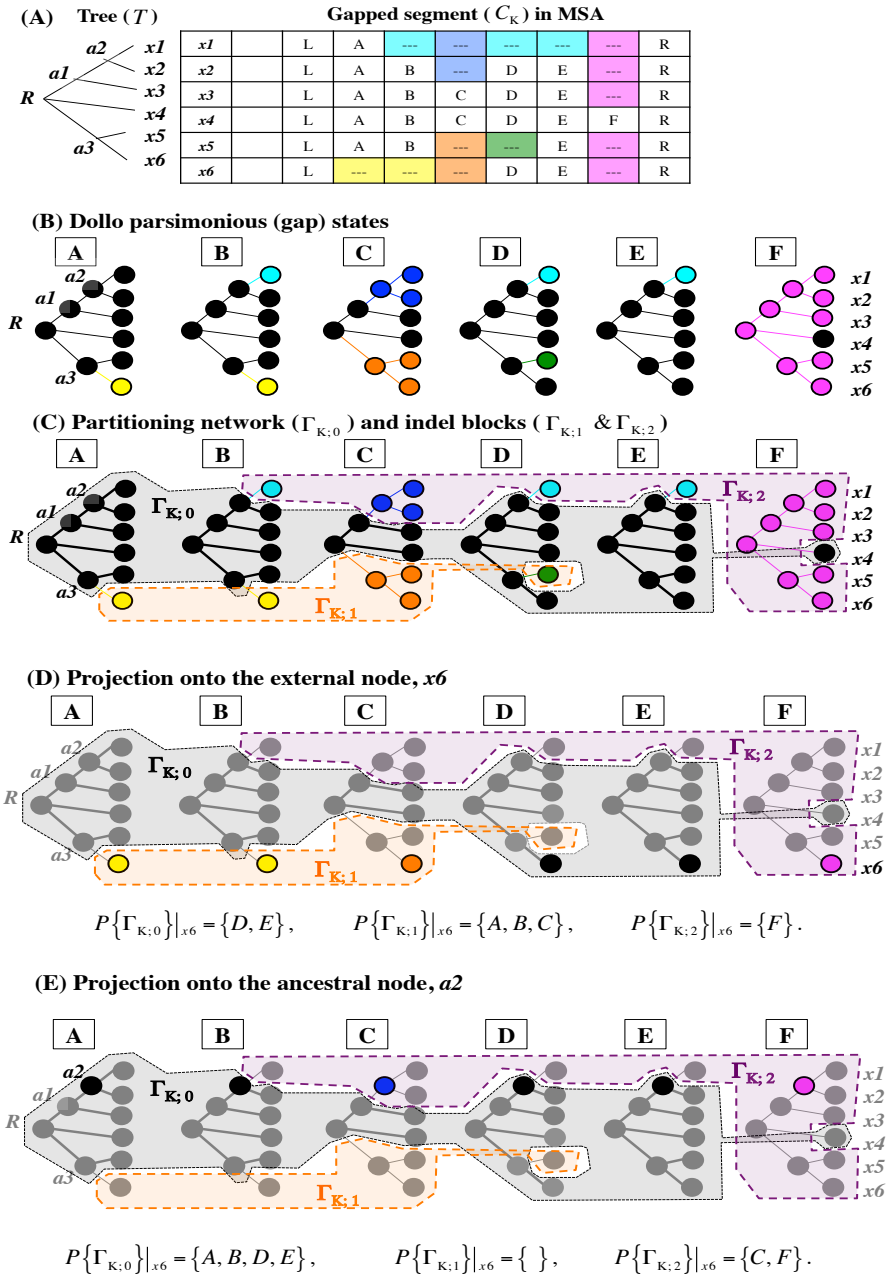


Figure 26: **Extended *vertical* partitioning of gapped segment. A.** An example of an input data set, consisting of a tree ($\mathcal{T}$) and a gapped segment (C_K) of a multiple sequence alignment (MSA). (In this figure, capital roman alphabets label the sites in the MSA.) **B.** Minimal gap states reconstructed under the Dollo parsimony principle [83, 2]. The black-filled circles represent the sites filled with residues at the nodes, and the color-shaded open circles represent the empty sites at the nodes. The colors correspond to the colors in panel A. And a black branch (edge) indicates that the site remains filled with a residue along the branch, whereas a colored branch indicates that the site could become (or remain) empty along the branch. **C.** Defining a ***partitioning network*** ($\Gamma_{K;0}$) and ***indel blocks*** ($\Gamma_{K;i}$, with $i = 1, ..., I_K$). (Here, $I_K = 2$.) The partitioning network is constructed by connecting all black-filled nodes and black branches. An indel block is constructed by connecting contiguous open nodes and colored branches. **D** & **E.** Projections of the partitioning network and indel blocks onto the external node, $x6$, and onto the ancestral node, $a2$, respectively. In these panels, the irrelevant nodes and branches are colored in grey. In each panel, the results of the projections are shown at the bottom.

implementation of this "extended phylogenetic factorization" will use the Dollo parsimonious gap states as the reference states, $\{s_0(n)\}_{n\in \mathrm{N}^{IN}}$.) Then, using the identity:

$$\delta R_X^{ID}(s^A(b),\ s_0^{Root},\ \tau)[C_K] = \delta R_X^{ID}(s_0^A(b),\ s_0^{Root},\ \tau)[C_K] + \delta R_X^{ID}(s^A(b),\ s_0^A(b),\ \tau)[C_K]\ ,$$

where $s_0^A(b) = s_0(n^A(b))$, [88] we can rewrite Eq. 29 supplemented with Eq. 30 as follows:

$$\begin{aligned}
& \breve{\mathrm{M}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \\
\equiv\ & \mathrm{M}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ C_K \mid \mathcal{T}\Big] \times \mu_P[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ C_K] \times \\
& \times \exp\Big\{-\Phi_0\Big[\mathcal{T},\ \{s_0(n)\}_{n\in \mathrm{N}^{IN}},\ C_K;\ \{\Theta^{ID}(b)\}_{b\in\mathcal{T}}\Big]\Big\} \times \\
& \times \exp\Big\{-\sum_{b\in\{b\}_{\mathcal{T}}} \int_{t(n^A(b))}^{t(n^D(b))} d\tau \delta R_X^{ID}(s^A(b),\ s_0^A(b),\ \tau)[C_K]\Big\}\ ,
\end{aligned} \tag{41}$$

Here,

$$\Phi_0\Big[\mathcal{T},\ \{s_0(n)\}_{n\in \mathrm{N}^{IN}},\ C_K;\ \{\Theta^{ID}(b)\}_{b\in\mathcal{T}}\Big] \stackrel{\mathrm{def}}{=} \sum_{b\in\{b\}_{\mathcal{T}}} \int_{t(n^A(b))}^{t(n^D(b))} d\tau \delta R_X^{ID}(s_0^A(b),\ s_0^{Root},\ \tau)[C_K] \tag{42}$$

is the "reference" phase factor determined uniquely by the tree ($\mathcal{T}$), the gapped segment (C_K) and the reference sequence states ($\{s_0(n)\}_{n\in \mathrm{N}^{IN}}$) (and the indel model ($\{\Theta^{ID}(b)\}_{b\in\mathcal{T}}$); the factor is independent of any other specific local indel histories that are compatible with the MSA within C_K , hence it can be computed easily (and fairly quickly).

Second, we re-define **effectively independent indel blocks** (, which were referred to as "partial indel history zones" in Figure 24 C (in K.3.1,) so that they can also cover more complex cases including insertions. For this purpose, let us recall that we are now considering gap-state configurations within the direct product space, $C_K \times \mathcal{T}$ (as represented by the array of trees in Figure 26 B), or more simply, $C_K \times \mathrm{N}^{IN}(\mathcal{T})$. The latter is usually adequate for the current purpose, because we are now considering *all* possible sets of ancestral gap-states *compatible with* the local MSA, *instead of* all possible indel histories compatible with the

[88] Remember that $n^A(b)$ denotes the node at the upper (*i.e.*, "ancestral") end of branch b. Incidentally, $n^D(b)$ denotes the node at the lowe (*i.e.*, "descendant") end of branch b.

local MSA. (Please remember that gap-states at external nodes ($n \in \mathrm{N}^X$) are *already* fixed, given a local MSA.)

Now, remember that the Dollo parsimonious ancestral state contains the *smallest* number of ancestral sites *occupied* by residues, because the Dollo parsimonious history consists of the *shortest* paths (along the tree) connecting residue-occupied sites at external nodes (see, *e.g.*, Figure 4 (especially panel A) of [131]). In other words, all other MSA-compatible ancestral gap-states can be made from the Dollo parsimonious state by filling some empty sites (*i.e.*, gaps) at ancestral nodes with residues *in a phylogenetically consistent manner* (*e.g.*, [129, 130]), which is enabled by extending some "paths" of residue-occupied sites from the network of such sites representing the Dollo parsimonious states (see, *e.g.*, Figure 4 (especially panel B) of [131]).

This "minimal" nature of the Dollo parsimonious indel history enables it to *partition* each gapped segment ($C_K \times \mathcal{T}$) into some "**indel block**"s, $\Gamma_{K;i}$ ($i = 1, 2, ..., I_K$), each of which can accommodate some indels, and a "**partitioning network**" of residue-occupied sites of the Dollo parsimonious states extended across the tree, denoted as $\Gamma_{K;0}$. (Figure 26 C illustrates such a partitioning.) This partitioning can be represented in an abstract equation using the symbol, "$\cup$," for a union of sets:

$$C_K \times \mathcal{T} = \Gamma_{K;0} \cup \left\{ \bigcup_{i=1}^{I_K} \Gamma_{K;i} \right\} . \tag{43}$$

(Here, it should be noted that the sets involved in the right-hand side are mutually disjoint from one another.) Importantly, **each of $\Gamma_{K;i}$ ($i = 1, 2, ..., I_K$) *defines* an "effectively-independent indel block"** (previously referred to as a "partial indel history zone"), or an "**indel block**" for short. [89] The following is the reason. First, each point in $C_K \times \mathcal{T}$ is specified by a site (in C_K) and a node (in $\mathcal{T}$); henceforth, we refer to such a point as a

[89] We consider that two (site, node)-points belong to the same indel block if they are connected *via* a path of (site, node)-points that are empty in the Dollo parsimonious history. Otherwise, that is, if they are clearly separated by at least a (site, node)-point that is occupied by a residue in the Dollo parsimonious history, we consider them as belonging to different indel blocks.

"(site, node)-point"). Now, by definition, every (site, node)-point belonging to $\Gamma_{K;0}$ must always be occupied by a residue, *regardless of* the indel histories (as long as they are compatible with the MSA). In contrast, each (site, node)-point belonging to $\Gamma_{K;i}$ $(i = 1, 2, ..., I_K)$ can be either empty or occupied with a residue depending on the specific MSA-compatible history, although it should always be empty in the Dollo parsimonious indel history. And it should also be noted that every indel event in every MSA-compatible parsimonious (or next-to-parsimonious) indel histories should be *completely* confined in one of $\Gamma_{K;i}$ $(i = 1, 2, ..., I_K)$; *otherwise*, the "partitioning network," $\Gamma_{K;0}$, is *not* working as it should, thus it *must* be re-defined.

Now, let us extend Eq. 35 (in K.3.1) for the purely vertical partitioning of a gapped segment, in order to create a formula suitable for the more general partitioning, Eq. 43. For this purpose, let $P\{\Gamma_{K;i}\}|_n$ be the projection of an indel block $\Gamma_{K;i}$ onto a node n(, which could be either internal or external), which is nothing other than the set of *all* sites (both empty and residue-occupied) in the ancestral sequence at n belonging to $\Gamma_{K;i}$. Similarly, let $P\{\Gamma_{K;0}\}|_n$ be the projection of the partitioning network ($\Gamma_{K;0}$) onto n. (Panels D & E of Figure 26 illustrate these projections.) Then, thanks to Eq. 43, the following decomposition always holds at every node n in $\mathcal{T}$:

$$C_K = P\{\Gamma_{K;0}\}|_n \cup \left\{ \bigcup_{i=1}^{I_K} P\{\Gamma_{K;i}\}|_n \right\} . \tag{44}$$

(Again, the sets involved in the right-hand side are mutually disjoint. It should also be noted that the $P\{\Gamma_{K;i}\}|_n$'s with some i's may be empty sets.)

Now, each element (denoted as $\{s(n) - s_0^{Root}\}_{\mathrm{N}^{IN}}[C_K]$) of $\Delta_\Sigma[C_K;\ s_0^{Root};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}]$, *i.e.*, a set of differences of internal gap states (within the region C_K) from the reference root state (s_0^{Root}), could be considered as an $|\mathrm{N}^{IN}|$-tuple [90], each of whose components is the difference of the gap (or "presence"/"absence") state ($s(n)$) at an internal node (n) from the "reference" s_0^{Root}; *hereafter*, the component will be denoted as

[90] The $|\mathrm{N}^{IN}|$ denotes the "size" of N^{IN}, *i.e.*, the number of all internal nodes.

$(s(n) - s_0^{Root})[C_K]$. [91] Because differences in distinct indel blocks are *effectively* independent of one another, we *almost* always have:

$$\begin{aligned} & (s(n) - s_0^{Root})[C_K] \\ = \; & (s_0(n) - s_0^{Root})[C_K] + (s(n) - s_0(n))[P\{\Gamma_{K;0}\}|_n] + \sum_{i=1}^{I_K} \Big\{ (s(n) - s_0(n))[P\{\Gamma_{K;i}\}|_n] \Big\} \; . \end{aligned} \tag{45}$$

Here, the $(s(n) - s_0(n))[P\{\Gamma_{K;i}\}|_n]$ is the gap-state difference within the indel block, $\Gamma_{K;i}$. Because the gap-state within the partitioning network ($\Gamma_{K;0}$) is *almost* always identical to that of $s_0(n)$, the second term on the right-hand-side *almost* always vanishes. Hence we have:

$$(s(n) - s_0^{Root})[C_K] = (s_0(n) - s_0^{Root})[C_K] + \sum_{i=1}^{I_K} \Big\{ (s(n) - s_0(n))[P\{\Gamma_{K;i}\}|_n] \Big\} \; . \tag{46}$$

Now, consider the $|\mathrm{N}^{IN}|$-tuple again. As already noted, its different components are *not* independent of one another. Nevertheless, *as long as the partitioning network remains intact,* it is *sufficient* to consider the phylogenetic consistency conditions among the components, $\Big\{ (s(n) - s_0(n))[P\{\Gamma_{K;i}\}|_n] \;\Big|\; n \in \mathrm{N}^{IN}(\mathcal{T}) \Big\}$, *within each indel block* ($\Gamma_{K;i}$). In such cases, components within different indel blocks can be treated independently of one another. Thus, the space of ancestral state differences, $\Delta_\Sigma[C_K;\; s_0^{Root};\; \alpha[s_1, s_2, ..., s_{N^X}];\; \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\; \mathcal{T}]$, can be *approximately* decomposed as follows. First, separate the constant differences between the reference ancestral states and the reference root state *from* the remaining variable parts:

$$\begin{aligned} & \Delta_\Sigma[C_K;\; s_0^{Root};\; \alpha[s_1, s_2, ..., s_{N^X}];\; \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\; \mathcal{T}] \\ = \; & \{s_0(n) - s_0^{Root}\}_{\mathrm{N}^{IN}(\mathcal{T})} + \Delta_\Sigma[C_K;\; \{s_0(n)\}_{\mathrm{N}^{IN}(\mathcal{T})};\; \alpha[s_1, s_2, ..., s_{N^X}];\; \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\; \mathcal{T}] \; . \end{aligned} \tag{47}$$

On the right-hand side, the first term represents the set of constant differences, and the second term represents the remaining variable parts. Then, the second term can be *approximately*

[91] The components of each $|\mathrm{N}^{IN}|$ -tuple are*not* mutually independent of each other, because they have to satisfy the phylogenetic consistency condition.

decomposed as follows:

$$\begin{aligned} & \Delta_{\Sigma}[C_K;\ \{s_0(n)\}_{\mathrm{N}^{IN}(\mathcal{T})};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}] \\ \approx & \ \overset{I_K}{\underset{i=1}{\times}} \Delta_{\Sigma}[\Gamma_{K;i}\ ;\ \{s_0(n)\}_{\mathrm{N}^{IN}(\mathcal{T})};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}]\ . \end{aligned} \tag{48}$$

Here, each element of the component space, $\Delta_{\Sigma}[\Gamma_{K;i}\ ;\ \{s_0(n)\}_{\mathrm{N}^{IN}(\mathcal{T})};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}]$, is a phylogenetically consistent set, $\{s(n) - s_0(n)\}_{\mathrm{N}^{IN}}[\Gamma_{K;i}] \overset{\text{def}}{=} \Big\{(s(n) - s_0(n))[P\{\Gamma_{K;i}\}|_n] \ \Big|\ n \in \mathrm{N}^{IN}(\mathcal{T})\Big\}$, of ancestral gap state differences within the given indel block, $\Gamma_{K;i}$.

Now, in order to extend the purely *vertical* factorization of the indel multiplication factor, Eq. 37 (or Eq. 40) in K.3.1, we need **two assumptions**, *in addition to* the approximate space decomposition given by Eqs. 47 & 48. **One** is the assumption that the *region-wise* increment of the exit rate can also be *further* decomposed (*at least* approximately) just as in Eq. 46:

$$\delta R_X^{ID}(s(n),\ s_0(n),\ \tau)[C_K] \approx \sum_{i=1}^{I_K} \Big\{\delta R_X^{ID}(s(n),\ s_0(n),\ \tau)[\Gamma_{K;i}]\Big\}\ . \tag{49}$$

(Here we omitted the increment confined in $\Gamma_{K;0}$, because the ancestral gap states within the partitioning network are *almost* always unchanged.) **The other** is the assumption that the (multiplicative) increment of the prior probability of the root state confined in C_K can also be *further* factorized (*at least* approximately) as:

$$\begin{aligned} & \mu_P[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ C_K] \\ \approx & \ \mu_P\Big[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ P\{\Gamma_{K;0}\}|_{n^{Root}}\Big] \times \prod_{i=1}^{I_K} \mu_P\Big[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ P\{\Gamma_{K;i}\}|_{n^{Root}}\Big]\ . \end{aligned} \tag{50}$$

Whether Eqs. 49 & 50 indeed hold or not is non-trivial *in general*, especially when the considered model allows indel variation across sites (or regions). Here, however, we *assume* that they hold *at least approximately*, and go on the mathematical argument.

Now, let's use Eqs.(SSA-1.9,10), the fact that different indel blocks ($\Gamma_{K;i}$'s) are *almost* always independent of each other, and the fact that the partitioning network ($\Gamma_{K;0}$) *almost* always remains intact. Then, the multiplication factor, Eq. 29 supplemented with Eq. 30 (both in K.3.1), which has been rewritten here as Eq. 41 supplemented with Eq. 42 can be *further* factorized (*at least approximately*) as follows:

$$\begin{aligned}
&\breve{\mathrm{M}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \\
\approx\ &\breve{\mathrm{M}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ \Gamma_{K;0} \mid \mathcal{T}\Big] \times \\
&\times \prod_{i=1}^{I_K} \breve{\mathrm{M}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ \Gamma_{K;i} \mid \mathcal{T}\Big]\ . \qquad (51)
\end{aligned}$$

Here,

$$\begin{aligned}
&\breve{\mathrm{M}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ \Gamma_{K;0} \mid \mathcal{T}\Big] \\
\stackrel{\text{def}}{=}\ &\mu_P\Big[s_0(n^{Root}),\ s_0^{Root},\ n^{Root};\ P\{\Gamma_{K;0}\}|_{n^{Root}}\Big] \times \exp\Big\{-\Phi_0\Big[\mathcal{T},\ \{s_0(n)\}_{n\in\mathrm{N}^{IN}},\ C_K;\ \{\Theta^{ID}(b)\}_{b\in\mathcal{T}}\Big]\Big\} \qquad (52)
\end{aligned}$$

is the portion of the multiplication factor associated with the "reference" ancestral gap states, $\{s_0(n)\}_{\mathrm{N}^{IN}}$, within the gapped segment (C_K) that is *contributed from* the partitioning network ($\Gamma_{K;0}$) . It should be noted that this portion *remains* unchanged regardless of the ancestral gap states, $\{s(n)\}_{\mathrm{N}^{IN}}$. And

$$\begin{aligned}
&\breve{\mathrm{M}}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ \Gamma_{K;i} \mid \mathcal{T}\Big] \\
\stackrel{\text{def}}{=}\ &\mathrm{M}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ \Gamma_{K;i} \mid \mathcal{T}\Big] \times \mu_P\Big[s(n^{Root}),\ s_0^{Root},\ n^{Root};\ P\{\Gamma_{K;i}\}|_{n^{Root}}\Big] \times \\
&\times \exp\Big\{-\sum_{b\in\{b\}_{\mathcal{T}}} \int_{t(n^A(b))}^{t(n^D(b))} d\tau \delta R_X^{ID}(s^A(b),\ s_0^A(b),\ \tau)[\Gamma_{K;i}]\Big\} \qquad (53)
\end{aligned}$$

is the portion of the multiplication factor *contributed from* the indel block, $\Gamma_{K;i}$, with

$$\begin{aligned}
&\mathrm{M}_P\Big[\alpha[s_1,s_2,...,s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ \Gamma_{K;i} \mid \mathcal{T}\Big] \\
\stackrel{\text{def}}{=}\ &\prod_{b\in\{b\}_{\mathcal{T}}} \Big\{ \prod_{\gamma_{\kappa_b}(b)\subseteq\Gamma_{K;i}} \tilde{\mu}_P\Big[(\tilde{\Lambda}^{ID}[\gamma_{\kappa_b}(b);\ \alpha(s^A(b),\ s^D(b))],\ b)\Big|(s^A(b),n^A(b))\Big]\Big\}\ . \qquad (54)
\end{aligned}$$

Then, substituting Eq. 51 into Eq. 28 in K.3.1, and using Eqs. 47 & 48, we get the *approximate* factorization of the *entire* multiplication factor for the local MSA within C_K:

$$\begin{aligned}
& \tilde{\breve{\mathrm{M}}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ s_0^{Root};\ C_K \mid \mathcal{T}\Big] \\
\approx\ & \breve{\mathrm{M}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ \Gamma_{K;0} \mid \mathcal{T}\Big] \times \\
& \times \prod_{i=1}^{I_K} \tilde{\breve{\mathrm{M}}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ \Gamma_{K;i} \mid \mathcal{T}\Big] \ . \qquad (55)
\end{aligned}$$

Here, the $\breve{\mathrm{M}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ s_0^{Root};\ \Gamma_{K;0} \mid \mathcal{T}\Big]$ has already been defined in Eq. 52, and

$$\begin{aligned}
& \tilde{\breve{\mathrm{M}}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ \Gamma_{K;i} \mid \mathcal{T}\Big] \\
\stackrel{\text{def}}{=} & \sum_{\substack{\{s(n)-s_0(n)\}_{\mathrm{N}^{IN}}[\Gamma_{K;i}] \\ \in \Delta_\Sigma\left[\Gamma_{K;i};\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}\right]}} \breve{\mathrm{M}}_P\Big[\alpha[s_1, s_2, ..., s_{N^X}];\ \{s(n)\}_{\mathrm{N}^{IN}};\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ \Gamma_{K;i} \mid \mathcal{T}\Big]
\end{aligned} \qquad (56)$$

(with the summands defined in Eq. 53) [92] is the *total* multiplicative contribution from the indel block, $\Gamma_{K;i}$.

Because each of the multiplicative factors in the approximate factorization formula, Eq. 55, can in principle be calculated independently of one another, this should facilitate the computation of the entire multiplication factor for each gapped segment, and also the computation of its change in response to a move of the gap-pattern in the gapped segment and its neighbors. Thus, this extension of the vertical partitioning (Eqs. 37 & 38 in K.3.1) will resolve the three drawbacks of its predecessor mentioned at the top of this sub-subsection.

[**NOTE:** Some of the computations based on the results of this sub-subsection are implemented in the subroutines in the modules, "MyTreeMap_indels_spt_odr_finer.pm" & "MyTreeMap_indels_ML_hs_finer.pm," in ANEX, in an *incomplete* manner, in the sense that

[92] The range of summation, $\Delta_\Sigma\Big[\Gamma_{K;i};\ \{s_0(n)\}_{\mathrm{N}^{IN}};\ \alpha[s_1, s_2, ..., s_{N^X}];\ \{n \in \mathrm{N}^{IN}(\mathcal{T})\};\ \mathcal{T}\Big]$, is "defined" below Eq. 48.

they can*not* yet *fully* utilize the outputs of LASTPIECE [4]; the subroutines do *not* incorporate the case-(i) multiplication factors into the computation of MSA multiplication factors; another problem is that these subroutines does *not fully* resolve the problem of homology structures when inferring the parsimonious ancestral states. (These incomplete features must be rectified, by using similar subroutines in the modules, "MyTreeMap_indels_spt_odr_hs.pm" and "MyTreeMap_indels_ML_hs_hs_wLP.pm," as a guide.)]

K.3.3 Toward *finer* phylogenetic partitioning of gapped segment

The "extended" phylogenetic partitioning derived in K.3.2 is expected to be *quite useful* for computing indel multiplication factors, as well as their changes caused by the moves of gap-blocks. Actually, we could, and should, go further, by ***separately dealing with*** the **gap-blocks** that, *physically*, vertically overlap (and horizontally adjoin or overlap) each other but that are *effectively* **non-interfering** with each other.

There are two typical cases of such *physically* overlapping (or adjoining) yet *effectively* non-interfering (composite) gap-blocks. In the first case, a pair of "ONN" gap-blocks [93] are horizontally adjoining (Figure 27 A). In this case, each (effective-)indel history formed by the interaction of the two gap-blocks has at least two more (effective-)indels than the parsimonious (effective-)indel histories, and such additional (effective-)indels must be *exquisitely coordinated* (Figure 27 B & C). And, in the second case, a pair of gap-blocks, one of which vertically includes the other, and whose delimiting branches are separated by *at least* two branches (or three trivalent nodes), are horizontally adjoining or nesting (Figure 27 D). In this case, too, you need at least two more *exquisitely coordinated* (effective-)indels, if you want a(n) (effective-)indel history in which the two gap-blocks interact (Figure 27 E).

Therefore, *generally* in these cases (Figure 27 A & D), *independently* handling these gap-blocks could be a good approximation; hence, an approximate factorization, like Eq.55 in K.3.2, should hold also for these cases.

[93] See appendix A for the definition of the "ONN" gap-blocks.

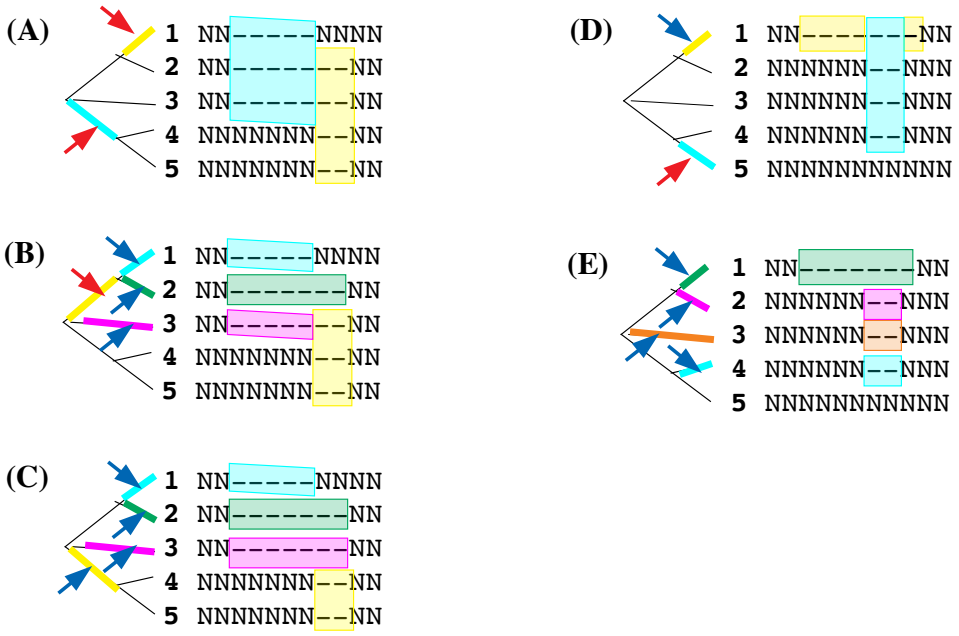


Figure 27: ***Physically* overlapping (or adjoining) yet *effectively* non-interfering gap-blocks. A.** A pair of horizontally adjoining "ONN" gap-blocks, and the parsimonious (effective-)indel history. **B & C.** The simplest (effective-)indel histories where the gap-blocks in panel A interact. (Actually, there is yet another history, which is similar to B but the deletion spanning the two gap-blocks is along branch 3 instead of 2.) **D.** A pair of horizontally nesting gap-blocks, one of which is vertically including the other, with their delimiting branches separated by two branches; accompanied by the parsimonious (effective-)indel history. **E.** The simplest (effective-)indel history where the gap-blocks in panel D interact. In each panel, an (effective-)indel history is shown on the tree on the left, with red and blue arrows represent (effective-)insertions and (effective-)deletions, respectively; in the MSA on the right, each colored rectangle represents the effect of an (effective-)indel, which occurred along the branch with the same hue (or color). [**NOTE:** The colored rectangles coincide with the gap-blocks *only in* panels A and D.]

However, we may have to consider *more carefully* whether such an approximate factorization indeed holds or not *even if* the gap-block configuration is more complex (for example, cases like Figures 27 A & D, but with each gap-block replaced with a *composite* gap-block). Once we confirm that this is indeed the case, extending the derivation of the (approximate) factorization formula, Eq. 55 in K.3.2, should be relatively easy.

[**NOTE:** We have *not* implemented this "*finer* phylogenetic partitioning" yet. Once the theory (as briefly described above) is established, however, you should be able to implement it by extending the implementation described at the bottom of K.3.2.]

L Conceivable problems

As with any brand-new method, there could be a number of problems that ANEX suffers from. Here we discuss three major problems, namely, those of (1) how to provide the program with accurate model parameters (including the tree), (3) autocorrelation, and (3) overfitting, as well as possible solutions to them.

L.1 Model parameters (including phylogenetic tree)

Currently, ANEX requires a fixed set of model parameters, including the phylogenetic tree. *Usually*, however, we do *not* know the *correct* parameters before conducting an analysis, and thus we have to estimate the parameters from the input data at hand. Theoretically, an ideal way would be to estimate the ***joint* probability distribution** of MSAs and model parameters including phylogenetic trees (e.g., [58, 105, 60, 61]). At present, however, this is generally very time-consuming and impractical *even if we use unrealistically simplistic probabilistic models of indels (, most of which are not even evolutionary models, rigorously speaking).* [94]

[94] Especially, the ML inference of the phylogenetic tree(s) is one of the most time-consuming problems in the study of molecular evolution, although such methods have been advanced greatly also in speed (*e.g.*,

Therefore, *at least* for the moment, it would be *inevitable* to *first* obtain a point estimation of the set of model parameters (including the tree), maybe using the input MSA, and to obtain the approximate MSA distribution under such a point-estimation. If, however, the aligned sequences are known to be the **orthologous ones** collected from organisms with *known* phylogenetic relationships, we could *safely* use (at least the topology of) the phylogenetic tree of the organisms. Moreover, if it is almost certain that the aligned sequences have undergone **neutral evolution**, we could *borrow* evolutionary parameters, such as the branch lengths and the indel rates, that were already estimated in the past (hopefully genome-wide) analyses. These cases will surely mitigate the problems caused by mis-estimated parameters, and thus they should be sought for.

Or, in the future, we may develop a more sophisticated method that *alternately* (and *iteratively*) estimates (1) the set of model parameters (or their approximate distributions, maybe via bootstrapping) and (2) the *approximate* MSA distribution, where the estimation of one uses (the previous estimate of) the other as input. This may improve the accuracy of the parameter estimation. Actually, a similar method was developed for the reconstruction of a single-optimum MSA, and was indeed shown to improve the accuracy (*e.g.*, [134]).

L.2 Autocorrelations

It would be *virtually* impractical, in terms of both computational time and memory space, to perform an analysis by using the (approximate) probability distribution of MSAs of *entire* sequences, because the number of alternative MSAs would be *super-astronomical*. Therefore, it would be inevitable to perform a **window analysis**, using the *approximate* probability distribution of MSAs obtained *for* each window created as in subsection 2.4 (and in appendix C), and to *integrate* the results of *all* windows to obtain the **"summary" result** for the MSAs of *entire* sequences. Here comes the potential problem of **autocorrelations**, because each window is fairly likely to *overlap* the neighboring ones. Solving this problem will be

[132, 133]).

very important for obtaining accurate results of the analyses.

A simplest solution would be to tile the entire sequence with a set of non-overlapping windows. Such a set should be devoid of auto-correlations, although using such a set may *not* exploit the merit of overlapping windows. A next simplest solution would be to compute the MSA probabilities in regions larger than single windows using a Markov-chain-like approximation, as explained in footnote 40 in a different context. Regarding the latter solution, we need to examine whether it actually works or not, *e.g.*, *via* thorough *in silico* "experiments."

L.3 Overfitting

At least in principle, our theoretical formulation of sequence evolution *via* indels [1, 2] can incorporate regional variation of indel rates to some extent. When actually incorporating this feature, however, we will need to be careful about the problem of **overfitting**, because, in general, sequences in an MSA experienced *much fewer* indels than substitutions during the evolution that created the *true* MSA. Thus, it is *very* likely that, if we *simply* employ a *single* optimum set of indel parameters for each MSA region, the **stochastic fluctuation** of an indel process should be *misinterpreted* as an indel rate variation.

A possible way to mitigate this problem would be to give an **approximate distribution of the sets of indel parameters**, giving a discrete number of typical sets (or classes) and their relative probabilities in the *entire* MSA. The relative probabilities may be based on some well-known (continuous) distribution, such as a gamma-distribution. Such a method has been developed for substitution models [86, 87] and it is commonly used for molecular evolution analyses these days (*e.g.*, [26]). And we expect that the method, possibly with some modifications, will also be *applicable to* the problem of sequence evolution *via* indels. [95]

[95] A recent program, SpartaABC [104], provides an approximate distribution of indel model parameters, *via* an approximate Bayesian computation method. Thus, it may be useful when pursuing this direction.

References

[1] K Ezawa. General continuous-time Markov model of sequence evolution via insertions/deletions: are alignment probabilities factorable? *BMC Bioinformatics.*, 17:304, 2016. Erratum in: BMC Bioinformatics (2016) 17:457.

[2] K Ezawa. General continuous-time Markov model of sequence evolution via insertions/deletions: local alignment probability computation. *BMC Bioinformatics.*, 17:397, 2016.

[3] K Ezawa. Characterization of multiple sequence alignment errors using complete-likelihood score and position-shift map. *BMC Bioinformatics.*, 17:133, 2016.

[4] K Ezawa. New perturbation method to compute probabilities of mutually adjoining insertion-type and deletion-type gaps in ancestor-descendant pairwise sequence alignment under *genuine* sequence evolution model with *realistic* insertions/deletions: the "last piece of the puzzle.". preprint (KEZW_BI_ME00005.lastpiece.pdf) available at: https://www.bioinformatics.org/ftp/pub/anex/Documents/Preprints/., 2020.

[5] D Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, Cambridge, UK, 1997.

[6] S Kumar and A Filipski. Multiple sequence alignment: in pursuit of homologous DNA positions. *Genome Res.*, 17:127–135, 2007.

[7] MR Aniba, O Poch, and JD Thompson. Issues in bioinformatics benchmarking: the case study of multiple sequence alignment. *Nucleic Acids Res.*, 38:7353–7363, 2010.

[8] A Löytynoja. Alignment methods: strategies, challenges, benchmarking, and comparative overview. In M Anisimova, editor, *Evolutionary Genomics. Methods in Molecular Biology (Methods and Protocols), vol. 855*, pages 203–235. Humana Press, Totowa, NJ, 2012.

[9] S Iantorno, K Gori, N Goldman, M Gil, and C Dessimoz. Who watches the watchmen? an appraisal of benchmarks for multiple sequence alignment. In D Russell, editor, *Multiple Sequence Alignment Methods. Methods in Molecular Biology (Methods and Protocols), vol. 1079*, pages 59–73. Humana Press, Totowa, NJ, 2014.

[10] T Warnow. *Computational Phylogenetics: An introduction to designing methods for phylogeny estimation*, chapter 9. Cambridge University Press, 2017.

[11] E Margulies and E Birney. Approaches to comparative sequence analysis: towards a functional view of vertebrate genomes. *Nat Rev Genet.*, 9:303–313, 2008.

[12] JR Lupski. Genomic rearrangements and sporadic disease. *Nat Genet.*, 39(7 Suppl):S43–S47, 2007.

[13] M Lynch. *The Origins of Genome Architecture*. Sinauer Associates, Sunderland, MA, 2007.

[14] W Gu, F Zhang, and JR Lupski. Mechanisms for human genomic rearrangements. *Pathogenetics.*, 1:4, 2008.

[15] TH Lee, J Kim, JS Robertson, and AH Paterson. Plant genome duplication database. In A. van Dijk, editor, *Plant Genomic Databases. Methods in Molecular Biology, vol. 1533*, pages 267–277. Humana Press, New York, NY, 2017.

[16] CN Dewey. Whole-genome alignment. In M Anisimova, editor, *Evolutionary Genomics. Methods in Molecular Biology (Methods and Protocols), vol. 855*, pages 237–257. Humana Press, Totowa, NJ, 2012.

[17] D Earl, N Nguyen, G Hickey, RS Harris, S Fitzgerald, and K et. al. Beal. Alignathon: a competitive assessment of whole-genome alignment methods. *Genome Res.*, 24:2077–2089, 2014.

[18] JD Thompson. *Statistics for Bioinformatics: Methods for Multiple Sequence Alignment, 1st Edition*, chapter 6. ISTE Press - Elsevier, 2016.

[19] T Yada. Genome alignment. In S Ranganathan, K Nakai, and C Schonbach, editors, *The Encyclopedia of Bioinformatics and Computational Biology*, pages 268–283. Elsevier, Amsterdam, NL, 2019.

[20] J Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol.*, 17:368–376, 1981.

[21] J. Felsenstein. *Inferring Phylogenetics*. Sinauer, Sunderland, Massachusetts, 2004.

[22] K Arnold, L Bordoli, J Kopp, and T Schwede. The SWISS-MODEL workspace: a Web based envoronment for protein structure homology modeling. *Bioinformatics.*, 22:195–201, 2006.

[23] JA Eisen. Plhylogenomics: improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Res.*, 8:163–187, 1998.

[24] RD Finn, J Mistry, J Tate, P Coggill, A Hedger, JE Pollington, et al. The Pfam protein families database. *Nucleic Acids Res.*, 38:D211–D212, 2009.

[25] PP Gardner, J Daub, J Tabe, BL Moore, IH Osuch, S Griffiths-Jones, et al. Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Res.*, 39:D141–D145, 2011.

[26] Z Yang. PAML: a program package for phylogenetic analysis by maximum likelihood. *Mol Biol Evol.*, 24:1586–1591, 2007.

[27] JD Thompson, F Plewniak, and O Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, 27:2682–2690, 1999.

[28] C Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Comput Biol.*, 3:e123, 2007.

[29] JD Thompson, DG Higgins, and TJ Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[30] C Notredame, DG Higgins, and J Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment. *J Mol Biol.*, 302:205–217, 2000.

[31] K Katoh, K Misawa, K Kuma, and T Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Res.*, 30:3059–3066, 2002.

[32] K Katoh, K Kuma, H Toh, and T Miyata. MAFFT version 5: implevement in accuracy of multiple sequence alignment. *Nucleic Acids Res.*, 33:511–518, 2005.

[33] K Katoh and H Toh. Recent developments in the MAFFT multiple sequence alignment program. *Brief Bioinform.*, 9:286–298, 2008.

[34] RC Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32:1792–1797, 2004.

[35] CB Do, MS Mahabhashyam, M Brudno, and S Batzoglou. ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15:330–340, 2005.

[36] A Löytynoja and N Goldman. An algorithm for progressive multiple alignment of sequences with insertions. *Proc Natl Acad Sci USA.*, 102:10557–10562, 2005.

[37] A Löytynoja and N Goldman. Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science.*, 320:1632–1635, 2008.

[38] J Pei and NV Grishin. MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structural information. *Nucleic Acids Res.*, 34:364–374, 2006.

[39] U Roshan and DR Livesay. Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics.*, 22:2715–2721, 2006.

[40] LM Wallace, O O'Sullivan, DG Higgins, and C Notredame. M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Res.*, 34:1692–1699, 2006.

[41] AR Subramanian, M Kaufumann, and B Morgenstern. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms Mol Biol.*, 3:6, 2008.

[42] K Kryukov and N Saitou. MISHIMA–a new method for high speed multiple alignment of nucleotide sequences of bacterial genome scale data. *BMC Bioinformatics.*, 11:142, 2010.

[43] EA O'Brien and DG Higgins. Empirical estimation of the reliability of ribosomal rna alignments. *Bioinformatics.*, 14:830–838, 1998.

[44] KM Wong, MA Suchard, and JP Huelsenbeck. Alignment uncertainty and genome analysis. *Science.*, 319:473–476, 2008.

[45] G Landan and D Graur. Characterization of pairwise and multiple sequence alignment errors. *Gene.*, 441:141–147, 2009.

[46] DA Morrison and JT Ellis. Effects of nucleotide sequence alignment on phylogeny estimation: a case study of 18S rDNAs of apicomplexa. *Mol Biol Evol.*, 14:428–441, 1997.

[47] RE Hickson, C Simon, and SW Perry. The performance of several multiple sequence alignment programs in relation to secondary-structure features for an rrna sequence. *Mol Biol Evol.*, 17:530–539, 2000.

[48] TH Ogden and MS Rosenberg. Multiple sequence alignment accuracy and phylogenetic inference. *Syst Biol.*, 55:314–318, 2006.

[49] P Markova-Raina and D Petrov. High sensitivity to aligner and high rate of false positives in the estimates of positive selection in the 12 Dorosophila genomes. *Genome Res.*, 21:863–874, 2011.

[50] O Westesson, G Lunter, B Paten, and I Holmes. Accurate reconstruction of insertion-deletion histories by statistical phylogenetics. *PLoS One.*, 7:e34572, 2012.

[51] G Lunter, A Rocco, N Mimouni, A Heger, A Caldeira, and J Hein. Uncertainty in homology inferences: assessing and improving genomic sequence alignment. *Genome Res.*, 18:298–309, 2008.

[52] RA Cartwright. Problems and solutions for estimating indel rates and length distributions. *Mol Biol Evol.*, 26:473–480, 2009.

[53] J Hein, C Wiuf, B Knudsen, MB Møller, and G Wibling. Statistical alignment: computational properties, homology testing and goodness-of-fit. *J Mol Biol.*, 302:265–279, 2000.

[54] MJ Bishop and EA Thompson. Maximum likelihood alignment of DNA sequences. *J Mol Biol.*, 190:159–165, 1986.

[55] JL Thorne, H Kishino, and J Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences. *J Mol Biol.*, 33:114–124, 1991.

[56] JL Thorne, H Kishino, and J Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *J Mol Biol.*, 34:3–16, 1992.

[57] B Knudsen and MM Miyamoto. Sequence alignments and pair hidden Markov models using evolutionary history. *J Mol Biol.*, 333:453–460, 2003.

[58] I Holmes and WJ Bruno. Evolutionary HMMs: a Bayesian approach to multiple alignment. *Bioinformatics.*, 17:803–820, 2001.

[59] I Holmes. Using guide trees to construct multiple-sequence evolutionary HMMs. *Bioinformatics.*, 19(Suppl 1):i147–i157, 2003.

[60] MA Suchard and BD Redelings. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics.*, 22:2047–2048, 2006.

[61] Á Novák, I Miklós, R Lyngsø, and J Hein. StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. *Bioinformatics.*, 24:2403–2404, 2008.

[62] B Paten, J Herrero, S Fitzgerald, K Beal, P Flicek, I Holmes, and E Birney. Genome-wide nucleotide-level mammalian ancestor reconstruction. *Genome Res.*, 18:1829–1843, 2008.

[63] RK Bradley, A Roberts, M Smoot, S Juvekar, J Do, C Dewey, I Holmes, and I Pachter. Fast statistical alignment. *PLoS Comput Biol.*, 5:e1000392, 2009.

[64] JL Herman, Á Novák, R Lyngsø, A Szabó, I Miklós, and J Hein. Efficient representation of uncertainty in multiple sequence alignments using directed acyclic graphs. *BMC Bioinformatics.*, 16:108, 2015.

[65] N De Maio. The cumulative indel model: fast and accurate statistical evolutionary alignment. *Syst Biol.*, 2020. (available as E-pub).

[66] I Holmes. Application of indel evolution by differential calculus of finite state automata. available in bioRxiv with doi: 10.1101/2020.06.29.178764., 2020.

[67] R Durbin, SR Eddy, A Krogh, and G Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK, 1998.

[68] GH Gonnet, MA Cohen, and SA Benner. Exhaustive matching of the entire protein sequence database. *Science.*, 256:1443–1445, 1992.

[69] SA Benner, MA Cohen, and GH Gonnet. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J Mol Biol.*, 229:1065–1082, 1993.

[70] X Gu and WH Li. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. *J Mol Evol.*, 40:464–473, 1995.

[71] Z Zhang and M Gerstein. Patterns of nucleotide substitution, insertion and deletion in the human genome inferred from pseudogenes. *Nucleic Acids Res.*, 31:5338–5348, 2003.

[72] MS Chang and SA Benner. Empirical analysis of protein insertions and deletions determining parameters for the correct placement of gaps in protein sequence alignments. *J Mol Biol.*, 341:617–631, 2004.

[73] K Yamane, K Yano, and T Kawahara. Pattern and rate of indel evolution inferred from whole chloroplast intergenic regions in sugarcane, maize and rice. *DNA Res.*, 13:197–204, 2006.

[74] Y Fan, W Wang, G Ma, L Liang, Q Shi, and S Tao. Patterns of insertion and deletion in mammalian genomes. *Curr Genomics.*, 8:370–378, 2007.

[75] I Miklós and Z Toroczkai. An improved model for statistical alignment. *WABI 2001.*, LNCS 2149:1–10, 2001.

[76] I Miklós, GA Lunter, and I Holmes. A "long indel" model for evolutionary sequence alignment. *Mol Biol Evol.*, 21:529–540, 2004.

[77] RA Cartwright. DNA assembly with gaps (Dawg): simulating sequence evolution. *Bioinformatics.*, 21 (Suppl. 3):iii31–iii38, 2005.

[78] W Fletcher and Z Yang. INDELible: A flexible simulator of biological sequence evolution. *Mol Biol Evol.*, 26:1879–1888, 2009.

[79] CL Strope, K Abel, SD Scott, and EN Moriyama. Biological sequence simulation for testing complex evolutionary hypothesis: indel-Seq-Gen version 2.0. *Mol Biol Evol.*, 26:2581–2593, 2009.

[80] E Levy Karin, H Ashkenazy, J Hein, and T Pupko. A simulation-based approach to statistical alignment. *Syst Biol.*, 68:252–266, 2019.

[81] J Kim and S Sinha. Indelign: a probabilistic framework for annotation of insertions and deletions in a multiple alignment. *Bioinformatics.*, 23:289–297, 2007.

[82] K Ezawa. Substitutional Residue-Difference Map (SRD Map) to help locate mis-alignments in Multiple Sequence Alignment (MSA): toward Artificial-Intelilgence-assisted probability distribution of alternative MSAs. preprint (KEZW_BI_ME00007.srdmap.pdf) available at: https://www.bioinformatics.org/ftp/pub/anex/Documents/Preprints/., 2020.

[83] JS Farris. Phylogenetic analysis under dollo's law. *Syst Zool.*, 26:77–88, 1977.

[84] R. Sedgewick. *Algorithms in C.* Addison-Wiley, Massachusetts, US, 1990. Translated into Japanese by K. Noshita, M. Hoshi, H. Satou, and A. Taguchi, and published by Kindai Kagaku Sha (Tokyo, Japan) in 1996.

[85] H Watanabe, A Fujiyama, M Hattori, TD Taylor, A Toyoda, et al. DNA sequence and comparative analysis of chimpanzee chromosome 22. *Nature.*, 429:382–388, 2004.

[86] Z Yang. A space-time process model for the evolution of DNA sequences. *Genetics.*, 139:993–1005, 1995.

[87] Z. Yang. *Computational Molecular Evolution.* Oxford Univ. Press, Oxford, UK, 2006.

[88] K Tamura and M Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol.*, 10:512–526, 1993.

[89] TH Jukes and CR Cantor. Evolution of protein molecules. In HN Munro, editor, *Mammalian protein metabolism*, pages 21–123. Academic Press, New York, US, 1969.

[90] M Kimura. A simple method for estimating evolutionary rate of base-substitution through comparative studies of nucleotide sequences. *J Mol Evol.*, 16:111–120, 1980.

[91] M Hasegawa, H Kishino, and T Yano. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol.*, 22:160–174, 1985.

[92] S Tavaré. Some probabilistic and statistical problem on the analysis of DNA sequences. *Lec Math Life Sci.*, 17:57–86, 1986.

[93] Z Yang. Estimating the pattern of nucleotide substitution. *J Mol Evol.*, 39:105–111, 1994.

[94] X Gu and WH Li. A general additive distance with time-reversibility and rate variation among nucleotide sites. *Proc Natl Acad Sci. USA.*, 93:4671–4676, 1996.

[95] Z Yang and S Kumar. Approximate methods for estimating the pattern of nucleotide substitution and the variation of substitution rates among sites. *Mol Biol Evol.*, 13:650–659, 1996.

[96] KM Roskin, B Paten, and D Haussler. Meta-alignment with Crumble and Prune: partitioning very large alignment problems for performance and parallelization. *BMC Bioinformatics.*, 12:144, 2011.

[97] BP Blackburne and S Whelan. Class of multiple sequence alignment algorithm affects genome analisis. *Mol Biol Evol.*, 30:642–653, 2013.

[98] K Ezawa. Alingment Neighborhood EXplorer (ANEX): First attempt to apply *genuine* sequence evolution model with realistic insertions/deletions to Multiple Sequence Alignment reconstruction problem. preprint (KEZW_BI_ME00006.anex.pdf) available at: https://www.bioinformatics.org/ftp/pub/anex/Documents/Preprints/., 2020.

[99] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, New York, NY, 2006. ISBN: 9780387310732, 9781493938438.

[100] Ameet Talwalkar Mehryar Mohri, Afshin Rostamizadeh. *Foundations of Machine Learning, 2nd edition*. MIT Press, Cambridge, MA, 2018. ISBN: 0262351366, 9780262351362.

[101] Ethern Alpaydin. *Introduction to Machine Learning, 4th edition (Adaptive Computation and Machine Learning series)*. MIT Press, Cambridge, MA, 2020. ISBN: 0262043793, 9780262043793.

[102] Y Lecun, Y Bengio, and G Hinton. Deep Learning. *Nature.*, 521:436–444, 2015.

[103] Aaron Courville Ian Goodfellow, Yoshua Bengio. *Deep Learning (Adaptive computation and machine learning series)*. MIT Press, Cambridge, MA, 2016. ISBN: 0262337371, 9780262337373.

[104] E Levy Karin, D Shkedy, H Ashkenazy, RA Cartwright, and T Pupko. Inferring rates and length-distributions of indels using Approximate Bayesian Computation. *Genome Biol Evol.*, 9:1280–1294, 2017.

[105] G Lunter, I Miklós, A Drummond, JL Jensen, and J Hein. Baysian coestimation of phylogeny and sequence alignment. *BMC Bioinformatics.*, 6:83, 2005.

[106] M Morgante, E De Paoli, and S Radovic. Transposable elements and the plant pangenomics. *Curr Opin Plant Biol.*, 10:1039–1049, 2007.

[107] D Chalopin, M Navile, F Plard, D Galiana, and JN Vollff. Comparative analysis of transposable elements highlights mobilome diversity and evolution in vertebrates. *Genome Biol Evol.*, 7:567–580, 2015.

[108] J Ellegren. Microsatellites: simple sequences with complex evolution. *Nat Rev Genet.*, 5:435–445, 2004.

[109] R Sainudiin, RT Durrett, CF Aquadro, and R Nielsen. Microsatellite mutation models: insights from a comparison of humans and chimpanzees. *Genetics.*, 168:383–395, 2004.

[110] F Famoud, M Schwartz, and J Bruck. Estimation of duplication history under a stochastic model for tandem repeats. *BMC Bioinformatics.*, 20:64, 2019.

[111] N Saitou and T Kitano. The PNarec method for detection of ancient recombinations through phylogenetic network analysis. *Mol Phylogen Evol.*, 66:507–514, 2013.

[112] JM Chen, DN Cooper, N Chuzhanova, C Férec, and GP Patrionos. Gene conversion: mechanisms, evolution and human disease. *Nat Rev Genet.*, 8:762–775, 2007.

[113] K Ezawa, K Ikeo, T Gojobori, and N Saitou. Evolutionary pattern of gene homogenization between primate-specific paralogs after human and macaque speciation using the 4-2-4 method. *Mol Biol Evol.*, 27:2152–2171, 2010.

[114] JA Fawcett and H Innan. The role of gene conversion in preserving rearrangement hotspots in the human genome. *Trends in Genetics.*, 29:561–568, 2013.

[115] A Löytynoja and N Goldman. Short template switch events explain mutation clusters in human genome. *Genome Res.*, 27:1039–1049, 2017.

[116] ST Lovett. Template-switching during replication fork repair in bacteria. *DNA Repair (Amst)*, 56:118–128, 2017.

[117] B Lavi, E Levy Karin, T Pupko, and E Covo-Hazkani. The prevalence and evolutionary conservation of inverted repeats in proteobacteria. *Genome Biol Evol.*, 10:918–927, 2018.

[118] K Ezawa, D Graur, and G Landan. Perturbative formulation of general continuous-time Markov model of sequence evolution via insertions/deletions, Part IV: incorporation of substitutions and other mutations. available in bioRxiv with doi: 10.1101/023622., 2015.

[119] G Tesler. GRIMM: genome rearrangements web server. *Bioinformatics.*, 18:492–493, 2002.

[120] O Gascuel, editor. *Mathematics of Evolution and Phylogeny.* Oxford University Press, New York, NY, 2005.

[121] J Ma, A Ratan, BJ Raney, BB Suh, W Miller, and D Haussler. The infinite sites model of genome evolution. *Proc Natl Acad Sci USA.*, 105:14254–14261, 2008.

[122] Y Lin and BM Moret. A new genomic evolutionary model for rearrangements, duplications, and losses that applies both eukaryotes and prokaryotes. *J Comput Biol.*, 18:1055–1064, 2011.

[123] PH da Silva, R Machado, S Dantas, and MD Braga. Restricted DCJ-indel model: sorting linear genomes with DCJ and indels. *BMC Bioinformatics.*, 13(Suppl 19):S14, 2012.

[124] M Shao, Y Lin, and BM Moret. Sorting genomes with rearrangements and segmental duplications through trajectory graphs. *BMC Bioinformatics.*, 14 (Suppl 15):S9, 2013.

[125] CG Ghiurcuta and BME Moret. Evaluating synteny for improved comparative studies. *Bioinformatics.*, 30:i9–i18, 2014.

[126] MD Braga and J Stoye. Sorting linear genomes with rearrangements and indels. *IEEE/ACM Trans Comput Biol Bioinform.*, 12:500–506, 2015.

[127] K Ezawa. (Approximate) Solutions to some technical issues on alignment probability calculation under *genuine* sequence evolution model with realistic insertions/deletions. (in preparation.).

[128] G Lunter, I Miklós, A Drummond, J Jensen, and J Hein. Baysian phylogenetic inference under a statistical indel model. *Lecture Notes in Bioinformatics.*, 2812:228–244, 2003.

[129] L Chindelevitch, Z Li, E Blais, and M Blanchette. On the inference of parsimonious evolutionary scenarios. *J Bioinform Comput Biol.*, 4:721–744, 2006.

[130] AB Diallo, V Makarenkov, and M Blanchette. Exact and heuristic algorithms for the indel maximum likelihood problem. *J Comput Biol.*, 14:446–461, 2007.

[131] K Ezawa, D Graur, and G Landan. Perturbative formulation of general continuous-time Markov model of sequence evolution via insertions/deletions, Part I: Theoretical basis. available in bioRxiv with doi: 10.1101/023598., 2015.

[132] S Guindon, JF Dufayard, V Lefort, M Anisimova, W Hordijk, and O Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Syst Biol.*, 59:307–321, 2010.

[133] A Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics.*, 30:1312–1313, 2014.

[134] O Gotoh. Significant improvement in accuracy of multiple sequence alignments by iterative refinement as assessed by reference to structural alignments. *J Mol Biol.*, 264:823–838, 1996.