



UNIL | Université de Lausanne

Unicentre

CH-1015 Lausanne

<http://serval.unil.ch>

Year : 2019

Progressive Multiple Sequence Alignment with Indel Evolution

Maiolo Massimo

Maiolo Massimo, 2019, Progressive Multiple Sequence Alignment with Indel Evolution

Originally published at : Thesis, University of Lausanne

Posted at the University of Lausanne Open Archive <http://serval.unil.ch>

Document URN : urn:nbn:ch:serval-BIB_D24577D3A8854

Droits d'auteur

L'Université de Lausanne attire expressément l'attention des utilisateurs sur le fait que tous les documents publiés dans l'Archive SERVAL sont protégés par le droit d'auteur, conformément à la loi fédérale sur le droit d'auteur et les droits voisins (LDA). A ce titre, il est indispensable d'obtenir le consentement préalable de l'auteur et/ou de l'éditeur avant toute utilisation d'une oeuvre ou d'une partie d'une oeuvre ne relevant pas d'une utilisation à des fins personnelles au sens de la LDA (art. 19, al. 1 lettre a). A défaut, tout contrevenant s'expose aux sanctions prévues par cette loi. Nous déclinons toute responsabilité en la matière.

Copyright

The University of Lausanne expressly draws the attention of users to the fact that all documents published in the SERVAL Archive are protected by copyright in accordance with federal law on copyright and similar rights (LDA). Accordingly it is indispensable to obtain prior consent from the author and/or publisher before any use of a work or part of a work for purposes other than personal use within the meaning of LDA (art. 19, para. 1 letter a). Failure to do so will expose offenders to the sanctions laid down by this law. We accept no liability in this respect.



UNIL | Université de Lausanne

Faculté de biologie
et de médecine

Département de biologie computationnelle

Progressive Multiple Sequence Alignment with Indel Evolution

Thèse de doctorat ès sciences de la vie (PhD)

présentée à la

Faculté de biologie et de médecine
de l'Université de Lausanne

par

Massimo Maiolo

Master of Science SUPSI in Engineering

Jury

Prof. Jérôme Goudet, Président
Prof. Christophe Dessimoz, Directeur de thèse
Prof. Nicolas Salamin, Co-directeur
Dr. Maria Anisimova, Co-directeur
Prof. Olivier Gascuel, expert
Prof. Anna-Sapfo Malaspinas, expert

Lausanne 2019

Imprimatur

Vu le rapport présenté par le jury d'examen, composé de

Président·e	Monsieur Prof. Jérôme Goudet
Directeur·rice de thèse	Monsieur Prof. Christophe Dessimoz
Co-directeurs·rices	Monsieur Prof. Nicolas Salamin
	Madame Dre Maria Anisimova
Experts·es	Monsieur Prof. Olivier Gascuel
	Madame Prof. Anna-Sapfo Malaspinas

le Conseil de Faculté autorise l'impression de la thèse de

Monsieur Massimo Maiolo

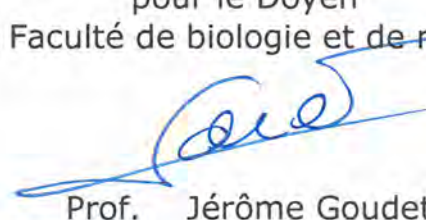
Master en ingénierie SUPSI, Lugano

intitulée

**Progressive multiple sequence alignment
with indel evolution**

Lausanne, le 12 février 2019

pour le Doyen
de la Faculté de biologie et de médecine



Prof. Jérôme Goudet

© 2018 by Massimo Maiolo. All rights reserved



Acknowledgments

“Gratitude is when memory is stored in the heart and not in the mind.”

– Lionel Hampton

The outcomes of this thesis could not have been attained without the many people who have been involved in my life throughout the years. In different ways these people have encouraged, mentored, guided, instructed, influenced and loved me.

Firstly, I have great pleasure in acknowledging my gratitude to Maria Anisimova and Manuel Gil, my PhD advisors, for supporting and assisting me during the past four years. I have been fortunate to have been advised by you.

I want to express my gratitude to Lorenzo Gatti who gave me valuable inputs in many occasions. We have spent together many many hours coding and debugging our software package. I am particularly indebted to Simone Ulzega for helping me a lot especially with the STFT project. He is the colleague that I have my morning ‘doppio espresso’ with. I would like to express my thanks to Xiaolei Zhang, our brilliant Master student, for the great collaboration in the 3D-DP algorithm design. I have appreciated the collaboration with Tiziano Leidi and Diego Frei, for the impressive experience and for inspirational discussions with me regarding the code parallelisation.

I am grateful for conversations with Victor Garcia, Spencer Bliven and Norman Juchler. They have been helpful in giving advises many times during my time at ZHAW. I’m also glad to have worked with our master students, Ta Cam Phuong Nguyesan and Jithin Mathew Peechatt.

The members at the Institute of Applied Simulations have contributed to my professional development, they have been always very kind and friendly as well as good collaborator. In particular, I am grateful to the administrative assistants Rita Schoepfer Braschler, Simone Frei, Cyril Burkhard and Natyra Ajvazi at ZHAW along with Susanna Bachmann at UZH, Livia Ioni and Alexandra Cassoli at UNIL. They kept me organized and were always ready to help.

I am much obliged also to my former colleagues at SUPSI, I miss Melissa Gajewsky for the heated discussions and the fun we had together, she has been a special inspiration to me. I will forever be indebted to my former research advisor Prof. Andrea Danani, I am very grateful for his scientific advise and many insightful discussions and suggestions. I still think fondly of my time spent in his lab. He has been the one letting me pursuing a career in research. I really enjoyed the time spent with Andrea Bernasconi. The joy and enthusiasm he has for scientific research was motivational for me. I earnestly thank him for his influence on my professional life. I am especially thankful to Alberto Vancheri for the excellent example he has provided as person and admirable researcher. Alberto was and remains my best role model as scientist, mentor, and teacher. I never encountered another person so clever and at the same time so modest. He is extremely knowledgeable in just about everything, that's the reason I used to call him 'Vancher-pedia'.

I especially thank my advisors, Prof. Christian von Mering at the university of Zurich and Prof. Christophe Dessimoz at the University of Lausanne. Also, the members of my oral defense committee deserve special thanks, Prof. Olivier Gascuel, Prof. Anna-Sapfo Malaspinas, Prof. Nicolas Salamin and Prof. Gerome Goudet. Prof. Andreas Wagner deserve my heartfelt thanks as being an additional member of my oral exam committee at the University of Zurich.

I have greatly appreciated also the contributions of time and suggestions of Prof. Ziheng Yang. I would also like to acknowledge Prof. Marco Comini for sharing the L^AT_EX template I used for writing this thesis. I acknowledge the agency that funded my PhD studies. I was financial supported by the Swiss National Science Foundation.

I would like to express my deepest gratitude to my mom Helen and my dad Vincenzo. My hard-working parents have sacrificed their lives for my brother, my sisters and myself. They instilled in me care for and curiosity in the world around us. It is very sad that my mom left so soon, she would have been proud of this important achievement. It would be inappropriate if I omit to mention also my wife's family, Rosa and Federico, who all have been supportive and caring.

Nevertheless, the most special thank goes to my wife Teresa and my daughter Noemi. Definitely, Teresa is the right person for me. All these years, Teresa has been a true and great supporter. She unconditionally love me during my good and bad times. She has faith in me and my intellect, and she always encouraged me in all possible ways from my Bachelor to my PhD studies. These past several years have not been an easy ride, both academically, financially and personally. But she was always at my side. We have both

learned a lot from each other about life and this experience has strengthened our relationship. Noemi, my darling little princess, she gave and still gives me a reason to fight in the most difficult moments. Decisively, she is the light that brightens my heart. I hope, I can give a better future now that I attained this important milestone. I love you dearly and thank thoroughly for always fostering me. Your presence was indispensable in a process that is often perceived as considerably solitaire. My family has always been someone I can count on. There are no words to convey how deeply I love you.

GOD, I am truly grateful for your exceptional love during this hard but wondrous journey.

*Thank
you!*

Massimo Maiolo
University of Lausanne
December 2018

To my wife Teresa: for
your unconditional love, end-
less support and constant encourage-
ment. You always believed in me and trusted
the decisions I made. You empowered me to
work hard for the things I aspire to achieve.
You are my perfect wife and mother of our
child. To my daughter Noemi: your smile is
the source of my joy and pride. You gave
me the energy I needed for dealing with
all this work. I love you all dearly. You
both are very special and lovely,
the most precious and valu-
able persons in my life. I
am truly thankful
for having
you!

Abstract

Here we present, for the first time, a frequentist progressive Multiple Sequence Alignment (MSA) method based on a rigorous and explicit mathematical formulation of insertions and deletions, namely the Poisson Indel Process (PIP). Having designed our algorithm in the Maximum Likelihood (ML) framework has enabled us to avoid the time-consuming Markov Chain Monte Carlo sampling of alignments. Our proposed algorithm aligns two homology paths, represented by their corresponding MSAs in polynomial time by ML under the PIP model (Maiolo et al. 2017). The procedure has been integrated into a progressive procedure that, traversing a given phylogenetic tree, produces at each internal node the optimal pairwise MSA ending at the root with the alignment of all the input sequences. The integration of PIP equations into a Dynamic Programming (DP) approach is not straightforward since the marginal likelihood is non-monotonic in the alignment length. Therefore, to account for the dependence on alignment length we have extended the DP matrices with a third dimension.

In order to reduce the computational complexity, the algorithm predicts candidate homologous segments for the purpose of filtering out non-promising regions in the DP matrix prior to the effective alignment process. Although our method has been strongly inspired by MAFFT, we have introduced a number of improvements like for example the use of a multi-scale short-time Fourier transform (STFT) for the automatic detection of candidate homologous patterns. Moreover, the use of the multiple-resolution STFT rather than the Fourier transform improves the detection of homologous regions especially in the presence of noise and in case of relative short patterns. We have also defined a more sophisticated and general approach to generate logically sound paths to connect homologous blocks and resolve overlaps between them.

To mitigate the intrinsic greediness brought by the progressive DP approach we have also implemented a Stochastic backtracking (Mueckstein et al. 2002) version of the algorithm under the PIP model. In this way, our progressive algorithm generates at each visited node a distribution of candidate

sub-optimal alignments. Aligning sub-optimal solutions increases the chances to escape from local maxima and in our opinion provides a valid strategy to reduce the progressive bias.

Finally, to account for the among-sites substitution rate variation (ASRV) we have applied a Gamma distribution to all the rates, insertion and deletion rates included. However, further analysis are still needed to investigate the impact of ASRV on the inferred alignments. Our hope is that this feature could mimic to some extent a long insertion, that is, an insertion of more than a single character at a time.

The use of a sound mathematical model of indel, namely the Poisson Indel Process model, is providing more realistic and accurate estimates of MSAs, phylogenies and model parameters. As a consequence, our new algorithms will allow not only more accurate phylogeny and alignment inference but it will also facilitate the estimation of statistical supports of inferred tree partitions and the ancestral reconstruction of insertions-deletions and substitution history. Our tool has been developed in a user-friendly software package and is applicable to large genomic and metagenomic datasets.

Résumé

Nous présentons ici, pour la première fois, une méthode d'alignement progressif de séquences multiples (MSA) basée sur une formulation mathématique rigoureuse et explicite des insertions et des suppressions, à savoir le procédé Poisson Indel (PIP). Le fait d'avoir conçu notre algorithme dans le cadre du maximum de vraisemblance (ML) nous a permis d'éviter l'échantillonnage fastidieux des alignements par la chaîne de Markov Monte Carlo. L'algorithme que nous proposons aligne deux trajectoires homologues, représentées par leurs MSA correspondantes en temps polynomial par ML sous le modèle PIP (Maiolo et al. 2017). La procédure a été intégrée dans une procédure progressive qui, en traversant un arbre phylogénétique donné, produit à chaque nœud interne le MSA optimal par paires se terminant à la racine avec l'alignement de toutes les séquences d'entrée. L'intégration des équations PIP dans une approche de programmation dynamique (DP) n'est pas simple puisque la probabilité marginale est non monotone dans la longueur de l'alignement. Par conséquent, pour tenir compte de la dépendance à la longueur d'alignement, nous avons étendu les matrices DP avec une troisième dimension.

Afin de réduire la complexité des calculs, l'algorithme prédit les segments homologues candidats afin de filtrer les régions non prometteuses dans la matrice DP avant le processus d'alignement efficace. Bien que notre méthode soit fortement inspirée du MAFFT, nous avons introduit un certain nombre d'améliorations comme par exemple l'utilisation d'une transformée de Fourier multi-échelle de courte durée (STFT) pour la détection automatique des modèles homologues candidats. De plus, l'utilisation de la STFT multi-résolution plutôt que de la transformée de Fourier améliore la détection des régions homologues, en particulier en présence de bruit et dans le cas de motifs relativement courts. Nous avons également défini une approche plus sophistiquée et plus générale pour générer des chemins logiquement sains pour connecter des blocs homologues et résoudre les chevauchements entre eux.

Pour atténuer l'avidité intrinsèque apportée par l'approche progressive de la DP, nous avons également mis en œuvre une version rétrospective stochas-

tique (Mueckstein et al. 2002) de l'algorithme selon le modèle PIP. De cette façon, notre algorithme progressif génère à chaque nud visité une distribution des alignements sous-optimaux candidats. L'alignement de solutions sous-optimales augmente les chances d'échapper aux maxima locaux et, à notre avis, fournit une stratégie valable pour réduire le biais progressif.

Enfin, pour tenir compte de la variation du taux de substitution entre les sites (ASRV), nous avons appliqué une distribution gamma à tous les taux, taux d'insertion et de suppression inclus. Toutefois, une analyse plus approfondie est encore nécessaire pour étudier l'impact de l'ASRV sur les tracés déduits. Nous espérons que cette caractéristique pourrait imiter dans une certaine mesure une insertion " longue ", c'est-à-dire une insertion de plus d'un caractère à la fois.

L'utilisation d'un modèle mathématique solide de l'indel, à savoir le modèle du processus de Poisson Indel, fournit des estimations plus réalistes et plus précises des ASM, des phylogénies et des paramètres du modèle. Par conséquent, nos nouveaux algorithmes permettront non seulement une phylogénie et une inférence d'alignement plus précises, mais ils faciliteront également l'estimation des supports statistiques des partitions d'arbres inférées et la reconstruction ancestrale des insertions-suppressions et de l'historique des substitutions. Notre outil a été développé dans un progiciel convivial et est applicable aux grands ensembles de données génomiques et métagénomiques.

Contents

List of Symbols	xxvii
List of Acronyms	xxxv
Introduction	xxxvii
Main result of the thesis	lv
Outline	lvii
I Progressive Multiple Sequence Alignment with In- del Evolution	1
1 Progressive Dynamic Programming under PIP model	3
1.1 Introduction	3
1.2 Likelihood computation of a column $p(c)$	5
1.3 Likelihood computation of a column $p(c_\emptyset)$	7
2 3D Dynamic Programming under PIP	11
2.1 Introduction	11
2.2 Alignment at node v_3	14
2.3 Alignment at node v_5	20
2.4 Alignment at node Ω	24
2.5 Tracebacking	25
2.6 Early stop condition	27
3 Stochastic backtracking DP algorithm	29
3.1 Introduction	29
3.2 Partition functions	30
3.3 The partition functions \mathbf{Z}^M , \mathbf{Z}^X and \mathbf{Z}^Y	31
3.4 Forward recursion	33
3.5 Backward recursion	38
3.6 The temperature parameter	43

4	Marginal Likelihood with Rate Variation Across Sites	49
4.1	Introduction	49
4.2	PIP equations under ASRV	51
5	Multi-scale STFT based homologous blocks detection	55
5.1	Introduction	55
5.2	Fourier transform	58
5.3	Short-time Fourier transform	59
5.3.1	The Heisenberg Uncertainty Principle	62
5.4	Algorithm overview	63
5.4.1	Sequence residues to signal conversion	68
5.4.2	Signal padding	70
5.4.3	Fourier transform based homology detection	71
5.4.4	Multi-scale STFT based homology detection	73
5.4.5	Homology matrix and optimal path	83
5.4.6	Final alignment	91
5.5	STFT vs. FT approach for block detection	93
5.5.1	Noise sensitivity	93
5.5.2	Pattern length sensitivity	94
6	Progressive bias analysis	97
6.1	Introduction	97
6.2	Alignment with unbalanced tree	99
6.2.1	Global alignment	99
6.2.2	Progressive alignment	101
6.2.3	Progressive bias with unbalanced tree	102
6.3	Alignment with balanced tree	104
6.3.1	Global alignment	104
6.3.2	Progressive alignment	104
	Discussion & Conclusions	107
II	Appendices	113
A	Some technicalities	115
A.1	Detailed derivation of the marginal likelihood function $\varphi(v)$. . .	115
A.2	Detailed derivation of the survival probability function $\beta(v)$. . .	120
A.3	Overview of PIP equations	123

B	Reversibility of TKF91 and PIP	125
B.1	Introduction to TKF91	125
B.2	Time-reversible evolutionary process	128
B.3	Reversibility of TKF91	129
B.4	Reversibility of PIP	133
C	Characterization of Indel rates	137
C.1	Introduction	137
C.2	Inferring indel rates from a given MSA	139
D	Doob-Gillespie method and PIP description	145
D.1	Doob-Gillespie method	145
D.1.1	When does the next event happen?	146
D.1.2	What kind of event happens next?	148
D.2	Local PIP description	148
E	Grantham's distance	151
E.1	Introduction	151
E.2	Grantham's distance computation	152
F	Homologous blocks overlap resolution	159
G	Multiple sequence alignment evaluation	165
G.1	Overview	165
G.2	Benchmarks	167
	Bibliography	173
III	Journal article	189
	Progressive multiple sequence alignment with indel evolution	191

List of Figures

1	Tree of life.	xxxviii
2	Genetic variations at the molecular level.	xl
3	DNA, RNA and protein synthesis.	xlii
4	MSA and corresponding homology paths.	xlv
5	Progressive Dynamic Programming.	xlviii
1.1	Rooted topology used to illustrate the PIP-DP formulation.	4
2.1	Phylogenetic tree τ	12
2.2	Four three-dimensional sparse DP matrices.	12
2.3	An example of $\varphi(m)$	13
2.4	Cells computed at layer $m = 0$	14
2.5	Homology paths at layer $m = 0$	16
2.6	Cells computed layer $m = 1$	17
2.7	Homology path representing a match.	18
2.8	Homologous scenarios at the node v_3 . 1.	19
2.9	Homologous scenarios at the node v_3 . 2.	19
2.10	Cells computed at layer $m = 2$	21
2.11	Homology paths at node v_5 . 1.	22
2.12	Homology paths at node v_5 . 2.	23
2.13	Homology paths at node v_5 . 3.	24
2.14	Homology paths at node v_5 . 4.	24
2.15	Homology paths at the root Ω	25
2.16	Possible starting points of the traceback phase.	26
2.17	Cells with likelihood different from 0.	27
2.18	Likelihood values of the last column.	28
3.1	Six partition functions.	33
3.2	Paths connected to $\mathbb{P}^M(3, 3, 4)$	34
3.3	Cell dependencies in the 3D-DP matrices.	34
3.4	Possible homologous paths at $\mathbf{Z}^M(3, 3, 4)$	37
3.5	Possible homologous paths at $\mathbf{Z}^X(3, 3, 4)$	38
3.6	Possible homologous paths at $\mathbf{Z}^Y(3, 3, 4)$	39
3.7	Last column in the 3D-DP matrices.	40
3.8	Backtracking phase.	41

3.9	Probability distortion as function of the temperature.	44
3.10	Probability distortion at $T = 0.1$	45
3.11	Probability distortion at $T = 0.3$	45
3.12	Probability distortion at $T = 10$	46
3.13	Stochastic backtracking paths at different temperatures.	47
4.1	Gamma probability distribution functions.	51
5.1	Theoretical speed-up as function of the number of blocks.	57
5.2	Windowed signal.	60
5.3	Tiling of the time-frequency space.	63
5.4	FT coefficients f_k	65
5.5	Column-gap content.	66
5.6	Overlapping and non-overlapping blocks.	67
5.7	Padding scheme with $S_w = 64$	71
5.8	Padding scheme with $S_w = 32$	71
5.9	Window functions examples.	74
5.10	Boundaries effects at different resolution scales.	75
5.11	STFT spectrogram.	76
5.12	First analysis at two different window size.	77
5.13	Window functions at different resolution level.	78
5.14	Spectrogram slice at different sliding length.	79
5.15	Noise threshold convergence.	80
5.16	Multi-scale STFT surfaces.	81
5.17	Blocks boundaries analyzed at different scales.	82
5.18	Multi-resolution boundary detection.	84
5.19	Example of overlapping blocks.	85
5.20	Tree of blocks representing possible paths.	86
5.21	Example of overlapping blocks and their resolution. i).	87
5.22	Example of overlapping blocks and their resolution. ii).	87
5.23	Example of overlapping blocks and their resolution. iii).	88
5.24	Example of overlapping blocks and their resolution. iv).	88
5.25	Overlap-free paths.	89
5.26	Homologous and linking blocks.	90
5.27	Magnification of connecting corner.	90
5.28	Homologous and linking blocks and magnification.	90
5.29	MSA obtained with/without STFT.	92
5.30	Noisy pattern experiment.	93
5.31	Noise sensitivity.	94
5.32	Variable pattern experiment.	94

5.33	Pattern length.	95
6.1	Topologies used to quantify the progressive bias.	98
6.2	Marginal likelihood curves at the root node Ω	101
6.3	Marginal likelihood curves at the root node v_2	103
6.4	Marginal likelihood curves at the root node v_1	103
6.5	Marginal likelihood curves at node v_1	105
6.6	Marginal likelihood curves at the root Ω	106
6.7	Progressive bias analysis.	106
A.1	Graphical representation of U and W	120
B.1	Time reversibility.	129
B.2	Four pairwise alignments.	131
B.3	Two pairwise alignments.	132
B.4	Topology used to test the reversibility of PIP.	133
B.5	Representation of likelihood under PIP.	135
C.1	Possible fate of characters.	138
C.2	Number of characters through time.	140
C.3	n_H , n_G and $n_H + n_G$ as function of I	142
C.4	n_H , n_G and $n_H + n_G$ as function of τ	142
C.5	n_H , n_G and $n_H + n_G$ as function of λ	143
C.6	n_H , n_G and $n_H + n_G$ as function of μ	143
D.1	Doob-Gillespie method under PIP. Step 1.	150
D.2	Doob-Gillespie method under PIP. Step 2.	150
D.3	Doob-Gillespie method under PIP. Step 3.	150
D.4	Doob-Gillespie method under PIP. Step 4.	150
E.1	Standardized and non standardized volumes.	153
E.2	Cross-correlation vs. Grantham's distance approach.	154
E.3	Grantham's distance coefficients.	157
F.1	Overlaps table.	163
G.1	BALiBASE benchmark dataset RV911-115: phylogenetic tree.	168
G.2	Evolutionary vs Structural alignment. Columns (1-200).	168
G.3	Evolutionary vs Structural alignment. Columns (201-400).	169
G.4	Evolutionary vs Structural alignment. Columns (401-600).	169
G.5	Evolutionary vs Structural alignment. Columns (601-800).	170
G.6	Evolutionary vs Structural alignment. Columns (801-1000).	170

- G.7 Evolutionary vs Structural alignment. Columns (1001-1200). . . 171
- G.8 Evolutionary vs Structural alignment. Columns (1190-...). . . 171

List of Tables

5.1	Amino-acids physiochemical properties.	69
6.1	MSAs potentially be generated at node v_1	99
6.2	MSAs potentially be generated at node v_2	99
6.3	MSAs potentially be generated at the root Ω	100
G.1	PIP based MSA from BAliBASE datasets.	172

List of Algorithms

1	Doob-Gillespie-PIP procedure	149
2	Overlap resolution procedure	160
3	Create block tree procedure	160
4	Expand node procedure	161
5	Resolve all overlaps procedure	161
6	Resolve single path procedure	161
7	Resolve pairs procedure	162
8	Resolve overlap pairs procedure	162
9	Search the best path procedure	162

List of Symbols

- $H(v)$ denotes an homology path of a single character generated by a substitution-deletion continuous-time Markov Chain process along a the phylogeny. 123
- $L(\rho(v_j, v_k))$ length of the path that connects v_j to v_k . Corresponds to the sum of branch length $b(v)$ for all the edges connecting the two end points (v_j and v_k). 3
- Γ symbol for the gamma distribution function. The density probability function is defined as $f(x) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} e^{-\frac{x}{\theta}} = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$ with (k, θ) and (α, β) positive numbers. 51
- Ω phylogenetic tree root. The root represents the most recent common ancestor of all of the taxa in the tree. 52–54, 123
- \mathbf{S}^M sparse 3-dimensional Dynamic Programming matrix. It contains the likelihood values for *match* state. xxvii, 14, 16, 18, 27
- $\mathbf{S}^{M,X,Y}$ short notation for \mathbf{S}^M , \mathbf{S}^X and \mathbf{S}^Y . 16, 18, 21, 26–28
- \mathbf{S}^T sparse 3-dimensional Dynamic Programming traceback matrix. It stores at each position (i, j, k) which matrix contains the highest likelihood value. By convention the index 1 stands for \mathbf{S}^M , 2 for \mathbf{S}^X and 3 for \mathbf{S}^Y . 14, 25–27
- \mathbf{S}^X sparse 3-dimensional Dynamic Programming matrix. It contains the likelihood values for *gapX* state. xxvii, 14, 16, 23, 27
- \mathbf{S}^Y sparse 3-dimensional Dynamic Programming matrix. It contains the likelihood values for *gapY* state. xxvii, 14, 16, 27
- $\|\nu\|$ normalizing measure for the Poisson process. It corresponds to $\|\nu\| = \lambda \|\tau\| + \frac{\lambda}{\mu}$. 17, 18, 20, 116, 139, 140
- $\|\tau\|$ tree length obtained summing up all the branch lengths $b(v)$ for $v \in \mathcal{V}$. xxxiii, 6, 18, 52, 116, 133, 134, 138–141

- $\alpha(v)$ corresponds to the sum of all the prior insertion probabilities $\iota(v)$ and survival probabilities $\beta(v)$ from the root to a particular node v on the path $\rho(v, \Omega)$. It follows that for $v \in \mathcal{V}$: $\alpha(v) = \frac{1/\mu}{\|\tau\|+1/\mu}$. 7
- * hadamard product between two arrays. The hadamard product between two arrays x and y of same size corresponds to $(\mathbf{x} * \mathbf{y})(i) = x(i) \cdot y(i)$. 5, 17, 22, 23, 133
- $\beta(v)$ survival probability associated to the node v given an insertion, on a random location, on the edge (pa \rightarrow v) of length $b(v)$. See Appendix A.2 for its detailed derivation.. xxviii, 4, 120, 123
- $\boldsymbol{\pi}_\epsilon$ extended steady state frequencies. It is represented as an array of size $|\mathcal{A}_\epsilon| \times 1$ obtained as the quasi-stationary distribution of \mathbf{Q}_ϵ . $\boldsymbol{\pi}_\epsilon(i)$ contains the background frequency of the character at the position i in the extended alphabet \mathcal{A}_ϵ . The steady state probability of a gap is always 0. See also $\boldsymbol{\pi}$. 17, 133–135, 150
- $\boldsymbol{\pi}$ steady state frequencies. For instance using a nucleotide alphabet, the stationary frequencies, $\pi_A, \pi_C, \pi_G, \pi_T$ correspond to equilibrium base compositions of the four states. The background frequencies are stored into an array of size $|\mathcal{A}| \times 1$ and computed as the limiting distribution when $t \rightarrow \infty$, that is $\boldsymbol{\pi} \exp(\mathbf{Q}t) = \boldsymbol{\pi}$. xxviii
- o dot product between two arrays. The scalar (dot) product between two arrays x and y of same size corresponds to $\mathbf{x} \circ \mathbf{y} = \sum_i x(i) \cdot y(i)$. 5, 15, 17, 20–24, 52, 54, 133
- $\delta_\Omega(dt)$ (continuous) dirac function defined on the topology. It returns 1 at the root node and 0 elsewhere. 150
- ◇ it means ‘putatively homologue to’. Hence, $Ai \diamond Bj$ means that Ai is (putatively) homologue to Bj . 16, 18, 19, 22, 31, 32, 36, 85
- ϵ it denotes a gap state which is an absorbing state in the Markov chain under the PIP model. It means that once the system enters in gap state it cannot change anymore, or in other words, a gap never gives birth to any character. 15, 123
- $\eta(v)$ recursive formula associated to the node v that computes the character deletion probability on the sub-tree rooted at node v . 8

- $\iota(v)$ prior probability of a single character insertion on the branch associated to v with $\sum_{v \in \mathcal{V}} \iota(v) = 1$. The prior insertion probability of a character along the edge belonging to $v \in \mathcal{V} \setminus \Omega$ is proportional to its length $b(v)$. xxviii
- λ single character insertion rate. λ is kept constant during the entire evolutionary process (static or time-invariant parameter) and does not depend on the particular character inserted. 116, 138–141
- \mathbb{E} denotes the expectation function. 50, 51, 115
- \mathbb{P} symbol for the probability function. 126
- \mathbf{Q}_ϵ extended substitution rate matrix with an extra column and an extra row to account for the gap character (absorbing state ϵ). With the extended alphabet of nucleotide for instance \mathbf{Q}_ϵ has dimension of 5×5 . See also **Q**. 52, 123
- Q** infinitesimal generator matrix completely describing the Markov process. **Q** contains the rates a continuous time Markov chain moves between states. Therefore $\mathbf{Q}_{i,j}$ is the rate at which character i mutates into character j . By using the nucleotide for instance **Q** is a matrix of size 4×4 . xxix
- X** set of input sequences. A sequence is a string of characters belonging to a given alphabet \mathcal{A} . Typically there are three kind of standard alphabets: nucleotides, amino-acids or codons alphabet. 115
- $\mathbf{Z}^{\mathbf{M},\mathbf{X},\mathbf{Y}}$ short notation for $\mathbf{Z}^{\mathbf{M}}$, $\mathbf{Z}^{\mathbf{X}}$ and $\mathbf{Z}^{\mathbf{Y}}$. 33, 36, 38, 41, 43
- $\mathbf{Z}^{\mathbf{M}}$ sparse 3-dimensional dynamic matrix containing the partition function of alignments ending in state *match*. 31, 32, 35, 36, 40
- $\mathbf{Z}^{\mathbf{X}}$ sparse 3-dimensional dynamic matrix containing the partition function of alignments ending in state *gapX*. 31, 32, 36, 37
- $\mathbf{Z}^{\mathbf{Y}}$ sparse 3-dimensional dynamic matrix containing the partition function of alignments ending in state *gapY*. 31, 32, 36, 37
- Z** total partition function obtained summing up $\mathbf{Z}^{\mathbf{M}}$, $\mathbf{Z}^{\mathbf{X}}$ and $\mathbf{Z}^{\mathbf{Y}}$. 35
- \mathcal{A}_ϵ extended alphabet. It is defined as $\mathcal{A}_\epsilon = \mathcal{A} \cup \epsilon$ where ϵ is an absorbing state (gap character). Under PIP a gap is considered as an extra state added to the canonical ones. See also **A**. xxviii, xxxii, liii, 5, 123

- \mathcal{A} represents the alphabet. A sequence is commonly represented as a string of characters belonging to a given alphabet. The three alphabets commonly used are of type *nucleotides*, *amino-acids* or *codons*. For instance the nucleotide alphabet contains the four letters $\{A, T, G, C\}$. xxix, xxxi, liii, 3, 123, 133
- \mathcal{B} set of candidate homologous blocks detected by the Fourier Transform or the short-time Fourier Transform. Blocks are then aligned as independent DP sub-problems.. 83, 159
- \mathcal{E} set of edges. An edge is a link connecting v with $\text{pa}(v)$ and has associated a length indicated as $b(e) = b(v)$. xxxi, xxxiii
- \mathcal{I} set of vertices $\mathcal{I} \cup \mathcal{V}$ where an insertion could have happened. It corresponds to the set of vertices having a felsenstein's weight different from 0. liv, 52, 123
- \mathcal{L} set of leaves where $\mathcal{L} \subset \mathcal{V}$. Leaves are nodes without children. The input sequences are associated at the leaves before aligning them for instance with a progressive DP procedure. 8, 52, 123
- \mathcal{P} set of candidate paths between candidate homologous blocks. Prior to independently align the single blocks an algorithm has to connect logically, all the blocks belonging to a path \mathcal{P} that respect the order of the columns and avoid any duplication.. xxxii, 83, 86, 159, 160
- \mathcal{T} nth tree that represents all possible paths connecting in a meaningful way the detected homologous blocks.. 83
- $\mathcal{U}(0, t)$ uniform (continuous) distribution on the interval $(0, t)$, with $t > 0$. 120
- \mathcal{V} finite subset of vertices (branching points in the topology) where $\mathcal{V} \subset \tau$, in fact under PIP the topology is considered as a continuous set of points. 52, 54, 116, 123
- $\mathcal{W}(\mu)$ exponential distribution with parameter μ , describes for instance the (waiting) time between consecutive events in a Poisson point process. 120
- \mathcal{W}^L denotes the Lambert-W function. The Lambert-W function is a set of functions (branches) of the function $f(z) = z \exp(z)$ where z is a complex number, therefore $z = f^{-1}(z \exp(z)) = \mathcal{W}^L(z \exp(z))$. For any complex number $z_0 = z \exp(z)$ we get $z_0 = \mathcal{W}^L(z_0) \exp(\mathcal{W}^L(z_0))$. 141

- $\mathbb{1}$ indicator function. The indicator function returns the value 1 if the function argument is true, 0 otherwise. liv, 3–5, 52
- character symbol for the gap character, its state is denoted with the simbol ϵ . 15
- Poi stands for Poisson probability/distribution. 137, 138, 150
- $\text{child}(v)$ child node of v , the next node (below), directed linked to v . The topologies considered here are always binary trees, it follows that each internal node $v \in \mathcal{V} \setminus \Omega$ has degree 3 (2 children). The root Ω has degree 2 (no parent node) and leaves do not have any children (degree 1). Accordingly, $\text{pa}(\text{child}(v)) = v$ for $v \in \mathcal{V} \setminus \Omega$. xxxi
- $\text{pa}(v)$ parent node of v , the next node (above), directed linked to v . Only the root Ω does not have any parent node. It follows that $\text{child}(\text{pa}(v)) = v$ for $v \in \mathcal{V} \setminus \Omega$ (see also $\text{child}(v)$). 3, 4
- μ single character deletion rate. μ is kept constant during the entire evolutionary process (static or time-invariant parameter) and does not depend on the particular character deleted. 116, 120, 121, 126, 130, 132, 140, 141
- $\nu(dt)$ Poisson process intensity defined on the topology, the latter considered a continuous set of point. Under PIP this intensity is defines as $\nu(dt) = \lambda \cdot (\tau(dt) + \frac{1}{\mu} \cdot \delta_{\Omega}(dt))$. 150
- $\rho(v_j, v_k)$ it denotes the set of nodes in the path v_j to v_k where $v_j, v_k \in \mathcal{V}$. A path is a connected sequence of edges $e \in \mathcal{E}$ and is defined on the topology. 3
- \setminus the set difference operator. It is defined as $A \setminus B = \{x : x \in A \text{ and } x \notin B\}$. 27, 52–54, 123
- σ represent a character from the set of characters of a given alphabet \mathcal{A} or \mathcal{A}_{ϵ} . 123
- $\tau(dt)$ Lebesgue measure on the topology. Its value corresponds to the distance measured from a given point to the root. 150
- τ phylogenetic tree. Under PIP the phylogeny τ is a continuous set of points, its topology is denoted by $(\mathcal{L}, \mathcal{E})$, where $\mathcal{V} \subset \tau$ is equal to the finite subset of vertices (branching points), the leaves $\mathcal{L} \subset \mathcal{V}$ and the root Ω , and where \mathcal{E} is the set of edges. 11, 51, 115, 138, 141

- θ is the collection of model parameters (branch lengths, substitution rate matrix, insertion/deletion rate, ...). 126, 127, 130
- $\varphi(p(c_\emptyset), |m|)$ marginal likelihood of non-observable empty columns for an alignment of length $|m|$ where $p(c_\emptyset)$ is the likelihood of a single MSA column full of gaps. See Appendix A.1 for its detailed derivation. 53, 115, 116
- $|\mathcal{A}_\epsilon|$ cardinality (number of elements) of the (extended) alphabet \mathcal{A}_ϵ . For the nucleotide alphabet, for instance, $|\mathcal{A}_\epsilon| = 5$. See also \mathcal{A}_ϵ . 5
- $|\mathcal{B}|$ cardinality of the set \mathcal{B} which corresponds to the total number of homologous blocks detected by the Fourier Transform or short-time Fourier Transform. 83
- $|m|$ number of (observed) columns in an alignment m . An alignment m contains n columns, where $|m|$ are observable and $n - |m|$ are full of gaps and hence in general are not represented in an alignment. 13, 14, 17, 21, 51, 115–117, 140
- $\widehat{\mathbf{Z}}^M$ sparse 3-dimensional dynamic matrix containing the partition function of alignments starting in state *match* and ending once all the characters have been inserted into the alignment. 31
- $\widehat{\mathbf{Z}}^X$ sparse 3-dimensional dynamic matrix containing the partition function of alignments starting in state *gapX* and ending once all the characters have been inserted into the alignment. 31
- $\widehat{\mathbf{Z}}^Y$ sparse 3-dimensional dynamic matrix containing the partition function of alignments starting in state *gapY* and ending once all the characters have been inserted into the alignment. 31
- $\widetilde{\mathcal{P}}$ set of overlap-free paths. See Appendix F for a description of the algorithm that resolve the block overlap for a path $p \in \mathcal{P}$. 87, 160
- $\xi(v)$ it represents the non-survival probability associated to the node v obtained as $\xi(v) = 1 - \zeta(v)$ which corresponds therefore at the complementary probability to the survival probability ζ . 4
- $\zeta(v)$ ‘pure’ survival probability associated to the node v given an insertion, on a random location, on the path ($\text{pa} \rightarrow \Omega$). Differently from the function $\beta(v)$ the character is already present at $\text{pa}(v)$ whereas in $\beta(v)$ the character is inserted on a random location along the edge belonging to v . 3, 4

- $b(v)$ branch length associated to node v , which is the length of the edge from $\text{pa}(v)$ to v . 123
- c_\emptyset MSA column containing only gap characters. Such columns are not observable in an MSA and the number thereof is unknown. xxxiii, 9, 13–15, 18, 21, 39, 51, 53, 115, 123
- e it denotes an edge with $e \in \mathcal{E}$ between v and $\text{pa}(v)$ with $b(e) = b(v)$. 3, 120, 121
- f_v felsenstein’s weight. It corresponds to the likelihood under the given topology of a given MSA column accounting for the process of substitution and deletion. See Appendix A.3 for more details. 123
- n_G number of columns containing gaps (but not full of gaps), corresponds to $n_G = \lambda \cdot \|\tau\| + \frac{\lambda}{\mu} \left[1 - \exp(-\|\tau\|\mu) \right]$. See Appendix C for more details. 141
- n_H number of columns without gaps, where $n_H = \frac{\lambda}{\mu} \exp(-\|\tau\|\mu)$. See Appendix C for more details. 141
- $p_\tau(m)$ marginal likelihood function under PIP of an alignment m given a phylogenetic tree τ , marginalized over ancestral states (see the original paper for more details [13]). 115
- $p_v(c_\emptyset)$ likelihood of an MSA column full of gaps computed rooting the topology at node v . When omitted, $v = \Omega$. 9
- \top array or matrix transposition operator. 5

List of Acronyms

- ASRV** Across Sites Rate Variation. lii, 11, 49
- CDF** cumulative distribution function. 146
- CTMC** Continuous-Time Markov Chains. 125, 149
- DP** Dynamic Programming. xlvii, 11, 23, 29, 33, 46, 56
- FFT** Fast Fourier Transform. 56, 57, 64, 68, 151
- indel** stands for insertions and/or deletions. xli, 140
- ML** Maximum Likelihood. 56
- MSA** Multiple Sequence Alignment. xlv, xlvi, 29
- PDF** probability distribution function. 51, 121, 147
- PIP** Poisson Indel Process (see [13]). 31, 146
- SB** Stochastic Backtracking. 30, 31, 34, 36–39, 44–46
- SP** sum-of-pairs. xlvi, xlvii, 165
- STFT** short-time Fourier Transform. 57, 64, 67, 73

Introduction

“We are at the very beginning of time for the human race. It is not unreasonable that we grapple with problems. But there are tens of thousands of years in the future. Our responsibility is to do what we can, learn what we can, improve the solutions, and pass them on.”

– Richard P. Feynman



EVOLUTION is the genetic code alteration of a population over time mainly driven by natural selection. The modern theory of evolution is supported with evidences by many scientific domains, not only from biological sciences but also, to mention a few, from geology, anthropology, chemistry, physics, mathematics, astrophysics, psychology as well as behavioral and social sciences. In 1859, *Charles Darwin* published the seminal book *On the Origin of Species* where he elaborated the scientific theory of evolution by natural selection. Since then, many other discoveries in the life sciences endorsed or extended his hypothesis, that is, all organisms are to some extent related, species evolve as response to natural selection. To quote *Theodosius Dobzhansky*, a prominent Ukrainian-American geneticist and evolutionary biologist,

“Nothing in biology makes sense except in the light of evolution”.

Nonetheless, natural selection is not the single evolutionary driving force for the biodiversity, variations occur for example also via mutations, sexual selection, genetic drift, and artificial selection. However, selection can occur only if traits exhibits some genetic variability within a population. Although evolution happens at the level of population rather than at the single individual level, heritable changes take place in individuals and those bringing any phenotypic advantage are more likely to be passed at the next generations. Therefore, the frequency of advantageous alleles increases with the time while that of disadvantageous ones tends to diminish. These individual phenotypic changes within a population are modifying the distribution of phenotypic values over time. When the phenotypic shift brings some benefit to the fitness, than it is likely that this modification is conserved over generations.

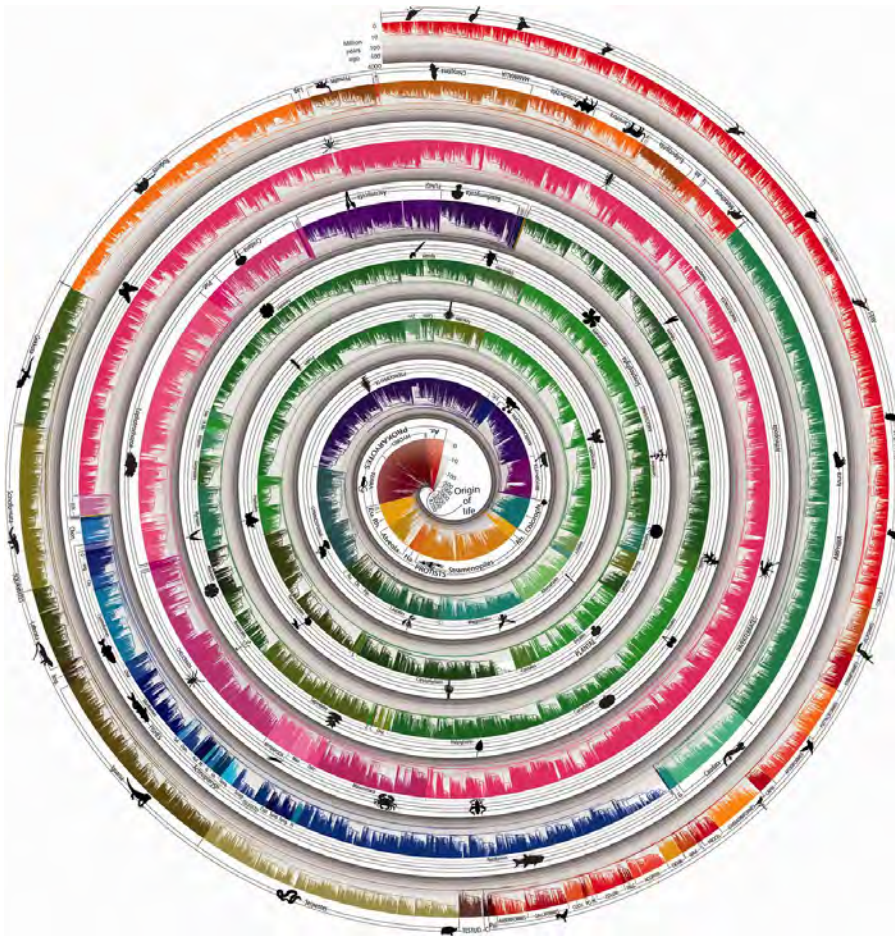


Figure 1: Tree of life reveals clock-like speciation and diversification. A timetree of 50,632 species synthesized from times of divergence published in 2,274 studies. Evolutionary history is compressed into a narrow strip and then arranged in a spiral with one end in the middle and the other on the outside. Therefore, time progresses across the width of the strip at all places, rather than along the spiral. Time is shown in billions of years on a log scale and indicated throughout by bands of gray. Major taxonomic groups are labeled and the different color ranges correspond to the main taxonomic divisions of our tree.

Image and caption reproduced with permission from Blair Hedges et al. [51].

Along with natural selection processes, also catastrophic events or environmental changes may induce changes in the gene pools of a population. This population gene diversity is actually crucial to survive within a dynamically changing environment. A wide assortment of genes increases the probability that some individuals possess the differentiated allele needed to better adapt

to the environment. In small population sizes the genetic variability in general decreases as the inbreeding or the mating between individuals having similar genetic makeup is more likely to occur. Less biodiversity implies, in turn, less chances of coping with environmental variability.

Genetic variation increases with the spontaneous generation of new alleles but also because of gene flowing (horizontal/lateral gene transfer) from other individuals which introduce in this way new traits from foreign populations. These phenomena along with natural selection, establish part of the natural process yielding towards evolution and speciation. Indeed, life continues to evolve through speciations and extinctions, two key mechanisms controlling biodiversity. These events are depicted graphically by means of phylogenetic trees. An example is provided in Figure 1.

However, speciation and extinction are events happening at different rates. Speciation and extinction rates establish the frequency at which new species originate and are lost over evolutionary time. Typically, species that promote speciation are also highly specialized to a particular environment, are often isolated or have a low population size. The same characteristics are associated also with extinction. Actually, specialized species are susceptible to environmental change and therefore are predisposed to disappear rapidly. Speciation can occur slowly and gradually often described by point mutations or appear with silent periods interrupted by event “bursts”. As a result, when two populations that have diverged from a common ancestor become reproductively distincts, i.e. capable of interbreeding and exchanging genetic information to produce fertile offspring, a new species arise.

The carriers of the genetic information are DNA and RNA, they constitute the building elements of any genetic code and of different pathways. These “building blocks” are conserved across all the species while the encoded information may changes. Observing the conserved core processes and the essential features shared among all organisms, both extinct or extant, suggests that all organisms descent from a common ancestry (see Figure 1). In fact, some of these mechanisms are so sophisticated that the probability that they emerged independently several times is almost negligible. All living beings endowing these features, and from which diversification occurred, led to the development of the three domains: Archaea, Bacteria, and Eucarya. The universal common ancestor had as its genetic material the DNA which, through transcription and translation, expressed its genetic code. Phylogenetic trees show graphically the evolutionary history of acquired and lost traits during the evolution.

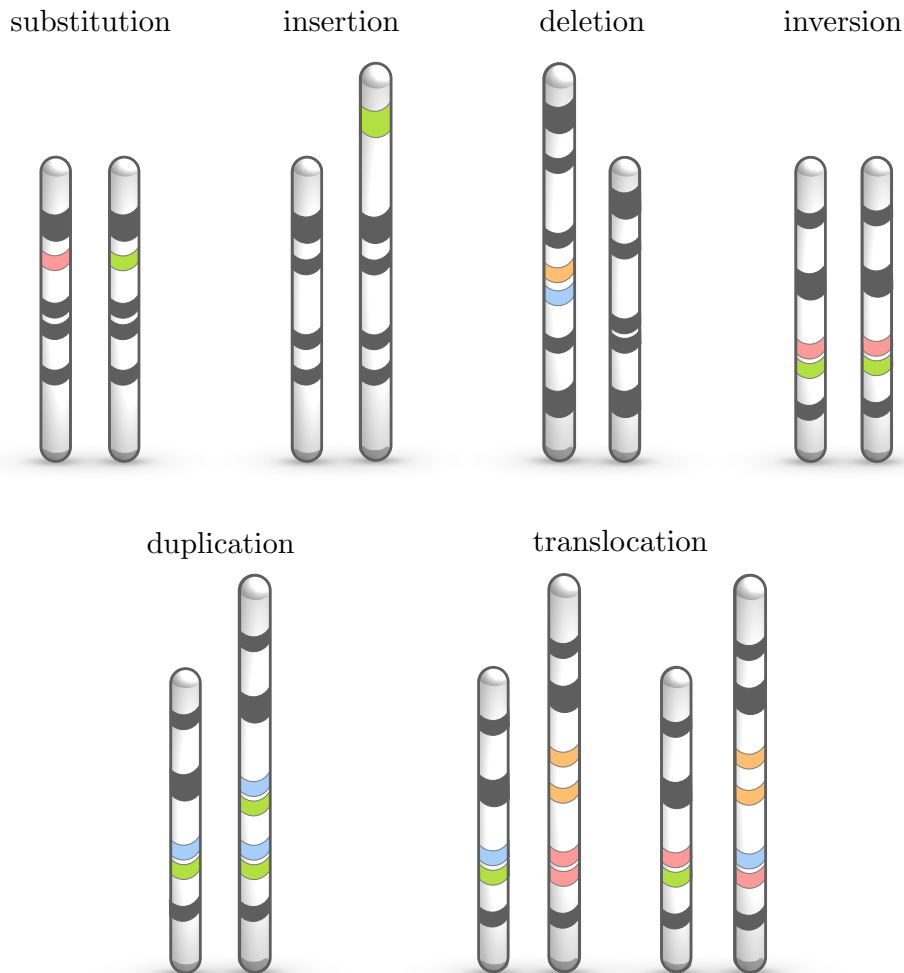


Figure 2: Genetic variations at the molecular level. The main processes leading to a genetic variation are substitution (point mutations), insertions (of one or more characters), deletions (of one or more existing characters), translocations (the change of location of a chunk of characters), duplications (the copying of a block of characters one or many times, notably tandem repeats when the copies are placed at the immediate adjacent locations) and inversions (the rearrangement in which a segment of is reversed end to end).

At the molecular level the main processes leading to a genetic variation are *substitution* (point mutations), *insertions* (of one or more characters), *deletions* (of one or more existing characters), *translocations* (the change of location of a chunk of characters), *duplications* (the copying of a block of characters one

or many times, notably tandem repeats when the copies are placed at the immediate adjacent locations) and *inversions* (the rearrangement in which a segment of is reversed end to end). See Figure 2 for a graphic representation. While substitutions keep the sequences at a constant length, insertions and deletions (indel) are breaking the collinear order. Another cause of sequence-length modification has to be ascribed at tandem repeats (see for example [56, 90, 128]) whose mathematical description, among other things, is still very poor, yet. For the same reason small inversions are frequently treated by the inference tools as multiple adjacent substitutions [47, 64, 65, 123].

Usually mutations are considered as random events. This is true when we are considering whether they bring or not any advantage. However, not all types of mutations happen with equal probability, some appear more frequently than others. Such changes are modeled mathematically with the goal of providing evidence for the evolution and bringing new insights about its dynamics. Unfortunately, the mathematical modeling of the evolution is not so precise and detailed as how it is achieved for example in theoretical physics. Ultimately, one of the main goal of the evolutionary models are not meant to predict future outcomes but rather to reconstruct past histories. Such models aim to to build a general framework for understanding various phenomena at the molecular level and the analysis of genomics data [143]. Evolutionary-based methods are thus key elements for gaining insights in many exciting area of biological research spanning from developmental biology [133] tree of life inference [28], studies of epidemiology and virulence [8], drug design [71], human genetics [130], cancer [169], biodiversity [29], secondary [114] and tertiary structure analysis of RNAs and proteins [20], and protein function prediction analysis [66], just to mention a few.

Multiple sequence alignments

New emerging technologies are quickly providing genomic sequence data at an exponential rate producing Giga basepairs per machine per day [21, 79]. Among the most important projects in that area one can mention, for instance, the Human Genome Project [55] and the Genome 10K Project [110]. The collected data can be used for studying genome evolution and genetic diversity. In this context genome alignment has become a key instrument for shedding new light on the evolution dynamics, for instance for detecting the presence of rearrangements, duplications, and large-scale sequence gain and loss. Then, from the inferred alignment, one can build the phylogenies, which in turn can

be used to determine orthology, or to find recently duplicated regions as well as to identify species-specific DNA.

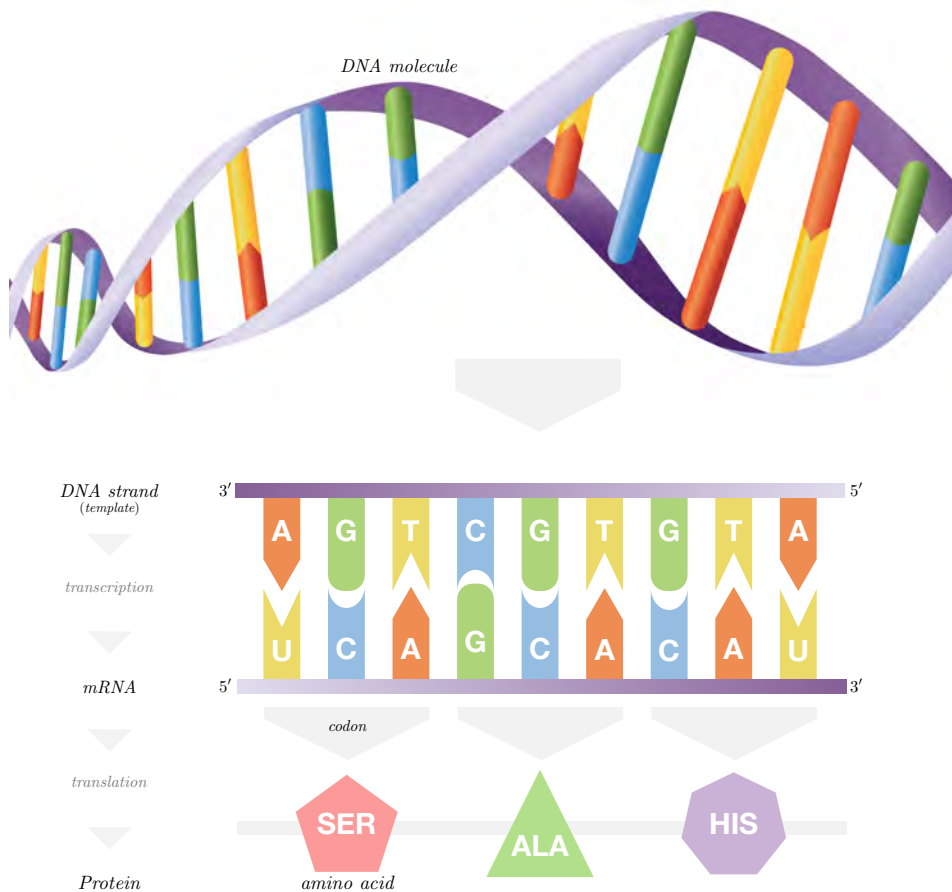


Figure 3: DNA, RNA and protein synthesis. The nucleotide triplet that encodes an amino acid is named a codon. Each nucleotide triplet encodes one amino acid. There are 64 combinations of 4 nucleotides taken three at a time and only 20 amino acids, this means that the translation code is degenerate (there are more codons per amino acid). The transcription is the process by which DNA is copied to RNA, while the translation is the process by which RNA is used to produce proteins. Source of the DNA image on top: Colourbox:ID# 5823998.

Indeed, the development of molecular phylogenetics and the increasing availability of molecular data have enabled unprecedented understanding of evolution at levels from genes to genomes offering a great potential to discover new cellular mechanisms and previously unknown relationships. An integrated study of evolutionary phenomena can unveil their underlying mechanisms that

of course are results of the interplay of phylogeny, form, function and molecular evolution. Evolutionary thinking provides a powerful framework for disentangling underlying biological mechanisms, thus revealing and understanding the patterns and processes of microevolution.

An integral part of contemporary phylogenetics is the development of mathematical models and effective algorithmic solutions to tackle high-complexity computational problems of building evolutionary scenarios, inferring patterns of coevolution of molecules, pathways, regulation systems, and species and assembling of sequence and tree data. A robust phylogenetic hypothesis is intricately linked to the quality of sequence alignments. Sequence alignment in a phylogenetic analysis is about assessing homologies: the residues aligned should be homologous in the evolutionary sense. Accurate alignments of multiple genomics sequences are of chief importance for many evolutionary analyses. However, lacking an explicit evolutionary insertion/deletion model, these methods are prone to cause biased gap placement and evolutionary rate inferences, negatively affecting downstream analyses.

In a multiple sequence alignment, a set of related sequences are organized into a matrix where “similar” (structurally or evolutionary) residues are placed in the same column (see Figure 4 for an example of an evolutionary MSA). This strings vector, whose components are characters representing the residues encoded from a one-to-one mapping alphabet (usually nucleotide, amino acids or codons, see for example Figure 3), is obtained by using *edit operations*. In the classic approach the edit operations consist of match, substitution, insertion and deletion operations. A match operation consists in putting in the same column equal residues while a substitution aligns different characters together. Insertions and deletions on the other side change the length of the alignment by shifting the columns and placing gaps so that “similar” residues stay in the same column. The multiple sequence alignment problem translates therefore in finding the set of insertions, deletions and point mutations that transform one sequence to another (pairwise alignment) or to more than two sequences (multiple sequence alignment).

From the postulated MSA, the sequences homology is derived and a phylogenetic tree can be inferred which is used to assess the sequences’ shared evolutionary origins. The multiple sequence alignment is also exploited to study related genes or proteins in order to envisage evolutionary relationships between them identifying for example shared patterns among functionally or structurally related genes. Multiple sequence alignments are also used to distinguish orthologs from paralogs, or to identify pattern of conservation and to

determine the key functional domains as well as to develop a generic representation (by means of a profile) of a protein family. The latter, for example, applied to database searching to find new related sequences in order to characterize for example a protein. In this context the multiple sequence alignment (MSA) inference problem along with the phylogenetic inference and all the consecutive downstream analysis are essential requisites for evolutionary-based research [78, 85, 163].

Multiple sequence alignment inference

Despite the fact that humans are better pattern matching than computers [87], modern sequencing datasets are becoming unaffordable to be manually aligned. Indeed, humans eyes can easily detect mis-matched regions in an alignment, nevertheless manual curated alignments are highly subject to personal judgement which lack of standard rules and therefore not replicable [32]. Complex alignments, showing sophisticated gappy patterns, are extremely difficult to align and even experts would not be able to evaluate the entire alignment space to select the most plausible one. Inexplicably, however, manual adjustments are still currently performed by a considerable number of scientists [96].

Whilst new technologies generate an overwhelming amounts of data, the performance of current algorithms needed to process it are falling dramatically behind and are not able keep up the pace with modern high throughput sequencing systems. In fact, aligner packages are asked to align not only more and more sequences together but also of increasingly length [30]. Moreover, modern MSA software packages are required to be also biological accurate in inferring homologies and of lowest possible computational complexity [21]. In this context, the accuracy concerns the similarity or closeness between the true alignment and the inferred one regarding the postulated evolutionary events. The computational complexity is measured as the total amount of resources that the inference tool needs to produce the solution, that are in general number of CPUs/GPUs, run time and RAM memory.

The phylogeny estimation as the alignment inference are biological challenging problems by nature since in general the true evolutionary history cannot be directly observed. A pragmatic solution could be the in-silico evaluation by means of evolution simulation studies. Therefore, to be of any practical use, mathematical models describing the evolution must be sufficiently realistic to mimic the essential dynamic. Whether the inferred tree/alignment is evalu-

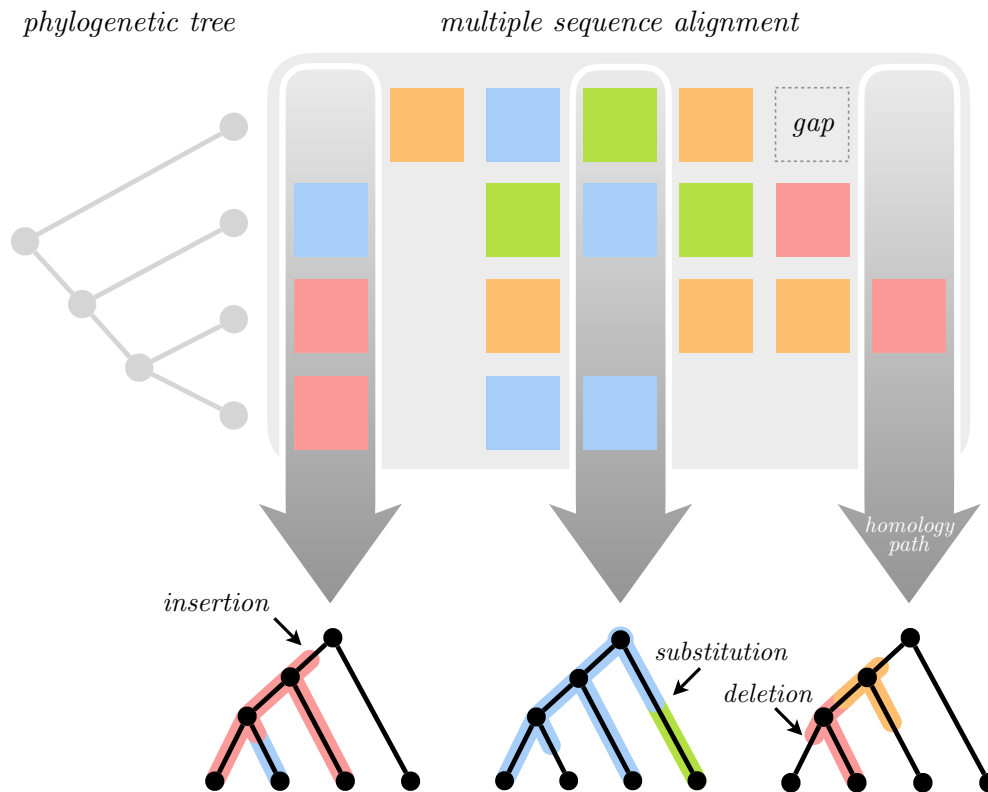


Figure 4: Multiple sequence alignment and corresponding homology paths. A multiple sequence alignment is a mapping from a set of sequences to a strings vector. Sequences are placed in row and characters in column. In an evolutionary multiple sequence alignment, characters evolved from the same common ancestor are disposed in the same column. Characters aligned in a column are embodying a set of compatible homology paths. In this Figure we are showing only for three columns a single homologous scenario compatible with the observations. Gaps are added in an alignment in order to shift the characters in their “correct” (postulated) position.

ated against a simulated or a biological benchmark, both approaches enable to quantify the performance error. Alternatively, one can evaluate the error based on the effects on downstream analysis. For instance, by measuring the effects on phylogenetic trees estimated on the basis of an alignment [58]. Nonetheless, it must be acknowledged, that despite the effort in developing new fast and accurate aligners and considering the already great availability of different MSA inference tool none of them is and won’t (probably) be able to produce an evolutionary perfect alignment [32, 107].

Nowadays, the supply of multiple sequence alignment tools is impressively wide and the decision of which tool to use depends mostly on individual preferences, the type of data to be processed and the size of the dataset. One can identify four main strands of multiple sequence alignment purposes, that are *structure prediction*; *database searching*; *sequence comparison*; and *phylogenetic analysis* requiring, underneath, different mathematical models [97, 98]. The multiple sequence alignment tools can be further classified by the implied inference methods as local/global alignment, using iterative refinement feature, based on sum-of-pairs (SP) optimality criterion or probabilistic (posterior probability or maximum likelihood). These references provide a quick link to further information: [4, 25, 32, 66, 86, 98, 107, 111, 116, 120, 144, 155, 161].

In this thesis we will focus only on **evolutionary alignment** that represent character homology between related sequences. They, for instance, are fundamental prerequisite for phylogenetic inference [98]. Multiple sequence alignment are conventionally represented in a table where taxa are placed in succession in rows and the sequence characters are the columns of the table. In this representation a character state is obtained by intersecting row and columns [98]. Gaps are added for the correct positioning of the homologous characters – sharing a common ancestor – together in the same column. It is clear at this point that the fundamental units for an homology hypothesis are the characters which translates in the a proper gaps disposal inside the table [38]. It is worth mentioning the distinction between the information represented in the phylogeny and the one in a multiple sequence alignment. A phylogenetic tree reproduce the relationship among taxa defined by their sequences, whereas an alignment depicts in columns the relationship among characters [98]. When studying homologies we have to focus therefore on the evolutionary events on characters [98], in this sense, homology exists only if referred to phylogeny [6, 16].

An important aspect frequently underestimated is the difference between the homology based on structural features and the homology established by evolutionary histories. The former is called “structural homology” while the second “evolutionary homology” or also “positional homology”. In case of convergent evolution structural alignments are often relatively similar to evolutionary alignments, albeit characters putted in the same column in a structural alignment imply homology in an evolutionary alignment even if that specific homology does not occur [58, 125].

There is a plethora of different computational algorithms that have been proposed for the MSA inference problem. They come in different flavors and

using different techniques and heuristics [21]. For pairwise sequence alignment usually the score of a candidate solution is obtained by summing up a fixed score for each match, and a penalty for each substitution and gap eventually distinguishing between gap-opening/gap-extending penalty. When there are more than only two sequences to be aligned, let say N , then the multiple sequence alignment score is obtained by summing the $N(N - 1)/2$ pairwise alignments obtained extracting all the pairs from the original MSA. The so obtained score is called sum-of-pairs [101] (SP). The optimal pairwise alignment, given a fixed scoring set, can be efficiently computed by means of dynamic programming algorithms in linear space and quadratic (in the average sequence length) time [45, 102, 103, 132]. This topic is discussed in the next section.

Progressive Dynamic Programming

The alignment of multiple sequences with an SP scoring model is known to belong to the class of NP-complete problems [33, 61, 159]. Solving the problem by means of a Dynamic Programming (DP) algorithm aligning at the same time all the sequences together would result in an algorithm that need time and space exponential in the number of taxa. A Dynamic programming algorithms is a mathematical optimization method which exploits the Bellman equation [10]. The basic idea behind that technique is the simplification of a complex problem that, though, must satisfy an optimal substructure, by breaking it into sub-problems. These last are recursively solved and the corresponding optimal sub-solutions are then recombined yielding the global optimal of the original problem.

Dynamic programming techniques are usually applied to global alignments by using methods such as the Needleman-Wunsch algorithm [103] and local alignments by employing the Smith-Waterman algorithm [132]. The computational complexity of aligning by means of a classic DP algorithm of N sequences of L characters in average scales as $\mathcal{O}(L^N)$. Therefore, even though this method can be extended to more than two sequences, its adverse complexity makes it inadvisable for practical problems [62]. A practical solution to find an optimal alignment by means of DP methods is by combining it with the idea developed by Feng and Doolittle [36] named “progressive alignment” [36, 155].

Progressive approaches are employed to break intractable problems into a series of tractable subproblems [25]. They work traversing a given topology

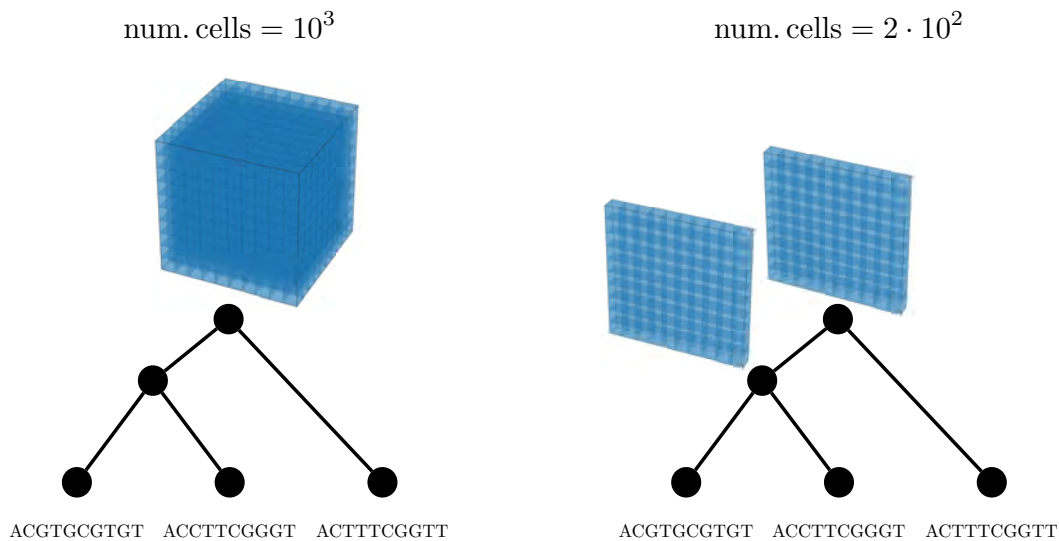


Figure 5: *Progressive Dynamic Programming.* Left: The computational complexity of aligning by means of a classic DP algorithm of N sequences of L characters in average scales as $\mathcal{O}(L^N)$. The progressive procedure runs at each internal node a DP algorithm for the pairwise alignment. This procedure results therefore in an algorithm that scales as $\mathcal{O}(m \cdot L^2)$ with m the number of internal nodes.

from the bottom (leaves) to the top (root) running an instance of pairwise DP alignment at each visited node. Then, at each node the algorithm assigns an optimal sub-MSA for the corresponding sub-tree. The most similar sequences are aligned first and, according to the tree, then the more distant ones are gradually added. At the root, the associated MSA corresponds to the solution of the inference problem. This approach results therefore in an algorithm that scales as $\mathcal{O}(m \cdot L^2)$, with m the number of internal nodes. Figure 5 depicts on the left panel the size of the corresponding DP matrix that aligns at the same time all the sequences at the leaves of the tree. In this example the sequences are composed of 10 characters each, so the DP matrix requires 10^3 entries to infer their alignment. On the right panel, the progressive approach needs to compute at each internal node only 10^2 for a total of 200 entries.

Though, the progressive solution obtained by combining optimal pairwise alignment is in general different from the optimal solution that one would obtain by running a full DP with all the sequences at the same time. Indeed, the progressive approach belongs to greedy heuristics and – as is often the case – they make decisions based on local/partial information. At each internal node the algorithm preserves the two sub-alignments associated at the corre-

sponding children nodes and aligns a pair of sub-alignments rather than all the sequences. Unfortunately, a column mistakenly aligned at an early stage cannot be anymore recovered and hence this error is propagate upwards affecting negatively the successive stages of the alignment algorithm. Several software packages to align sequences are exploiting a progressive approach, among others we can mention ClustalW [140], Clustal Omega [129], MAFFT [62], Kalign [75], Proalign [83], MUSCLE [30], DIALIGN [94], PRANK [85], FSA [14], T-Coffee [108, 148], ProbCons [23], DFALIGN [36], MULTAL [138, 139], PCMA [115], and MSAProbs [83].

To recover, to some extent, errors introduced at first run, iterative approaches repeatedly re-align the sequences, by means of dynamic programming, searching for better solutions [99]. However because of the computational cost they are limited to few hundred sequences [129]. The most popular iterative alignment algorithms include PRRP [46], MUSCLE [30], Dialign [94], SAGA [105], and T-COFFEE [148]. Another proposed approach to improve the alignment accuracy is the use of already existing protein structural information. The reason behind that idea is that structures evolve slowly and they bring a kind of stability in the alignment [77]. Familiar aligner structure based are for instance 3D-COFFEE [112], EXPRESSO [5] and MICAlign [164]. Another class of MSA aligner are algorithm that search for motif, that are short stretch of amino acids, in the long sequences. PHI-Blast [170] and Gapped Local Alignments of Motifs (GLAM2) [40] belong to this class. The last category that we want to mention here is short sequence alignment algorithms that are developed to align short read genome sequencing data, among them we can find Maq [80], SOAP [81], and Bowtie [72].

Phylogenies and alignments

Typically, the phylogeny inference starts by first evaluating a multiple sequence alignment, once there is an initial MSA then also the tree can be computed. Progressive alignments depends on the quality of the input guide tree (see for example [19, 82, 104, 119, 131, 149, 160]), the better the guide the more accurate the obtained MSA. The opposite is also true, phylogenies are sensitive to the quality of the starting alignment ([34]) and therefore by changing the alignments one can expect a different inferred tree [11, 53, 95, 163].

Nevertheless, a perfect guide tree does not guarantee an error-free alignment, in fact errors may be introduced at each node of the pairwise alignment

[25]. Consistency-based objective functions aim at improving the accuracy in inferring the match by adding information from outgroup sequences [44, 152–154]. The idea of using information from outgroup sequences is the following: suppose we are aligning two sequences \mathbf{X} and \mathbf{Y} at position i and j , respectively. If the outgroup sequence \mathbf{Z} at position k matches both characters X_i and Y_j then by transitivity also X_i and Y_j must be aligned. Following this principle the scoring functions update their parameters during the alignment to incorporate in the alignment that information. Example of such tools are T-Coffee [108, 148], DIALIGN [94], PROBCONS [23], PCMA [115], MUMMALS [117], PROMALS [118], and Align-m[156, 158].

There are tools, like for instance CLUSTALW, MUSCLE, and MAFFT, that are adjusting the gap penalty based on the specific position of the latter. The rationale is to avoid an over-penalization of extending gaps once the first one has been added. When aligning globular protein sequences, the hydrophobic/hydrophilic property of a residue is used to adjust the gap penalties. Software packages that make use of such heuristic are for example CLUSTALW and MUSCLE. It has been shown that indeed the hydrophobicity-based gap scoring improves the MSA accuracy for distant sequences [24] and by using profile alignments also improves the detection of homology [84].

Progressive multiple sequence alignment with indel evolution

All state of the art MSA programs nowadays use an evolutionary model to describe changes between homologous characters, providing a more realistic description of molecular data and thus more accurate inferences. However, a mathematical formulation of the insertion-deletion (indel) process still remains a critical issue. Describing the indel process in probabilistic terms is more challenging: unlike substitutions, indels often involve several sites, vary in length and may overlap obscuring the underlying mechanisms. Instead, the popular PRANK program adopts a pragmatic approach; it uses an outgroup to distinguish insertions from deletions during the progressive alignment procedure, so that insertions are not overpenalized [85]. As a result, PRANK produces exceptionally accurate alignments, notably with densely sampled data and given an accurate guide tree. Still the method lacks a mathematical model describing the evolution of indels. Indeed, the computation of the marginal likelihood under the classical indel models TKF91 [145] and TKF92 [146] is exponential in the number of taxa due to the absence of site independence assumption.

A recent modification of TKF91 describes the evolution of indels on a phylogenetic tree as a Poisson process, thus dubbed the Poisson indel process or the PIP model [13]. The indels occur uniformly within a sequence. Standard mathematical results, particularly the Poisson thinning, allow to achieve linear time complexity for computing the joint marginal probability of a tree and an MSA. This includes analytic marginalisation of unobservable homologous paths which occur whenever an ancestral character is inserted and subsequently deleted, and consequently cannot be detected in the extant sequences. For a given MSA and a tree, a likelihood score under PIP can be computed in linear time. This score can be used to find the maximum a posteriori tree-alignment solution. Remarkably, this breakthrough allows for a necessary rigorous way of combining models of substitutions and indels, and a tractable computation of the marginal likelihood function. At the moment the algorithm has only been applied in a Bayesian framework via tree-alignment space sampling.

With this project we have developed a new progressive Dynamic programming algorithm under the PIP evolutionary model. Our tool for the inference of multiple sequence alignment has been framed in the frequentist framework, and is implemented in a user-friendly software package. We have re-framed the original PIP equations into a dynamic programming (DP) approach. It aligns two MSAs –represented by their homology paths on the two corresponding subtrees– by maximum likelihood (ML) in polynomial time. The progressive algorithm traverses a guide tree in postorder; at each internal node the DP is applied to align the two sub-alignments at the child nodes. The procedure terminates at the root of the guide tree, with the complete MSA and the corresponding likelihood, which by construction is the likelihood under the PIP model. We have implemented the progressive MSA algorithm in a prototype program and verified its correctness by simulation. The use of a sound mathematical model of indel, namely the Poisson Indel Process model, is providing realistic and accurate estimates of MSAs and phylogenies. As a consequence, our new algorithms will allow not only more accurate phylogeny and alignment inference but it will also facilitate the estimation of statistical supports of inferred tree partitions and the ancestral reconstruction of insertions-deletions and substitution history.

Similarly to MAFFT, we detect homologous regions by means of a multi-scale short-time Fourier transform, which greatly reduces the algorithm complexity and allows further parallelizations. The resulting alignments display phylogenetically meaningful gap patterns and are of similar length compared to PRANK. To mitigate the intrinsic greediness brought by Dynamic program-

ming approach we have implemented the Stochastic backtracking algorithm of Mueckstein et al. [100] under the PIP model. In this way, our progressive algorithm generates at each visited node a distribution of candidate sub-optimal alignments, which may contains also the optimal one, rather than only the single optimal. Aligning sub-optimal solutions increases the chances to escape from local maxima and in our opinion provides a valid strategy to lessen the progressive bias. The greediness niveau is tuned with the a parameter that control our sake of sub-optimals. Finally, to account for substitution rate variation among sites (ASRV) we have implemented the Gamma distribution that applies to all the rates, insertion and deletion included. Further analysis are needed to investigate the impact of ASRV on the inferred alignments. Our hope is that this feature could mimic to some extent a ‘long’ insertion, that is, an insertion of more than a single character at a time.

To our knowledge, this is the first progressive MSA algorithm with polynomial time complexity, using a mathematical formulation of an explicit indel process. Note that an equivalent formulation under TKF91 or TKF92 –i.e. using the full marginal likelihood along the subtrees in question– would have exponential time complexity. Quadratic time complexity under the TKF models could be obtained [54] by representing sequences at internal nodes through probability profiles, and aligning those. However, this approach does not consider the evolutionary history in the subtrees.

The Poisson Indel Process

In this section we give a brief review of the basic definitions and results of PIP model [13]. Let $\tau = (\mathcal{V}, \mathcal{E}, b)$ represent a rooted binary phylogenetic tree with N leaves. τ is a directed, connected, labelled acyclic graph, with a finite set of branching points \mathcal{V} of cardinality $|\mathcal{V}| = 2N - 1$ and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Leaves $\mathcal{L} \subset \mathcal{V}$ denotes N observed taxa, represented by strings of characters from a finite alphabet Σ (nucleotides, amino acids or codons). There are $N - 1$ internal vertices $v \in \mathcal{V}$ whereof the root Ω is the most recent common ancestor of all leaves. Branch length $b(v)$ associated with node $v \in \mathcal{V}$ spans from v to its parent node $\text{pa}(v)$. The total tree length $\|\tau\|$ is a sum of all the branch lengths.

The PIP model describes a string-valued evolutionary process along the branches of τ . We denote the distance from the root to a given point on the tree, by the same symbol τ . Atomic insertions are Poisson events with rate

measure $\nu(dt) = \lambda(\tau(dt) + \mu^{-1}\delta_\Omega(dt))$, where λ is the insertion rate, μ the deletion rate, and $\delta_\Omega(\cdot)$ Dirac's delta function. This formulation guarantees that the expected sequence length remains constant during the whole evolutionary process. Point substitutions and deletions are modelled by a continuous-time Markov process on $\mathcal{A}_\epsilon = \mathcal{A} \cup \{\epsilon\}$, where ϵ is the deletion symbol. Accordingly, the generator matrix \mathbf{Q}_ϵ of the combined substitution and indel process extends the instantaneous substitution rate matrix \mathbf{Q} by a row and a column to include ϵ , which is modelled as an absorbing state as there can be no substitutions after a deletion event. The quasi-stationary distribution of \mathbf{Q}_ϵ is denoted by $\boldsymbol{\pi}_\epsilon$. Root Ω has a virtual infinite length stem, reflecting the equilibrium steady state distribution at the root.

For an internal node v , the single character prior insertion probability $\iota(v)$ associated to the branch $\text{pa}(v) \rightarrow v$ is proportional to its branch length $b(v)$. For $v \neq \Omega$ it is given by $\iota(v) = b(v)/(\|\tau\| + \mu^{-1})$; at the root atomic mass point probability $\iota(\Omega) = \mu^{-1}/(\|\tau\| + \mu^{-1})$ so that $\sum_{v \in \mathcal{V}} \iota(v) = 1$. The survival probability $\beta(v)$ associated with an inserted character on branch $\text{pa}(v) \rightarrow v$ is given by $\beta(\Omega) = 1$ and $\beta(v) = (1 - \exp(-\mu b(v)))/(\mu b(v))$ for $v \neq \Omega$.

The marginal likelihood $p_\tau(m)$ of MSA m of length $|m|$ is computable in $O(N \cdot |m|)$ and can be expressed as

$$p_\tau(m) = \varphi(p(c_\emptyset), |m|) \prod_{c \in m} p(c), \quad (1)$$

where $p(c)$ is the likelihood of a single column c , and $p(c_\emptyset)$ is the likelihood of an unobservable character history, represented by a column c_\emptyset with a gap at every leaf. The factor in (1)

$$\varphi(p(c_\emptyset), |m|) = \|\nu\|^{|m|} \exp\left(\|\nu\|(p(c_\emptyset) - 1)\right)/|m|! \quad (2)$$

is the marginal likelihood over all unobservable character histories, where $\|\nu\|$ is the normalising measure of the Poisson process.

The column likelihood can be expressed as

$$p(c) = \sum_{v \in \mathcal{V}} \iota(v) f_v, \quad (3)$$

where f_v denotes the probability of the homology path underlying column c , given that the corresponding character was inserted at v . This probability can be computed in $O(N)$ using a variant of Felsenstein's peeling recursion [35].

Let \mathcal{S} be the set of leaves that do not have a gap in column c , and \mathcal{I} the set of nodes ancestral to \mathcal{S} . Then

$$f_v = \begin{cases} \mathbb{1}[v \in \mathcal{I}] \beta(v) \sum_{\sigma \in \mathcal{A}} \pi_\epsilon(\sigma) \tilde{f}_v(\sigma) & \text{if } c \neq c_\emptyset \\ 1 - \beta(v) + \beta(v) \sum_{\sigma \in \mathcal{A}} \pi_\epsilon(\sigma) \tilde{f}_v(\sigma) & \text{otherwise,} \end{cases} \quad (4)$$

where

$$\tilde{f}_v(\sigma) = \begin{cases} \mathbb{1}[c(v) = \sigma] & \text{if } v \in \mathcal{L} \\ \prod_{w \in \text{child}(v)} \left[\sum_{\sigma' \in \mathcal{A}_\epsilon} \exp(b(w) \mathbf{Q}_\epsilon)_{\sigma, \sigma'} \tilde{f}_w(\sigma') \right] & \text{otherwise,} \end{cases} \quad (5)$$

and $\mathbb{1}[\cdot]$ is the indicator function. In equation 4, the term $1 - \beta(v)$ accounts for the probability that the inserted character does not survive till the first node below the insertion point. The recursive function \tilde{f}_v computes the probability of the substitution-deletion process of a single character.

Main result of the thesis

“To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science.”

– Albert Einstein

In this thesis we present for the first time, a frequentist progressive multiple sequence alignment method under an explicit evolutionary model of substitutions, insertions and deletions. Modeling indel evolution using the Poisson Indel Process reduces the computational complexity from exponential to linear (Bouchard-Côté & Jordan 2013). Our new method, allows to infer progressive MSAs under an explicit evolutionary model of indels. With respect to MSA inference, this is the first major methodological advance since the development of the phylogeny-aware PRANK.

At each internal node of the guide-tree we align the two children MSAs by full maximum likelihood approach. This has not been done before. Our method is highly competitive with phylogeny-aware PRANK and outperforms other popular aligners by a good margin. In addition to high quality our method allows to infer phylogenetically meaningful gaps alternative to PRANK, while producing alignments of similar length. Our algorithm is of polynomial time complexity and can be further parallelised, which makes it suitable for large-scale analyses.

To date only the Bayesian methods implement evolutionary indel models, but are limited to just a few sequences due to very costly MCMC sampling algorithms. With this thesis, we open a number of novel avenues in the analysis of genomics sequences. For example, more realistic model features could be implemented allowing for variation of indel rates along sequences and over time. Moreover, including the alignment uncertainty estimation, it will be possible to extend our tool to include the joint estimation of MSAs and trees both in a frequentist framework under a unique explicit and sound model of indel evolution.

Outline

“Equipped with his five senses, man explores the universe around him and calls the adventure Science.”

– Edwin Hubble

Part I. Progressive DP under the PIP model

1 In Section **Progressive Dynamic Programming under P-IP model** we present a new formulation of the PIP equations which have been integrated in our progressive Dynamic Programming algorithm. The result is a new algorithm which returns the maximum likelihood multiple sequence alignments under the PIP model in a progressive fashion.

2 In Section **3D Dynamic Programming under PIP** we show, step-by-step, by means of a small example the equations involved in our progressive aligner to compute the likelihood of the different homologous scenarios implied by an MSA column. We illustrate in particular how the different three-dimensional dynamic programming matrices are structured and finally how the traceback algorithm builds the optimal pairwise alignment.

3 Classic dynamic programming algorithms by construction produce only a single optimal solution. Unfortunately, since dynamic programming techniques are often framed in a progressive approach, the obtained solution may be biased. Furthermore, for certain applications one is interested in more than a unique solution, for example for testing different hypothesis. In Section **Stochastic backtracking DP algorithm** we present the Stochastic Backtracking algorithm of Mueckenstein et al. [100] under the PIP evolutionary model. In this way, at each internal node the algorithm generates a distribution of alignment. We illustrate also the effects of different values of the ‘temperature’ parameter on the achievable traceback paths.

4

To take into account for the among-site rate variations (ASRV) we have extended the PIP equations with the classic Gamma distribution. In Section **Marginal Likelihood with Rate Variation Across Sites** we present the equations for the computation of the marginal likelihood marginalized both over ancestral states and over the discrete Gamma rate categories.

5

To save computational time we have implemented a short-time Fourier transform based homologous blocks detector. The purpose of predicting candidate homologous segments is the successive elimination of the non-promising regions in the DP matrix prior to the effective alignment process. This new algorithm, presented in Section **Multi-scale STFT based homologous blocks detection**, has been inspired by MAFFT [62] but differently from the original idea we have implemented a multi-scale time-frequency transform for the automatic detection of candidate homologous regions providing simultaneously the positional shift lag and the relative position of similar patterns inside the two sequences. We have also extended the algorithm which resolves the potential overlaps between different postulated homologous regions.

6

Progressive approaches considerably reduce the alignment complexity by traversing a given binary phylogenetic tree from bottom towards the root and aligning at each internal node only two sub-alignments. Although progressive approaches strongly accelerate the generation of the final solution they potentially introduce biases at each iteration. The reason of this bias lies, on one hand in the fact that the information available at each internal node is deliberately not complete, on the other because the alignment is constrained to the previously computed solutions. In Section **Progressive bias analysis** we have quantified, with a very small example, the bias introduced by the progressive procedure using two different topologies.

Part II. Appendix

A

In Appendix **Detailed derivation of the marginal likelihood function** $\varphi(v)$ we step-by-step retrace, also for non mathematicians, the construction of a closed form expression

of the marginal likelihood of empty columns, and in Appendix **Detailed derivation of the survival probability function** $\beta(v)$ the derivation of the survival probability associated to a given node v . Section **Overview of PIP equations** highlights the main equations involved in the computation of the marginal likelihood under PIP [13].

B

In Appendix **Reversibility of TKF91 and PIP**, we first give a concise introduction to the TKF91 model and we highlight the main equations involved in the definition time-reversible evolutionary models. Then, by means of two examples, we check whether TKF91 and PIP model are both time-reversible models.

C

Appendix **Characterization of Indel rates** starts by recapitulating the definition of the four different fates of a character on a phylogeny [13]. Using these definitions we propose an approach to infer the indel rates from the input data. Finally, we illustrate graphically the effects of the various model parameters on the number of fully conserved and gappy columns in a given multiple sequence alignment.

D

In Appendix **Doob-Gillespie method and PIP description** we aim to illustrate the connection between the Doob-Gillespie representation and the PIP model, and thus also TKF91. In this section we clarify the equations involved in the generation of homologous sequences under the PIP model. Afterwards, we schematically represent the algorithm that synthesizes the evolution of a string of character under PIP.

E

In Appendix **Grantham's distance** we propose a new formulation which exploits a mathematically sound Grantham's distance based metric defined on the space of the physicochemical properties of amino acids. The Grantham's distance is used as a measure of the physicochemical similarity of two molecules, which in turn is interpreted as a proxy for the likelihood that the two molecules might undergo a substitution in an evolutionary process.

F

In Appendix F we report the pseudocode of the algorithm that resolve the overlaps between candidate homologous blocks and

builds paths that connect them. Finally among the possible overlap-free path the algorithm returns the one bearing the largest homologous regions, that is, the largest total number of columns.

G

In Appendix **Multiple sequence alignment evaluation** we compare a BAliBASE reference alignment with an evolutionary alignment generated under PIP. Structural alignments are in practice the standards against which alignments tools are often measured. We critically discuss this routine of assessing evolutionary alignment software packages against BAliBASE benchmarks.

I

**Progressive Multiple Sequence
Alignment with Indel Evolution**

1

Progressive Dynamic Programming under PIP model

“The important thing is not to stop questioning. Curiosity has its own reason for existence. One cannot help but be in awe when he contemplates the mysteries of eternity, of life, of the marvelous structure of reality. It is enough if one tries merely to comprehend a little of this mystery each day.”

– Albert Einstein, *LIFE Magazine* (2 May 1955)

1.1 Introduction

In this chapter we propose a new formulation of the PIP equations adapted for the progressive Dynamic Programming approach. To simplify the exposure of the new formulas we will refer, from now on, to the topology of Figure 1.1. Moreover, we will assume that the algorithm is running at node v_7 to align the sub-alignments associated to v_3 and v_6 . To simplify the notation let define $\rho(v_j, v_k)$ as the set of nodes in the path v_j to v_k . In our simple example, shown in Figure 1.1, the path $\rho(v_3, \Omega)$ is the set of nodes starting at node v_3 and ending at the root Ω following the topology, i.e. $\rho(v_3, \Omega) = \{v_3, v_7, v_9, \Omega\}$. The length of the path $\rho(v_j, v_k)$, represented by the function $L(\rho(v_j, v_k))$, corresponds to the sum of branch lengths of all the edges between the two end points. From our example: $L(\rho(v_3, \Omega)) = e(v_3) + e(v_7) + e(v_9) = b(v_3) + b(v_7) + b(v_9)$.

We introduce also the *pure* survival probability $\zeta(v)$ associated to the node v

$$\zeta(v) = \sum_{\sigma \in \mathcal{A}} \exp(b(v)\mathbf{Q}_\sigma) \mathbb{1}(c(v) = \sigma) = \exp(-\mu b(v)) \quad (1.1)$$

as the probability that a character survives along the edge $e = (v \rightarrow \text{pa}(v))$ of length $b(e) = b(v)$ given that the character was inserted along the path

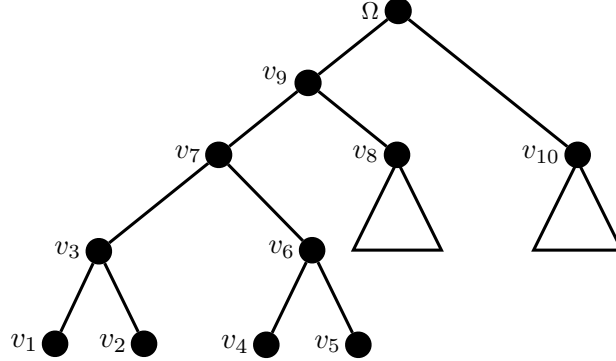


Figure 1.1: Rooted topology used to illustrate the new PIP formulation for the progressive Dynamic Programming approach.

$\text{pa}(v) \rightarrow \Omega$. In equation 1.1, the function $\mathbf{1}(\cdot)$ represents the indicator function¹. Note that the summation in equation 1.1 is going over all the characters in the canonical alphabet, this means that the gap character is never envisaged.

Equation 1.1 can be interpreted as the marginal probability that the character survives along the edge e of length $b(v)$, marginalized over all states but the gap. The ‘pure’ survival probability $\zeta(v)$ differs from the definition of $\beta(v)$ as regards the insertion location. While in $\zeta(v)$ the character is already present at $\text{pa}(v)$, in βv the character is inserted on a random location along the edge associated to v . In both cases, however, the character is required to survive till the node v .

The *pure* survival probability associated to a path $\rho(v_j, v_k)$ is equal to

$$\zeta(\rho(v_j, v_k)) = \prod_{v_i \in \rho(v_j, v_k)} \zeta(v_i) = \exp\left(-\mu L(\rho(v_j, v_k))\right). \quad (1.2)$$

For sake of compactness we introduce also the non-*pure*-survival probability $\xi(v)$, defined as

$$\xi(v) = 1 - \zeta(v) \quad (1.3)$$

and the non-*pure*-survival probability computed for a path it gets

$$\xi(\rho(v_j, v_k)) = 1 - \zeta(\rho(v_j, v_k)). \quad (1.4)$$

¹The indicator function returns the value 1 if the function argument is true, 0 otherwise.

1.2 Likelihood computation of a column $p(c)$

Our Dynamic Programming algorithm computes the likelihood of a single column c by marginalizing over all possible insertion scenarios that may have created that column. The marginalization goes over all the nodes along the path that connects the root to the actual node. Let us suppose that we are at node v_7 , then the marginal likelihood for matching the two columns (sub-alignment columns) at the two children nodes v_3 and v_6 is computed by means of the following equation²

$$p_{v_7}(c) = \iota(\Omega)\beta(\Omega) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(\Omega)) + \iota(v_9)\beta(v_9) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_9)) + \iota(v_7)\beta(v_7) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_7)) \quad (1.5)$$

with

$$\begin{aligned} \mathbf{f}(\Omega) &= \mathbf{f}(v_9) * \mathbf{f}(v_{10}) & \mathbf{f}(v_1) &= \mathbf{1}(v_1) & \mathbf{f}(v_5) &= \mathbf{1}(v_5) \\ \mathbf{f}(v_9) &= \mathbf{f}(v_7) * \mathbf{f}(v_8) & \mathbf{f}(v_2) &= \mathbf{1}(v_2) & \mathbf{f}(v_8) &= [1, \dots, 1]^\top \\ \mathbf{f}(v_7) &= \mathbf{f}(v_3) * \mathbf{f}(v_6) & \mathbf{f}(v_6) &= \mathbf{f}(v_4) * \mathbf{f}(v_5) & \mathbf{f}(v_{10}) &= [1, \dots, 1]^\top \\ \mathbf{f}(v_3) &= \mathbf{f}(v_1) * \mathbf{f}(v_2) & \mathbf{f}(v_4) &= \mathbf{1}(v_4) & & \end{aligned} \quad (1.6)$$

and where $\mathbf{1}(v)$ takes value 1 only at the position corresponding to the observed character at the leaf. For instance, using an extended nucleotide alphabet built in this order $\mathcal{A}_\epsilon = \{A, C, G, T, -\}$, and assuming a character ‘T’ at the leaf v_1 then its indicator array becomes $\mathbf{1}(v_1) = [0, 0, 0, 1, 0]^\top$. For a gap it will be $\mathbf{1}(v_1) = [0, 0, 0, 0, 1]^\top$.

It is worth noting, however, that at node v_7 there is no information regarding the states at the leaves of the respective sub-trees rooted at v_8 and v_{10} . Hence, the two felsenstein’s arrays $\mathbf{f}(v)$ (of size $|\mathcal{A}_\epsilon| \times 1$) are set to $[1, \dots, 1]^\top$. In this way the algorithm accounts for all possible scenarios.

Let assume, for instance, that the leaves are associated with the following characters: $v_1 = \{G\}$, $v_2 = \{A\}$, $v_4 = \{T\}$, and $v_5 = \{C\}$. At node v_7 the felsenstein’s array gets

$$\begin{aligned} \mathbf{f}(v_7) &= \mathbf{f}(v_3) * \mathbf{f}(v_6) \\ &= \dots \end{aligned}$$

²Here for simplicity we have used an algebraic notation. The symbol \circ denotes the scalar product between vector arrays and the symbol $*$ represents the hadamard product. Variables in bold express multi-dimensional arrays.

$$\begin{aligned}
&= \dots \\
&= \left\{ \exp(\mathbf{Q}_\epsilon \cdot b(v_3)) \cdot \left[(\exp(\mathbf{Q}_\epsilon \cdot b(v_1)) \cdot \mathbf{f}(v_1)) * (\exp(\mathbf{Q}_\epsilon \cdot b(v_2)) \cdot \mathbf{f}(v_2)) \right] \right\} * \left\{ \exp(\mathbf{Q}_\epsilon \cdot b(v_6)) \cdot \left[(\exp(\mathbf{Q}_\epsilon \cdot b(v_4)) \cdot \mathbf{f}(v_4)) * (\exp(\mathbf{Q}_\epsilon \cdot b(v_5)) \cdot \mathbf{f}(v_5)) \right] \right\} \\
&= \left\{ \exp(\mathbf{Q}_\epsilon \cdot b(v_3)) \cdot \left[\left(\exp(\mathbf{Q}_\epsilon \cdot b(v_1)) \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right) * \left(\exp(\mathbf{Q}_\epsilon \cdot b(v_2)) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \right] \right\} * \left\{ \exp(\mathbf{Q}_\epsilon \cdot b(v_6)) \cdot \left[\left(\exp(\mathbf{Q}_\epsilon \cdot b(v_4)) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right) * \left(\exp(\mathbf{Q}_\epsilon \cdot b(v_5)) \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \right] \right\}.
\end{aligned}$$

The likelihood function of equation 1.5 computed at node v_7 for a column c in state *match* rewrites

$$\begin{aligned}
p_{v_7}(c) &= \iota(\Omega)\beta(\Omega) [\boldsymbol{\pi}_\epsilon \circ (\mathbf{f}(v_9) * \mathbf{f}(v_{10}))] + \iota(v_9)\beta(v_9) [\boldsymbol{\pi}_\epsilon \circ (\mathbf{f}(v_7) * \mathbf{f}(v_8))] + \\
&\quad + \iota(v_7)\beta(v_7) [\boldsymbol{\pi}_\epsilon \circ (\mathbf{f}(v_3) * \mathbf{f}(v_6))] \\
&= \left[\iota(\Omega)\beta(\Omega)\zeta(\rho(v_7, \Omega)) + \iota(v_9)\beta(v_9)\zeta(\rho(v_7, v_9)) + \iota(v_7)\beta(v_7) \cdot \right. \\
&\quad \left. \cdot \zeta(\rho(v_7, v_7)) \right] \cdot (\boldsymbol{\pi}_\epsilon \circ (\mathbf{f}(v_3) * \mathbf{f}(v_6))) \\
&= \left[\iota(\Omega)\beta(\Omega)\zeta(v_7)\zeta(v_9) + \iota(v_9)\beta(v_9)\zeta(v_7) + \iota(v_7)\beta(v_7) \cdot 1 \right] \cdot \\
&\quad \cdot (\boldsymbol{\pi}_\epsilon \circ (\mathbf{f}(v_3) * \mathbf{f}(v_6))). \tag{1.7}
\end{aligned}$$

The equation 1.7 can be simplified by considering that

$$\left[\iota(\Omega)\beta(\Omega)\zeta(v_7)\zeta(v_9) + \iota(v_9)\beta(v_9)\zeta(v_7) + \iota(v_7)\beta(v_7) \cdot 1 \right] = \frac{1/\mu}{\|\tau\| + 1/\mu}. \tag{1.8}$$

Therefore equations 1.7, applying the result of equation 1.8, becomes

$$p_{v_7}(c) = \alpha(v_7) (\boldsymbol{\pi}_\epsilon \circ (\mathbf{f}(v_3) * \mathbf{f}(v_6))) \tag{1.9}$$

where

$$\alpha(v) = \frac{1/\mu}{\|\tau\| + 1/\mu} \quad \text{for } v \in \mathcal{V}. \tag{1.10}$$

The function $\alpha(v)$ does not depend on the particular node v and its value corresponds to the total sum of prior insertion probabilities and survival probabilities from the root to a particular node v on the path $\rho(v, \Omega)$. The consequence for our DP algorithm is that during the alignment procedure we are allowed to re-root the tree at the current node v .

1.3 Likelihood computation of a column $p(c_\emptyset)$

The tree re-rooting strategy just described above, is however allowed only for the computation of the likelihood of columns $c \neq c_\emptyset$, namely columns containing at least one character different from the gap. In other words the finding in equation 1.8 cannot be applied for a column c_\emptyset full of gaps. The reason lies in the fact that the function $\alpha(\cdot)$ considers the history of a character present at the root surviving till node v whereas for the computation of the likelihood of an empty column we must consider also scenarios where a character does not survive till that node. Thus, for the computation of the likelihood of an empty column we have to consider the original root of the tree.

Referring to the topology in Figure 1.1, by re-rooting the tree at the node v_7 we would miss the likelihood of homologous paths where *i*) a character inserted at the root or *ii*) along the path $\rho(v_7, \Omega)$ dies before hitting v_7 .

The probability *i*) that a character inserted at the root does not survive till v_7 is computed as

$$p = \iota(\Omega)\beta(\Omega)(1 - \zeta(v_7)\zeta(v_9)) = \alpha(\Omega)\xi(\rho(v_7, \Omega)). \quad (1.11)$$

The prior probability of an single character insertion along the path $\rho(v_7, \Omega)$ is

$$\iota(\rho(v_7, \Omega)) = \frac{b(v_7)}{\|\tau\| + 1/\mu} + \frac{b(v_9)}{\|\tau\| + 1/\mu} = \frac{L(\rho(v_7, \Omega))}{\|\tau\| + 1/\mu} \quad (1.12)$$

and in general

$$\iota(\rho(v_i, v_j)) = \frac{L(\rho(v_i, v_j))}{\|\tau\| + 1/\mu} \quad v_i, v_j \in \mathcal{V}. \quad (1.13)$$

The corresponding survival probability – the probability that a character inserted on a random location along the path $\rho(v_7, \Omega)$ survives till v_7 – is

$$\beta(\rho(v_7, \Omega)) = \frac{1}{\mu L(\rho(v_7, \Omega))} \left(1 - \exp\left(-\mu L(\rho(v_7, \Omega))\right) \right) \quad (1.14)$$

and for a generic path $\rho(v_i, v_j)$

$$\beta(\rho(v_i, v_j)) = \frac{1}{\mu L(\rho(v_i, v_j))} \left(1 - \exp\left(-\mu L(\rho(v_i, v_j))\right) \right) \quad v_i, v_j \in \mathcal{V}. \quad (1.15)$$

From equations 1.11-1.14 we can compute the probability $p(c)$ that a character inserted either at the root or along the path $v_7 \rightarrow \Omega$ does not reach v_7 . This reads

$$p(c) = \alpha(\Omega)\xi(\rho(v_7, \Omega)) + \iota(\rho(v_7, \Omega))(1 - \beta(\rho(v_7, \Omega))). \quad (1.16)$$

At this point it remains to compute the probability that a character, present at v_7 , is deleted on both children sub-trees. This in general correspond to the following calculus:

- (i) *probability of deletion on the direct left edge **and** probability of deletion on the direct right edge **or***
- (ii) *probability of deletion on the direct left edge **and** probability of survival on the right edge but deletion on the right sub-tree **or***
- (iii) *probability of survival on the left edge but deletion on the left sub-tree **and** probability of deletion on the direct left edge **or***
- (iv) *probability of survival on the left edge but deletion on the left sub-tree **and** probability of survival on the right edge but deletion on the right sub-tree.*

For node v_7 this translates to the following probability

$$\begin{aligned} p_{v_7}(c_\emptyset) = & [\xi(v_3)] \cdot [\xi(v_6)] + \\ & + [\zeta(v_3) \cdot \xi(v_1) \cdot \xi(v_2)] \cdot [\xi(v_6)] + \\ & + [\xi(v_3)] \cdot [\zeta(v_6) \cdot \xi(v_4) \cdot \xi(v_5)] + \\ & + [\zeta(v_3) \cdot \xi(v_1) \cdot \xi(v_2)] \cdot [\zeta(v_6) \cdot \xi(v_4) \cdot \xi(v_5)] \end{aligned}$$

which can be written by means of the following general recursive formula

$$\eta(v) = \begin{cases} 0 & v \in \mathcal{L} \\ \prod_{w \in \text{child}(v)} [\zeta(w)(\eta(w) - 1) + 1] & \text{otherwise.} \end{cases} \quad (1.17)$$

Putting equation 1.16 and equation 1.17 together the likelihood of a single column full of gaps at a node v can be computed as

$$p_v(c_\emptyset) = \alpha(\Omega)\xi(\rho(v, \Omega)) + \iota(\rho(v, \Omega))(1 - \beta(\rho(v, \Omega)) + \alpha(v)\eta(v). \quad (1.18)$$

2

3D Dynamic Programming under PIP

“The most beautiful experience we can have is the mysterious. It is the fundamental emotion that stands at the cradle of true art and true science.”

– Albert Einstein, *The World as I See It*

2.1 Introduction

In this Chapter, we are showing the computations involved in the Dynamic Programming algorithm under the Poisson Indel model [13] using the results of Chapter 1. To simplify the exposure we are considering here only the case of constant rate variation among sites ASRV and the formulas are expressed in the natural space rather than the log space.

Let τ be the phylogenetic tree that relates the four sequences associated at the leaves $\{v_1, v_2, v_4, v_6\}$ as depicted in Figure 2.1. In order to infer their multiple sequence alignment, our progressive DP algorithm traverses the tree τ in post-order, starting from the leaves upwards till it reaches the root. At each internal node the algorithm produces the optimal pairwise alignment of the two sub-alignments present at its children nodes. This approach, called *progressive alignment*, explores at each node the space of pairwise alignments constraints on the previously inferred alignments. From our example, the algorithm starts at the first internal node v_3 and aligns the sequences at the leaves v_1 and v_2 , then, at the node v_5 , aligns the optimal pairwise alignment obtained at v_3 with

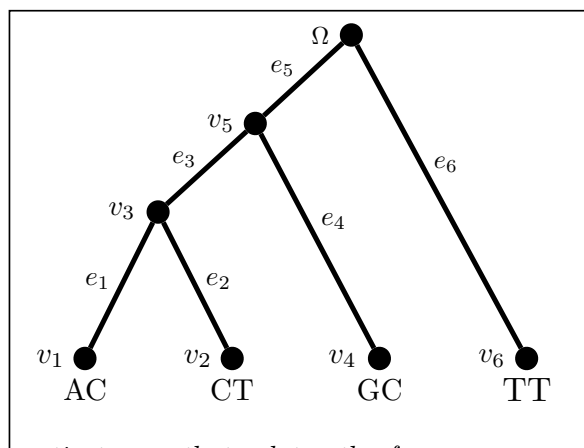


Figure 2.1: Phylogenetic tree τ that relates the four sequences at the leaves. The tree is rooted at Ω , the most recent common ancestor of all the sequences at the tips.

the sequence present at v_4 and finally at the root Ω the algorithm aligns the optimal sub-alignment obtained at v_5 with the sequence associated at node v_6 . The computations performed to align at node v_3 and v_5 are shown in the next

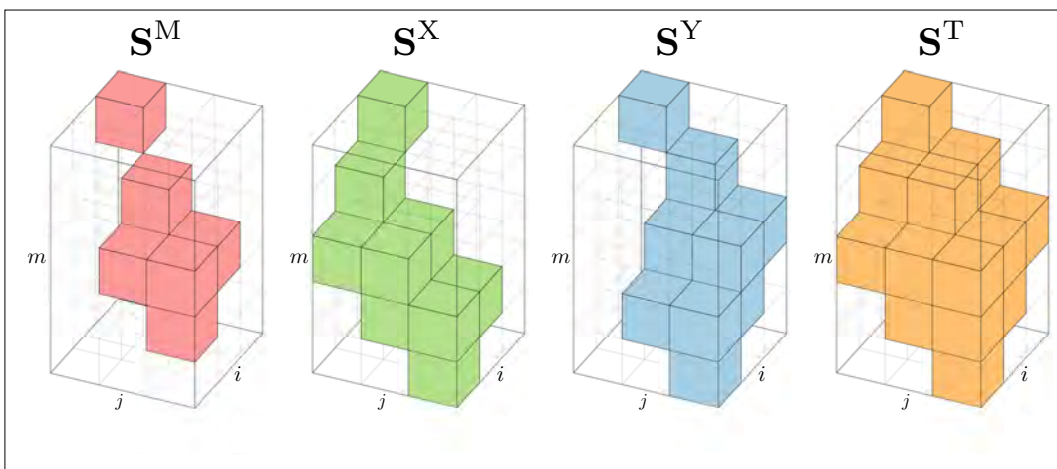


Figure 2.2: Four three-dimensional sparse DP matrices S^M (match), S^X (gapX), S^Y (gapY) and S^T (traceback). Colored cells represent entries different from 0, i.e. the ones accessed by the aligner algorithm. There is no permitted path in the DP that connects an empty cell (colourless) at a given position (i, j, m) to the corner $(0, 0, 0)$ of length m .

As presented in the original paper [13] and outlined in Appendix A.1, under PIP the likelihood of an alignment is composed of the product of the MSA

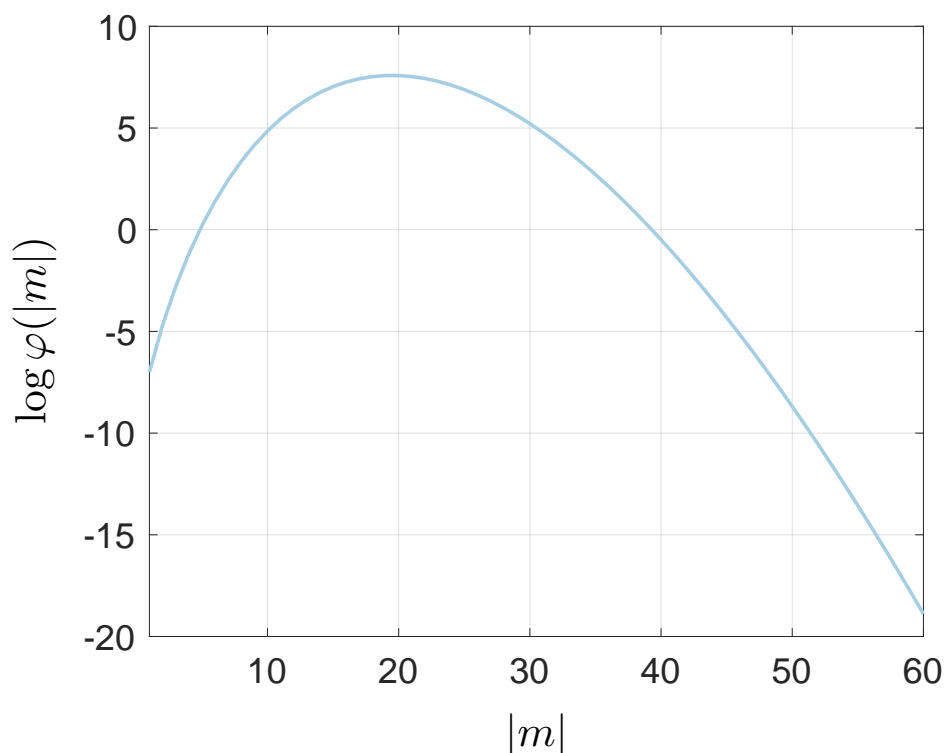


Figure 2.3: An example of $\varphi(|m|)$ (equation 2.1), i.e. the marginal likelihood of all non-observable histories, as a function of MSA length $|m|$. The parameters are: $\|\tau\| = 1$, $\lambda = 10$, $\mu = 1$, $p_{c_0} = 0.5$.

In Figure 2.2 one can see a small example of the four three-dimensional sparse DP matrices \mathbf{S}^M (*match*), \mathbf{S}^X (*gapX*), \mathbf{S}^Y (*gapY*) and \mathbf{S}^T (*traceback*) needed by our aligner. The colored cells represent entries with a likelihood different from 0, the others entries are never computed because they correspond to meaningless alignment. Indeed, there is no permitted path that connects an empty cell at a given position (i, j, m) to the corner $(0, 0, 0)$ of length m .

2.2 Alignment at node v_3

Let be $\mathbf{X} = \text{AC}$ the sequence associated at the leaf v_1 and $\mathbf{Y} = \text{CT}$ the sequence at the leaf v_2 , and let denote with $|\mathbf{X}|$ and $|\mathbf{Y}|$ their respective length (see Figure 2.1). Hence, for this simple example $|\mathbf{X}| = |\mathbf{Y}| = 2$. The progressive DP algorithm requires four three-dimensional sparse matrices \mathbf{S}^M , \mathbf{S}^X , \mathbf{S}^Y and \mathbf{S}^T each of size $(|\mathbf{X}| + 1) \times (|\mathbf{Y}| + 1) \times (|\mathbf{X}| + |\mathbf{Y}| + 1)$, dimensions indexed by i , j and m , respectively. The third dimension in the DP matrices, indexed by m , indicates the MSA length.

Then, one layer at a time along the third dimension m , the algorithm fills with likelihood values the three DP matrices, by keeping track in the traceback matrix for each non empty cell which matrix, among the three, contains the highest value. In the following sections we are showing, by means of the simple example of Figure 2.1, the computations required to align the input sequences and the homologous scenarios implied by a given MSA column

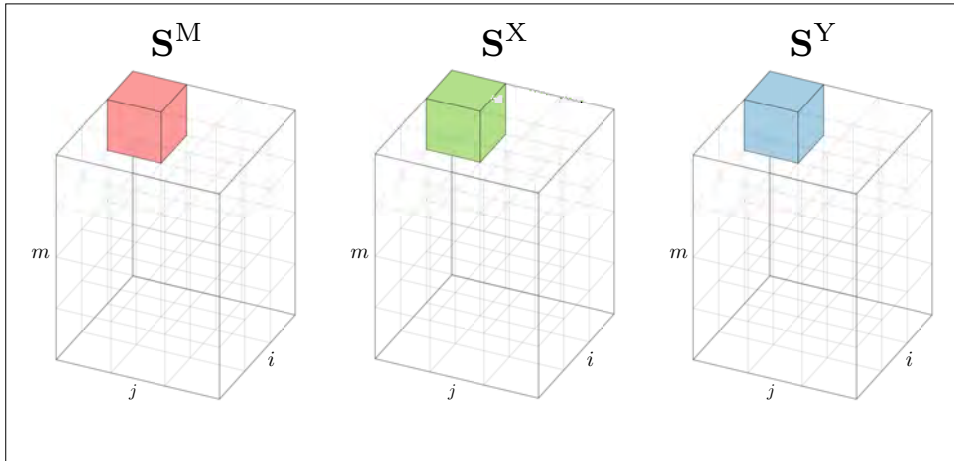


Figure 2.4: At layer $m = 0$ only one cell for each matrix is computed. These entries contain the marginal likelihood for empty columns $\varphi(p_{v_3}(c_\emptyset), 0)$ for an alignment of size $|m| = 0$.

Layer $m = 0$

The algorithm starts filling at position $(i, j, m) = (0, 0, 0)$ the three matrices \mathbf{S}^M , \mathbf{S}^X and \mathbf{S}^Y (see Figure 2.4) with the likelihood value

$$\varphi(p_{v_3}(c_\emptyset), 0) = \frac{1}{0!} \|\nu\|^0 \exp\left(\|\nu\|(p_{v_3}(c_\emptyset) - 1)\right) = \exp\left(\|\nu\|(p_{v_3}(c_\emptyset) - 1)\right) \quad (2.1)$$

where the likelihood of a single column containing only gaps, $p_{v_3}(c_\emptyset)$, in this case $[-]$, is computed by means of

$$p_{v_3}(c_\emptyset) = \alpha(\Omega)\xi(\rho(v_3, \Omega)) + \quad (\text{i})$$

$$+ \iota(\rho(v_3, \Omega)) \left(1 - \beta(\rho(v_3, \Omega))\right) + \quad (\text{ii})$$

$$+ \alpha(v_3)\eta(v_3) + \quad (\text{iii})$$

$$+ \iota(v_1) \left(\left(1 - \beta(v_1) + \beta(v_1)(\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_1))\right) \right) + \quad (\text{iv})$$

$$+ \iota(v_2) \left(\left(1 - \beta(v_2) + \beta(v_2)(\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_2))\right) \right). \quad (\text{v}).$$

The equation numbers, expressed in Roman numerals, refer to the homology paths depicted graphically in Figure 2.5. The likelihood of a single column full of gaps $p_{v_3}(c_\emptyset)$ requires the marginalizing over five possible homologous scenarios all producing no extant characters at the leaves. The five homologous paths resulting in an MSA column $c_\emptyset = [-]$ at v_3 are schematically depicted in Figure 2.5 and correspond to

- (i) an insertion of a character at the root Ω and the non-survival of this character till the node v_3 (Figure 2.5(i));
- (ii) an insertion of a character along the path $\rho(v_3, \Omega)$ and the non-survival till v_3 (Figure 2.5(ii));
- (iii) an insertion on the node v_3 and a deletion along the branches e_1 and e_2 (Figure 2.5(iii));
- (iv) an insertion along the edge e_1 with non-survival till v_1 (Figure 2.5(iv));
- (v) an insertion along the edge e_2 with non-survival till v_2 (Figure 2.5(v)).

See Appendix 1 for more details.

At layer $m = 0$ the only value different from 0 is located at position $(0, 0, 0,)$ and takes the value $\varphi(p_{v_3}(c_\emptyset), 0)$ which corresponds to the likelihood of an alignment of observable length 0 and an undefined and possibly infinite number of empty columns.

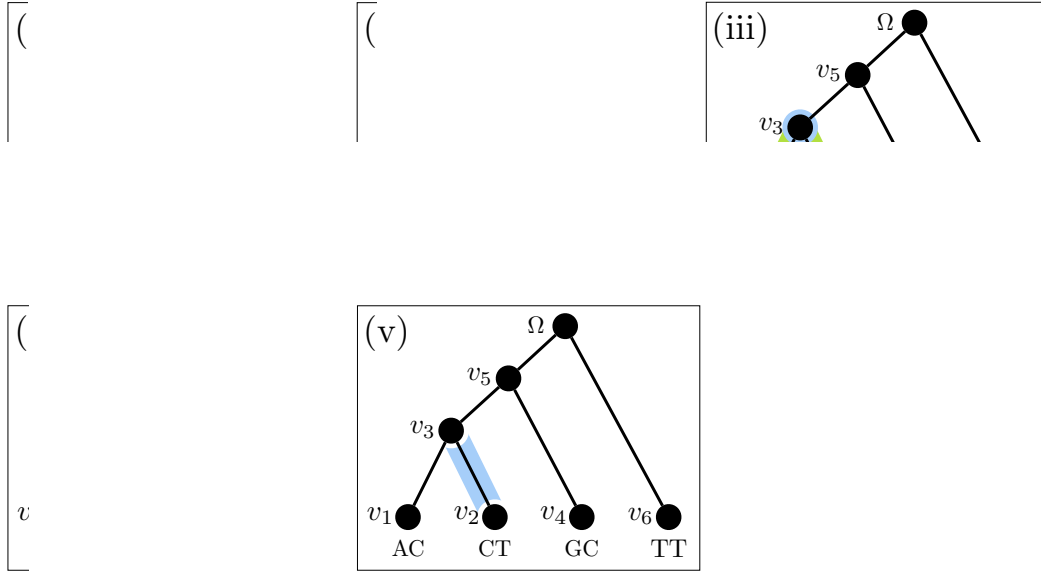


Figure 2.5: Schematic representation of all the homology paths resulting in an alignment column full of gaps at v_3 . In blue are shown the possible insertion locations and in green the edges where substitutions and deletions might occur.

Layer $m = 1$

A cell, in the three-dimensional DP *match* matrix $\mathbf{S}^M(i, j, m)$, depends only on the values contained in $\mathbf{S}^{M,X,Y}(i-1, j-i, m-1)$ ¹, a cell in the *gapX* matrix $\mathbf{S}^X(i, j, m)$ depends only on the values contained in $\mathbf{S}^{M,X,Y}(i-1, j, m-1)$ and a cell in the *gapY* matrix $\mathbf{S}^Y(i, j, m)$ depends only on the values contained in $\mathbf{S}^{M,X,Y}(i, j-i, m-1)$. Thus, cells laying in the same layer are independent among them which allows us to compute them in parallel.

At layer $m = 1$, the three matrices are extending the optimal alignment obtained at the previous layer ($m - 1$). Since at $m = 0$, there is only a single value at position $(0, 0, 0)$, the algorithm computes at $m = 1$ only three values, that are $\mathbf{S}^M(1, 1, 1)$, $\mathbf{S}^X(1, 0, 1)$ and $\mathbf{S}^Y(0, 1, 1)$. All the other values are equal to 0 (see Figure 2.6 for a graphical representation). The likelihood at $\mathbf{S}^M(1, 1, 1)$ corresponds to the likelihood of matching $\mathbf{X}_1 = A$ with $\mathbf{Y}_1 = C$, i.e. $(\mathbf{X}_1 \diamond \mathbf{Y}_1) \Rightarrow p\left(\begin{bmatrix} A \\ C \end{bmatrix}\right)$. The likelihood of the implied homology path,

¹where $\mathbf{S}^{M,X,Y}$ stands for \mathbf{S}^M , \mathbf{S}^X and \mathbf{S}^Y .

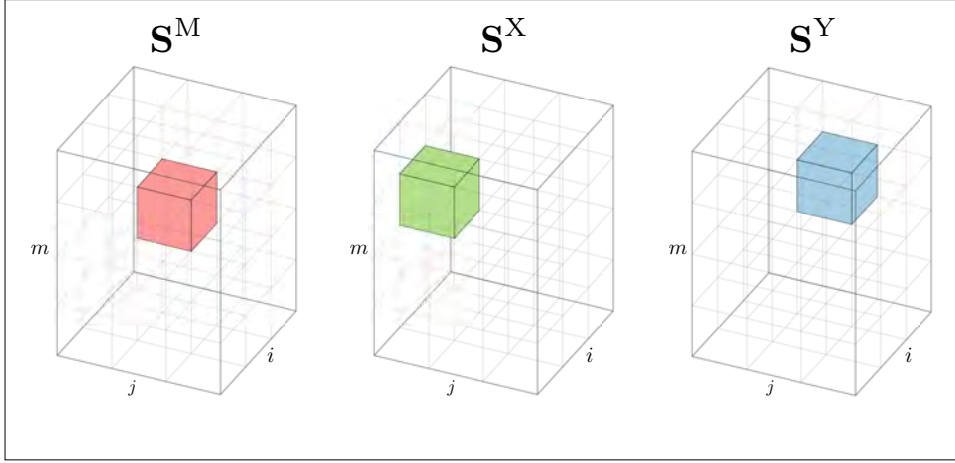


Figure 2.6: At layer $m = 1$, which corresponds to an alignment of size $|m| = 1$, the algorithm computes at $\mathbf{S}^M(1, 1, 1)$ the likelihood of matching \mathbf{X}_1 with \mathbf{Y}_1 . $\mathbf{S}^M(1, 1, 1)$ extends an alignment whose likelihood is contained at $\mathbf{S}^{M,X,Y}(0, 0, 0)$. $\mathbf{S}^X(1, 0, 1)$ contains the likelihood of the best alignment obtained at $m = 0$ at position $(0, 0, 0)$ and the likelihood of matching \mathbf{X}_1 with a gap on the right sub-tree. At $\mathbf{S}^Y(0, 1, 1)$ the algorithm computes the likelihood of the optimal alignment at layer $m = 0$ located at position $(0, 0, 0)$ with the likelihood of matching a gap on the left sub-tree with \mathbf{Y}_1 .

represented in Figure 2.7, is computed as

$$\mathbf{S}^M(1, 1, 1) = \frac{1}{1} \cdot \|\nu\| \cdot p\left(\begin{bmatrix} \text{A} \\ \text{C} \end{bmatrix}\right) \cdot \max \begin{cases} \mathbf{S}^M(0, 0, 0) \\ \mathbf{S}^X(0, 0, 0) \\ \mathbf{S}^Y(0, 0, 0) \end{cases} \quad (2.2)$$

where the *match* probability gets

$$p\left(\begin{bmatrix} \text{A} \\ \text{C} \end{bmatrix}\right) = \alpha(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)), \quad (2.3)$$

and $\mathbf{f}(v_3)$ is equal to

$$\mathbf{f}(v_3) = \left(\exp(b(v_1)\mathbf{Q}_\epsilon)\mathbf{f}(v_1)\right) * \left(\exp(b(v_2)\mathbf{Q}_\epsilon)\mathbf{f}(v_2)\right). \quad (2.4)$$

At the leaves v_1 and v_2 the felsenstein's weights are $\mathbf{f}(v_1) = [1, 0, 0, 0, 0]^\top$ and $\mathbf{f}(v_2) = [0, 1, 0, 0, 0]^\top$ ², respectively.

²Here we are considering an extended nucleotide alphabet in this order $\mathcal{A}_\epsilon = \{\text{A}, \text{C}, \text{G}, \text{T}, -\}$.

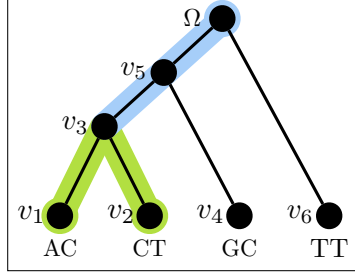


Figure 2.7: Homology path representing a match of two characters at the leaves v_1 and v_2 . A match at the internal node v_3 consists of an insertion of a character along the path $\rho(v_3, \Omega)$ and its survival till v_3 (whose likelihood is computed by the function $\alpha(\cdot)$) and the independent evolution along the two edges e_1 and e_2 through a Markov process of substitution and deletion.

The function $\alpha(v_3)$ represents the total prior insertion probabilities and survival probabilities of a character along the path $\rho(v_3, \Omega)$ which is equal to $(1/\mu)/(\|\tau\| + 1/\mu)$. π_ϵ is an array of size $|\mathcal{A}_\epsilon| \times 1$ containing the steady state probabilities of all the characters in the extended alphabet $\mathcal{A}_\epsilon = \mathcal{A} \cup \epsilon$.³ $\mathbf{f}(v_1)$ and $\mathbf{f}(v_2)$ are arrays of size $|\mathcal{A}_\epsilon| \times 1$ full of zeros and a 1 at the index corresponding to the position of the observed character at the leaf in the alphabet. Hence, a character A is associated to the index 0, C \rightarrow 1, G \rightarrow 2, T \rightarrow 3 and ‘-’ \rightarrow 4. The factor $\frac{1}{1} \cdot \|\nu\|$ in equation 2.2 stands for the marginal likelihood

$$\varphi(p_{v_3}(c_\emptyset), 1) = \frac{1}{1!} \|\nu\|^1 \exp(\|\nu\|(p_{v_3}(c_\emptyset) - 1))$$

where the factor $\exp(\|\nu\|(p_{v_3}(c_\emptyset) - 1))$ is already contained at $\mathbf{S}^{\mathbf{M}, \mathbf{X}, \mathbf{Y}}(0, 0, 0)$.

The entry $\mathbf{S}^{\mathbf{X}}(1, 0, 1)$ encloses the likelihood of the two homologous histories compatible with the MSA column $\begin{bmatrix} \text{A} \\ - \end{bmatrix}$ and is computed as the sum of the likelihood of the two scenarios depicted in Figure 2.8. The likelihood of aligning \mathbf{X}_1 with a gap on the right sub-tree, i.e. $(\mathbf{X}_1 \diamond c_\emptyset) \Rightarrow p\left(\begin{bmatrix} \text{A} \\ - \end{bmatrix}\right)$ is obtained as

$$\mathbf{S}^{\mathbf{X}}(1, 0, 1) = \frac{1}{1} \cdot \|\nu\| \cdot p\left(\begin{bmatrix} \text{A} \\ - \end{bmatrix}\right) \cdot \max \begin{cases} \mathbf{S}^{\mathbf{M}}(0, 0, 0) \\ \mathbf{S}^{\mathbf{X}}(0, 0, 0) \\ \mathbf{S}^{\mathbf{Y}}(0, 0, 0) \end{cases} \quad (2.5)$$

³More precisely the quasi-limiting distribution $\pi_\epsilon = [\pi_A, \pi_C, \pi_G, \pi_T, 0]^\top$

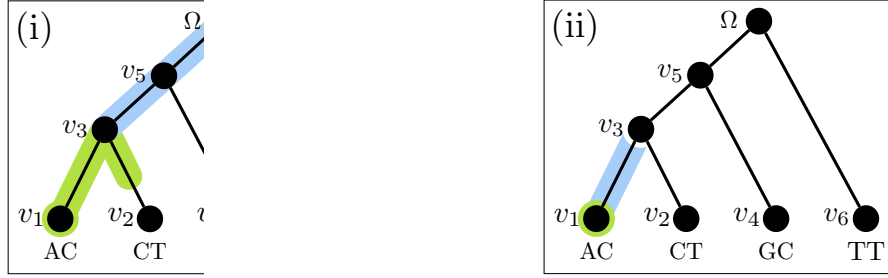


Figure 2.8: At the node v_3 , there are two possible homologous scenarios that are compatible with the alignment column $[A, -]^T$. The first one is an insertion along the common path $\rho(v_3, \Omega)$ and then, successively deletion along the edge e_2 . The second scenario is given by an insertion along the edge e_1 . The likelihood of both possible histories is computed and summed together at $\mathbf{S}^X(1, 0, 1)$.

where

$$p\left(\begin{bmatrix} A \\ - \end{bmatrix}\right) = \alpha(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) + \quad (\text{i})$$

$$+ \iota(v_3) \beta(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) \quad (\text{ii})$$

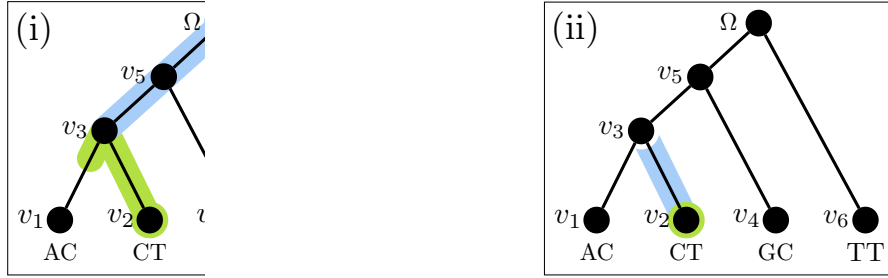


Figure 2.9: At the node v_3 , there are two possible homologous scenarios that are compatible with the alignment column $[-, C]^T$. The first one is an insertion along the common path $\rho(v_3, \Omega)$ and then, successively a deletion along the edge e_1 . The second scenario is given by an insertion along the edge e_2 . The likelihood of both possible histories is computed and summed together at $\mathbf{S}^Y(0, 1, 1)$.

The entry $\mathbf{S}^Y(0, 1, 1)$ is calculated by summing the likelihood of the homologous scenarios depicted in Figure 2.9. The likelihood of aligning \mathbf{Y}_1 with a gap on the left sub-tree, i.e. $(c_\emptyset \diamond \mathbf{Y}_1) \Rightarrow p\left(\begin{bmatrix} - \\ C \end{bmatrix}\right)$ is calculated with the

following equation

$$\mathbf{S}^Y(0, 1, 1) = \frac{1}{1} \cdot \|\nu\| \cdot p\left(\begin{bmatrix} - \\ C \end{bmatrix}\right) \cdot \max \begin{cases} \mathbf{S}^M(0, 0, 0) \\ \mathbf{S}^X(0, 0, 0) \\ \mathbf{S}^Y(0, 0, 0) \end{cases} \quad (2.6)$$

where

$$p\left(\begin{bmatrix} - \\ C \end{bmatrix}\right) = \alpha(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) + \quad (i)$$

$$+ \iota(v_2)\beta(v_2) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_2)) \quad (ii)$$

and $\mathbf{f}(v_3)$ is computed as in equation 2.4 with $\mathbf{f}(v_1) = [0, 0, 0, 0, 1]^\top$ and by setting at node v_2 the felsenstein's array as $\mathbf{f}(v_2) = [0, 1, 0, 0, 0]^\top$.

Layer $m = 2, \dots$

At layer $m = 2$ and at all the following layers the likelihood is computed in the same way as outlined above. The cells with a likelihood different from 0, at $m = 2$, are represented graphically in Figure 2.10 and they are located at

- (i) in the match matrix at positions (2, 1, 2), (1, 2, 2) and (2, 2, 2);
- (ii) in the gapX matrix at positions (2, 0, 2), (2, 1, 2) and (1, 1, 2) and
- (iii) in the matrix gapY at positions (1, 1, 2), (1, 2, 2) and (0, 2, 2).

2.3 Alignment at node v_5

The DP algorithm generates at each internal node an optimal pairwise alignment under the PIP model. At node v_5 the algorithm builds the new optimal pairwise alignment starting from the optimal sub-alignment obtained at v_3 and the input sequence associated at v_4 . Let's suppose that the optimal pairwise alignment originated at node v_3 is the following

$$\mathbf{X} = \begin{bmatrix} A & C & - \\ - & C & T \end{bmatrix} \quad (2.7)$$

and that the input sequence associated at the leaf v_4 is $\mathbf{Y} = \text{GC}$. The equations involved in the alignment process at node v_5 for $m = 0$ and $m = 1$ are shown in the next sections.

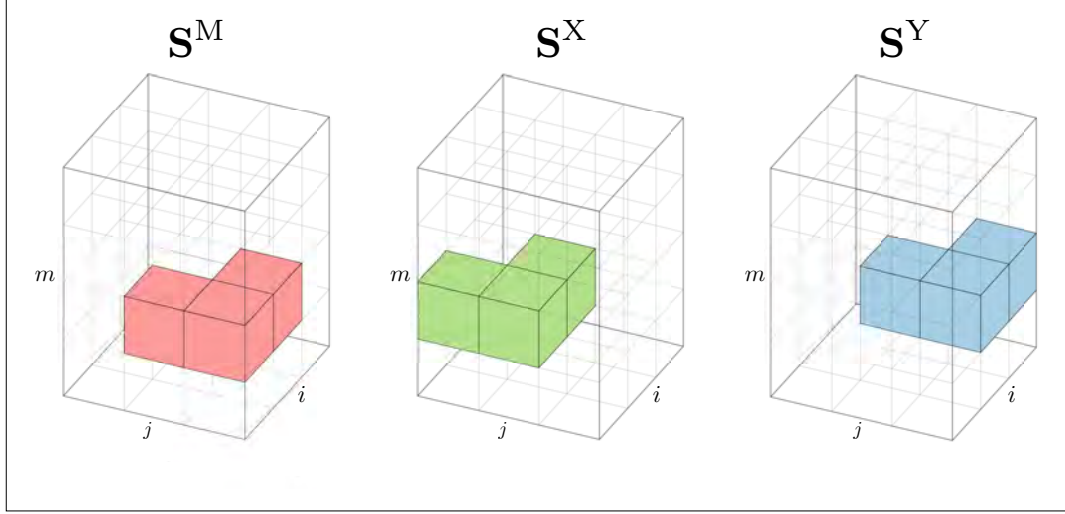


Figure 2.10: At layer $m = 2$ the DP algorithm computes the likelihood of all possible alignments of size $|m| = 2$. The entries containing a value different from 0 are colored, all the others positions corresponds to unattainable alignments.

Layer $m = 0$

Like as at v_3 , the algorithm starts at $m = 0$ by computing the marginal likelihood of empty columns $\varphi(p_{v_5}(c_\emptyset), 0)$ for an alignment of no observable columns, i.e. $|m| = 0$ (see equation 2.1) and puts this result at position $(0, 0, 0)$ in all three matrices $\mathbf{S}^{M,X,Y}$. The likelihood of a single column full of gaps at v_5 gets

$$p_{v_5}(c_\emptyset) = \alpha(\Omega)\xi(\rho(v_5, \Omega)) + \quad (\text{i})$$

$$+ \iota(\rho(v_5, \Omega)) \left(1 - \beta(\rho(v_5, \Omega))\right) + \quad (\text{ii})$$

$$+ \alpha(v_5)\eta(v_5) + \quad (\text{iii}), (\text{iv})$$

$$+ \iota(v_1) \left((1 - \beta(v_1) + \beta(v_1)(\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_1))) \right) + \quad (\text{vi})$$

$$+ \iota(v_2) \left((1 - \beta(v_2) + \beta(v_2)(\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_2))) \right) + \quad (\text{vii})$$

$$+ \iota(v_3) \left((1 - \beta(v_3) + \beta(v_3)(\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3))) \right) + \quad (\text{v}), (\text{ix})$$

$$+ \iota(v_4) \left((1 - \beta(v_4) + \beta(v_4)(\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_4))) \right). \quad (\text{viii})$$

The equation numbers, expressed in Roman numerals, refer to the homology paths depicted graphically in Figure 2.11.

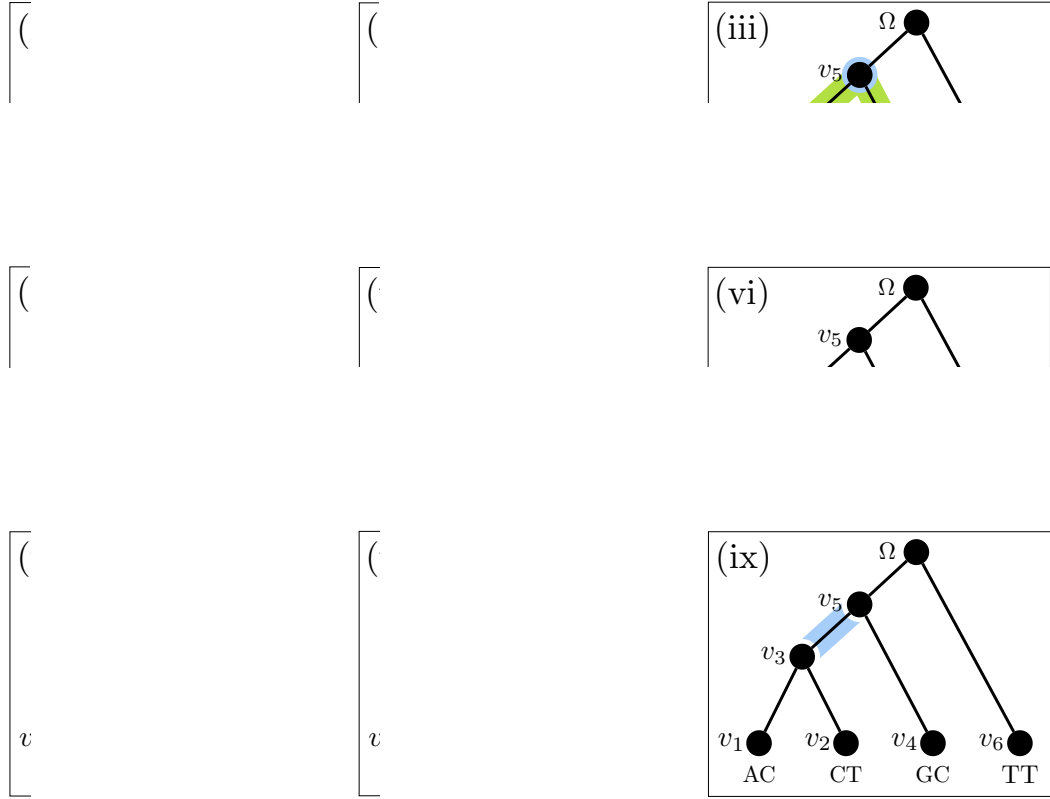


Figure 2.11: At node v_5 there are 9 homology paths (i) – (ix) that produce an MSA column full of gaps at the leaves.

Layer $m = 1$

At $m = 1$, the algorithm computes for example in $\mathbf{S}^X(1, 0, 1)$ the likelihood of the homology path $(\mathbf{X}_1 \diamond c_\emptyset)$ resulting in the MSA column $\begin{bmatrix} A \\ - \\ - \end{bmatrix}$ by means of the following formula

$$\begin{aligned}
 p\left(\begin{bmatrix} A \\ - \\ - \end{bmatrix}^\top\right) &= \alpha(v_5) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_5)) + & \text{(i)} \\
 &+ \iota(v_3)\beta(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) + & \text{(ii)} \\
 &+ \iota(v_1)\beta(v_1) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_1)) & \text{(iii)} \quad (2.8)
 \end{aligned}$$

where $\mathbf{f}(v_5)$ is computed, as shown in equation 2.4, by

$$\begin{aligned}
 \mathbf{f}(v_1) &= [1, 0, 0, 0, 0]^\top & \mathbf{f}(v_4) &= [0, 0, 0, 0, 1]^\top & \mathbf{f}(v_2) &= [0, 0, 0, 0, 1]^\top \\
 \mathbf{f}(v_5) &= \mathbf{f}(v_3) * \mathbf{f}(v_4) & \mathbf{f}(v_3) &= \mathbf{f}(v_1) * \mathbf{f}(v_2)
 \end{aligned}$$



Figure 2.12: There are three scenarios at v_5 that are compatible with an MSA column $[A, -, -]^T$. The likelihood of all these scenarios are summed together at $\mathbf{S}^X(1, 0, 1)$.

The corresponding homology paths are illustrated in Figure 2.12.

To save computational time – during the forward phase – the algorithm caches in the current node the computed likelihood values that will be reused in a successive phase of the alignment (at the parent node). For instance, in equation 2.8, the second and third terms of the summation, i.e. $\iota(v_3)\beta(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) + \iota(v_1)\beta(v_1) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_1))$, have been already computed at node v_3 to obtain the likelihood of the alignment $[\frac{A}{-}]$. To be more precise, at node v_3 the algorithm computes $\alpha(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) + \dots$ but for further reuse stores $\iota(v_3)\beta(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) + \dots$

A similar situation appears for example in the gapX matrix \mathbf{S}^X for the computation of the column likelihood (Figure 2.13) $[-, T, -]^T$ which is

$$\begin{aligned}
 p\left([-, T, -]^T\right) &= \alpha(v_5) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_5)) + & \text{(i)} \\
 &+ \iota(v_3)\beta(v_3) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_3)) + & \text{(ii)} \\
 &+ \iota(v_2)\beta(v_2) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_2)) & \text{(iii)} \quad (2.9)
 \end{aligned}$$

where

$$\begin{aligned}
 \mathbf{f}(v_1) &= [0, 0, 0, 0, 1]^T & \mathbf{f}(v_2) &= [0, 0, 0, 1, 0]^T & \mathbf{f}(v_4) &= [0, 0, 0, 0, 1]^T \\
 \mathbf{f}(v_3) &= \mathbf{f}(v_1) * \mathbf{f}(v_2) & \mathbf{f}(v_5) &= \mathbf{f}(v_3) * \mathbf{f}(v_4).
 \end{aligned}$$

Also in equation 2.9, the second and third terms have been already computed during the forward phase at node v_3 . These values are stored, during the backward phase, at node v_3 to be re-used from its parent node, namely v_5 .

At $m = 2$, the DP computes the column likelihood of the alignment $[\frac{-}{G}]$ by



Figure 2.13: There are three scenarios at v_5 that are compatible with an MSA column $[-, T, -]^T$. The likelihood of all these scenarios are summed together in the matrix \mathbf{S}^X .

means of the following equation

$$\begin{aligned}
 p\left([-,-,G]^T\right) &= \alpha(v_5) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_5)) + && \text{(i),(ii)} \\
 &+ \iota(v_4)\beta(v_4) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_4)) && \text{(iii)} \quad (2.10)
 \end{aligned}$$

where the corresponding homologous paths are represented in Figure 2.14.



Figure 2.14: There are three scenarios at v_5 that are compatible with an MSA column $[-,-,G]^T$. The likelihood of all these scenarios are summed together in the matrix \mathbf{S}^Y .

2.4 Alignment at node Ω

At the root the algorithm works in the same way as shown above. The likelihood of an empty column at the root node marginalizes over the homologous scenarios represented graphically in Figure 2.15.

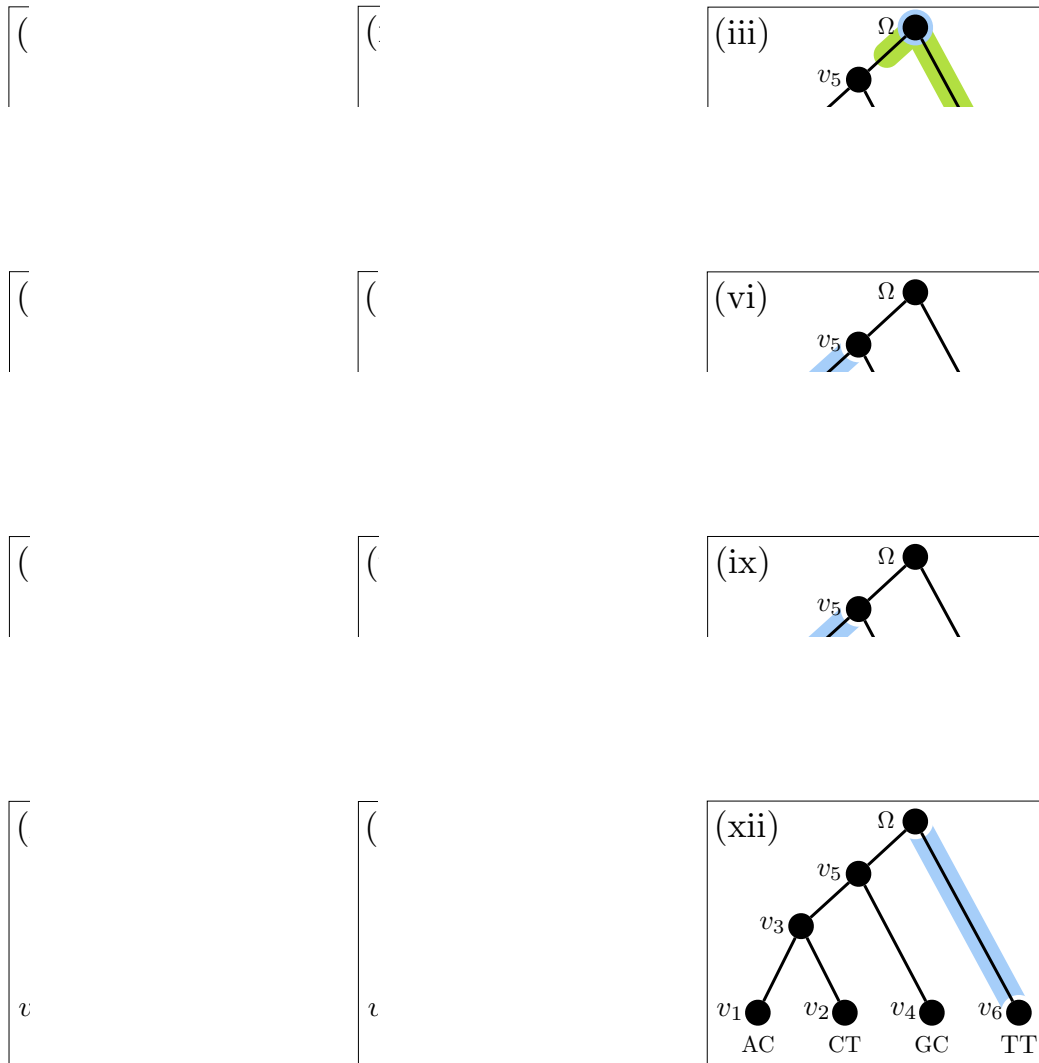


Figure 2.15: At the root Ω the DP algorithm computes the likelihood of all these scenarios which are all producing a column full of gaps in the alignment. The column likelihood is then used in the function $\varphi(\cdot)$ to compute the marginal likelihood for an infinite and unobservable number of empty columns.

2.5 Tracebacking

An optimal alignment is determined by backtracking along a trace-back matrix \mathbf{S}^T of size $(|\mathbf{X}| + 1) \times (|\mathbf{Y}| + 1) \times (|\mathbf{X}| + |\mathbf{Y}| + 1)$. In the forward phase, \mathbf{S}^T records at position (i, j, m) the name⁴ of the DP matrix with highest likelihood

⁴we have assigned the following integer names : $\mathbf{S}^M = 1$, $\mathbf{S}^X = 2$ and $\mathbf{S}^Y = 3$.

at the same position, i.e. (i, j, m) . If the maximum is not unique then a uniform random choice is made. The backtracking algorithm starts at position $\mathbf{S}^T(|\mathbf{X}|, |\mathbf{Y}|, k_0)$, where

$$k_0 = \arg \max_k \left(\mathbf{S}^{\text{M},\text{X},\text{Y}}(|\mathbf{X}|, |\mathbf{Y}|, k) \right), \quad (2.11)$$

and $k = \max(|\mathbf{X}|, |\mathbf{Y}|), \dots, (|\mathbf{X}| + |\mathbf{Y}|)$. In equation 2.11 k_0 corresponds to the length of the best scoring (ML) alignment. Figure 2.16 highlights the column containing all

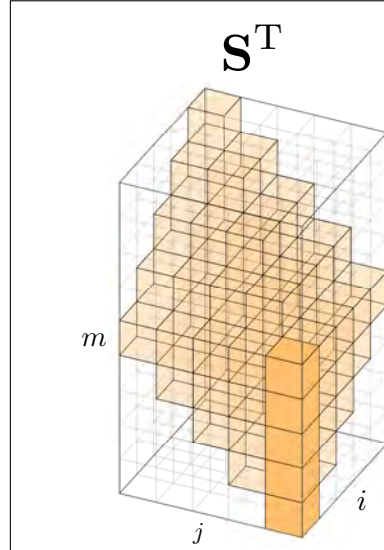


Figure 2.16: Possible starting points of the traceback phase. The backtracking algorithm starts the traceback path at the position containing the maximum likelihood value (one of those highlighted in the column). The level, k_0 , where the maximum has been identified, corresponds to the MSA length while the value corresponds to the corresponding MSA likelihood.

If k_0 is not unique a random uniform choice is made. \mathbf{S}^T is then traversed from $(|\mathbf{X}|, |\mathbf{Y}|, k_0)$ to $(0, 0, 0)$. Suppose the algorithm is at position (i, j, k) . If $\mathbf{S}^T(i, j, k) = 1$ (which corresponds to *match*) then the columns \mathbf{X}_i and \mathbf{Y}_j are matched and all the indices are decremented, i.e. $i \leftarrow i - 1$, $j \leftarrow j - 1$, $m \leftarrow m - 1$. If $\mathbf{S}^T(i, j, m)$ is set to 2 (*gapX*) then the column \mathbf{X}_i is matched with a column of only gaps and the indices i and m are decremented, and, if $\mathbf{S}^T(i, j, m)$ contains the value 3 (*gapY*) then the column \mathbf{Y}_j is matched with a column of gaps and the indices j and m are decremented. Figure 2.17 shows the entries filled in the traceback matrix \mathbf{S}^T at $m = 0, 1, 2$.

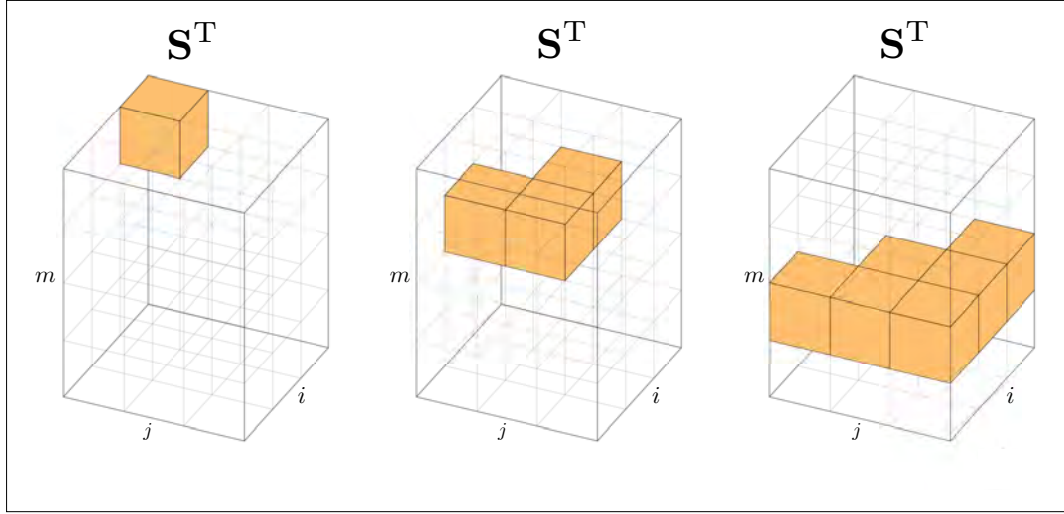


Figure 2.17: Colored cells show the entries different from 0 in the traceback matrix \mathbf{S}^T at layer $m = 0$ (left), $m = 1$ (middle) and $m = 2$ (right). They correspond to all the entries different from 0 in any of the matrices \mathbf{S}^M , \mathbf{S}^X or \mathbf{S}^Y . A ‘1’ at a given position indicates that at the same position the matrix containing the highest likelihood value is the match matrix \mathbf{S}^M . The same applies for \mathbf{S}^X with code ‘2’ and \mathbf{S}^Y with code ‘3’.

2.6 Early stop condition

We have observed, empirically, that the likelihood values in the three dynamic programming matrices at $\mathbf{S}^{M,X,Y}(|\mathbf{X}|, |\mathbf{Y}|, m)$ with $m = \max(|\mathbf{X}|, |\mathbf{Y}|), \dots, (|\mathbf{X}| + |\mathbf{Y}|)$ exhibits only a single global optima. In Figure 2.18 we have plotted the highest likelihood values of the “last column” (among $\mathbf{S}^{M,X,Y}$) for a symmetric 16-taxon topology with uniform branch lengths. The three plots refer to the results obtained with dataset *close*, *intermediate* and *distant* (see reference [85] for more details). Each curve in Figure 2.18 represents an instance of the pairwise DP alignment. Therefore, for a 16-taxon topology, the progressive approach visits 15 internal nodes and for each of them runs a DP instance. The values plotted in Figure 2.18 correspond to

$$\text{lk}_v[i] = \max\left(\mathbf{S}^{M,X,Y}(|\mathbf{X}|, |\mathbf{Y}|, m)\right) \quad (2.12)$$

for $i = \max(|\mathbf{X}|, |\mathbf{Y}|), \dots, (|\mathbf{X}| + |\mathbf{X}|)$ and $v \in \mathcal{V} \setminus \mathcal{L}$.

It is interesting to note that the curves lk_v are smooth and present only a single maximum value. This empirical observation suggests a practical termination condition – called *early stop condition* – that may interrupt the compu-

tation in advance and consequently saving CPU time. The algorithm stores, for each layer m , the highest likelihood value lk_v at $\mathbf{S}^{\mathbf{M},\mathbf{X},\mathbf{Y}}(|\mathbf{X}|, |\mathbf{Y}|, m)$. When the highest likelihood decreases k times consecutively (k set by the user), the algorithm stops returning the optimal pairwise MSA. By terminating the algorithm few steps after the maximum likelihood value has been found can, in the best case, halve the computational complexity, that is from $\mathcal{O}(\frac{1}{3}L^3)$ to $\mathcal{O}(\frac{1}{6}L^3)$ (L is the average sequence length). We did not demonstrate that the functions lk_v present a single optima but there are indications that support this empirical observation, in particular the trend $\varphi(\cdot)$ and the fact that $p(c)$ – a generic MSA column likelihood – is always positive.

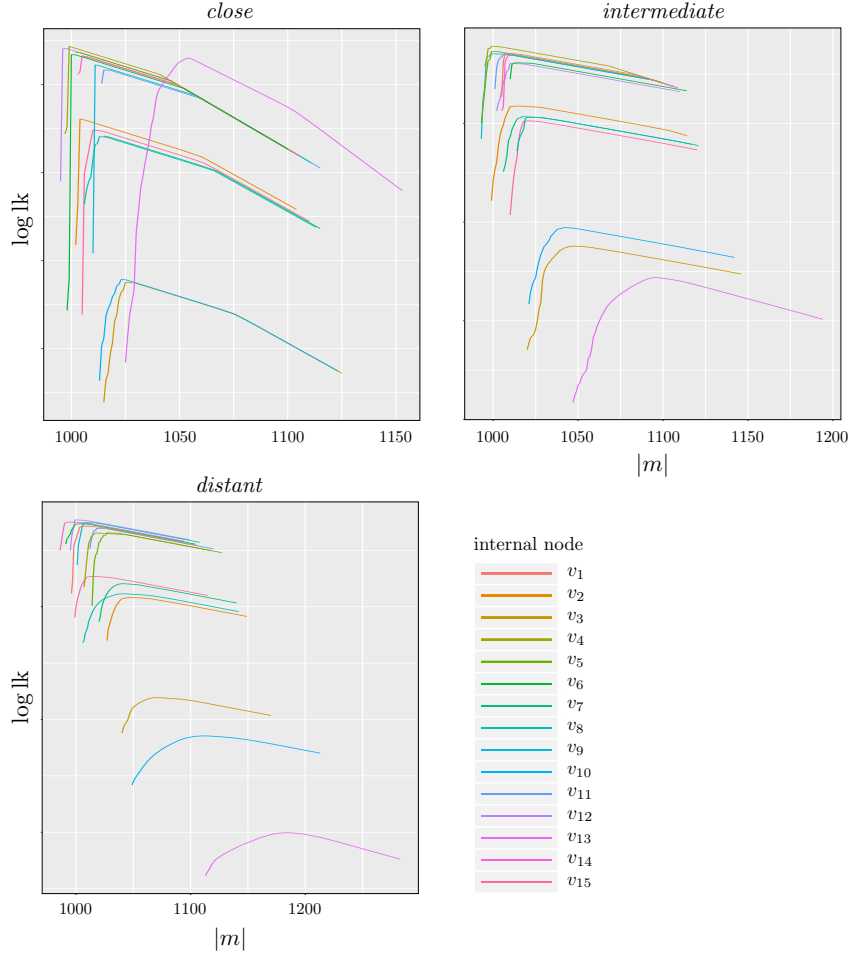


Figure 2.18: Highest log likelihood values as function of the MSA length (depth m in the dynamic programming matrices) as defined in equation 2.12. The curves refer to the 15 internal nodes of a symmetric, balanced 16-taxon tree.

3

Stochastic backtracking DP algorithm

“If I have seen further it is by standing on the shoulders of Giants.”

– Isaac Newton , *The Correspondence Of Isaac Newton*

3.1 Introduction

A classic DP algorithm returns by construction only a single optimal solution. In our context, the DP method produces the maximum likelihood pairwise alignment at each internal node of a given phylogenetic tree. The optimal pairwise alignment, thought, conditioned on the two sub-alignments present at the children node. To reduce the overall complexity, our DP alignment procedure has been framed into a progressive approach. Whilst, from one side the progressive procedure considerably reduces the search space where to find the optimal alignment, and thus the computational complexity, on the other side it might introduce a bias (see also Appendix 6: [Progressive bias analysis](#)). Indeed, this greedy heuristics works by picking, at each instance, the best partial solution and step-by-step builds the final multiple sequence alignment of all the sequences. Unfortunately, as is often the case, the progressive approach gets stuck into sub-optimal MSAs. The reason lies in the fact that the algorithm has to make decisions at early stages based on partial information and being constrained onto a sub-space of the entire multiple sequence alignments space. Thus, to mitigate the algorithm greediness we have implemented a stochastic backtracking dynamic programming algorithm [100] which generates, at each internal node, not a single optimal pairwise alignment but rather an en-

semble of sub-optimal candidate solutions, the latter distributed according to their probability. Then, at the parent node, instead of computing the pairwise alignment based on only two sub-MSAs, the aligner generates an ensemble of MSA combining samples from the distributions at the children nodes. This approach decreases the chances to be trapped in local optima engendered by the greedy nature of the ‘pure’ progressive DP procedure.

Our stochastic backtracking (SB) DP approach has been inspired on the method proposed in 2002 by Mueckstein et al. [100]. However, of course, our proposed algorithm computes the pairwise alignment under PIP based on the three-dimensional DP matrices.

3.2 Partition functions

At the core of the idea proposed by Mueckstein et al. [100], there is the calculation of three *partition functions* which represent a particular statistical ensemble of alignments. In the alignment context, the partition function $Z(T)$, at a given ‘temperature’ T (explained later), is defined as:

$$Z(T) = \sum_{\mathcal{M}} \exp\left(\frac{S(\mathcal{M})}{kT}\right) = \sum_{\mathcal{M}} \exp(\beta S(\mathcal{M})) \quad (3.1)$$

with $\beta = (kT)^{-1}$. The summation in equation 3.1 goes over all possible alignments \mathcal{M} and $S(\mathcal{M})$ represents an alignment scoring function¹. The parameter k is the analogous of the Boltzmann constant (a model dependent parameter) and the temperature T – in this context – tunes the “stochastic deviation” from the optimal alignment (see also Section 3.6 for more details and examples). In practice, in our algorithm we are considering kT as a single parameter, called T , since in the alignment problem there is no connection to the real temperature value.

The probability of a given alignment \mathcal{M} at the ‘temperature’ T becomes therefore

$$\mathbb{P}(\mathcal{M}, T) = \frac{1}{Z(T)} \exp(\beta S(\mathcal{M})). \quad (3.2)$$

The partition function $Z(T)$, in equation 3.2, plays the role of normalizing constant. In fact

$$\sum_{\mathcal{M}} \frac{1}{Z(T)} \exp(\beta S(\mathcal{M})) = 1. \quad (3.3)$$

¹the analogous of the negative energy in a thermodynamic context

To efficiently compute the summation in equation 3.1, the SB algorithm has been framed into a dynamic programming procedure [100].

In Section 3.4 and 3.5 we will explain the *forward* and *backward recursion*, respectively, for the computation of the partition functions under PIP.

3.3 The partition functions Z^M , Z^X and Z^Y

Let \mathbf{A} and \mathbf{B} be two sequences or sub-alignments at the children nodes of v and $|\mathbf{A}|$, $|\mathbf{B}|$ their length. Moreover, let denote $\mathbf{A}[1, \dots, i]$ and $\mathbf{B}[1, \dots, j]$ the sub-sequence or sub-alignment from column 1 to i and j , respectively. We can at this point define three different partition functions, namely Z^M , Z^X and Z^Y ([100]), which will be used in the *forward phase*, that are

- (i) $Z^M(i, j, m)$ indicates the partition function of all the alignments of sub-sequences / sub-alignments $\mathbf{A}[1, \dots, i]$ and $\mathbf{B}[1, \dots, j]$ ending – at position (i, j, m) – in state *match*. That means that column $\mathbf{A}(i) \diamond \mathbf{B}(j)$: $\mathbf{A}(i)$ is homologue to $\mathbf{B}(j)$. The probability of matching $\mathbf{A}(i)$ with $\mathbf{B}(j)$ is denoted $\mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)}$;
- (ii) $Z^X(i, j, m)$ indicates the partition function of all the alignment of sub-sequences / sub-alignments $\mathbf{A}[1, \dots, i]$ and $\mathbf{B}[1, \dots, j]$ ending in state *gapX* at position (i, j, m) . Column $\mathbf{A}(i)$ is aligned with a column full of gaps on the right sub-tree. The gapX probability of matching $\mathbf{A}(i)$ with a column of gaps is denoted $\mathbb{P}_{\mathbf{A}(i), \epsilon}$;
- (iii) $Z^Y(i, j, m)$ indicates the partition function of all the alignment of sub-sequences / sub-alignments $\mathbf{A}[1, \dots, i]$ and $\mathbf{B}[1, \dots, j]$ ending in state *gapY* at position (i, j, m) . Column $\mathbf{B}(j)$ is aligned with a column full of gaps on the left side. The gapY probability of matching $\mathbf{B}(j)$ with a column of gaps is denoted $\mathbb{P}_{\epsilon, \mathbf{B}(j)}$.

Similarly we define three partition functions, namely \widehat{Z}^M , \widehat{Z}^X and \widehat{Z}^Y , that will be used in the *backward phase*:

- (i) $\widehat{Z}^M(i, j, m)$ indicates the partition function of all the alignments of sub-sequences / sub-alignments $\mathbf{A}[i, \dots, |\mathbf{A}|]$ and $\mathbf{B}[j, \dots, |\mathbf{B}|]$ starting in *match* state – at position (i, j, m) – and containing all the characters / columns from i to $|\mathbf{A}|$ and from j to $|\mathbf{B}|$;

- (ii) $\widehat{\mathbf{Z}}^X(i, j, m)$ indicates the partition function of all the alignments of sub-sequences / sub-alignments $\mathbf{A}[i, \dots, |\mathbf{A}|]$ and $\mathbf{B}[j, \dots, |\mathbf{B}|]$ starting in *gapX* state – at position (i, j, m) – and containing all the characters / columns from i to $|\mathbf{A}|$ and from j to $|\mathbf{B}|$;
- (iii) $\widehat{\mathbf{Z}}^Y(i, j, m)$ indicates the partition function of all the alignments of sub-sequences / sub-alignments $\mathbf{A}[i, \dots, |\mathbf{A}|]$ and $\mathbf{B}[j, \dots, |\mathbf{B}|]$ starting in *gapY* state – at position (i, j, m) – and containing all the characters / columns from i to $|\mathbf{A}|$ and from j to $|\mathbf{B}|$.

The six partition functions defined above are graphically represented in Figure 3.1 by means of a simple example. Accordingly to the stated above definitions we can write the *match* probability at position (i, j, m) , notably $\mathbb{P}^M(i, j, m)$, which thus implies that $\mathbf{A}(i) \diamond \mathbf{B}(j)$, as

$$\mathbb{P}^M(i, j, m) = \frac{\mathbf{Z}^M(i, j, m) \cdot \widehat{\mathbf{Z}}^M(i, j, m)}{\mathbf{Z}} \cdot \frac{1}{\mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)}}}; \quad (3.4)$$

the *gapX* probability, at position (i, j, m) , as

$$\mathbb{P}^X(i, j, m) = \frac{\mathbf{Z}^X(i, j, m) \cdot \widehat{\mathbf{Z}}^X(i, j, m)}{\mathbf{Z}} \cdot \frac{1}{\mathbb{P}_{\mathbf{A}(i), \epsilon}}; \quad (3.5)$$

and *gapY* probability, at position (i, j, m) , as

$$\mathbb{P}^Y(i, j, m) = \frac{\mathbf{Z}^Y(i, j, m) \cdot \widehat{\mathbf{Z}}^Y(i, j, m)}{\mathbf{Z}} \cdot \frac{1}{\mathbb{P}_{\epsilon, \mathbf{B}(j)}}. \quad (3.6)$$

In equations 3.4 - 3.6, \mathbf{Z} indicates the total partition function, that is $\mathbf{Z} = \mathbf{Z}^M + \mathbf{Z}^X + \mathbf{Z}^Y$. $\mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)}$ is the probability of matching column $\mathbf{A}(i)$ with column $\mathbf{B}(j)$, $\mathbb{P}_{\mathbf{A}(i), \epsilon}$ represents the probability of aligning the column $\mathbf{A}(i)$ with a column full of gaps on the right side and $\mathbb{P}_{\epsilon, \mathbf{B}(j)}$ is the probability of aligning a the column full of gaps on the left sub-tree with the column $\mathbf{B}(j)$ (more details in Section 1: [Progressive Dynamic Programming under PIP model](#)). It is worth noting, that in equations 3.4 - 3.6 the factors $(\mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)})^{-1}$, $(\mathbb{P}_{\mathbf{A}(i), \epsilon})^{-1}, (\mathbb{P}_{\epsilon, \mathbf{B}(j)})^{-1}$, have been added to remove the respective probabilities *match*, *gapX* and *gapY* which otherwise appear twice, both in $\mathbf{Z}^{M, X, Y}(i, j, m)$ and in $\widehat{\mathbf{Z}}^{M, X, Y}(i, j, m)$.

For illustrative purposes, Figure 3.2 graphically portrays the different paths connected to the cell $\mathbb{P}^M(3, 3, 4)$ (colored cell). The sum over all paths starting at $(0, 0, 0)$ and ending at position $(3, 3, 4)$ is computed by the function $\mathbf{Z}^M(3, 3, 4)$. The sum over all paths starting at position $(3, 3, 4)$ and ending at $(|\mathbf{A}|, |\mathbf{B}|, m)$ is computed by the function $\widehat{\mathbf{Z}}^M(3, 3, 4)$.

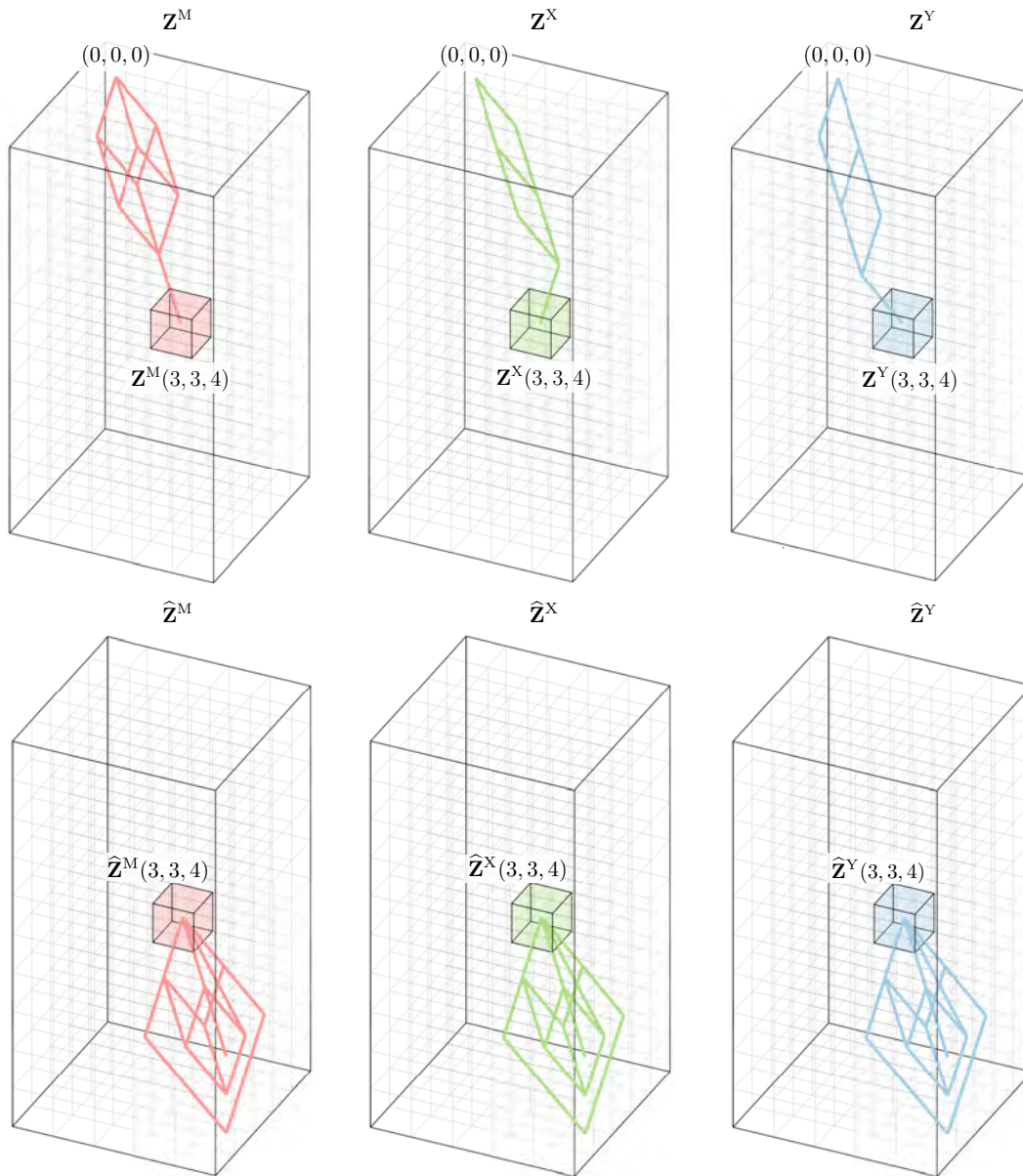


Figure 3.1: Example of the six partition functions, defined in the text, evaluated at the position $(3, 3, 4)$, i.e. $\hat{Z}^{M,X,Y}(3, 3, 4)$ and $\hat{Z}^{M,X,Y}(3, 3, 4)$.

3.4 Forward recursion

During the forward phase the algorithm incrementally computes the three partition functions $Z^{M,X,Y}$. Differently from a classic DP approach, the stochastic

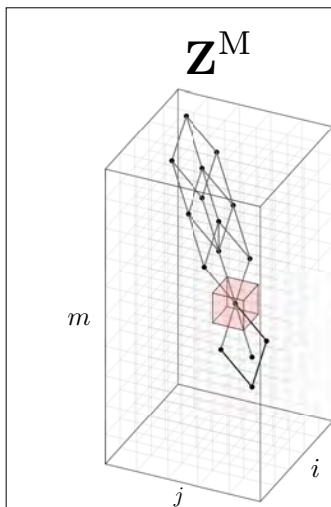


Figure 3.2: Paths connected to $\mathbb{P}^M(3, 3, 4)$. The sum over all paths starting at $(0, 0, 0)$ and ending at position $(3, 3, 4)$ is computed by the function $\mathbf{Z}^M(3, 3, 4)$. Such paths are upwards of the cell $(3, 3, 4)$. The sum over all paths starting at position $(3, 3, 4)$ and ending at $(|\mathbf{A}|, |\mathbf{B}|, m)$ (at any layer m when all the characters have been aligned) is computed by the function $\hat{\mathbf{Z}}^M(3, 3, 4)$. These paths are downstream of the cell $(3, 3, 4)$.

backtracking (SB) algorithm does not keep track of the matrix containing the highest likelihood. The reason lies in the fact that the backtracking phase builds paths stochastically rather than deterministically, based on the partition functions

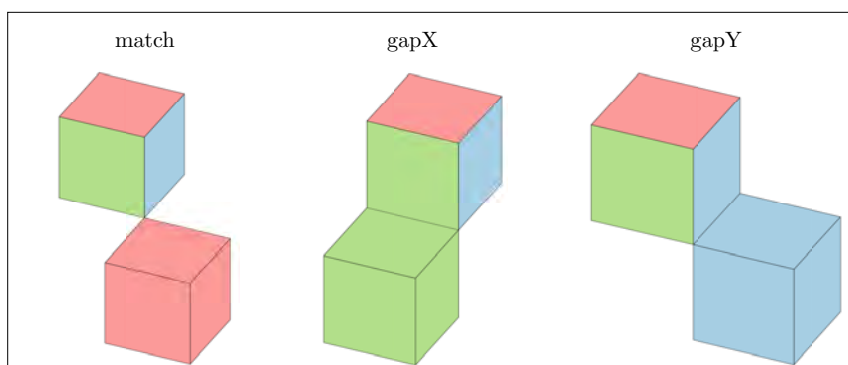
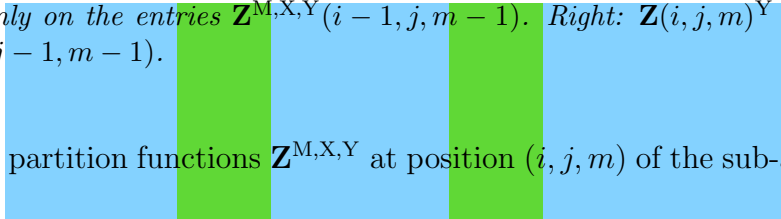


Figure 3.3: Left: A cell in $\mathbf{Z}^M(i, j, m)$ depends only on the three entries $\mathbf{Z}^{M,X,Y}(i-1, j-1, m-1)$ represented here with three colors. Middle: A cell in $\mathbf{Z}(i, j, m)^X$ depends only on the entries $\mathbf{Z}^{M,X,Y}(i-1, j, m-1)$. Right: $\mathbf{Z}(i, j, m)^Y$ depends on $\mathbf{Z}^{M,X,Y}(i, j-1, m-1)$.

The three partition functions $\mathbf{Z}^{M,X,Y}$ at position (i, j, m) of the sub-alignments



$\mathbf{A}[1, \dots, i]$ and $\mathbf{B}[1, \dots, j]$ can be computed by means of the recursive formulas:

$$\begin{aligned} \mathbf{Z}^{\mathbf{M}}(i, j, m) = & (\mathbf{Z}^{\mathbf{M}}(i-1, j-1, m-1) + \mathbf{Z}^{\mathbf{Y}}(i-1, j-1, m-1) + \\ & + \mathbf{Z}^{\mathbf{X}}(i-1, j-1, m-1)) \cdot \mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)} \end{aligned} \quad (3.7)$$

$$\begin{aligned} \mathbf{Z}^{\mathbf{X}}(i, j, m) = & (\mathbf{Z}^{\mathbf{M}}(i-1, j, m-1) + \mathbf{Z}^{\mathbf{Y}}(i-1, j, m-1) + \\ & + \mathbf{Z}^{\mathbf{X}}(i-1, j, m-1)) \cdot \mathbb{P}_{\mathbf{A}(i), \epsilon} \end{aligned} \quad (3.8)$$

$$\begin{aligned} \mathbf{Z}^{\mathbf{Y}}(i, j, m) = & (\mathbf{Z}^{\mathbf{M}}(i, j-1, m-1) + \mathbf{Z}^{\mathbf{Y}}(i, j-1, m-1) + \\ & + \mathbf{Z}^{\mathbf{X}}(i, j-1, m-1)) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(j)} \end{aligned} \quad (3.9)$$

for $i = 1, \dots, |\mathbf{A}|$, $j = 1, \dots, |\mathbf{B}|$ and $m = 1, \dots, |\mathbf{A}| + |\mathbf{B}|$.

At position $(0, 0, 0)$ the matrices are initialized with

$$\mathbf{Z}^{\mathbf{M}}(0, 0, 0) = 1, \quad \mathbf{Z}^{\mathbf{X}}(0, 0, 0) = 0, \quad \mathbf{Z}^{\mathbf{Y}}(0, 0, 0) = 0. \quad (3.10)$$

The total partition function \mathbf{Z} at position (i, j, m) comprehends the total probability over all alignments ending in either of the three states M, X, Y at that position. Hence, \mathbf{Z} is obtained by summing the three partitions functions as follow

$$\mathbf{Z}(i, j, m) = \mathbf{Z}^{\mathbf{M}}(i, j, m) + \mathbf{Z}^{\mathbf{X}}(i, j, m) + \mathbf{Z}^{\mathbf{Y}}(i, j, m). \quad (3.11)$$

At $m = 0$, only the position $(0, 0, 0)$ needs to be computed. After the initialization, shown in formulas 3.10, the algorithm iterates through $m = 1, \dots, |\mathbf{A}| + |\mathbf{B}|$ and computes the likelihood values as described below.

At layer $m = 1$, the algorithm calculates

$$\begin{aligned} \mathbf{Z}^{\mathbf{M}}(1, 1, 1) &= (\mathbf{Z}^{\mathbf{M}}(0, 0, 0) + \mathbf{Z}^{\mathbf{X}}(0, 0, 0) + \mathbf{Z}^{\mathbf{Y}}(0, 0, 0)) \cdot \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(1)} = \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(1)} \\ \mathbf{Z}^{\mathbf{X}}(1, 0, 1) &= (\mathbf{Z}^{\mathbf{M}}(0, 0, 0) + \mathbf{Z}^{\mathbf{X}}(0, 0, 0) + \mathbf{Z}^{\mathbf{Y}}(0, 0, 0)) \cdot \mathbb{P}_{\mathbf{A}(1), \epsilon} = \mathbb{P}_{\mathbf{A}(1), \epsilon} \\ \mathbf{Z}^{\mathbf{Y}}(0, 1, 1) &= (\mathbf{Z}^{\mathbf{M}}(0, 0, 0) + \mathbf{Z}^{\mathbf{X}}(0, 0, 0) + \mathbf{Z}^{\mathbf{Y}}(0, 0, 0)) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(1)} = \mathbb{P}_{\epsilon, \mathbf{B}(1)}. \end{aligned}$$

At $m = 2$, the partition function $\mathbf{Z}^{\mathbf{M}}$ for all the alignments ending in state *match* results in

$$\begin{aligned} \mathbf{Z}^{\mathbf{M}}(2, 2, 2) &= (\mathbf{Z}^{\mathbf{M}}(1, 1, 1) + \mathbf{Z}^{\mathbf{X}}(1, 1, 1) + \mathbf{Z}^{\mathbf{Y}}(1, 1, 1)) \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(2)} \\ &= \mathbf{Z}^{\mathbf{M}}(1, 1, 1) \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(2)} = \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(2)} \\ \mathbf{Z}^{\mathbf{M}}(2, 1, 2) &= (\mathbf{Z}^{\mathbf{M}}(1, 0, 1) + \mathbf{Z}^{\mathbf{X}}(1, 0, 1) + \mathbf{Z}^{\mathbf{Y}}(1, 0, 1)) \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(1)} \\ &= \mathbf{Z}^{\mathbf{X}}(1, 0, 1) \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(1)} = \mathbb{P}_{\mathbf{A}(1), \epsilon} \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(1)} \\ \mathbf{Z}^{\mathbf{M}}(1, 2, 2) &= (\mathbf{Z}^{\mathbf{M}}(0, 1, 1) + \mathbf{Z}^{\mathbf{X}}(0, 1, 1) + \mathbf{Z}^{\mathbf{Y}}(0, 1, 1)) \cdot \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(2)} \\ &= \mathbf{Z}^{\mathbf{Y}}(0, 1, 1) \cdot \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(2)} = \mathbb{P}_{\epsilon, \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(2)} \end{aligned}$$

the partition function \mathbf{Z}^X for all the alignments ending in state $gapX$ gets

$$\begin{aligned}\mathbf{Z}^X(2, 1, 2) &= (\mathbf{Z}^M(1, 1, 1) + \mathbf{Z}^X(1, 1, 1) + \mathbf{Z}^Y(1, 1, 1)) \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} \\ &= \mathbf{Z}^M(1, 1, 1) \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} = \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} \\ \mathbf{Z}^X(2, 0, 2) &= (\mathbf{Z}^M(1, 0, 1) + \mathbf{Z}^X(1, 0, 1) + \mathbf{Z}^Y(1, 0, 1)) \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} \\ &= \mathbf{Z}^X(1, 0, 1) \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} = \mathbb{P}_{\mathbf{A}(1), \epsilon} \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} \\ \mathbf{Z}^X(1, 1, 2) &= (\mathbf{Z}^M(0, 1, 1) + \mathbf{Z}^X(0, 1, 1) + \mathbf{Z}^Y(0, 1, 1)) \cdot \mathbb{P}_{\mathbf{A}(1), \epsilon} \\ &= \mathbf{Z}^Y(0, 1, 1) \cdot \mathbb{P}_{\mathbf{A}(1), \epsilon} = \mathbb{P}_{\epsilon, \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(1), \epsilon}\end{aligned}$$

and finally \mathbf{Z}^Y for all the alignments ending in state $gapY$ is

$$\begin{aligned}\mathbf{Z}^Y(1, 2, 2) &= (\mathbf{Z}^M(1, 1, 1) + \mathbf{Z}^X(1, 1, 1) + \mathbf{Z}^Y(1, 1, 1)) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} \\ &= \mathbf{Z}^M(1, 1, 1) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} = \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(1)} \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} \\ \mathbf{Z}^Y(1, 1, 2) &= (\mathbf{Z}^M(1, 0, 1) + \mathbf{Z}^X(1, 0, 1) + \mathbf{Z}^Y(1, 0, 1)) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(1)} \\ &= \mathbf{Z}^X(1, 0, 1) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(1)} = \mathbb{P}_{\mathbf{A}(1), \epsilon} \cdot \mathbb{P}_{\epsilon, \mathbf{B}(1)} \\ \mathbf{Z}^Y(0, 2, 2) &= (\mathbf{Z}^M(0, 1, 1) + \mathbf{Z}^X(0, 1, 1) + \mathbf{Z}^Y(0, 1, 1)) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} \\ &= \mathbf{Z}^Y(0, 1, 1) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} = \mathbb{P}_{\epsilon, \mathbf{B}(1)} \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)}.\end{aligned}$$

All the successive layers are filled in with likelihood values in the same manner as explained above.

Example

To make an example, let's have a look at the calculations implied at coordinates $(3, 3, 4)$ in the three matrices $\mathbf{Z}^{M, X, Y}$.

Likelihood computation at $\mathbf{Z}^M(3, 3, 4)$

At $\mathbf{Z}^M(3, 3, 4)$ the SB algorithm sums up the likelihood values of all the alignments of length $|m| = 4$ with $\mathbf{A}_3 \diamond \mathbf{B}_3$, which corresponds to

$$\begin{aligned}\mathbf{Z}^M(3, 3, 4) &= \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(3), \mathbf{B}(3)} + \\ &\quad + \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(1)} \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} \cdot \mathbb{P}_{\mathbf{A}(3), \mathbf{B}(3)} + \\ &\quad + \mathbb{P}_{\mathbf{A}(1), \epsilon} \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(1)} \cdot \mathbb{P}_{\epsilon, \mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(3), \mathbf{B}(3)} + \\ &\quad + \mathbb{P}_{\mathbf{A}(1), \epsilon} \cdot \mathbb{P}_{\epsilon, \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(3), \mathbf{B}(3)} + \\ &\quad + \mathbb{P}_{\epsilon, \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(1), \mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(2), \epsilon} \cdot \mathbb{P}_{\mathbf{A}(3), \mathbf{B}(3)} + \\ &\quad + \mathbb{P}_{\epsilon, \mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(1), \epsilon} \cdot \mathbb{P}_{\mathbf{A}(2), \mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(3), \mathbf{B}(3)}.\end{aligned}$$

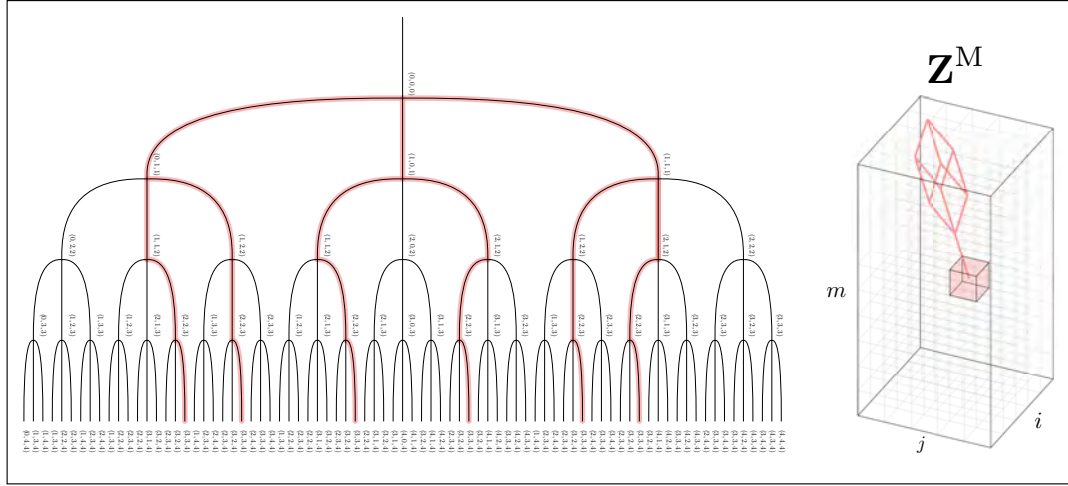


Figure 3.4: Possible homologous paths at $\mathbf{Z}^M(3,3,4)$. Left: The 6 possible paths are illustrated by means of a decision tree where at each coordinate (node) there are three directions where to move next: for match by adding the vector $(1,1,1)$ at the actual position; for gapX by adding the vector $(1,0,1)$ and for gapY by adding the vector $(0,1,1)$ to the current position coordinates. Right: The 6 possible paths are portrayed inside the dynamic programming matrix \mathbf{Z}^M .

Likelihood computation at $\mathbf{Z}^X(3,3,4)$

At $\mathbf{Z}^X(3,3,4)$ the SB algorithm sums up the likelihood values of all the alignments of length $|m| = 4$ with \mathbf{A}_3 aligned with a column full of gaps on the right side, hence

$$\begin{aligned} \mathbf{Z}^X(3,3,4) = & \mathbb{P}_{\mathbf{A}(1),\mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(2),\mathbf{B}(2)} \cdot \mathbb{P}_{\epsilon,\mathbf{B}(3)} \cdot \mathbb{P}_{\mathbf{A}(3),\epsilon} + \\ & + \mathbb{P}_{\mathbf{A}(1),\mathbf{B}(1)} \cdot \mathbb{P}_{\epsilon,\mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(2),\mathbf{B}(3)} \cdot \mathbb{P}_{\mathbf{A}(3),\epsilon} + \\ & + \mathbb{P}_{\epsilon,\mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(1),\mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(2),\mathbf{B}(3)} \cdot \mathbb{P}_{\mathbf{A}(3),\epsilon} \end{aligned}$$

Figure 3.5 depicts the 3 possible paths.

Likelihood computation at $\mathbf{Z}^Y(3,3,4)$

At $\mathbf{Z}^Y(3,3,4)$ the SB algorithm sums up the likelihood values of all the alignments of length $|m| = 4$ with \mathbf{B}_3 aligned with a column full of gaps on the left

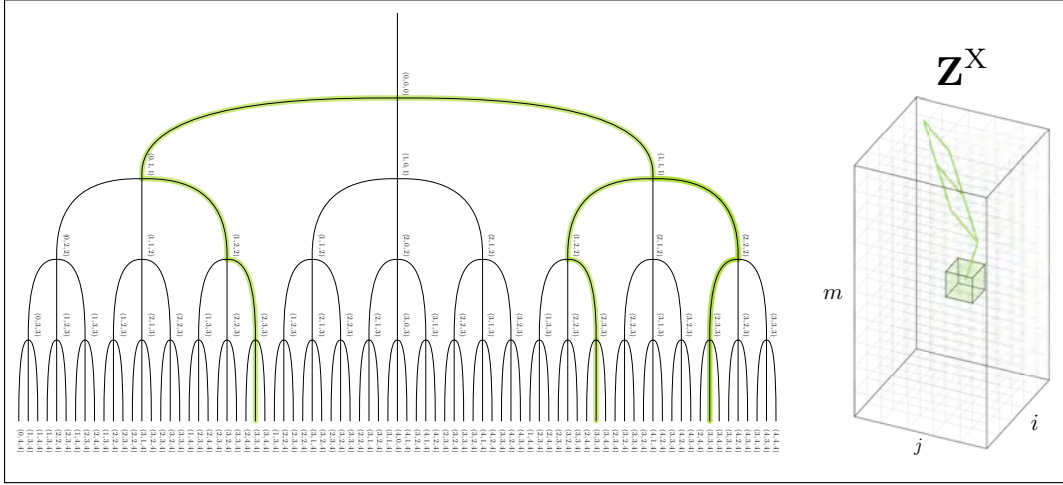


Figure 3.5: Possible homologous paths at $\mathbf{Z}^X(3,3,4)$. Left: The 3 possible paths are illustrated by means of a decision tree where at each coordinate (node) there are three directions where to move next: for match by adding the vector $(1,1,1)$ at the actual position; for gapX by adding the vector $(1,0,1)$ and for gapY by adding the vector $(0,1,1)$ to the current position coordinates. Right: The 3 possible paths are portrayed inside the dynamic programming matrix \mathbf{Z}^X .

side, hence

$$\begin{aligned} \mathbf{Z}^Y(3,3,4) = & \mathbb{P}_{\mathbf{A}(1),\mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(2),\mathbf{B}(2)} \cdot \mathbb{P}_{\mathbf{A}(3),\epsilon} \cdot \mathbb{P}_{\epsilon,\mathbf{B}(3)} + \\ & + \mathbb{P}_{\mathbf{A}(1),\mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(2),\epsilon} \cdot \mathbb{P}_{\mathbf{A}(3),\mathbf{B}(2)} \cdot \mathbb{P}_{\epsilon,\mathbf{B}(3)} + \\ & + \mathbb{P}_{\mathbf{A}(1),\epsilon} \cdot \mathbb{P}_{\mathbf{A}(2),\mathbf{B}(1)} \cdot \mathbb{P}_{\mathbf{A}(3),\mathbf{B}(2)} \cdot \mathbb{P}_{\epsilon,\mathbf{B}(3)} \end{aligned}$$

The 3 possible paths are illustrated in Figure 3.6.

3.5 Backward recursion

The backtracking recursion scheme has been modified to return a stochastic path, and hence a stochastic alignment, according to the total partition function computed in Section 3.4 : [Forward recursion](#). In the forward phase the SB algorithm builds in a very efficient way three distributions of likelihood. These distributions are used, in the backward phase, to generate stochastic paths where high probable paths are more likely to be generated than the others. The three partition function $\mathbf{Z}^{M,X,Y}$ are weighting, at each step of the backtracking phase, the probability to move on the three possible directions

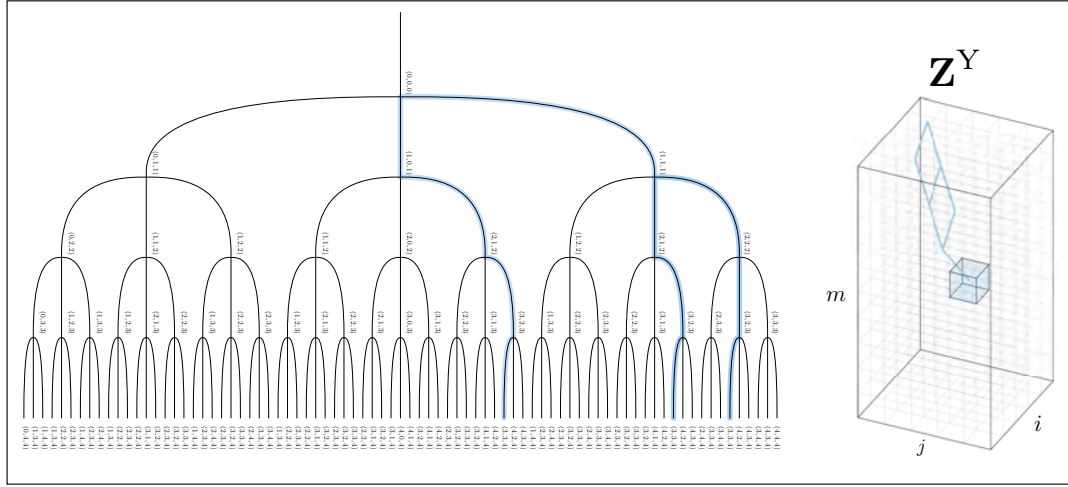


Figure 3.6: Possible homologous paths at $\mathbf{Z}^Y(3,3,4)$. Left: The 3 possible paths are illustrated by means of a decision tree where at each coordinate (node) there are three directions where to move next: for match by adding the vector $(1,1,1)$ at the actual position; for gapX by adding the vector $(1,0,1)$ and for gapY by adding the vector $(0,1,1)$ to the current position coordinates. Right: The 3 possible paths are portrayed inside the dynamic programming matrix \mathbf{Z}^Y .

corresponding to *match*, *gapX* and *gapY*. According to these probabilities a random choice is made.

An entry in $\mathbf{Z}^{M,X,Y}(|\mathbf{A}|, |\mathbf{B}|, m)$ for $m = \max(|\mathbf{A}|, |\mathbf{B}|), \dots, (|\mathbf{A}| + |\mathbf{B}|)$ contains the sum of all possible alignments of length $|m|$ between the sequences (or sub-alignments) \mathbf{A} and \mathbf{B} ending respectively in state *match*, *gapX* and *gapY*. In this computation we have expressly left out the marginal likelihood for all the non-observable empty columns. This likelihood component is added only at this stage in the following manner

$$\tilde{\mathbf{Z}}^M(|\mathbf{A}|, |\mathbf{B}|, m) = \mathbf{Z}^M(|\mathbf{A}|, |\mathbf{B}|, m) \cdot \varphi(p_{c_0}(v), m)$$

for $m = \max(|\mathbf{A}|, |\mathbf{B}|), \dots, (|\mathbf{A}| + |\mathbf{B}|)$. The same applies for $\tilde{\mathbf{Z}}^X(|\mathbf{A}|, |\mathbf{B}|, m)$ and $\tilde{\mathbf{Z}}^Y(|\mathbf{A}|, |\mathbf{B}|, m)$. At this point, the SB algorithm has to select firstly, from which matrix to start the tracebacking and secondly, the layer m where to start.

I) Selection of the starting state. To decide in which *state* (i.e. from which matrix) to start, the algorithm computes the total probability from the

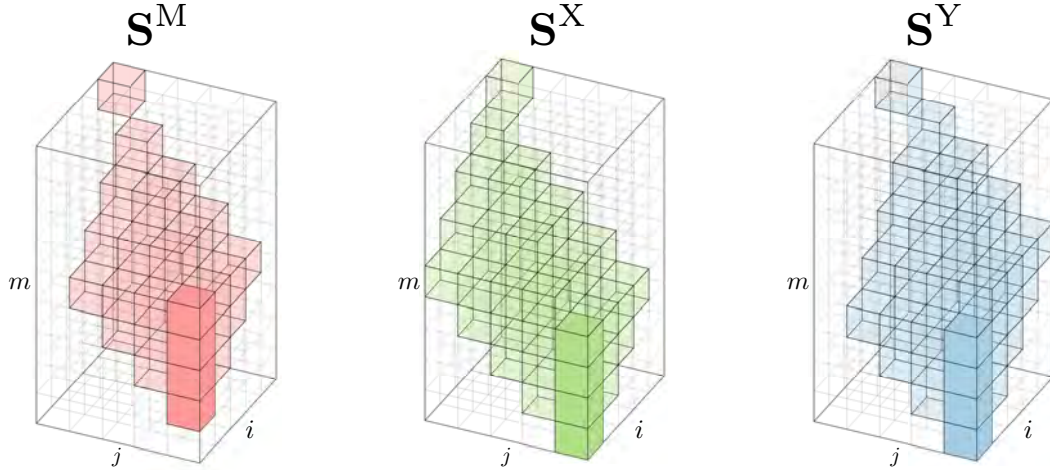


Figure 3.7: Darker cells represents entries containing likelihood of candidate solutions. At each layer corresponds an MSA with different length.

three distributions by summing the values contained in the “last column”². These columns, containing the likelihood of candidate solutions, are depicted as darker cells in Figure 3.7 and . For \mathbf{Z}^M the summation gets

$$z_M = \sum_{m=\max(|\mathbf{A}|, |\mathbf{B}|)}^{|\mathbf{A}|+|\mathbf{B}|} \tilde{\mathbf{Z}}^M(|\mathbf{A}|, |\mathbf{B}|, m).$$

The same applies for $\tilde{\mathbf{Z}}^X$ and $\tilde{\mathbf{Z}}^Y$, thus getting z_X and z_Y , respectively. In the columns highlighted in Figure 3.7 one can find the likelihood values of alignments ending in the corresponding states ‘M’, ‘X’, ‘Y’ and having a total length indicated by the depth m . To select in which *state* to start, a random number uniformly distributed $r = \mathcal{U}(0, 1)$ is drawn. If $r < z_M / (z_M + z_X + z_Y)$ than the first state is *match*, if $r < (z_M + z_X) / (z_M + z_X + z_Y)$ then the first state is *gapX* otherwise the algorithm starts in *gapY* state.

II) Selection of the starting level. A similar procedure is applied to choose at which layer m the tracebacking will start. Let us suppose that the algorithm starts in state *gapX*, then the starting position is selected by drawing a second random number uniformly distributed $r = \mathcal{U}(0, 1)$ and the algorithm iterates through the column $z_{m_0} = \tilde{\mathbf{Z}}^X(|\mathbf{A}|, |\mathbf{B}|, m_0) / z_X$ for $m_0 =$

²that is, the matrix column at the positions $\mathbf{Z}(|\mathbf{A}|, |\mathbf{B}|, m)^{M, X, Y}$ with $m = \max(|\mathbf{A}|, |\mathbf{B}|), \dots, (|\mathbf{A}| + |\mathbf{B}|)$.

$\max(|\mathbf{A}|, |\mathbf{B}|), \dots, |\mathbf{A}| + |\mathbf{B}|$ until the statement $z_{m_0} > r$ is no more true. The selected m_0 corresponds to the layer where the traceback starts and dictates the final MSA length.

III) Selection of stochastic path. The stochastic backtracking algorithm begins in the matrix selected in (I), at the layer chosen in (II), and then run through the three matrices until it reaches the upper-left corner at coordinates $(0, 0, 0)$. The algorithm selects the previous state based on the total likelihood of the mentioned paths at t

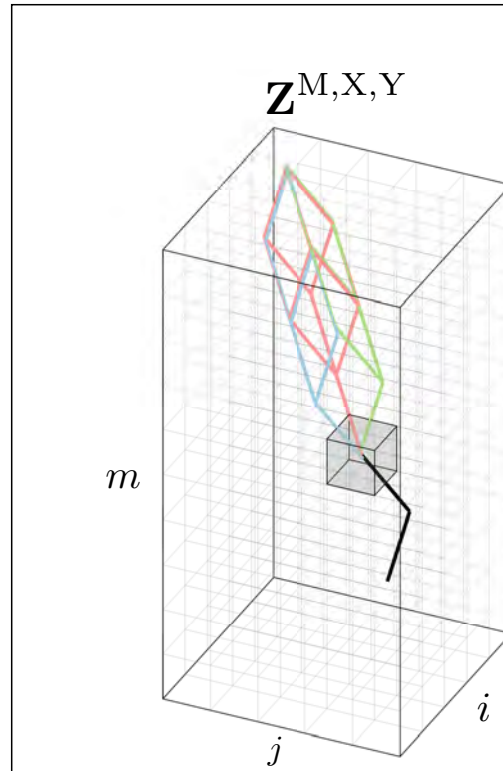


Figure 3.8: Backtracking phase. Graphical representation of the procedure of selecting the preceding state based on the partition functions $\mathbf{Z}^{M,X,Y}$. The set of paths linked to the current state at coordinates (i, j, m) (green box) are represented with different colors (in pink $\mathbf{Z}^M(i-1, j-1, m-1)$, in green $\mathbf{Z}^X(i-1, j, m-1)$ and in blue $\mathbf{Z}^Y(i, j-1, m-1)$). The algorithm select which is the previous state based on the total likelihood of the mentioned paths

The probability of being in state *match* at coordinates (i, j, m) coming from ‘M’, ‘X’ and ‘Y’, denoted $\mathbb{P}_m^M(i-1, j-1, m-1)$, $\mathbb{P}_m^X(i-1, j-1, m-1)$,

$\mathbb{P}_m^Y(i-1, j-1, m-1)$, respectively, is given by

$$\mathbb{P}_m^M(i-1, j-1, m-1) = \frac{\mathbf{Z}^M(i-1, j-1, m-1) \cdot \mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)}}}{\mathbf{Z}(i, j, m)} \quad (3.12)$$

$$\mathbb{P}_m^X(i-1, j-1, m-1) = \frac{\mathbf{Z}^X(i-1, j-1, m-1) \cdot \mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)}}}{\mathbf{Z}(i, j, m)} \quad (3.13)$$

$$\mathbb{P}_m^Y(i-1, j-1, m-1) = \frac{\mathbf{Z}^Y(i-1, j-1, m-1) \cdot \mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)}}}{\mathbf{Z}(i, j, m)} \quad (3.14)$$

with

$$\begin{aligned} \mathbf{Z}(i, j, m) = & (\mathbf{Z}^M(i-1, j-1, m-1) + \mathbf{Z}^X(i-1, j-1, m-1) + \\ & + \mathbf{Z}^Y(i-1, j-1, m-1)) \cdot \mathbb{P}_{\mathbf{A}(i), \mathbf{B}(j)}. \end{aligned}$$

The probability of being in state *gapX* at coordinates (i, j, m) coming from ‘M’, ‘X’ and ‘Y’, denoted $\mathbb{P}_x^M(i-1, j, m-1)$, $\mathbb{P}_x^X(i-1, j, m-1)$, $\mathbb{P}_x^Y(i-1, j, m-1)$, respectively, is given by

$$\mathbb{P}_x^M(i-1, j, m-1) = \frac{\mathbf{Z}^M(i-1, j, m-1) \cdot \mathbb{P}_{\mathbf{A}(i), \epsilon}}{\mathbf{Z}(i, j, m)} \quad (3.15)$$

$$\mathbb{P}_x^X(i-1, j, m-1) = \frac{\mathbf{Z}^X(i-1, j, m-1) \cdot \mathbb{P}_{\mathbf{A}(i), \epsilon}}{\mathbf{Z}(i, j, m)} \quad (3.16)$$

$$\mathbb{P}_x^Y(i-1, j, m-1) = \frac{\mathbf{Z}^Y(i-1, j, m-1) \cdot \mathbb{P}_{\mathbf{A}(i), \epsilon}}{\mathbf{Z}(i, j, m)} \quad (3.17)$$

with

$$\begin{aligned} \mathbf{Z}(i, j, m) = & (\mathbf{Z}^M(i-1, j, m-1) + \mathbf{Z}^X(i-1, j, m-1) + \\ & + \mathbf{Z}^Y(i-1, j, m-1)) \cdot \mathbb{P}_{\mathbf{A}(i), \epsilon}. \end{aligned}$$

The probability of being in state *gapY* at coordinates (i, j, m) coming from ‘M’, ‘X’ and ‘Y’, denoted $\mathbb{P}_y^M(i, j-1, m-1)$, $\mathbb{P}_y^X(i, j-1, m-1)$, $\mathbb{P}_y^Y(i, j-1, m-1)$, respectively, is given by

$$\mathbb{P}_y^M(i, j-1, m-1) = \frac{\mathbf{Z}^M(i, j-1, m-1) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(j)}}{\mathbf{Z}(i, j, m)} \quad (3.18)$$

$$\mathbb{P}_y^X(i, j-1, m-1) = \frac{\mathbf{Z}^X(i, j-1, m-1) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(j)}}{\mathbf{Z}(i, j, m)} \quad (3.19)$$

$$\mathbb{P}_y^Y(i, j-1, m-1) = \frac{\mathbf{Z}^Y(i, j-1, m-1) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(j)}}{\mathbf{Z}(i, j, m)} \quad (3.20)$$

with

$$\mathbf{Z}(i, j, m) = (\mathbf{Z}^M(i, j - 1, m - 1) + \mathbf{Z}^X(i, j - 1, m - 1) + \mathbf{Z}^Y(i, j - 1, m - 1)) \cdot \mathbb{P}_{\epsilon, \mathbf{B}(j)}.$$

Therefore, if the *current state* is match, the algorithm selects the *preceding state* using equations 3.12, if the *current state* is gapX one uses equations 3.15 otherwise equations 3.18. The *preceding state* is selected by drawing a random number uniformly distributed $r = \mathcal{U}(0, 1)$. Let us suppose that we are at coordinates (i, j, m) and the current state is ‘M’ than if

$$r < \frac{\mathbb{P}_m^M(i - 1, j - 1, m - 1)}{\mathbb{P}_m^{\text{MXY}}}$$

where $\mathbb{P}_m^{\text{MXY}} = (\mathbb{P}_m^M(i - 1, j - 1, m - 1) + \mathbb{P}_m^X(i - 1, j - 1, m - 1) + \mathbb{P}_m^Y(i - 1, j - 1, m - 1))$, then, preceding state is set to ‘M’, then $i \leftarrow i - 1$, $j \leftarrow j - 1$ and $m \leftarrow m - 1$;

a preceding state is set to ‘X’ if

$$r < \frac{(\mathbb{P}_m^M(i - 1, j - 1, m - 1) + \mathbb{P}_m^X(i - 1, j - 1, m - 1))}{\mathbb{P}_m^{\text{MXY}}}$$

then $i \leftarrow i - 1$ and $m \leftarrow m - 1$,

otherwise the preceding state is set to ‘Y’ and $j \leftarrow j - 1$ and $m \leftarrow m - 1$.

Figure 3.8 depicts graphically the procedure of selecting the preceding state based on the partition functions $\mathbf{Z}^{M,X,Y}$. The set of paths linked to the current state are represented with different colors. The algorithm select which is the previous state based on the total likelihood of the mentioned paths.

3.6 The temperature parameter

As we have mentioned, in Section 3.1 : Introduction, the parameter ‘Temperature’ tunes, to some degree, the deviation from the optimal alignment. In our context, by setting $T \rightarrow \infty$, each alignment becomes equiprobable, the solution is therefore purely random. On the contrary, by setting $T = 0$ the stochastic backtracking returns the optimal alignment. In the range $0 < T < \infty$ the parameter temperature controls the deviation from the optimal alignment allowing, gradually, the generation of sub-optimal alignments. Such behavior

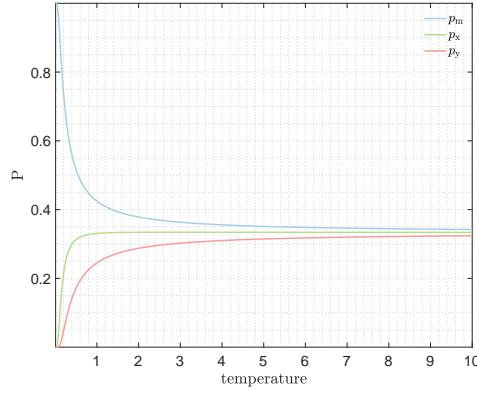


Figure 3.9: Probability distortion as function of the temperature. We have set $p_m = 0.80$, $p_x = 0.15$ and $p_y = 0.05$ and we let the temperature varies from 0.01 to 10. At low temperature the algorithm assigns a probability close to 1 at p_m and close to 0 at either p_x and p_y . At $T = 10$ the three states have almost the same chance to be selected with probability approaching $1/3$.

is obtained in this manner. Let's assume the algorithm is in state 'M', and let denote $p_m = \mathbb{P}_m^M(i-1, j-1, m-1)$, $p_x = \mathbb{P}_m^X(i-1, j-1, m-1)$ and $p_y = \mathbb{P}_m^Y(i-1, j-1, m-1)$, then the distorted probabilities become

$$\tilde{p}_m = \exp\left(\frac{p_m}{T}\right) \quad \tilde{p}_x = \exp\left(\frac{p_x}{T}\right) \quad \tilde{p}_y = \exp\left(\frac{p_y}{T}\right). \quad (3.21)$$

To guarantee that one of the three possible states is always selected we normalize them so that their sum gets always 1, that is

$$\hat{m} = \frac{m}{\tilde{p}_m + \tilde{p}_x + \tilde{p}_y} \quad \hat{x} = \frac{x}{\tilde{p}_m + \tilde{p}_x + \tilde{p}_y} \quad \hat{y} = \frac{y}{\tilde{p}_m + \tilde{p}_x + \tilde{p}_y}. \quad (3.22)$$

The probabilities in equations 3.12 - 3.20 are then re-weighted with the temperature T using equation 3.21 and 3.22.

Figure 3.9 shows the distortion of the probabilities as function of the temperature T . We have set $p_m = 0.60$, $p_x = 0.35$ and $p_y = 0.05$ and we let the temperature varies from 0.01 to 10. One can note that at low temperature the algorithm assigns a probability close to 1 at p_m and close to 0 at either p_x and p_y . Therefore at low temperature the SB becomes greedy reducing the chances of low probable alignments. As the temperature raises as more probable becomes that also one of the other two states is selected. Indeed, at $T = 10$ the three states have almost the same chance to be selected. Hence, at high temperature the algorithm is less greedy allowing the generation of sub-optimal alignments.

Temperature $T = 0.10$. In Figure 3.10 we are showing, for a fixed temperature, how the starting probabilities (on the left side) are re-weighted (on the right side) according to equations 3.21 and 3.22. For this plots we have set $T = 0.1$, p_m , p_x and p_y are linearly increased or decreased as depicted in the left plot. From the right plot it can be noted that at low temperature the algorithm promotes the highest value among the three and penalize the other two. Therefore at low temperature the SB behaves as a greedy algorithm.

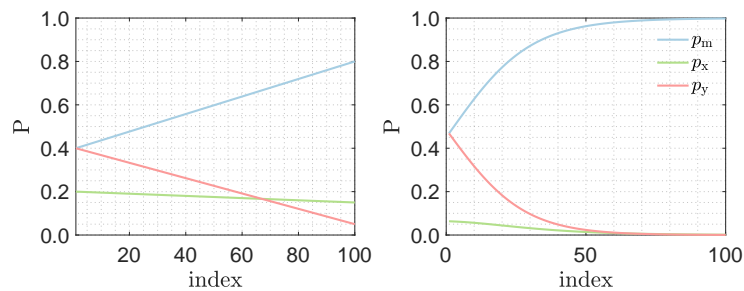


Figure 3.10: $p_m = 0.40, \dots, 0.80$, $p_x = 0.20, \dots, 0.15$ and $p_y = 1 - p_m - p_x$.

Temperature $T = 0.30$. Figure 3.11 depicts, at fixed temperature $T = 0.3$, how the starting probabilities (on the left side) are re-weighted (on the right side) according to equations 3.21 and 3.22. The *match*, *gapX* and *gapY* probabilities, p_m , p_x and p_y , respectively, are linearly increased or decreased as depicted in the left plot. From the right plot it can be noted that by increasing the temperature the algorithm becomes less greedy. In this example the probabilities are almost kept unchanged.

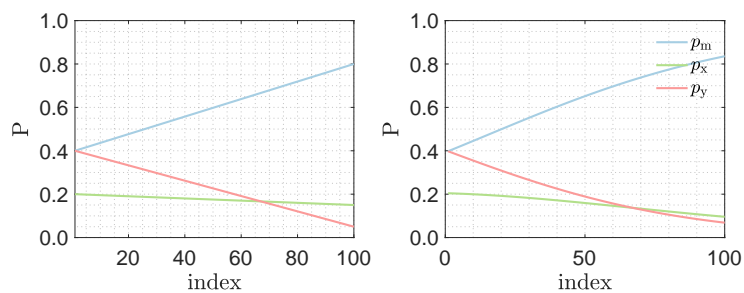


Figure 3.11: $p_m = 0.40, \dots, 0.80$, $p_x = 0.20, \dots, 0.15$ and $p_y = 1 - p_m - p_x$.

Temperature $T = 10.0$. In Figure 3.12 we are representing, at temperature $T = 10$, how the starting probabilities (on the left side) are re-weighted (on

the right side) according to equations 3.21 and 3.22. The probabilities p_m , p_x and p_y are linearly increased or decreased as depicted in the left plot. From the right plot it can be noted that at high temperature the algorithm squeezes all the probabilities around the value $1/3$. This means that each alignment has the same probability to be generated therefore the result is purely random.

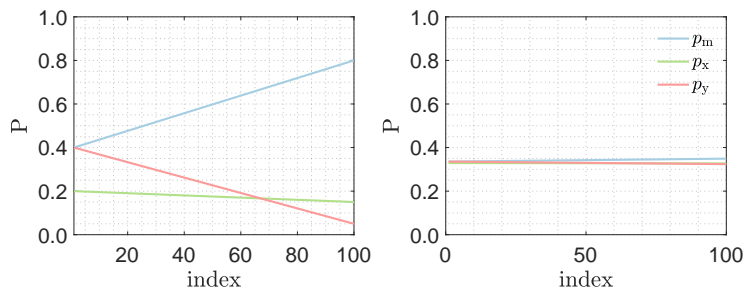


Figure 3.12: $p_m = 0.40, \dots, 0.80$, $p_x = 0.20, \dots, 0.15$ and $p_y = 1 - p_m - p_x$.

Applying the temperature distortion we have run the SB algorithm at three different temperatures. We have recorded the taken path through the matrices. Figure 3.13 shows the paths undertaken at $T = 0.01$, $T = 0.1$ and $T = 1$ by the SB algorithm, aligning a synthesized dataset. The DP matrix in 3.13 represents an hypothetical traceback sparse matrix (note that in SB there isn't any traceback matrix differently from the classic DP under PIP). It is interesting to see that at low temperature the traceback paths are less scattered than at high temperature. In fact, as the temperature increases as more the paths are dispersed in the traceback matrix. The alignments are always generated according to their likelihood but their likelihood is distorted so that very different likelihood values are transformed to look very similar.

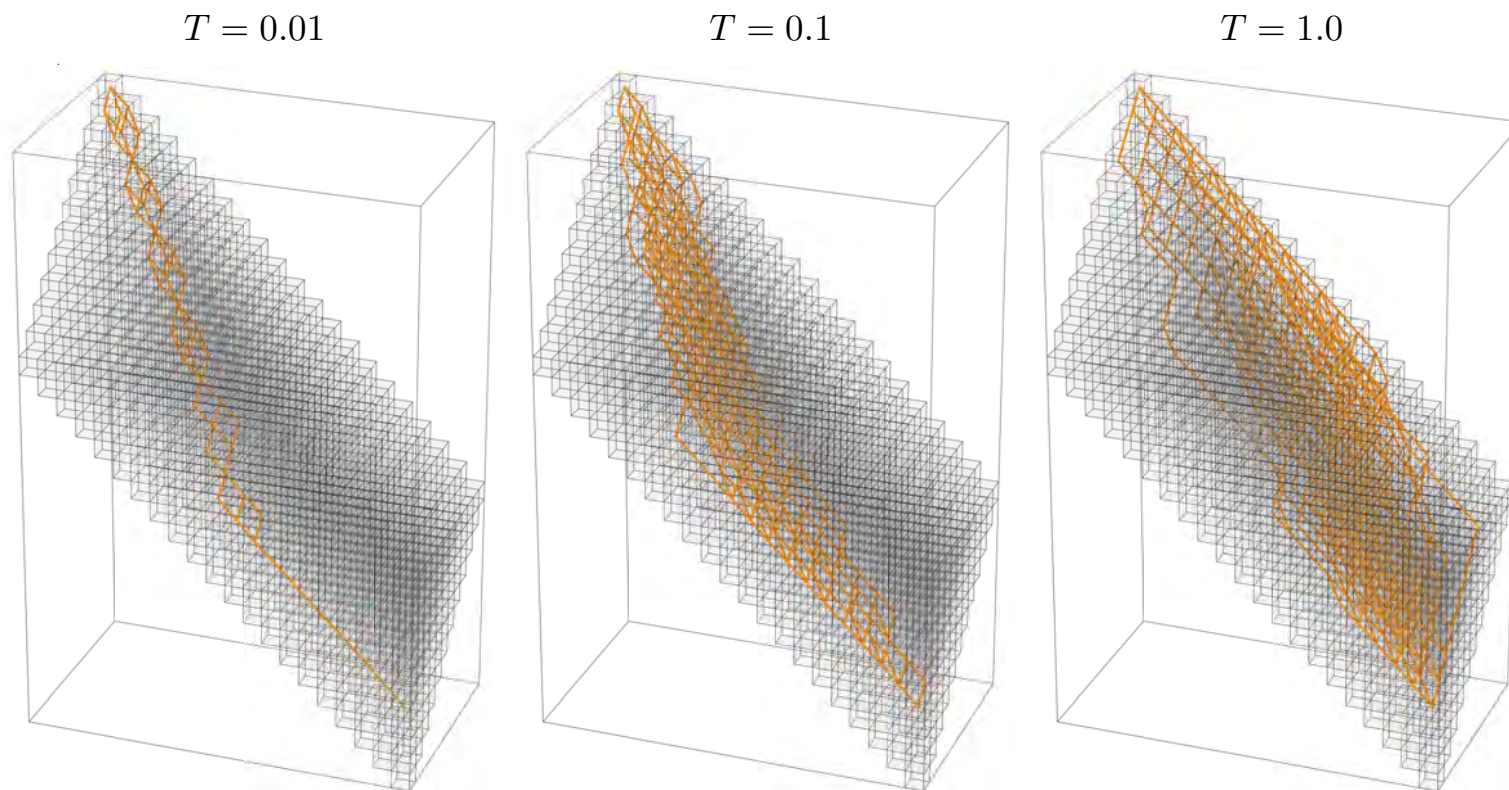


Figure 3.13: Stochastic backtracking paths at three different temperatures, $T = 0.01$, $T = 0.1$ and $T = 1.0$. It can be noticed that as the temperature growth, the paths diverge from the optimal path. In principle, at $T \rightarrow \infty$ any paths can be generated disregarding their likelihood.

4

Marginal Likelihood with Rate Variation Across Sites

“Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.”

– Marie Curie

4.1 Introduction

Evolutionary studies have shown that sites are substituted at different rates. Hence, it is meaningful to account for the Rate Variation Across Sites (ASRV) during the sequence alignment and the phylogeny inference [43, 49, 50, 137, 150, 166, 168]. To model the substitution with ASRV different continuous distributions have been suggested. The *mean-one discrete gamma distribution* is definitely the most-frequently used. In practice, each site evolves at any of the rates among the discrete rate classes. Since the specific rate / class of each site is unknown, one marginalizes over all the classes.

The discrete gamma distribution is applied in the following manner. Let us suppose that the substitution rate r associated to a character, inside our input sequence, is an i.i.d. random number according to a given distribution $g(r)$. Let also denote with k the site total number of substitutions during an evolutionary interval of time t . Assuming that substitutions are following a Poisson process (Markov process of substitutions) at a given rate r , implies

that k follows the distribution ([167])

$$f(k) = \int_0^{\infty} \frac{(r \cdot t)^k e^{-rt}}{k!} g(r) dr. \quad (4.1)$$

From equation 4.1 we can compute the expected value of k , $\mathbb{E}(k)$, as the first moment of the distribution $f(k)$ in equation 4.1, that is

$$\begin{aligned} \mathbb{E}(k) &= \sum_{k=0}^{\infty} (k \cdot f(k)) \\ &= \sum_{k=0}^{\infty} \left(k \cdot \int_0^{\infty} \frac{(r \cdot t)^k e^{-rt}}{k!} g(r) dr \right) \\ &= \int_0^{\infty} \sum_{k=0}^{\infty} \left(k \cdot \frac{(r \cdot t)^k e^{-rt}}{k!} \right) g(r) dr \\ &= \int_0^{\infty} r \cdot t \cdot g(r) dr \\ &= \mathbb{E}(r) \cdot t \end{aligned} \quad (4.2)$$

and the variance, $\text{Var}(k)$, as the second moment of $f(k)$ ([167])

$$\begin{aligned} \text{Var}(k) &= \sum_{k=0}^{\infty} (k^2 \cdot f(k)) - (\mathbb{E}(k))^2 \\ &= \sum_{k=0}^{\infty} \left(k^2 \cdot \int_0^{\infty} \frac{(r \cdot t)^k e^{-rt}}{k!} g(r) dr \right) - (\mathbb{E}(r) \cdot t)^2 \\ &= \int_0^{\infty} \sum_{k=0}^{\infty} \left(k^2 \cdot \frac{(r \cdot t)^k e^{-rt}}{k!} \right) g(r) dr - (\bar{r} \cdot t)^2 \\ &= \int_0^{\infty} (r^2 \cdot t^2 + r \cdot t) \cdot g(r) dr - (\bar{r} \cdot t)^2 \\ &= \text{Var}(r) \cdot t^2 + (\bar{r} \cdot t)^2 + \bar{r} \cdot t - (\bar{r} \cdot t)^2 \\ &= \text{Var}(r) \cdot t^2 + \bar{r} \cdot t \end{aligned} \quad (4.3)$$

where $\bar{r} = \mathbb{E}(r)$. From equation 4.2 and 4.3 we get

$$\bar{r} = \mathbb{E}(r) = \frac{\mathbb{E}(k)}{t} \quad \text{and} \quad \text{Var}(r) = \frac{\text{Var}(k) - \mathbb{E}(k)}{t^2}. \quad (4.4)$$

A classic choice for the distribution $g(r)$ is the gamma function, with probability density function (PDF) equal to

$$g(r) = \frac{(\alpha/\bar{r})^\alpha}{\Gamma(\alpha)} \cdot r^{\alpha-1} \cdot e^{-\alpha \cdot r/\bar{r}} \quad (4.5)$$

where as already mentioned above, $\bar{r} = \mathbb{E}(r)$ and α is the shape parameter of the function which is a measure of the level of among-site rate variation. Figure 4.1 displays some gamma distributions characterized by different shape parameters α . When α is relative small there will be a high among sites rate variation, when α is relative high then there will be a low degree of variation.

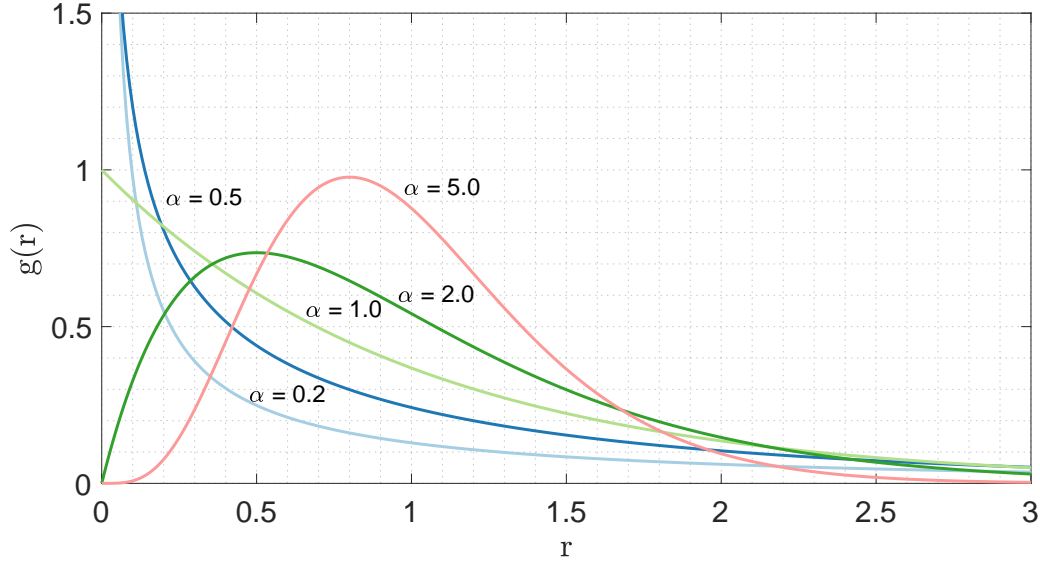


Figure 4.1: Gamma probability distribution function with different shape parameters for a fixed mean rate equal to 1.

4.2 PIP equations under ASRV

The marginal likelihood under the PIP model applying the gamma distribution to account for the among-site rate variation, denoted here with the subscript Γ , is computed with the following equation

$$p_{\Gamma, \tau}(m) = \varphi_{\Gamma}(p(c_{\emptyset}), |m|) \prod_c p_{\Gamma}(c) \quad (4.6)$$

$$= \sum_r \mathbb{P}(r) \varphi(p(c_{\emptyset}, r), |m|, r) \prod_c \sum_r \mathbb{P}(r) p(c, r) \quad (4.7)$$

where $\mathbb{P}(r)$ is the gamma density of the rate category r .

The likelihood of an MSA column c , with associated gamma rate r , becomes

$$p(c, r) = \sum_{v \in \mathcal{V}} \left[\mathbb{1}[v \in \mathcal{I}] \cdot \iota(v, r) \cdot \beta(v, r) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v, r)) \right]. \quad (4.8)$$

with

$$\mathbf{f}(v, r) = \begin{cases} \mathbb{1}[(v) = \sigma] & \text{if } v \in \mathcal{L} \\ \prod_{w \in \text{child}(v)} \exp(b(w) \cdot r \cdot \mathbf{Q}_\epsilon) \mathbf{f}(w, r) & \text{otherwise} \end{cases} \quad (4.9)$$

and \mathcal{I} , in equation 4.6, refers to the set of vertices with non zero felsenstein's weight which corresponds to the set of vertices where an insertion could have happened.

The normalizing measure of the Poisson process, applying the gamma distribution, becomes

$$\|\nu(r)\| = \lambda \cdot r \cdot \left(\|\tau\| + \frac{1}{\mu \cdot r} \right). \quad (4.10)$$

The insertion probability, in equation 4.8, on a given node $v \setminus \Omega$ is

$$\iota(v, r) = \frac{\lambda \cdot r \cdot b(v)}{\lambda \cdot r \left(\|\tau\| + \frac{1}{\mu \cdot r} \right)} = \frac{b(v)}{\|\tau\| + \frac{1}{\mu \cdot r}} \quad (4.11)$$

while at the root node gets

$$\iota(\Omega, r) = \frac{\frac{\lambda \cdot r}{\mu \cdot r}}{\lambda \cdot r \left(\|\tau\| + \frac{1}{\mu \cdot r} \right)} = \frac{\frac{1}{\mu \cdot r}}{\|\tau\| + \frac{1}{\mu \cdot r}}. \quad (4.12)$$

The survival probability under the gamma distribution can be derived by setting $\omega = \mu \cdot r$. Therefore, by computing the tail cumulative distribution $\mathbb{P}(Z > t)$ (see A.29 in Appendix A.2 for more details), we get

$$\begin{aligned} \mathbb{P}(W + U > t) &= \mathbb{P}(Z > t) \\ &= \int_t^\infty \frac{\exp(-\omega z) (\exp(\omega t) - 1)}{t} dz \\ &= \frac{1 - \exp(-\omega t)}{\omega t}. \end{aligned}$$

Substituting back ω with $\mu \cdot r$ we obtain the survival probability on a given node $v \setminus \Omega$

$$\beta(v, r) = \mathbb{P}(W + U > b(v)) = \frac{1 - \exp(-\mu \cdot r \cdot b(v))}{\mu \cdot r \cdot b(v)} \quad (4.13)$$

and at the root Ω is

$$\beta(\Omega, r) = 1. \quad (4.14)$$

The marginal likelihood for all empty columns (without ASRV) is computed by means of the following equation

$$\varphi(p(c_\emptyset), |m|) = \sum_{n=|m|}^{\infty} \mathbb{P}(|\mathbf{X}| = n) \cdot \binom{n}{|m|} \cdot p(c_\emptyset)^{n-|m|} \quad (4.15)$$

where

$$\mathbb{P}(|\mathbf{X}| = n) = \frac{\|\nu\|^n}{n!} e^{-\|\nu\|}$$

is the Poisson probability of observing an MSA of length n and $p(c_\emptyset)$ is the likelihood of a single column full of gaps. With gamma the marginal in equation 4.15 becomes

$$\begin{aligned} \varphi_\Gamma(p(c_\emptyset), |m|) &= \sum_r \mathbb{P}(r) \cdot \varphi(p(c_\emptyset, r), r, |m|) \\ &= \sum_r \mathbb{P}(r) \cdot \left[\sum_{n=|m|}^{\infty} \mathbb{P}(|\mathbf{X}| = n, r) \cdot \binom{n}{|m|} \cdot p(c_\emptyset, r)^{n-|m|} \right] \end{aligned} \quad (4.16)$$

where

$$\mathbb{P}(|\mathbf{X}| = n, r) = \frac{\|\nu(r)\|^n}{n!} e^{-\|\nu(r)\|} \quad (4.17)$$

and $p(c_\emptyset, r)$ is the likelihood of a column full of gaps under the gamma rate distribution (see equation 4.19). Using equation 4.16 and 4.17, the marginal

likelihood under PIP for all empty columns with gamma gets

$$\begin{aligned} \varphi_{\Gamma}(p(c_{\emptyset}), |m|) &= \sum_{n=|m|}^{\infty} \frac{(\sum_r \mathbb{P}(r) \|\nu(r)\|)^n}{n!} \exp\left(-\sum_r \mathbb{P}(r) \|\nu(r)\|\right) \cdot \binom{n}{|m|} \\ &\quad \cdot \left(\sum_r \mathbb{P}(r) p(c_{\emptyset}, r)\right)^{n-|m|} \\ &= \sum_{n=|m|}^{\infty} \frac{\Theta^n}{n!} \exp(-\Theta) \cdot \binom{n}{|m|} \cdot \left(\sum_r \mathbb{P}(r) p(c_{\emptyset}, r)\right)^{n-|m|} \end{aligned} \quad (4.18)$$

where Θ in 4.18 is calculated as $\Theta = \sum_r \mathbb{P}(r) \|\nu(r)\|$. And finally, the likelihood of a single empty column c_{\emptyset} with gamma rate r is computed by means of the following equation

$$\begin{aligned} p(c_{\emptyset}, r) &= \iota(v, r) \beta(v, r) (\boldsymbol{\pi}_{\epsilon} \circ \mathbf{f}(v, r)) + \sum_{v \in \mathcal{V} \setminus \Omega} \iota(v, r) \left(1 - \beta(v, r) + \right. \\ &\quad \left. + \beta(v, r) (\boldsymbol{\pi}_{\epsilon} \circ \mathbf{f}(v, r))\right). \end{aligned} \quad (4.19)$$

5

Multi-scale STFT based homologous blocks detection

“I would rather have questions that can’t be answered than answers that can’t be questioned.”

– Richard Feynman

5.1 Introduction

Today’s large genomics datasets provide a rich source of information and enable increasingly realistic models to be applied to study the underlying mechanisms shaping biological sequences. Such models however tend to be mathematically more sophisticated, and as a consequence, are computationally more demanding. In this context, one of the oldest and most fundamental problems is the alignment of related genomic sequences.

Due to the inherent computational complexity of the MSA inference, heuristic algorithms have been developed to enable this task as a part of routine sequence analyses. The progressive MSA heuristics simplify the problem by splitting it into a series of pairwise alignments guided by a tree structure representing the evolutionary relationship of the sequences. Each pairwise alignment is typically constructed by dynamic programming, which usually scales quadratically with the sequence length. Moreover, the computational complexity of a pairwise alignment with non-overlapping inversions becomes cubic with the sequence length [76].

A sound mathematical description of the evolutionary process of insertions

and deletions requires complex models such as the classical TKF91 [145] or the more recent Poisson Indel Process PIP [13]. The actual power of both models consists in describing the evolution of indels on a tree, and their computational complexity is largely determined by the evaluation of the marginal likelihood of an MSA. However, while TKF91 yields an exponential complexity in the number of taxa, PIP allows to reduce this computation to linear time.

In this thesis, we have presented our new progressive DP algorithm [88] which aligns two MSAs under the PIP model by maximum likelihood (ML). This algorithm runs through a given guide tree and computes at each internal node a column-wise likelihood for all the homology paths implied by the two sub-alignments observed at the children nodes (see Sections 1-2). The DP algorithm then returns the optimal ML pairwise alignment conditioned on the input. However, this approach requires sparse 3-dimensional DP matrices to account for the non-monotonicity of the marginal likelihood for non-observable scenarios. As a consequence, the computational complexity becomes cubic in the sequence length.

A possible approach to reduce the computational complexity in a DP framework is to predict candidate homologous segments for the purpose of filtering out non-promising regions in the DP matrix prior to the effective alignment process [62, 136]. This allows to constrain the possible candidate alignments for the overall problem thus reducing the problem complexity. One of the fastest and most accurate aligners, MAFFT [62, 62, 63], relies on a fast Fourier transform (FFT) for the segments detection. However, the usage of FFT for alignment can be traced back to *Felsenstein 1982* [35] who applied it to obtain ungapped pairwise alignments of entire sequences in $\mathcal{O}(L \log L)$, where L is the average sequence length. Nonetheless, the method was defined of “limited value” because of the impossibility of accommodating indels. In MAFFT ungapped homologous segments are used to constrain possible DP paths and, thus, exclude areas from the DP calculation. Gappy regions then link consecutive homologous regions thus yielding the final MSA. The resulting speed up increases with an increasing number of detected homologous segments. Figure 5.1 illustrates the theoretical speed-up factor that can be achieved by splitting the original 3D problem into a series of sub-problems. The speed-up factor refers to the number of entries in the original problem (total number of cells in the three 3D DP matrices) divided by the total number of cells of the sub-matrices considering the latter all of equal size.

In this chapter, we present a novel STFT-based approach to identify homologous segments and apply it in a progressive DP-PIP framework. Although

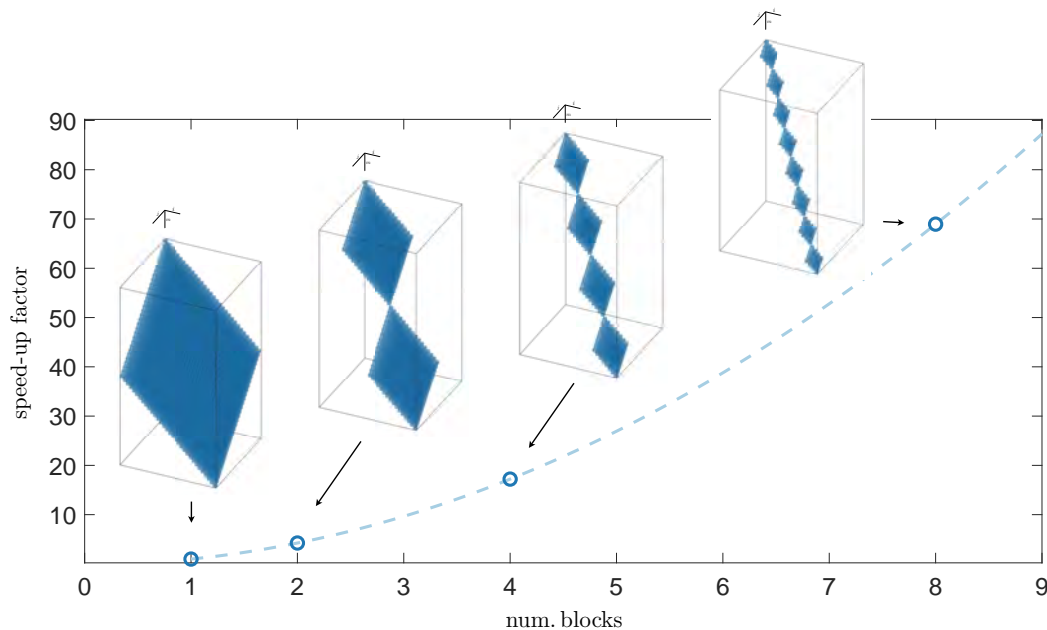


Figure 5.1: Theoretical speed-up as function of the number of blocks. A possible approach to reduce the computational complexity in a DP framework is to predict candidate homologous segments for the purpose of filtering out non-promising regions in the DP matrix prior to the effective alignment process. This allows to constrain the possible candidate alignments for the overall problem thus reducing the problem complexity.

our method is strongly inspired by MAFFT, we have introduced a number of improvements that are briefly summarized here.

- (i) The use of a multiple-resolution short-time Fourier transform (STFT) improves the detection of homologous regions especially in the presence of noise (see Section 5.5.1).
- (ii) The STFT is more effective in detecting relative short patterns than the classical FFT (see Section 5.5.2).
- (iii) The determination of the positional lag and the relative positions of the patterns inside the two signals are computed simultaneously and in a unique framework.
- (iv) We define a more sophisticated and general approach to generate logically sound paths to connect homologous blocks and resolve overlaps between them.

- (v) We compute several critical tuning parameters directly from the data at hand, instead of relying on arbitrary standard hard-coded values.
- (vi) The cardinality of the set of candidate positional lags for homologous regions is not fixed *a priori* but rather depends on the input data at hands. This set is constructed using a data-dependent and statistically robust noise threshold.

In order to understand the idea behind the use of the short-time Fourier transform to detect the homologous blocks, we start by first giving an overview of key concepts concerning the Fourier transform (Section 5.2) and the short-time Fourier transform (Section 5.3).

5.2 Fourier transform

Let us start considering a periodic function with period $T > 0$, i.e. $f(t+T) = f(t)$ and let restrict our analysis only on the space of square integrable periodic functions of period T , denoted $\mathbf{L}_T^2(\mathbb{R})$, and thus satisfying the following condition

$$\int_{t_0}^{t_0+T} |f_T(t)|^2 dt < \infty. \quad (5.1)$$

The periodic function f_T can be easily decomposed in a sum of complex basis functions, namely its Fourier series decomposition, which gets

$$f_T(t) = \sum_{j=-\infty}^{+\infty} a_j e^{i2\pi\omega_j t} \quad a_j \in \mathbb{R}, \quad (5.2)$$

where $w_j = \frac{j}{T}$ is kept constant. The set of basis functions $\{e^{i2\pi\omega_j s}, j \in \mathbb{Z}\}$ builds a complete and orthonormal set spanning the space $\mathbf{L}_T^2(\mathbb{R})$.

In equation 5.2, the different basis functions coefficients a_j are measuring the amplitude of the corresponding frequency component w_j – of their associated basis functions – contained in the function f_T . The coefficients a_j are computed as the orthogonal projection of f_T onto the vector space spanned by the basis functions, the coefficients being the projection on the associated basis function. The projection is obtained by computing the inner product

$$a_j = \int_0^T f_T(u) e^{i2\pi\omega_j u} du. \quad (5.3)$$

This implies that if the coefficient $a_j = 0$, the orthogonal projection of f_T onto its associated basis function is also 0 and therefore the corresponding frequency is not contained in the signal f_T . Thence, equation 5.2 completely describes the signal f_T by its frequency content.

The Fourier series can be extended to non-periodic functions, or alternatively said functions with infinite period T . The so obtained transform, named *Fourier transform*, is defined for arbitrary square integrable functions in $\mathbf{L}^2(\mathbb{R})$. The Fourier transform is an operator $\mathcal{F} = \hat{f} : \mathbf{L}^2(\mathbb{R}) \rightarrow \mathbf{L}^2(\mathbb{R})$, defined as

$$\mathcal{F}(f)(s) \equiv \hat{f}(s) = \int_{-\infty}^{+\infty} f(u)e^{-i2\pi su} du, \quad (5.4)$$

where $s = \omega_j$ and $s \in \mathbb{R}$ can assume any value. The coefficients, computed in the Fourier series by means of equation 5.3, now get

$$a(s) = \int_{-\infty}^{+\infty} f(u)e^{i2\pi su} du. \quad (5.5)$$

In equation 5.5 each value s has its associated periodic basis function $e^{i2\pi su}$ of given frequency s . Like in the periodic case of the Fourier series, the coefficients $a(s)$ (equation 5.5) are different from 0 if the frequency s is present in the signal f . The inverse Fourier transform is defined as

$$\mathcal{F}^{-1}(\hat{f})(t) = \int_{-\infty}^{+\infty} \hat{f}(s)e^{i2\pi\omega t} ds \quad (5.6)$$

where $f(t) = \mathcal{F}^{-1}(\hat{f})(t)$.

5.3 Short-time Fourier transform

For the purpose of getting an impression regarding the concept of time-frequency space, let us suppose that we are listening a piece of music and that the audio signal is represented as a real function f of the time. The auditory information we perceive takes place on time and frequency simultaneously. Our auditory system transforms the sound f onto a signal $\tilde{f}(t, \omega)$ that depends on both time and frequency. It is worth to mention, that the sound we have heard some time ago is no more influencing what we are hearing now. Therefore, it must exist a real number $t_0 > 0$ such that $\tilde{f}(t, \omega)$ – virtually computed in our auditory system – depends only on a finite interval of time $[t - t_0, t]$.

Mathematically this means that the *modulating function* used to analyze the frequency content of $\tilde{f}(t, \omega)$ is localized in time and hence, the analysis is concentrated only in a neighborhood of t . Denis Gabor proposed for the

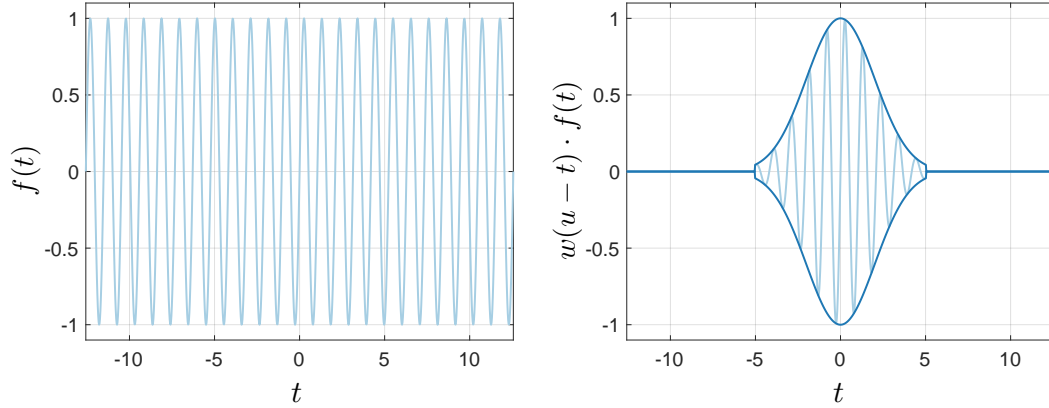


Figure 5.2: Windowing function. Left: original signal. Right: Function localized with a finite support Gaussian windowing function. When another function is multiplied by a window function the result is a zero-valued function outside the support of the windowing function, thereof its name.

One possible approach to analyze the function f on a finite interval is by using an auxiliary localized modulating function $w(u)$ which, in turn, localizes the otherwise infinite support basis function $e^{-i2\pi\omega u}$ of the Fourier transform. This reads as

$$w_{t,\omega}(u) = w(u-t)e^{-i2\pi\omega u} \quad (5.7)$$

and is depicted graphically with an example in Figure 5.2. If the windowing function w is time localized than the basis functions $w_{t,\omega}(u)$ result localized too. Let's first requiring that the window function $w(u)$ has norm 1, that is

$$\int_{-\infty}^{+\infty} w(u)du = 1 \quad (5.8)$$

and clearly also the shifted function $w(u-t)$ has norm 1, i.e.,

$$\int_{-\infty}^{+\infty} w(u-t)du = 1 \quad \forall t. \quad (5.9)$$

Given a signal $x(t) \in \mathbf{L}^2(\mathbb{R})$ and using equation 5.9 we can express $x(t)$

$$x(t) = x(t) \int_{-\infty}^{+\infty} w(u-t)du = \int_{-\infty}^{+\infty} x(t)w(u-t)du. \quad (5.10)$$

Starting from the Fourier transform definition

$$\mathcal{F}(\mathbf{x}) \equiv \mathbf{X}(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-i\omega t} dt \quad (5.11)$$

the short-time Fourier transform can be obtained substituting $x(t)$ from equation 5.9 into 5.11, which reads

$$\mathbf{X}(\omega) = \int_{-\infty}^{+\infty} \left[\int_{-\infty}^{+\infty} x(t)w(t-u)dt \right] e^{-i\omega u} du = \quad (5.12)$$

$$= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x(t)w(t-u)e^{-i\omega t} dt du. \quad (5.13)$$

By changing the order of integration

$$\mathbf{X}(\omega) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x(t)w(t-u)e^{-i\omega t} dt du \quad (5.14)$$

$$= \int_{-\infty}^{+\infty} \left[\int_{-\infty}^{+\infty} x(t)w(t-u)du \right] dt \quad (5.15)$$

$$= \int_{-\infty}^{+\infty} \mathbf{X}(t, \omega) dt. \quad (5.16)$$

The Fourier transform using the above mentioned windowing functions gets

$$\tilde{f}(t, \omega) = \int_{-\infty}^{+\infty} w(u-t)f(u)e^{-i2\pi\omega u} du = \int_{-\infty}^{+\infty} f(u)w_{t,\omega}(u)du = \langle \hat{w}_{t,\omega}, \hat{f} \rangle. \quad (5.17)$$

The transform $f \rightarrow \tilde{f}(t, \omega)$ in equation 5.17 is named short-time Fourier transform. It is worth to note that if $f \in \mathbf{L}^2(\mathbb{R})$, than $\tilde{f}(t, \omega) \in \mathbf{L}^2(\mathbb{R}^2)$. The short-time Fourier transform is therefore called *time-frequency transform* on the domain (t, ω) . If the windowing function w , in equation 5.17, has good localization properties, than also the transformed function \tilde{f} is well localized, i.e.,

$$\tilde{f}(t, \omega) = \langle w_{t,\omega}, t \rangle = \langle \hat{w}_{t,\omega}, \hat{f} \rangle. \quad (5.18)$$

This property translates in a transform which returns information both localized in the time and in the frequency domain.

5.3.1 The Heisenberg Uncertainty Principle

Unfortunately, one cannot analyze the function $f \in \mathbf{L}^2(\mathbb{R})$ at infinite finer resolution both in time and frequency domain (t, ω) simultaneously. There is a limit in the precision that one can achieve in localizing time and frequency jointly. This law is basically the same as the Heisenberg Uncertainty Principle that applies in physics for the simultaneous determination of positions and momenta of particles. Intuitively this principle translates in the following pragmatic property: *the more precise we want to know the frequency content the longer we must observe the process.*

Assuming that the norm $\mathbf{L}^2(\mathbb{R})$ of the window function w is 1, i.e. $\|w\|^2 = 1$ than, using the Plancherel theorem¹ it follows that also $\|\hat{w}\|^2 = 1$. Considering both w and \hat{w} (its corresponding Fourier transformed function) as probability distribution functions, than we can compute their first moment respectively as

$$\bar{t} = \int_{-\infty}^{+\infty} t \cdot |w(t)| dt \quad \text{and} \quad \bar{\omega} = \int_{-\infty}^{+\infty} \omega \cdot |\hat{w}(\omega)| d\omega. \quad (5.20)$$

Their standard deviations is obtained as

$$\delta_t^2 = \int_{-\infty}^{+\infty} (t - \bar{t})^2 \cdot |w(t)|^2 dt \quad \text{and} \quad \delta_\omega^2 = \int_{-\infty}^{+\infty} (\omega - \bar{\omega})^2 \cdot |\hat{w}(\omega)|^2 d\omega. \quad (5.21)$$

The standard deviations of equations 5.21 then are used as the two dimensions of the so called *Heisenberg boxes*, which represent a measure of the time-frequency localization of the transform. Indeed, geometrically the time-frequency domain localization (t, ω) of a given transform is often represented by rectangles of dimensions $\delta_t \times \delta_\omega$, which characterize its associated tiling of the time-frequency plane (see Figure 5.3). The Heisenberg principle defines a limit in the simultaneous precision achievable by a transform, which is

$$\delta_t^2 \delta_\omega^2 \geq \frac{1}{4\pi}. \quad (5.22)$$

Equation 5.22 is the classic *blanket that is too short*: when δ_ω is small, i.e. well localized in frequency, than δ_t must increase which translates in a loss in time localization and vice-versa. In other words there is a constraint in the simultaneous precision.

¹Plancherel equation states that

$$\|f\|^2 = \|\hat{f}\|^2 \quad (5.19)$$

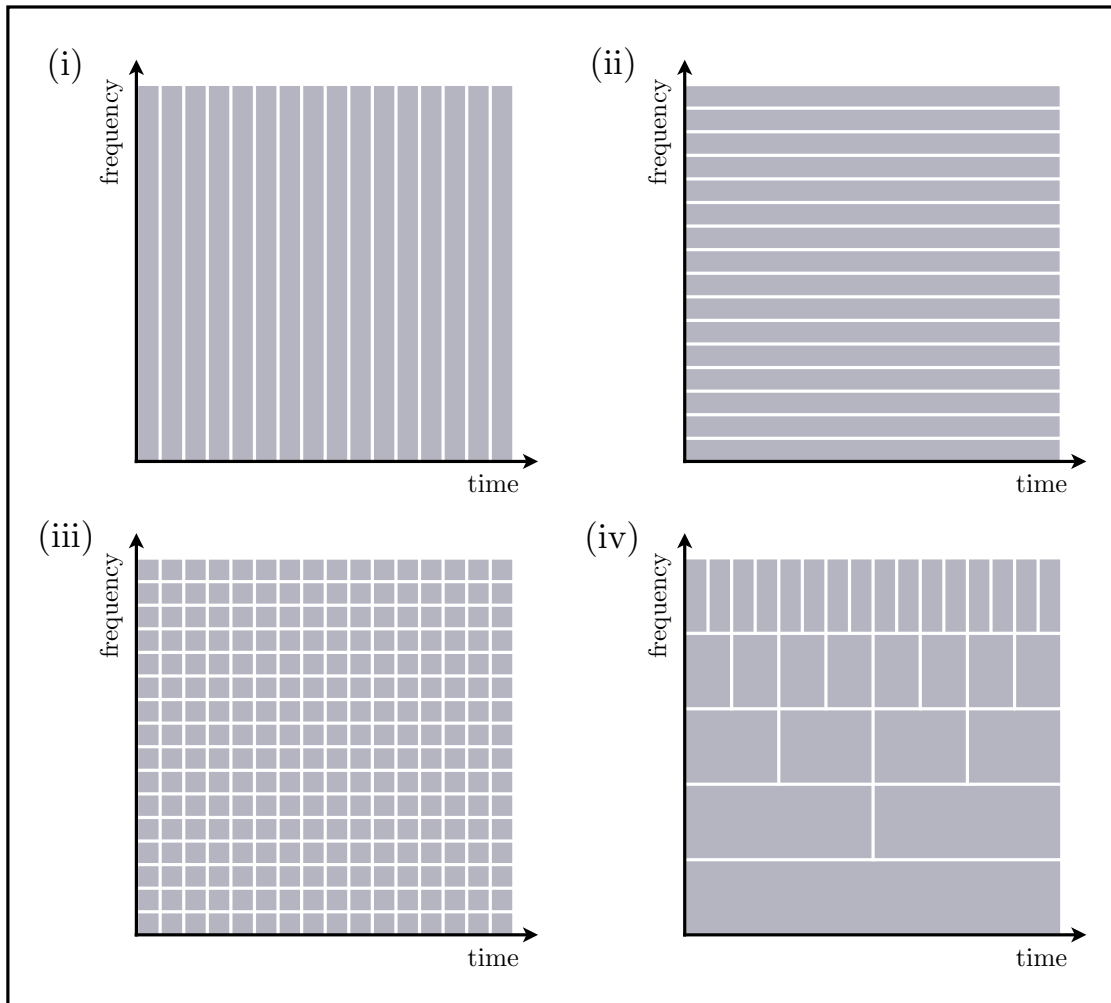


Figure 5.3: Tiling of the time-frequency plane for different transforms. *i)* Time-domain sampled signal representation. *ii)* Tiling of the time-frequency plane by the Fourier transform. It can be noted that the time information is completely missing. *iii)* The STFT returns a uniform rectangular tiling of the time-frequency plane, with Gaussian windows function it returns a squared tiling. *iv)* The wavelet transform produces a time-varying tiling of the time-frequency plane. At low frequency a better frequency resolution at high frequency a better time-domain resolution. The smallest possible Heisenberg box area is $1/4\pi$.

5.4 Algorithm overview

In order to reduce the computational complexity of aligning pairwise sequences by means of DP algorithms, candidate homologous segments are predicted for the purpose of filtering out non-promising regions from the computation, as

shown in Figure 5.1. The homologous segments detection is achieved by means of the cross-correlation of the sequences physicochemical properties. The computation of the cross-correlations is accelerated exploiting the convolution theorem of the Fourier transform which has been implemented in a fast Fourier transform (FFT) based algorithm. The cross-correlation computed by the FFT returns a discrete signal f_k that can then be represented as a function of the positional lag k between two input signals. Figure 5.4 shows the signal f_k computed with the original approach suggested by Katoh et al. [62] (top panel) and our proposed approach based on the Grantham's distance (bottom panel), see Appendix E for more details. The signal f_k identifies lags k for which the columns of the two mutually shifted sequences present a high degree of similarity. For example, in Figure 5.4 both approaches return $f_k = \{300, 310\}$ as positional shifts yielding candidate homologous segments. For a given positional shift one can find more than one homologous segment, the bigger the total homologous region the more pronounced is a peak.

Henceforth, we will consider such similarity (computed by means of the signal cross-correlation) as a proxy of a putative homology. However, since the Fourier transform analysis localizes the signal only in the frequency domain, as represented by the corresponding tiling of the time-frequency plane in Figure 5.3 (ii), the function f_k defines only the shifts k where homologies can be found, without providing information on the location of the actual homologous regions within the sequences. Katoh et al. proposed in their approach to score – column by column – the shifted sequences considering candidate segments regions having a score > 0.7 . We tried a similar approach computing the column marginal likelihood under PIP but unfortunately without success. The first problem was the definition of a suitable (data dependent) threshold that segregates the candidate segments from the rest. The second problem was that the column gap content depends on the model parameters like for example on the indel rates and the branch length. Therefore, it is not true that a high column likelihood value is directly related to a lower content of gaps. If the primary objective of the segment detection is to identify highly conserved blocks inside the sequences, than the marginal likelihood under PIP is not the appropriate tool to choose. A more naive instrument, although being more suitable for our purpose, could be the direct evaluation of the number of gaps contained in the shifted columns. This has been depicted graphically with a small example in Figure 5.5.

For addressing this problem, we propose a multi-resolution short-time Fourier transform (STFT) which provides information on both lag k and location of the detected homologous blocks. Indeed, as displayed in Figure 5.3, the

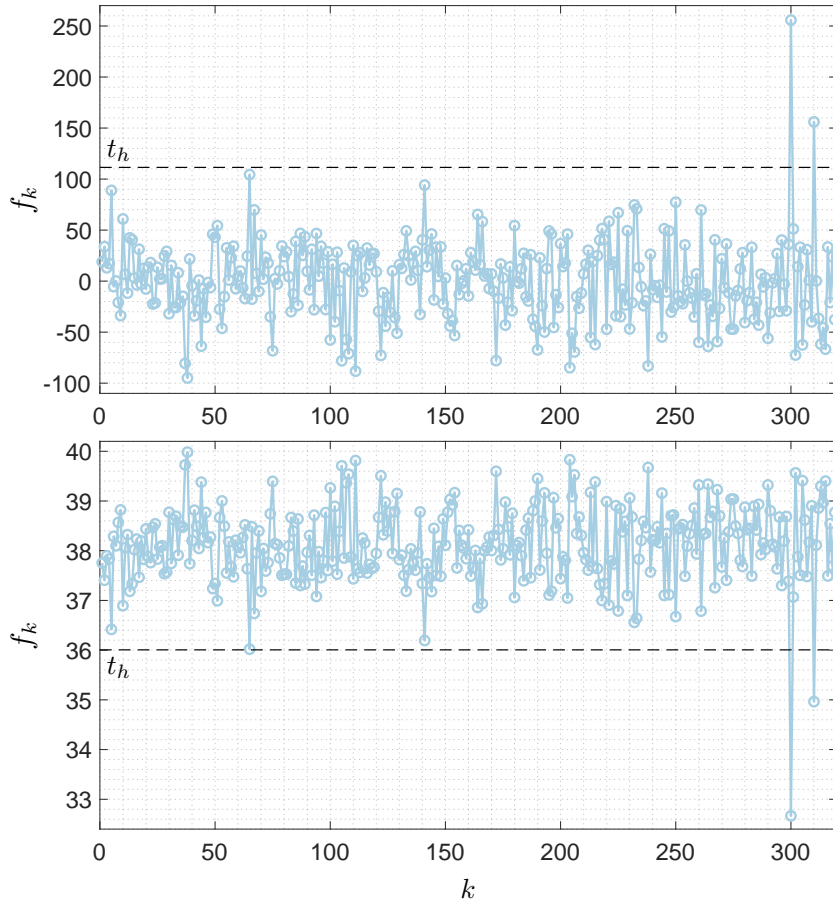


Figure 5.4: FT coefficients f_k . Top: Coefficients computed with the cross-correlation approach proposed by Katoh et al. [62]. Bottom: Coefficients computed with our proposed approach which exploits the Grantham's distance. For more details see also Appendix E. The dotted line represents the noise threshold t_h . The threshold is calculated by recomputing the cross-correlation matrix $f_w[m, k]$ after randomly permuting the residues of one of the two sequences. This statistically destroys any potential homologous patterns in the sequences and allows us to define an intrinsic 'noise' level as the maximum, respectively minimum, of the cross-correlation coefficients $f_w[m, k]$. In the cross-correlation (top) approach the coefficients bigger than the noise threshold are denoting similar patterns in the signals, in the Grantham's approach (bottom) are instead the coefficient smaller than t_h .

STFT differently from the FT is localized both in time and frequency. The size of this window characterizes the space-frequency resolution at which the analysis of the signal is performed. By progressively reducing the size of the window, the frequency resolution reduces while the space resolution increases,

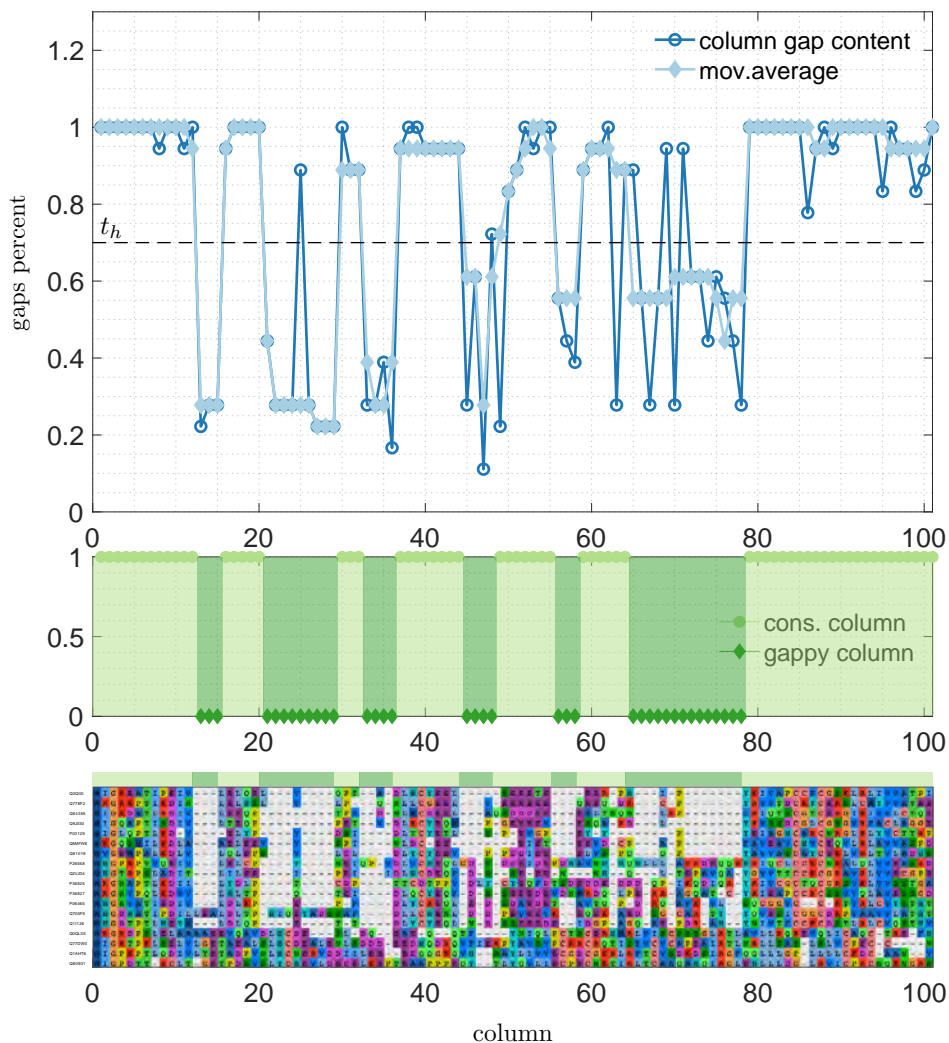


Figure 5.5: Column-gap content. Bottom: MSA portion of dataset RV912/Box49 (papillomavirus protein) taken from BALiBASE benchmarks. Top: Column-gap content computed from the alignment shown in the bottom panel. The gap content is then filtered with a median filtering in order to remove isolated spikes. The gap content threshold is set to $t_h = 0.7$ depicted with a dashed line. Middle: Regions having a gap content $< 30\%$, named “conserved columns”, are assigned to value 1 (highlighted in light green), regions exceeding 30% of gaps are assigned to value 0 (highlighted in dark green). Regions in light green constitute the candidate homologous blocks and are aligned first.

as expressed in equation 5.22 discussed in Section 5.3.1. This yields a more precise localization of the detected ‘similar’ patterns within the sequences,

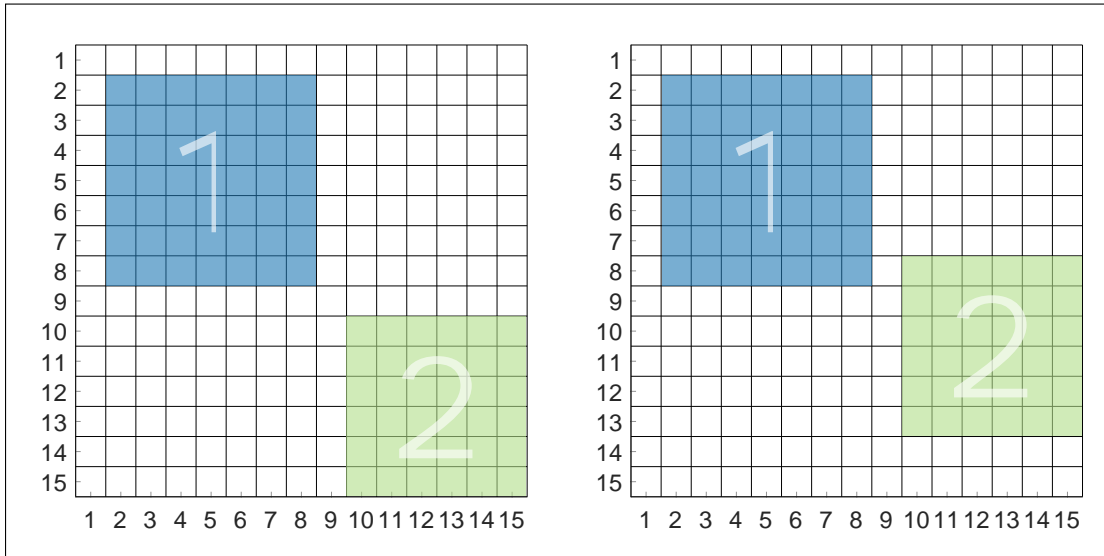


Figure 5.6: Non-overlapping (left) and overlapping (right) blocks. Overlapping blocks need to be resolved prior to be aligned. A more detailed explanation regarding our proposed algorithm to resolve overlaps is provided in Section 5.4.5 and Appendix F.

albeit at the cost of a more expensive computational effort. Although, there are strategies to reduce the computational effort required (briefly discussed below).

The candidate homologous regions thus detected by the STFT allow us to define a set \mathcal{B} of blocks in a MSA homology matrix analogous to the one introduced in Katoh et al. [62]. An example is shown in Figure 5.6, which represents a simplified configuration with non-overlapping blocks (left panel) and overlapping homologous blocks (right panel). The homologous blocks need to be selected, arranged and connected in a consistent way for designing a meaningful final alignment. To that purpose, within the set \mathcal{B} we first identify the set \mathcal{P} of all possible paths connecting the candidate homologous blocks and potentially yielding a final alignment. Each paths $p_i \in \mathcal{P}$ is then analyzed independently. As anticipated above, we first observe that along a given path p_i , two generic blocks $b_j, b_k \in \mathcal{B}$ might exhibit overlapping regions. Such overlaps must be removed so that b_j and b_k can be aligned as independent sub-matrices. In the presence of long sequences, which might exhibit a complex pattern of possibly overlapping homologous blocks, a consistent alignment can become a highly non-trivial task. This task is discussed in more details in 5.4.5 and Appendix F.

Finally, among the overlap-free paths $\tilde{\mathcal{P}}$ we select the optimal path p^* , including a set of homologous blocks \mathfrak{H} derived from the original set \mathcal{B} , according

to some criteria that need to be specified. At this point, the homologous blocks $\in \mathfrak{H}$ can be independently aligned following a DP approach (see also Sections 1 and 2). An additional set of intermediate *linking blocks* \mathfrak{L} is then introduced, when needed, to connect consecutive homologous blocks. The linking blocks $\in \mathfrak{L}$ are also independently aligned in a DP manner. The resulting tracebacks, from both \mathfrak{H} and \mathfrak{L} , are joined together yielding the final alignment.

It should be noted that in the homology matrix only the areas corresponding to the homologous and linking blocks of the previously selected optimal path p^* are retained, while the rest of the matrix is excluded from the calculations. This leads to a sparse matrix layout with important effects on the efficiency of the method especially in the framework of a 3-dimensional approach such as the one described in Maiolo et al.[88]. This aspect has been depicted graphically in Figure 5.1 where we have also have estimated the theoretical speed-up as function of the number of total blocks (homologous plus linking blocks).

The various steps of the algorithm, briefly outlined above, are considered sequentially in more detail in the next sections.

5.4.1 Sequence residues to signal conversion

In the MAFFT approach described by Katoh et al. [62], homologies between sequences are detected using a cross-correlation-based analysis of the physicochemical properties, i.e., volumes \mathbf{v} and polarities \mathbf{p} , of the sequence residues. The degree of cross-correlation between two sequences is used as a measure of the physicochemical similarity of the two molecules, which in turn is used as a proxy for the likelihood that the two molecules might undergo a substitution in an evolutionary process. Substitutions between physico-chemically similar amino acids tend indeed to preserve the structure of proteins, and they are therefore more likely to occur during evolution. The computational complexity of the cross-correlation is then reduced from L^2 to $L \cdot \log(L)$, where L is the average sequence length, by means of a FFT algorithm.

In our approach, the space of physicochemical properties has been extended to include also the chemical composition \mathbf{c} as suggested in *Grantham 1974* [48], besides the volume and polarity features already used in MAFFT. In Table 5.1 we have reported the physicochemical properties used in our algorithm. Subsequently, we have defined a *signal* \mathbf{s} as a $3 \times L$ multidimensional vector of the dimensionless physicochemical property values obtained from the input sequence, translating the residues with the values from Table 5.1, in the

AA	symbol	volume	polarity	chem. comp.
ALA	A	31.00	8.10	0.00
ARG	R	124.00	10.50	0.65
ASN	N	56.00	11.60	1.33
ASP	D	54.00	13.00	1.38
CYS	C	55.00	5.50	2.75
GLN	Q	85.00	10.50	0.89
GLU	E	83.00	12.30	0.92
GLY	G	3.00	9.00	0.74
HIS	H	96.00	10.40	0.58
ILE	I	111.00	5.20	0.00
LEU	L	111.00	4.90	0.00
LYS	K	119.00	11.30	0.33
MET	M	105.00	5.70	0.00
PHE	F	132.00	5.20	0.00
PRO	P	32.50	8.00	0.39
SER	S	32.00	9.20	1.42
THR	T	61.00	8.60	0.71
TRP	W	170.00	5.40	0.13
TYR	Y	136.00	6.20	0.20
VAL	V	84.00	5.90	0.00
-	-	0.00	0.00	0.00

Table 5.1: Amino-acids physiochemical properties used in our algorithm. The symbol ‘-’ refers to a gap character.

following way

$$\mathbf{s} = [\mathbf{v}, \mathbf{p}, \mathbf{c}]^T = \left[\begin{bmatrix} v_1 \\ p_1 \\ c_1 \end{bmatrix}, \begin{bmatrix} v_2 \\ p_2 \\ c_2 \end{bmatrix}, \dots, \begin{bmatrix} v_L \\ p_L \\ c_L \end{bmatrix} \right]. \quad (5.23)$$

In general, when the input is a sub-MSA rather than a sequence, the signal \mathbf{s} is obtained by averaging over the columns of the sub-MSA. Since the properties volume, polarity and chemical composition differ significantly in magnitude, we will use a standardized signal $\hat{\mathbf{s}}$ defined in terms of its standardized components, i.e. $\hat{\mathbf{v}}$, $\hat{\mathbf{p}}$ and $\hat{\mathbf{c}}$. Unlike MAFFT [62], however, we define a standardized physicochemical property as $\hat{\mathbf{v}} = (\mathbf{v} - \bar{v}) / \sigma_v$, where \bar{v} and σ_v are respectively the average and standard deviation computed over the input data rather than the 20 amino-acids in Table 5.1. Analogous definitions hold for $\hat{\mathbf{p}}$ and $\hat{\mathbf{c}}$. This

improved definition ensures that the standardized signals will fluctuate around zero even with non-homogeneous amino-acids distributions in the input data.

5.4.2 Signal padding

As mentioned above, when two sequences S_1 and S_2 have different lengths, a first *static* padding is required for having both signals of equal length $L = \max(L_1, L_2)$. Moreover, extra *dynamic* padding, dependent on the resolution level of the analysis (dictated by the window function support size), is necessary for avoiding boundary effects introduced by the sliding window w . The effects of the non-application of the dynamic padding have been quantified and are shown in Figure 5.10.

The dynamic padding can be achieved through different padding patterns, which allow us to tune the structure of the cross-correlation matrix $f_w[m, k]$ (see definition below). In order to center the cross-correlation coefficients in the spectrogram $f_w[m, k]$ we found convenient to use the following scheme. Given a window function w with a support of length S_w , signal $\hat{\mathbf{s}}_1$ is padded on the left side with an array of S_w column vectors of zeros, whereas $\hat{\mathbf{s}}_2$ is padded on both left and right sides with two arrays of $S_w/2$ column vectors of zeros. Padding in this case is dependent on the variable size of the window w and therefore it is referred to as *dynamic*. This is shown graphically below, where we have assumed $L = L_1 > L_2$,

$$\hat{\mathbf{s}}_1 = \left[\underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{S_w}, \underbrace{\begin{bmatrix} \hat{v}_{1,1} \\ \hat{p}_{1,1} \\ \hat{c}_{1,1} \end{bmatrix}, \begin{bmatrix} \hat{v}_{2,1} \\ \hat{p}_{2,1} \\ \hat{c}_{2,1} \end{bmatrix}, \dots, \begin{bmatrix} \hat{v}_{L_1,1} \\ \hat{p}_{L_1,1} \\ \hat{c}_{L_1,1} \end{bmatrix}}_L \right]$$

$$\hat{\mathbf{s}}_2 = \left[\underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{S_w/2}, \underbrace{\begin{bmatrix} \hat{v}_{1,2} \\ \hat{p}_{1,2} \\ \hat{c}_{1,2} \end{bmatrix}, \begin{bmatrix} \hat{v}_{2,2} \\ \hat{p}_{2,2} \\ \hat{c}_{2,2} \end{bmatrix}, \dots, \begin{bmatrix} \hat{v}_{L_2,2} \\ \hat{p}_{L_2,2} \\ \hat{c}_{L_2,2} \end{bmatrix}}_{L_2}, \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{L-L_2}, \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{S_w/2} \right].$$

Padding in this case is dependent on the variable size of the window w and therefore it is referred to as dynamic.

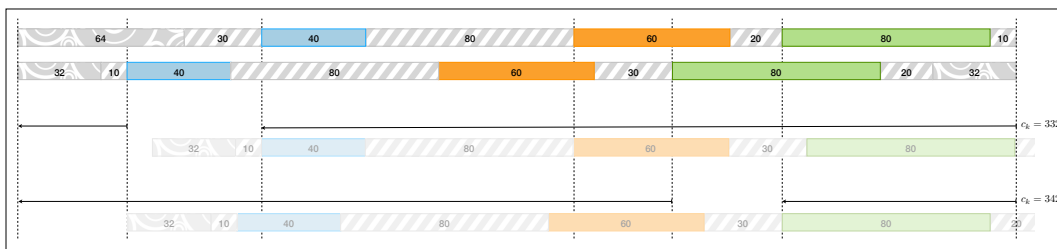


Figure 5.7: Padding scheme with $S_w = 64$. Assuming the analysis starts at level l_1 with a window function of size 64, the signal $\widehat{\mathbf{s}}_1$ is padded on the left side with an array of $S_w = 64$ column vectors of zeros, whereas $\widehat{\mathbf{s}}_2$ is padded on both left and right sides with two arrays of $S_w/2 = 32$ column vectors of zeros. The figure also shows $\widehat{\mathbf{s}}_2$ shifted on the left side (considering periodic boundary conditions) of 332 and 342 positions in order to match the respective patterns on $\widehat{\mathbf{s}}_1$.

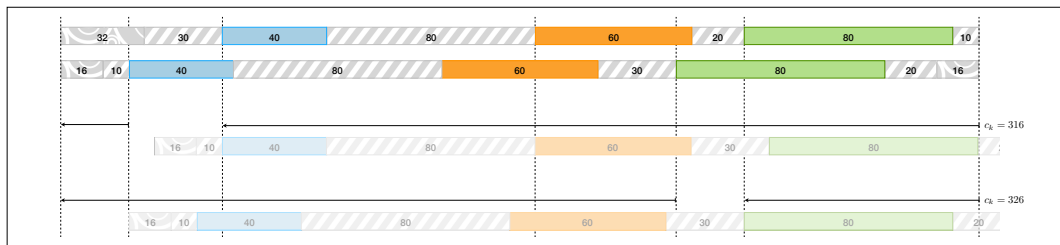


Figure 5.8: Padding scheme with $S_w = 32$. At level l_2 , using a window function of size 32, the signal $\widehat{\mathbf{s}}_1$ is padded on the left side with an array of $S_w = 32$ column vectors of zeros, whereas $\widehat{\mathbf{s}}_2$ is padded on both left and right sides with two arrays of $S_w/2 = 16$ column vectors of zeros. The figure also shows $\widehat{\mathbf{s}}_2$ shifted on the left side (considering periodic boundary conditions) of 316 and 326 positions in order to match the respective patterns on $\widehat{\mathbf{s}}_1$.

5.4.3 Fourier transform based homology detection

We use the cross-correlation between the physicochemical properties of two sequences as a measure of their similarity, which in turn is considered as a fast proxy for putative homologies [62]. Given two sequences S_1 and S_2 of equal length L , with associated signals $\widehat{\mathbf{s}}_1$ and $\widehat{\mathbf{s}}_2$ given by equation 5.23, the cross-correlation

$$f_k \equiv f[k] = \sum_{1 \leq i, i+k \leq L} \widehat{\mathbf{s}}_{i,1} \circ \widehat{\mathbf{s}}_{i+k,2}, \quad (5.24)$$

is computed as a function of the lag k between the sequences, where the subscripts 1 and 2 refer to the sequences, i and $i+k$ are column indexes in the

signals $\widehat{\mathbf{s}}_1$ and $\widehat{\mathbf{s}}_2$, respectively, and the symbol \circ denotes a standard dot product. Note that in general the two sequences S_1 and S_2 have two different lengths, say L_1 and L_2 , respectively. In that case $L = \max(L_1, L_2)$ and the shorter of the two sequences is padded with column vectors of zeros.

The convolution theorem allows us to compute the cross-correlation of equation 5.24 by means of the FT [62]. Indeed, the convolution theorem states that under suitable conditions, not discussed here, point-wise multiplication in one domain (e.g., the time domain) equals the convolution in the other domain (e.g., the frequency domain). So, the Fourier transform of a convolution of two signals is equivalent to the pointwise product of their Fourier transforms. Denoting the Fourier transform of a generic input signal \mathbf{x} as $\mathcal{F}(\mathbf{x}) \equiv \mathbf{X}[\omega]$, the volume component of the cross-correlation, for instance, can be written as,

$$\begin{aligned} f^{(v)}[k] &= \mathcal{F}^{-1}(\mathcal{F}(\mathbf{v}_1^*) \cdot \mathcal{F}(\mathbf{v}_2)) \\ &= \mathcal{F}^{-1}(\mathbf{V}_1^*[\omega] \cdot \mathbf{V}_2[\omega]) , \end{aligned} \quad (5.25)$$

where \mathcal{F}^{-1} represents the inverse Fourier transform, and the asterisk denotes the complex conjugate. Note that the latter in our case has no effect since all physicochemical properties are real values. Analogous definitions hold for the polarity and chemical composition components, $f^{(p)}[k]$ and $f^{(c)}[k]$, respectively. The cross-correlation is then given by the sum

$$f[k] = f^{(v)}[k] + f^{(p)}[k] + f^{(c)}[k] . \quad (5.26)$$

The use of the FFT in equation 5.25 reduces the computational complexity from $\mathcal{O}(L^2)$ to $\mathcal{O}(L \log L)$. The cross-correlation signal $f[k]$ is thus a discrete function of the positional lag k , whose peaks identify mutual shifts yielding highly-correlated regions (see Figure 5.4). As clearly shown in equation 5.25, the cross-correlation can be expressed in terms of Fourier transforms.

One major drawback of the Fourier transform is that its basis functions are complex exponentials, which are perfectly localized in frequency domain since they have a well-defined frequency (see for instance the tiling of the time-frequency plane in Figure 5.3). However, because of their infinite length, they have no time localization at all. In other words, time information is spread out over an infinite domain. As a consequence of the lack of time resolution of the Fourier transform, the cross-correlation of equation 5.26 can return only an indication of the shifts k where homologous regions can be found, without providing any information on the actual position of such regions within the sequences. The localization of the homologous regions has been addressed by Katoh et al. [62] by computing a column-wise scoring of the shifted sequences.

We have instead solved this issue with the computation of a time-frequency transform which provides simultaneously the information about the positional lag k and the relative position of the pattern in the sequences. In the next Section (5.4.4: [Multi-scale STFT based homology detection](#)) we are showing our proposed homologous block detection based on the multi-scale STFT.

5.4.4 Multi-scale STFT based homology detection

Here we propose an alternative solution that allows us to detect candidate shifts k and localize the corresponding homologous segments simultaneously within a unique conceptual framework, that is, the short-time Fourier Transform (STFT). The STFT of a generic discrete signal \mathbf{x} is defined as follows,

$$\mathcal{F}_{\text{ST}}(\mathbf{x}) \equiv \mathbf{X}[m, \omega] = \sum_{j=-\infty}^{\infty} x_j \cdot w_{j-m} \cdot e^{-i\omega j}, \quad (5.27)$$

where w represents a compact-support sliding window function, m is its displacement, and i is the imaginary unit. Typical choices for the window function are for instance Hamming, Hanning, Bartlett or Kaiser. Figure 5.9 depicts some example of window functions. For the purpose of detecting homologous blocks, we compute a windowed cross-correlation as,

$$f_w^{(x)}[m, k] = \mathcal{A}_w^{-1} \cdot \mathcal{F}^{-1}(\mathbf{X}_1^*[\omega] \cdot \mathbf{X}_2[m, \omega]), \quad (5.28)$$

where $\mathbf{X}_2[m, \omega]$ is the STFT defined in equation 5.27 and the signal \mathbf{x} can be the volume \mathbf{v} , polarity \mathbf{p} or chemical composition \mathbf{c} . The full cross-correlation $f_w[m, k]$ is given analogously to equation 5.26 by the sum of the three components. The term \mathcal{A}_w^{-1} is a normalization factor that we will define below. The index m defines the shift of the window function w with respect to the sequence to which the window is applied. For preventing boundary effects, the window w must slide over the entire length of the sequence. Therefore, $m = 1, \dots, L + S_w$, where S_w is the window support length. To qualitatively assess the effects on the boundaries induced by a missing padding we have computed equation 5.28 for a single component, let say the volume, of constant value for all the column (for our example we chose 0.25). To quantify the boundary effects as function of the window size, i.e. the resolution of the analysis, we have repeated the same experiment at three different resolution levels. We have set the window functions at size equal to 64, 32 and 16. The resulting spectrograms $f_w[m, k]$ are shown in Figure 5.10. One can note that the lower the resolution the “larger” (in magnitude and involved region) the

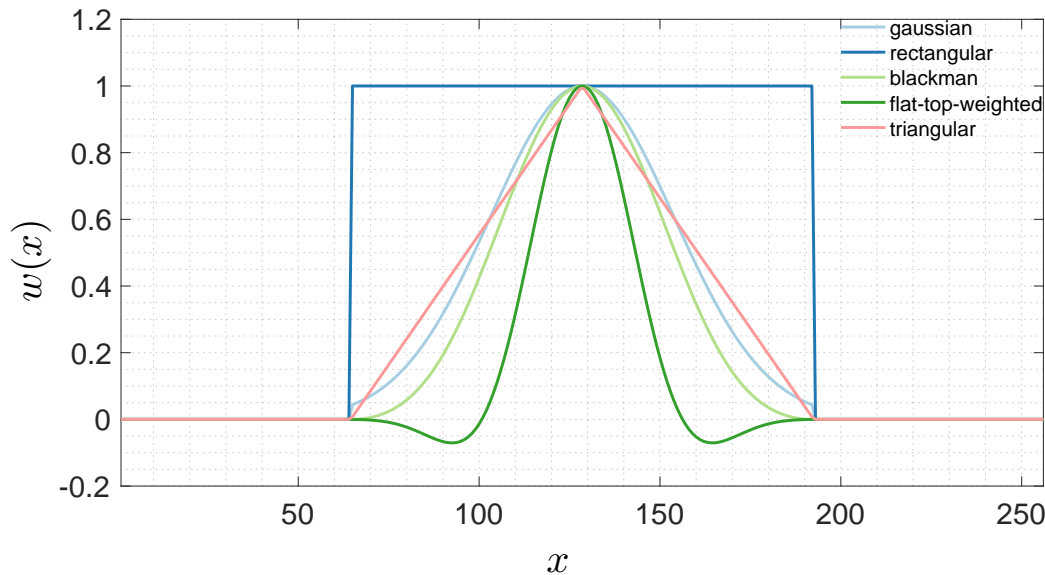


Figure 5.9: Different window functions. Window functions are mathematical functions that have a zero-value outside their support. They are normally symmetric around the middle of the support and usually the maximum is in the middle of the interval. Usually the window function is used to detect transient event in non-periodic functions and for the time-averaging of frequency spectra. The size of the support determines the time-frequency resolution. Window functions are also altering the original frequency content of the input signal by an effect called spectral leakage. Each type of window function acts differently on the spectral content, the choice of this function is made according to the particular needs.

edge effects. This observation is in accordance with what is known as the *Cone Of Influence* in Multiresolution analysis.

The STFT replaces the pure complex exponential functions of the classical Fourier transform with windowed basis functions. The latter yield a compact support which leads to a localization of the time information. The STFT thus returns information in both time and frequency domains. Indeed, the correlation $f_w^{(x)}[m, k]$ is now a 2-dimensional matrix, function of both the lag k and the moving window shift m . An example of spectrogram $f_w^{(x)}[m, k]$ is shown in Figure 5.11. On that Figure one can note three regions, marked M_1 , M_2 and M_3 at positional shifts k_1^* and k_2^* (more details below), corresponding to candidate homologous blocks.

We have exploited the time-frequency property of the STFT by a multi-scale analysis based on a progressive reduction of the window size S_w . At a

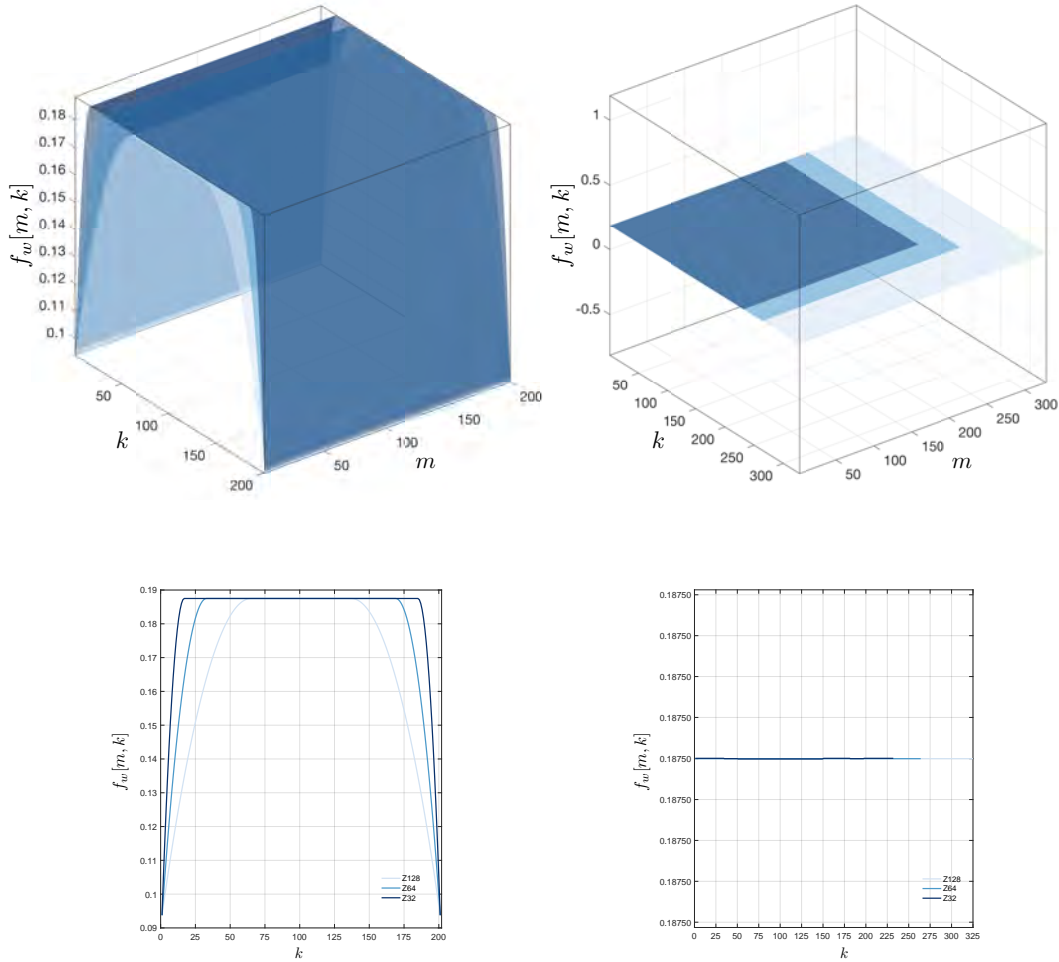


Figure 5.10: Boundaries effects at different resolution scales. We have computed the spectrogram $f_w[m, k]$ using equation 5.28 for the volume component. To focus our attention only on the effects on the boundaries we set a constant value for all the column, in this case equal to 0.25. To quantify the boundary effects as function of the window size, i.e. the resolution of the analysis, we have repeated the same experiment at three different resolution levels, i.e. using a window function of size 64, 32 and 16. Left: Experiment without padding the sequences. One can note that the lower the resolution the “larger” (in magnitude and involved region) the edge effects. Right: Padding the sequences as discussed in Section 5.4.2. It can be remarked that an appropriate padding remove or mitigate the effects on the edges.

large scale, i.e., a broad window support, the STFT provides low-resolution information on the localization of the homologous regions at a modest computational cost. Starting the analysis at large scale is also much less prone to

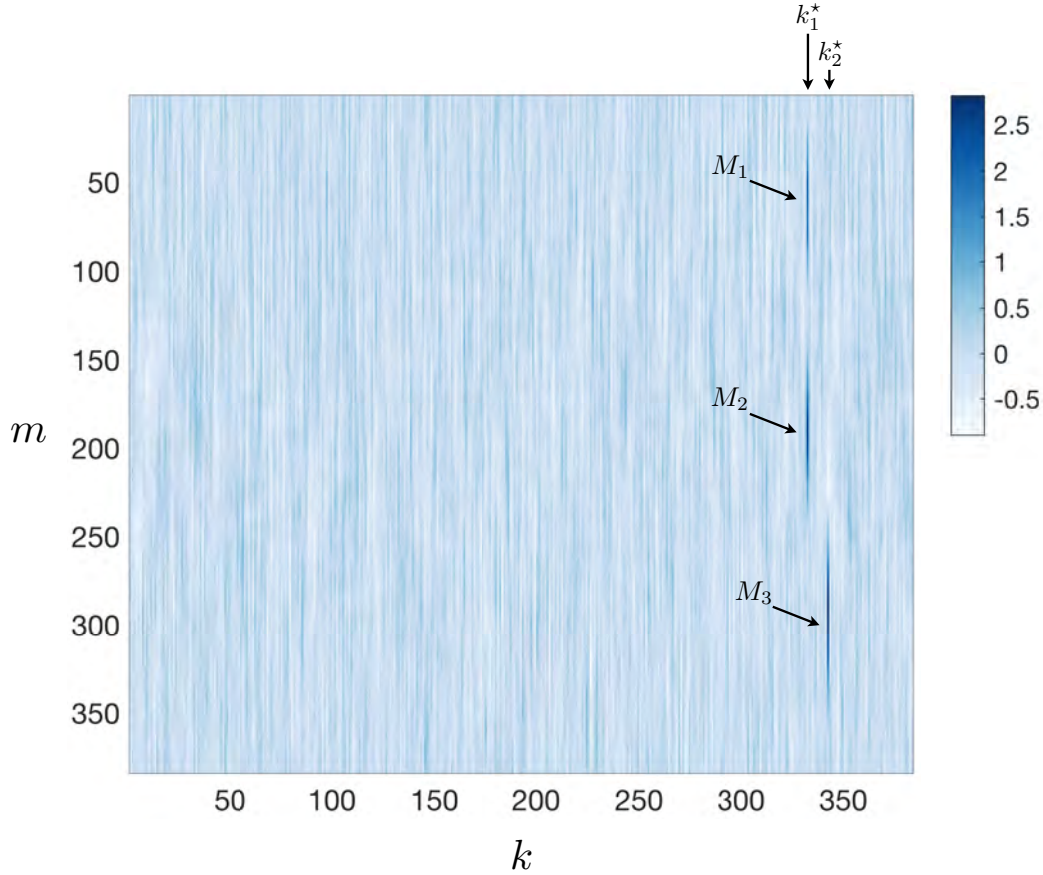


Figure 5.11: Graphical representation of the full cross-correlation matrix $f_w[m, k]$. One can identify two distinct shifts $k_{1,2}^*$ where the cross-correlation exceeds the noise threshold t_h , as explained in the text. The condition of equation 5.29 is fulfilled by the sets of window displacements M_1 and M_2 , when $k = k_1^*$, and by the set M_3 when $k = k_2^*$. The spectrogram refers to the signals depicted in Figure 5.7.

the artifacts that might arise at an excessively high resolution, that is, with an excessively short window (depicted in Figure 5.12). In the extreme limit of a window of length 1, for example, the correlation of equation 5.28 would return a misleadingly large number of false positives. Figure 5.13 shows shapes and positions of the window functions (in this case using Flat top weighted window functions²) used for the full STFT analysis at three increasingly resolution

²The Flat top weighted window is described by the following equation

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right) - a_3 \cos\left(\frac{6\pi n}{N-1}\right) + a_4 \cos\left(\frac{8\pi n}{N-1}\right)$$

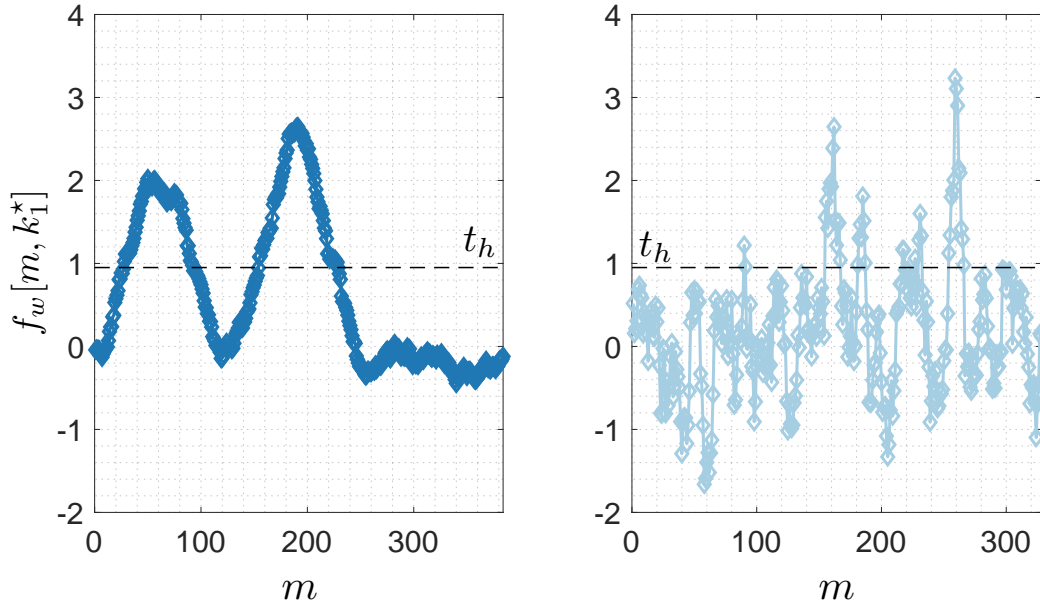


Figure 5.12: First analysis at two different window size. Left: Analysis with a window function of size equal to 64. The figure shows the slice $f_w[m, k_1^*]$. Right: Analysis with a window size equal to 8. At a large scale, the STFT provides low-resolution information on the localization of the homologous regions at a modest computational cost. Starting the analysis at large scale is also much less prone to the artifacts that might arise at an excessively high resolution, that is, with an excessively short window. Indeed, that slice should result in only two peaks, like in the left panel and not as in the right panel. Starting at too high resolution could detect false positive homologous blocks.

levels.

This first step thus translates into coarse but reliable and fast information. The window support size is then shrunk, typically by a factor 2, and the spatial resolution is thus increased (as shown in Figure 5.13). This leads to a more precise identification of the boundaries of the homologous regions (as shown in Figures 5.18 and 5.17). In principle, the process above can be iterated until a support size of one. However, practically, 2-3 iterations are normally sufficient to obtain a satisfactory resolution. Figure 5.13 shows 3 different resolution levels of a typical window function.

The approach described above provides simultaneously information on both the lags k and the positions of the homologous regions. However, this ad-

where $a_0 = 0.2156$, $a_1 = 0.4166$, $a_2 = 0.2773$, $a_3 = 0.0836$, $a_4 = 0.0070$.

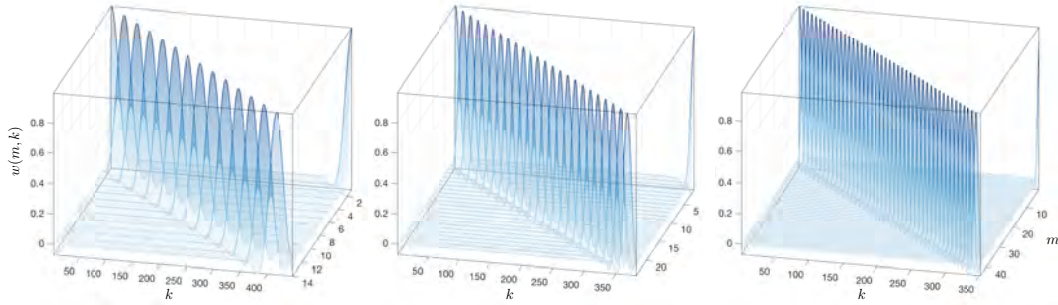


Figure 5.13: Window functions at different resolution level. From left to right: Flat top weighted window functions of size 128, 64 and 32 with a shift of respectively 32, 16 and 8 units.

vantage comes at a computational cost, which can be quantified for a naive implementation of the STFT by a complexity of $\mathcal{O}(L^2 \log L)$. Nevertheless, various algorithms have been proposed to reduce the intrinsic complexity of the STFT. For instance, for a sliding rectangular window, one can recycle most of the previously computed correlation coefficients. In addition, the sliding window can be moved in discrete steps larger than 1 while retaining most of the information carried by the signal. In Figure 5.13 window functions of length $S_w = 128$, $S_w = 64$ and $S_w = 32$, from left to right, respectively, have been shifted by a quantity equal to $S_w/4$. Figure 5.14 depicts the spectrogram slice at shift k_1^* , from the same example of Figure 5.11, i.e. $f_w[m, k_1^*]$ obtained with a window size of 64, at decreasing sliding length (from 1 to 16, doubling the length every time). Superimposing the curves at different step lengths, it can be noted that already at the coarsest sliding step the curve catches the most important features inside the signal. Indeed, the analysis using a sliding step of 1 contains redundant information and requires much more computational times than the coarsest one. Moreover, for further reducing the computational effort, each step of the analysis can be performed only on the regions emerged at the previous coarser iteration (as explained in detail below and depicted in Figure 5.17).

The workflow

Practically, the complete work-flow for the signal analysis and homologous blocks detection can be summarized as follows.

1. Starting at the lowest resolution (i.e., largest window support size S_w),

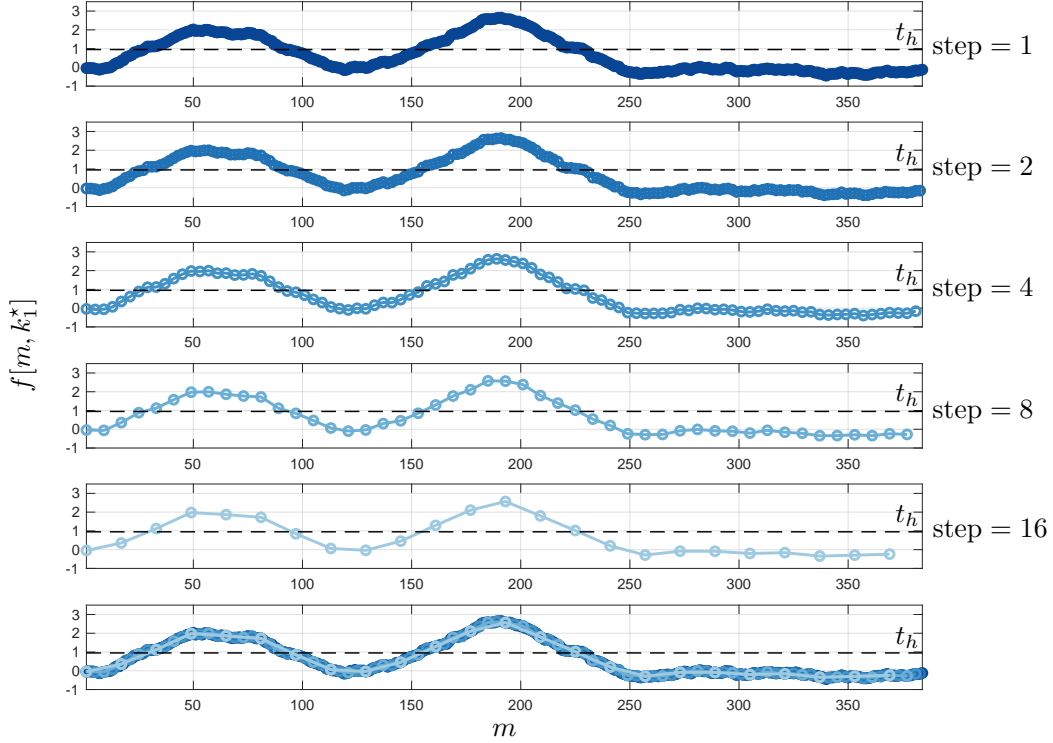


Figure 5.14: Spectrogram slice at different sliding length. From Top to bottom: spectrogram slice at shift k_1^* , i.e. $f_w[m, k_1^*]$ obtained with a window size of 64, at decreasing sliding length (from 1 to 16, doubling the length every time). In the bottom-most panel the different curves are superposed. It can be noted that already at the coarsest sliding step the curve catches the most important features inside the signal. Clearly, using a sliding length of 16 requires 16 times less computations than with 1.

we construct the full cross-correlation matrix $f_w[m, k]$, as described in equation 5.28. A cross-correlation is shown for example in Figure 5.11. We call this first resolution level l_1 .

2. At the current level l_1 we compute a noise threshold t_h , represented in Figure 5.17 by a dashed line. This threshold allows us to distinguish ‘noise’ from shifts k where true correlations, i.e., homologies, occur. The threshold is calculated by recomputing the cross-correlation matrix $f_w[m, k]$ after randomly permuting the residues of one of the two sequences. This statistically destroys any potential homologous patterns in the sequences and allows us to define an intrinsic ‘noise’ level as the maximum of the cross-correlation coefficients $f_w[m, k]$. This procedure is

eventually repeated until the noise level reaches a stationary value. Typi-

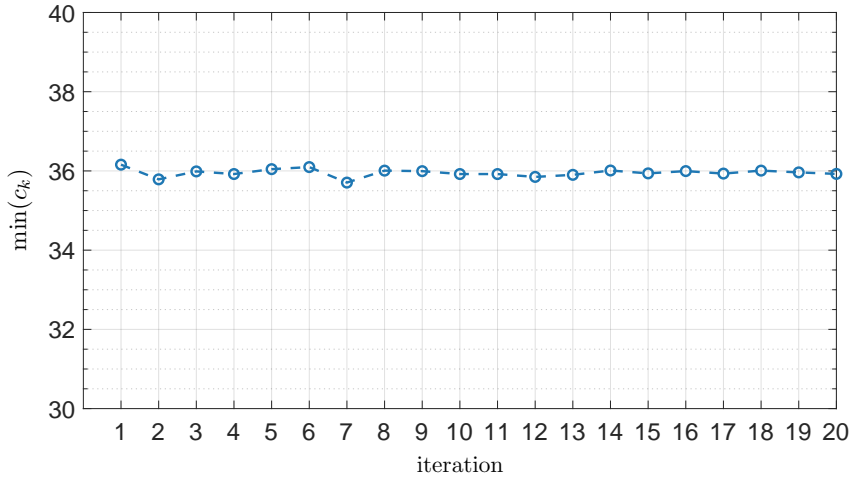


Figure 5.15: Noise threshold convergence. The threshold is calculated by recomputing the cross-correlation matrix $f_w[m, k]$ after randomly permuting the residues of one of the two sequences. This statistically destroys any potential homologous patterns in the sequences and allows us to define an intrinsic ‘noise’ level as the maximum of the cross-correlation coefficients $f_w[m, k]$. The plot represents the noise threshold estimated after several iterations. Typically, only a few iterations are sufficient for a good estimation of the noise threshold t_h .

at the first coarse resolution level l_1 and assumed constant through the different scales of the analysis outlined below.

3. We scan the matrix $f_w[m, k]$ to identify the set of shifts k^* where

$$f_w[m, k^*] > t_h. \quad (5.29)$$

4. For each shift k^* , the condition at point 3 is fulfilled by one or more sets M_i of window displacements m , with $i \geq 1$. Each set M_i corresponds to a specific pattern in the sequences characterized by a high correlation coefficient, i.e., a candidate homologous block. In Figure 5.11, for example,

one can identify three sets $M_{1,2,3}$ of window displacements corresponding to two shifts $k_{1,2}^*$. Figure 5.16 represents the spectrogram $f_w[m, k]$ as surface and the noise threshold t_h as plane (semi-transparent grey plane).

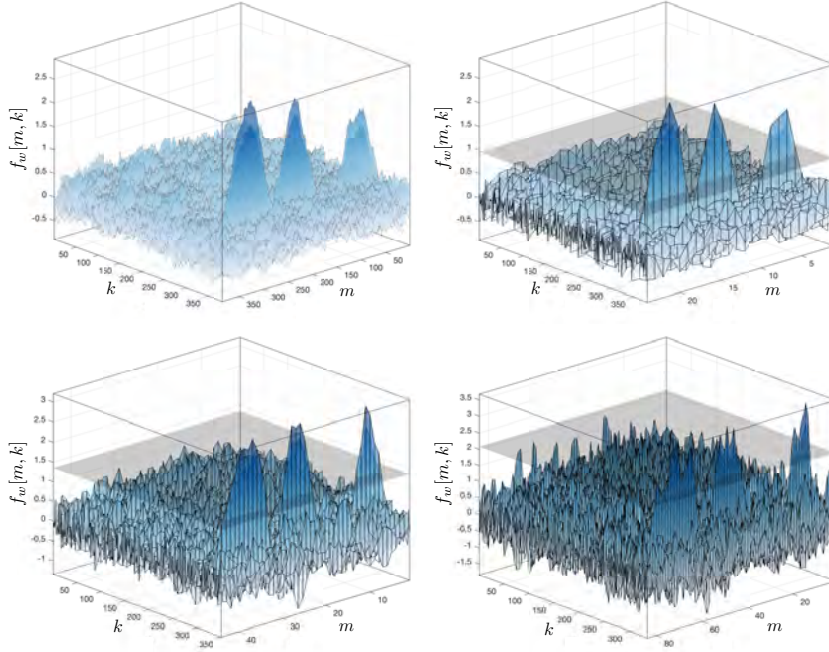


Figure 5.16: Multi-scale STFT surfaces. Top-left: Spectrogram surface obtained with a rectangular window function of size equal to 64 and sliding step of length 1. Top-right: Spectrogram surface with window size equal to 64 and sliding step of length 16. Bottom-left: Spectrogram surface using a rectangular window function of size 32 and sliding step of length 8. Bottom-right: Spectrogram surface with window size equal to 16 and sliding step of length 4.

5. We increase the resolution level from l_1 to l_2 , by halving the window size S_w .
6. For each k^* , and for each set of displacements M_i , we calculate the corresponding correlation coefficient using a slightly modified form of Eq. 5.24,

$$f_{k^*}[m] = \frac{1}{\mathcal{A}_w} \sum_{j, j+k^* \in M_i} \hat{\mathbf{s}}_{j,1} \circ \hat{\mathbf{s}}_{j+k^*,2}. \quad (5.30)$$

In this way, we restrict our analysis only to the regions of interest, that is, M_1 , M_2 , M_3 in the example of Fig. 5.11, while most part of the cross-correlation matrix is neglected. In order to guarantee a scale-invariant analysis through the sequential process described here, the cross-correlation function is normalized by the window area \mathcal{A}_w . In the discrete case this yields,

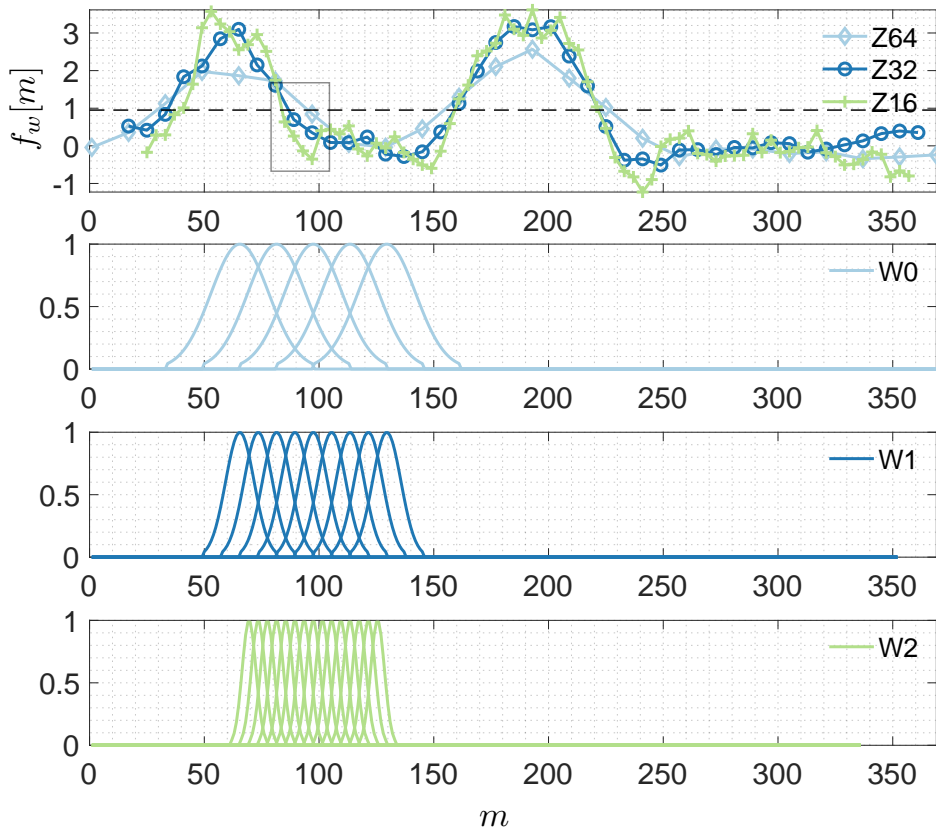


Figure 5.17: The boundaries of the blocks are analyzed at different scales. At the next level, higher resolution, the analysis is performed only in the neighbourhood of the boundary. Top: Slice of the spectrogram $f_w[m, k]$ at k_1^* analyzed with a window of size 64 (Z64), 32 (Z32) and 16 (Z16). The region highlighted in the box represent the boundary of the homologous block.

7. The one-dimensional cross-correlation of equation 5.30 allows us to refine the boundaries of the candidate homologous regions, as shown in Figure 5.17. We thus redefine the regions M_i of point 4 so that they now fulfill the condition $f_{k^*}[m] > t_h$.
8. The algorithm iterates the procedure from point 5 to 7, thus changing resolution level to l_3, l_4, \dots . Typically, a resolution l_3 is enough to achieve a good accuracy in the estimation of the homologous blocks.

5.4.5 Homology matrix and optimal path

The short-time Fourier Transform (see Section 5.4.4: Multi-scale STFT based homology detection) detects candidate homologous blocks which corresponds to regions in the two sub-alignments or sequences with high physicochemical similarity (high cross-correlations). These candidate homologous regions allow us to define a set \mathcal{B} of blocks in a MSA homology matrix analogous to the one introduced in MAFFT [62]. An example is shown in Figure 5.19, which represents a simplified configuration with five candidate homologous blocks. The homologous blocks need to be selected, arranged and connected in a consistent way for designing a meaningful final alignment. Prior to independently align the single blocks an algorithm has to connect them logically in a path that respect the order of the columns and avoid any duplication. This is achieved by first identifying within the set \mathcal{B} the set \mathcal{P} of all possible paths that connect the candidate blocks potentially yielding a final meaningful alignment. For illustrative purpose, Figures 5.25(i)-(iv) depicts four possible resolved paths leading to candidate alignments.

Firstly, one needs to identify the set \mathcal{P} of all possible consistent pathways through those blocks. A generic block $b_j \in \mathcal{B}$ is characterized by the coordinates of its upper-left (u) and bottom-right (v) vertexes, $\{[u_{j,1}, u_{j,2}], [v_{j,1}, v_{j,2}]\}$. The two subscripts 1, 2 refer to sequences S_1 and S_2 . Consider for example another block b_k adjacent to the first one with coordinates $\{[u_{k,1}, u_{k,2}], [v_{k,1}, v_{k,2}]\}$. The condition to connect the two blocks b_j and b_k to form a meaningful path is $(u_{k,1} \geq u_{j,1}) \wedge (u_{k,2} \geq u_{j,2})$. This procedure is iterated for all adjacent blocks finally yielding the sought pattern of possible paths \mathcal{P} . To simplify the management of plausible paths that connect the detected blocks the algorithm build an n th tree \mathcal{T} (see Figure 5.20). The tree nodes represent the blocks and the fulfillment of the aforementioned condition is described by edges connecting nodes. The root of \mathcal{T} has in total $|\mathcal{B}|$ children which is equal to the number

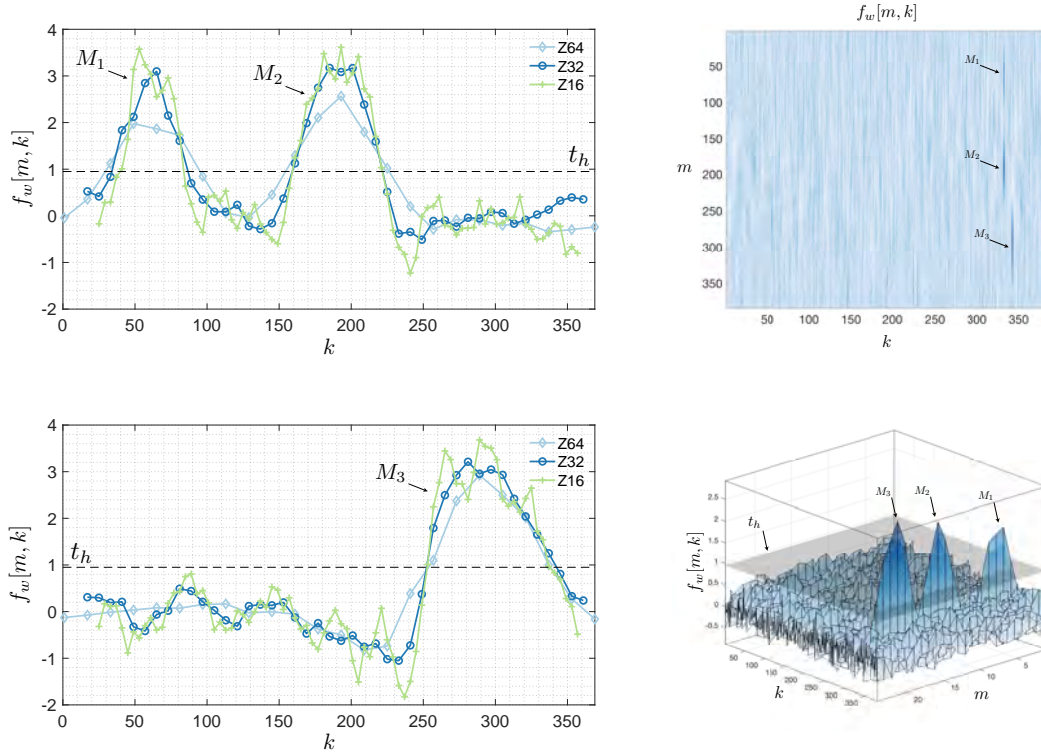


Figure 5.18: Multi-resolution boundary detection. Top right: Spectrogram $f_w[m, k]$ showing 3 candidate patterns, M_1 and M_2 at the positional shift k_1^* and M_3 shifting S_2 by k_2^* positions (on the left direction applying the periodic boundary conditions). Bottom right: Three dimensional view of the spectrogram $f_w[m, k]$. The plane at the z level t_h represents the noise threshold estimated by permuting the characters in S_2 and taking the highest coefficient in the so obtained spectrogram. This procedure is eventually repeated to improve the noise level estimation. Top-bottom left: slice of the spectrogram at position k_1^* and k_2^* . The dashed line represents the noise level t_h and the regions $f_w[m, k] > t_h$ are the candidate homologous pattern in the two signals. The analysis has been performed at three different level: using a window function of size 64 (Z64), 32 (Z32) and 16 (Z16) to improve the boundaries detection.

of blocks detected. This means that each block can be the starting block of a path p and also that a path may be composed by only one block. The total number of paths is given by the number of leaves. In Figure 5.20 for example one can observe 10 consistent connecting paths. Each of the selected paths $p_i \in \mathcal{P}$ will be analyzed independently. It can happen that two distinct blocks belonging to a given path p_i encompass one or more elements (columns) in common. We will refer to this condition as an *overlap*. For example, in Figure

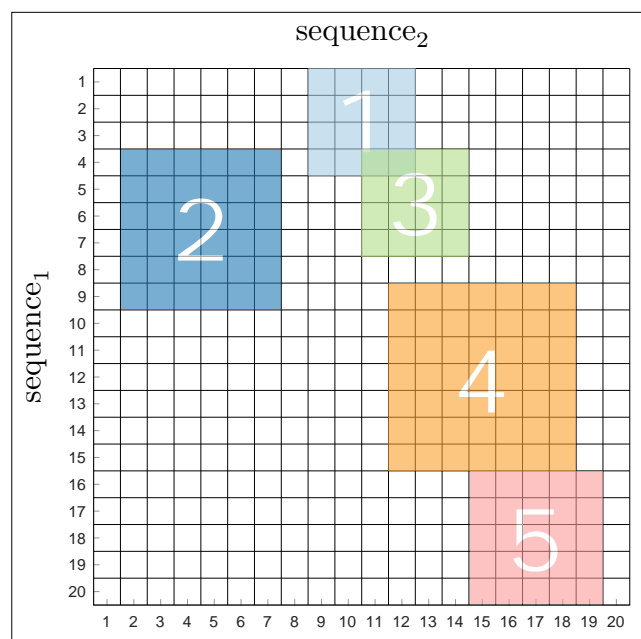


Figure 5.19: Example of overlapping homologous blocks. Blocks are represented in a bi-dimensional dynamic programming matrix where on one axis is placed sequence S_1 and on the other axis sequence S_2 .

5.19, $\text{sequence}_1(4) \diamond \text{sequence}_2(2)$ but also $\text{sequence}_1(4) \diamond \text{sequence}_2(12)$ (where \diamond refers to ‘putatively homologue to’). Since each block will be treated as an independent instance of a DP alignment procedure, then this overlap condition must be removed prior to the alignment phase to avoid to consider $\text{sequence}_1(4)$ twice in the final alignment. As described below, each block will be treated as an independent instance of a DP alignment procedure. For that reason possible overlaps between blocks must be removed prior to their alignment. The removal of the overlaps should alter in the least possible way the structure of homologous regions that was previously identified. Therefore, the overlaps are removed by replacing the overlapping blocks by two re-sized blocks that retain the largest possible part of the diagonal elements of the original ones without exhibiting any more overlaps. The diagonal corresponds to matches in the DP alignment and represents therefore the expected traceback path that we want to preserve as much as possible.

Our procedure allows us to remove overlaps only between two consecutive blocks. In Figures 5.21-5.24 we show some examples of how different types of overlaps can be resolved. More details are provided in Appendix F: [Homologous blocks overlap resolution](#). The reason is that in this way it is much

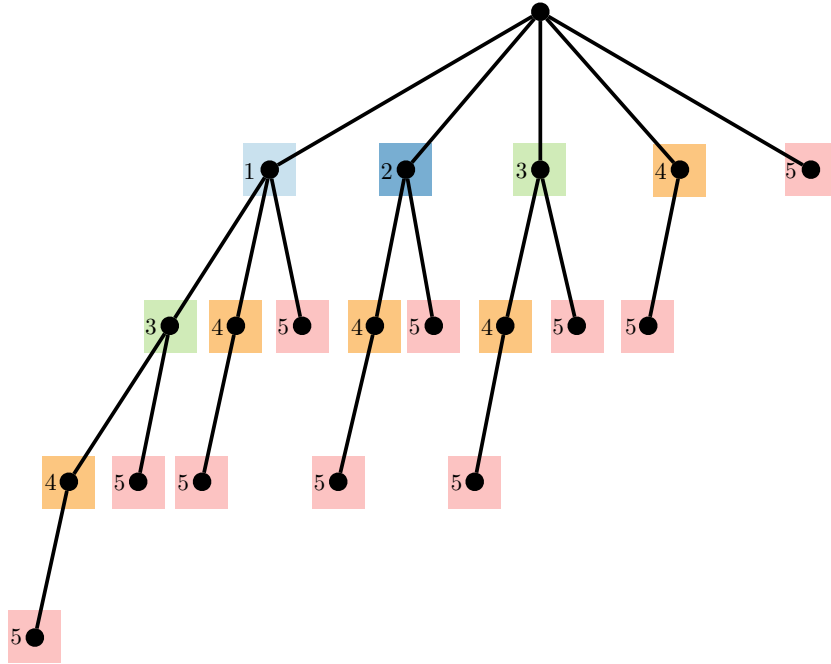


Figure 5.20: Tree of blocks representing possible the paths of the blocks shown in Figure 5.19. If the coordinates of two blocks b_j and b_k fulfill the condition $(u_{k,1} \geq u_{j,1}) \wedge (u_{k,2} \geq u_{j,2})$, then they are connected together to form a path. In the tree this is translated in an edge that connects the nodes representing the blocks. The procedure is iterated for all adjacent blocks finally yielding the sought pattern of possible paths \mathcal{P} , which is represented by a n th tree where the total number of paths is given by the number of leaves. In this example, one can observe 10 consistent connecting paths. Each of the selected paths $p_i \in \mathcal{P}$ will be analyzed independently.

simplier to generalize the procedure into an algorithm that works for any possible configuration at hand. An algorithm that solve simultaneously all the overlaps in a given path would be much more complicated to design, to debug and to test. To resolve the different types of overlaps they have to be first categorized. For this purpose, the algorithm checks the relative position of the corner coordinates $u_{1,2}$ and $v_{1,2}$ and with the aid of the table depicted in Figure F.1 assigns a type of overlap and its removal strategy.

Therefore given the set \mathcal{P} the algorithm iterates in each path p_i and for for all pairs of consecutive blocks in this path check and if needed resolve the overlaps, until all overlaps are removed. It should be noted that, as clearly shown in the tree of Figure 5.20, a given block can appear in multiple paths and in each of them it can present different overlaps with other blocks. Our

algorithm therefore modifies its structure in different ways in each of the paths where the block appears, depending on the type of overlap that needs to be removed. The algorithm generates in this way the set $\tilde{\mathcal{P}}$ of overlap-free paths (see Figure 5.25) in which we choose the optimal path p^* that maximizes a given target condition. From Figure 5.25: the path represented in *i*) connects blocks $\{b_1, b_4, b_5\}$ for a total of 6 columns; the path in *ii*) connects $\{b_1, b_3, b_5\}$ for a total of 9 columns; the path shown in *iii*) connects $\{b_2, b_4, b_5\}$ for a total of 12 columns and finally the path in *iv*) $\{b_3, b_5\}$ for a total of 9 columns. In our implementation of the algorithm the target is the path with the largest homologous regions (total sum of columns in a given path), therefore the path in *iii*) results optimal. We denote the final set of homologous blocks defining the optimal path p^* with \mathcal{H} . Finally, among the overlap-free paths $\tilde{\mathcal{P}}$ we select the optimal path p^* bearing the largest homologous regions, that is, the largest total number of columns.

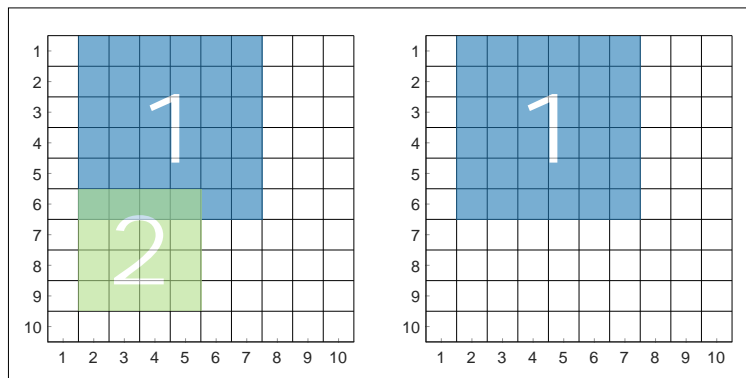


Figure 5.21: *i*) Example of overlapping blocks (left) and their resolution (right).

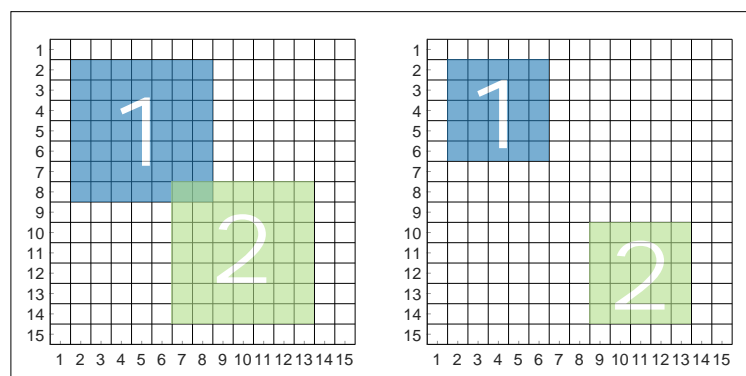


Figure 5.22: *ii*) Example of overlapping blocks (left) and their resolution (right).

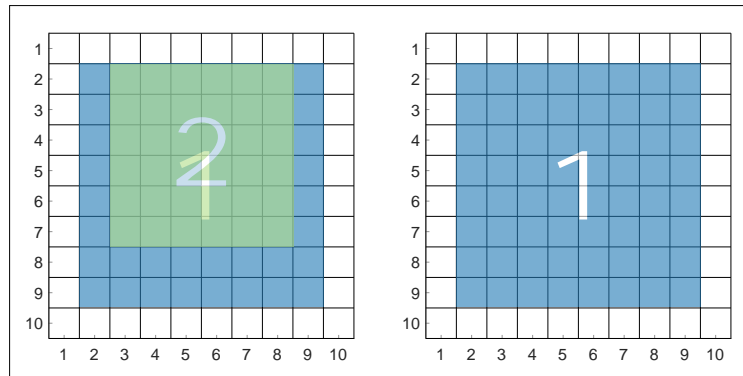


Figure 5.23: iii) Example of overlapping blocks (left) and their resolution (right).

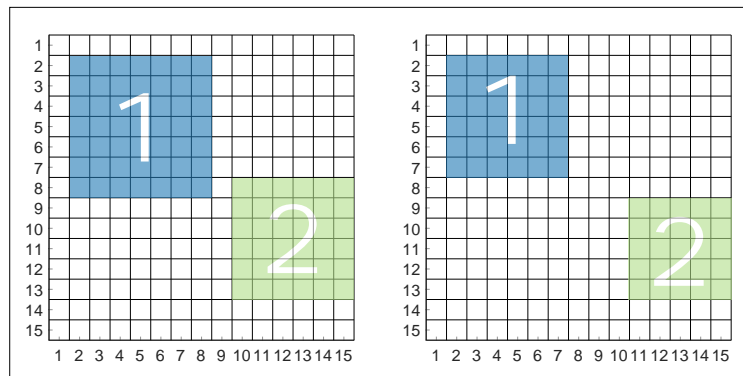


Figure 5.24: iv) Example of overlapping blocks (left) and their resolution (right).

Homologous blocks belonging to the optimal path p^* are connected by appropriate intermediate linking blocks. In Figure 5.28 *homologous blocks* are represented in light blue while *linking blocks* in dark blue. Homologous and linking blocks are then all aligned independently following a DP approach, and the resulting tracebacks are joined together to yield the final alignment. It should be noted that in the homology matrix only the areas corresponding to the homologous and linking blocks of the previously selected optimal path p^* are retained, while the rest of the matrix is excluded from the calculations. This leads to a sparse matrix layout with important effects on the efficiency of the method especially in the framework of a 3-dimensional approach such as the one described in *Maiolo et al. 2018* [88].

The next step consists in aligning the homologous blocks in p^* as independent DP sub-matrices. The alignment of the homologous blocks, carried out under the PIP scoring system, leads to a traceback path for each block. The

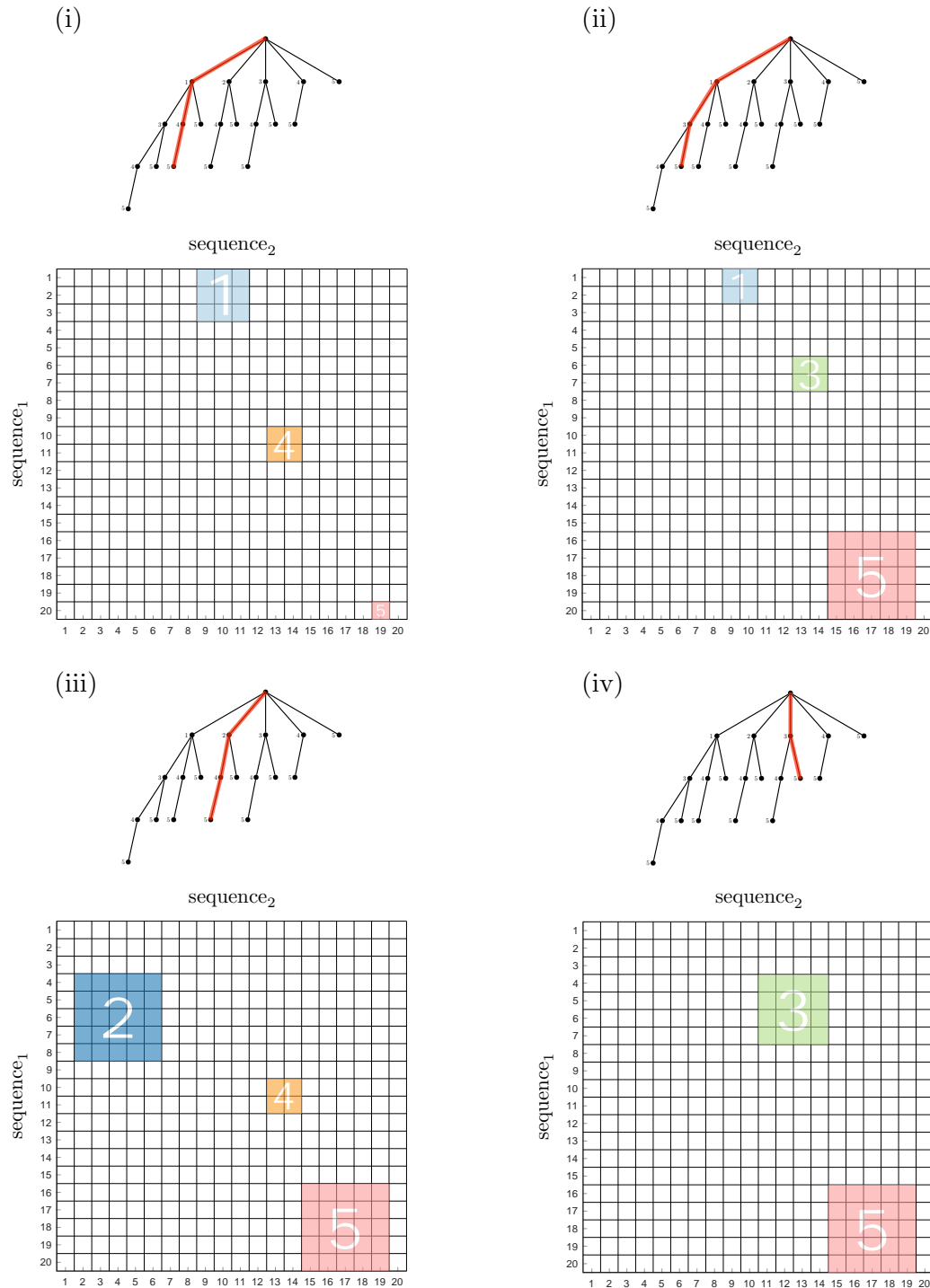


Figure 5.25: *Overlap-free paths. The four panels show overlap-free paths, both on the tree (in red) and the relative resolved blocks. The path represented in i) connects blocks $\{b_1, b_4, b_5\}$ for a total of 6 columns; the path in ii) connects $\{b_1, b_3, b_5\}$ for a total of 9 columns; the path shown in iii) connects $\{b_2, b_4, b_5\}$ for a total of 12 columns and finally the path in iv) $\{b_3, b_5\}$ for a total of 9 columns. In our implementation of the algorithm the target is the path with the largest homologous regions (total sum of columns in a given path), therefore the path in iii) results optimal.*



Figure
blocks.
sented
in dari

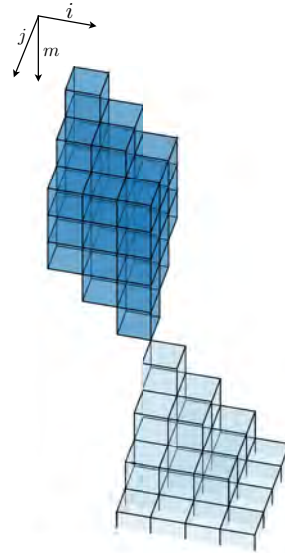


Figure 5.27: Magnification of connecting corner. Homologous and linking blocks are connected together at the corner.

Figure 5.28: Homologous and linking blocks and magnification of connecting corner. All blocks are aligned independently, first all the homologous block and afterwards all the linking ones. Once both homologous and linking blocks are aligned, all the independent tracebacks are merged to give the final alignment.

blocks are then resized so that the first and the last matches (going from left to right) of the previously computed tracebacks correspond to the top left and right bottom corners, respectively. Obviously, if the corners of a block are both matches, the block maintains its original size. Homologous blocks can then be connected by appropriate *linking blocks*, which will in turn be aligned as independent DP sub-matrices. This step leads again to a traceback path for each of the linking blocks. Resizing the homologous blocks so that they begin and end with matches has the advantage that the transitions between homologous and linking blocks then occur in a natural way along the diagonal, without the need for forcing any particular state in the final alignment. Since both homologous and linking blocks are treated as independent DP matrices, their alignments can be easily parallelized. Once both homologous and linking blocks are aligned, all the independent tracebacks are merged to give the final alignment.

5.4.6 Final alignment

The next step consists in aligning the homologous blocks in \mathcal{H} as independent DP sub-matrices. The alignment is carried out under the PIP scoring system and leads to a traceback path for each block. The blocks are then re-sized so that the first and the last matches (going from left to right) of the previously computed tracebacks correspond to the top left and bottom right corners, respectively. Obviously, if the corners of a block are both matches, the block maintains its original size. Consecutive disjoint homologous blocks, e.g., $h_j, h_k \in \mathcal{H}$, are then connected by appropriate linking blocks, for instance, $l_j \in \mathcal{L}$ with vertexes $\{[v_{j,1} + 1, v_{j,2} + 1], [u_{k,1} - 1, u_{k,2} - 1]\}$ (see Figure 5.28).

The linking blocks in \mathcal{L} will in turn be aligned as independent DP sub-matrices. This step leads again to a traceback path for each of the linking blocks. Re-sizing the homologous blocks so that they begin and end with matches has the advantage that the transitions between homologous and linking blocks then occur in a natural way along the diagonal, without the need for forcing any particular state in the final alignment.

Since both homologous and linking blocks are treated as independent DP matrices, their alignments can be easily parallelized. Homologous blocks in \mathcal{H} are first aligned in parallel and re-sized following the recipe outlined above. Afterwards, the alignment of the linking blocks \mathcal{L} can also be run in parallel. Once both homologous and linking blocks are aligned, all the independent tracebacks are merged to give the final alignment. Figure 5.29 shows an alignment under PIP obtained with and without the STFT based homologous detection. The two alignments agree with each other in the inferred homology of the conserved regions and to a large extent they agree also in the gappy regions. The MSA obtained with the STFT algorithm is two columns longer than the one obtained using the classic DP-PIP approach.

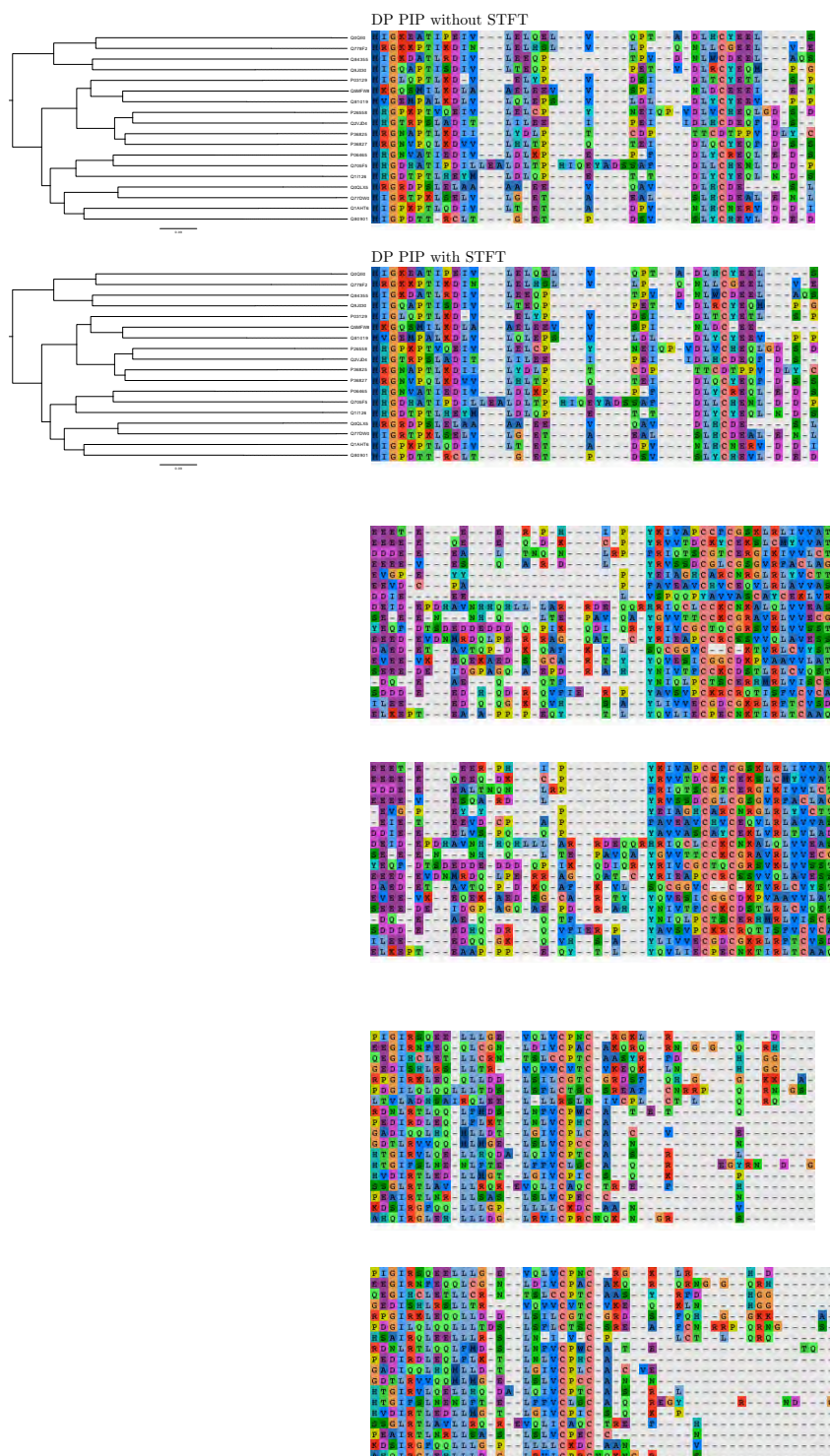


Figure 5.29: Multiple sequence alignment under PIP with and without the block detection based on the short-time Fourier transform. The two alignments agree with each other in the inferred homology of the conserved regions and to a large extent they agree also in the gappy regions. The MSA obtained with the STFT algorithm is two columns longer than the one obtained using the classic DP-PIP approach.

5.5 STFT vs. FT approach for block detection

We performed two tests to assess which from the two approaches, namely the Fourier transform and the short-time Fourier transform, is more effective in detecting a similar pattern within two signals. In the first test, in Section 5.5.1:Noise sensitivity, we have measured the performance in presence of a noisy pattern while in the second test, in Section 5.5.2:Pattern length sensitivity, we have quantified the ability of detecting a relative short pattern. In both tests we have evaluated the two approaches with respect to the signal to noise ratio, calculated as the cross-correlation value of the peak, at the known positional shift divided by the inferred noise threshold. The signal to noise ratio gives an indication about the ability to distinguish peaks raised by high correlated regions from the noise coefficients. In both tests the STFT performed better than the FT which is supporting our novel homologous detection approach.

5.5.1 Noise sensitivity

In this experiment we have synthesized two amino acid signals both of length 160 residues. The first signal contains a pattern of length 100 residues which has been corrupted with noise by changing at random positions and with random residues its content. After the noise addition, we have measured the number of equal residues in the two patterns which has been indicated in the plot in Figure 5.31 as noise percent. We have repeated this experiment with an increasing level of noise. For each new synthesized pair of signals we have computed the cross-correlation coefficients both with the FT and the STFT approach. In Figure 5.31 we are showing the signal to noise ratios obtained with the two methods as function of noise content. It can be noted that the STFT is more effective in finding noisy patterns than the FT. This is partic-

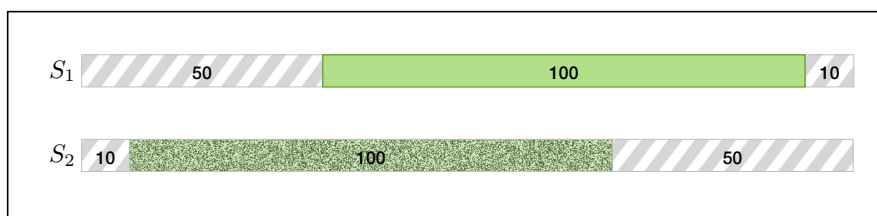


Figure 5.30: Noisy pattern experiment. In S_2 the pattern is corrupted with increasing level of noise.

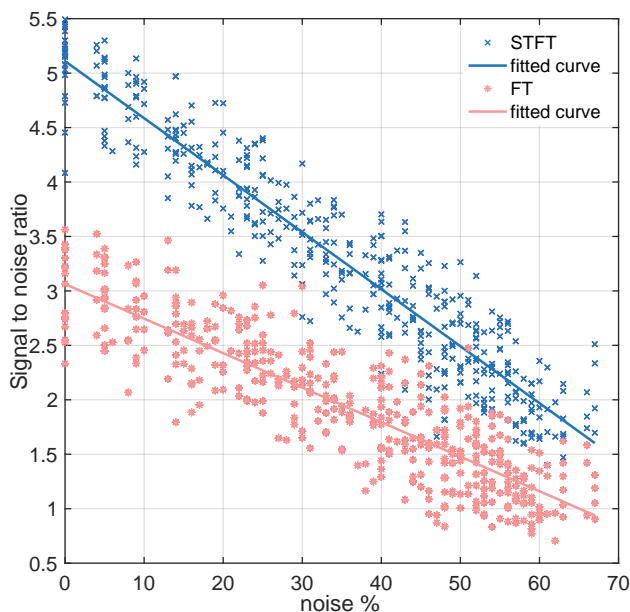


Figure 5.31: Noise sensitivity. Signal to noise ratio measurement of the two approaches in presence of noisy patterns which is mimicking the substitution process.

5.5.2 Pattern length sensitivity

In this experiment we have synthesized two amino acid signals of increasing length. The structure of the signals is depicted in Figure 5.32. The pattern has been incrementally extended from a length of 1 to a final length of 100 residues. For each pattern length we have computed the cross-correlation coefficients both with the FT and the STFT approach. In Figure 5.33 we are showing the signal to noise ratio obtained with the two methods as function of pattern

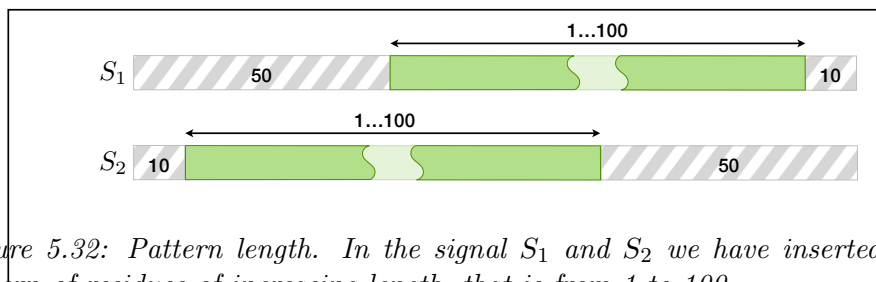


Figure 5.32: Pattern length. In the signal S_1 and S_2 we have inserted the same pattern of residues of increasing length, that is from 1 to 100.

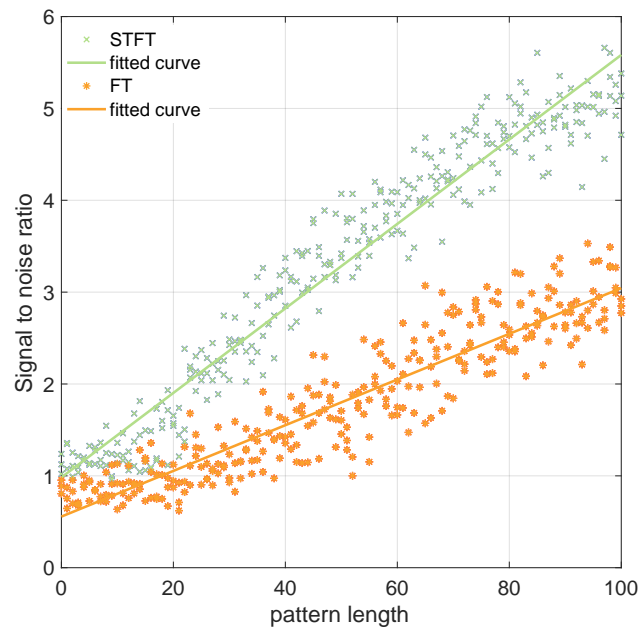


Figure 5.33: Pattern length sensitivity. Signal to noise ratio measurement of the two approaches for a variable pattern size.

6

Progressive bias analysis

“We are trying to prove ourselves wrong as quickly as possible, because only in that way can we find progress.”

– Richard P. Feynman

6.1 Introduction

Progressive approaches reduce the exponential complexity of aligning N sequences ($N > 2$) into polynomial-time algorithms by traversing a given binary phylogenetic tree from bottom towards the root and aligning at each internal node only two sub-alignments. Hence, the progressive aligner explores at each visited node only the space spanned by the two children’s sub-alignments. In this way the alignment space where to search for the optimal alignment is considerably reduced together with time complexity. At the tree root the algorithm returns the optimal alignment gradually built – an constraints on – the optimal pairwise alignments produced at each intermediate step of the procedure. Although progressive approaches strongly accelerate the generation of the final solution they potentially introduce biases at each iteration. The reason of this bias lies, on one hand in the fact that the information available at each internal node is deliberately not complete (smaller search space), and on the other hand because the current alignment is constrained on the previously computed partial solutions. Unfortunately though, an error introduced at an early stage of the tree traversal is propagated upwards into the next stages (at higher levels of the tree) deviating irreparably from the true optimal solution and providing potentially distorted homologous scenarios.

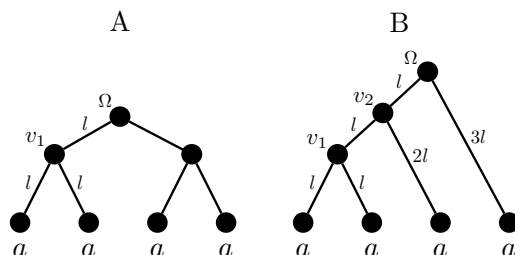


Figure 6.1: Topologies used to quantify the progressive bias. A) balanced tree B) unbalanced tree. In both trees the branch lengths are parameterized by l . The sequences at the leaves contain only a single character represented by a generic symbol ‘a’.

To quantify the bias introduced by the progressive procedure, called *progressive bias*, we have used two different topologies, namely *balanced* and *unbalanced* tree, both with branch lengths parameterized by l (see Figure 6.1). The progressive bias has been evaluated for both topologies by computing the marginal likelihood at the root for all possible alignments potentially be generated compared against the solution selected at each step by the progressive procedure. To facilitate the problem, but not the general conclusions we can draw from it, the sequences at the leaves contain only a single character represented by a generic symbol ‘a’. Simplifying the problem in this manner, allows us to compare the true optimal alignment, as function of the parameter l , with the partial solutions built at each stage of the progressive procedure. Indeed, one can check whether the greedy sub-alignment is actually part of the optimal MSA. From the results shown below, it can be noted that for certain values of the parameter l the solutions generated by the two approaches are different. Therefore, combining progressively optimal pairwise alignments is different from the optimal MSA that one would obtain by aligning all the sequences at the same time.

Test parameters and MSAs. For our test we have used a parameterized branch lengths with $l = 0.16, \dots, 0.4$ (expressed in expected number of substitutions per site). The substitution model used is JC69 [60] for a nucleotide alphabet. We have set $\lambda = 10\sqrt{2}$ and $\mu = \sqrt{2}$ which corresponds to expected sequence length $\mathbb{E} = \lambda/\mu = 10$ and intensity $I = \lambda\mu = 20$. In Section 6.2 we have compared the solutions using the unbalanced tree of Figure 6.1B and in Section 6.3 using the balanced tree of Figure 6.1A. In Table 6.1 we have listed all the MSAs potentially be generated at node v_1 starting from two sequences containing only a single character, represented here with a symbol ‘a’. In Table

6.2 we have included all the alignments generated with 3 sequences at v_2 and in Table 6.3 the alignments potentially obtained at the root node.

$\begin{bmatrix} a \\ a \end{bmatrix}$	$\begin{bmatrix} a & - \\ - & a \end{bmatrix}$
#1	#2

Table 6.1: MSAs potentially be generated at node v_1 .

$\begin{bmatrix} a \\ a \\ a \end{bmatrix}$	$\begin{bmatrix} a & - \\ a & - \\ - & a \end{bmatrix}$	$\begin{bmatrix} a & - \\ - & a \\ a & - \end{bmatrix}$	$\begin{bmatrix} a & - \\ - & a \\ - & a \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ - & a & - \\ - & - & a \end{bmatrix}$
#3	#4	#5	#6	#7

Table 6.2: MSAs potentially be generated at node v_2 .

6.2 Alignment with unbalanced tree

In this section we have analyzed the bias introduced during the progressive alignment using the unbalanced tree of Figure 6.1B. We first have computed in Section 6.2.1 the true optimal solution by considering all potential alignments be generated with four sequences comprehending only a single character. Then, in Section 6.2.2, we have generated the alignments from 2 and successively from 3 sequences as performed by the progressive aligner. At each step we have compared the solution that the DP algorithm would have selected, which consists in the maximum likelihood pairwise MSA, as function of the branch length, and finally we have checked whether the *progressive* optimal solution is actually contained in the true *global* optimal.

6.2.1 Global alignment

To analyse the bias introduced by the progressive approach we have computed the marginal likelihood of all the alignments listed in Table 6.3, which illustrates all the alignments potentially be produced with 4 sequences containing only a single character each. In this computation there is no approximation or bias introduced by the progressive approach. We refer this approach

$\begin{bmatrix} a \\ a \\ a \\ a \end{bmatrix}$	$\begin{bmatrix} a & - \\ a & - \\ a & - \\ - & a \end{bmatrix}$	$\begin{bmatrix} a & - \\ a & - \\ - & a \\ - & a \end{bmatrix}$	$\begin{bmatrix} a & - \\ - & a \\ - & a \\ - & a \end{bmatrix}$	$\begin{bmatrix} a & - \\ - & a \\ a & - \\ a & - \end{bmatrix}$
#8	#9	#10	#11	#12
$\begin{bmatrix} a & - \\ - & a \\ a & - \\ - & a \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ - & a & - \\ - & - & a \\ - & - & a \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ - & a & - \\ - & a & - \\ - & - & a \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ - & a & - \\ - & - & a \\ - & a & - \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ - & a & - \\ a & - & - \\ - & - & a \end{bmatrix}$
#13	#14	#15	#16	#17
$\begin{bmatrix} a & - \\ a & - \\ - & a \\ a & - \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ - & a & - \\ - & - & a \\ a & - & - \end{bmatrix}$	$\begin{bmatrix} a & - \\ - & a \\ - & a \\ a & - \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ a & - & - \\ - & a & - \\ - & - & a \end{bmatrix}$	$\begin{bmatrix} a & - & - \\ - & - & a \\ - & - & a \\ - & a & - \end{bmatrix}$
#18	#19	#20	#21	#22
$\begin{bmatrix} - & a & - \\ - & - & a \\ - & - & a \\ a & - & - \end{bmatrix}$	$\begin{bmatrix} a & - & - & - \\ - & a & - & - \\ - & - & a & - \\ - & - & - & a \end{bmatrix}$			
#23	#24			

Table 6.3: MSAs potentially be generated at the root Ω .

as the *global* alignment method since it explore the entire alignment space. For each alignment in the table we have computed the marginal likelihood under PIP as function of the parameter l , the results are plotted in Figure 6.2. From this figure one can notice that for $l < 0.125$ the multiple sequence alignment having highest marginal likelihood is $\text{MSA}_{\#8} = [a \ a \ a \ a]^T$. With $0.125 < l < 0.19$ the *global* optimal alignment is $\text{MSA}_{\#18} = \begin{bmatrix} a & a & - & a \\ - & - & a & - \end{bmatrix}^T$. With $0.19 < l < 0.325$ the best alignment is $\text{MSA}_{\#21} = \begin{bmatrix} a & a & - & - \\ - & - & a & - \\ - & - & - & a \end{bmatrix}^T$ and finally for $l > 0.325$ $\text{MSA}_{\#24} = \begin{bmatrix} a & - & - & - \\ - & a & - & - \\ - & - & a & - \\ - & - & - & a \end{bmatrix}$ is the *global* optimal alignment.

From the *global* optimal alignments¹ at the root node it is possible to assess

¹Here the alignments are function of the parameterized branch length l . Since the

whether an alignment being generated at a lower level in the tree is part of the final solution and for which value of the parameter l a given choice is made. The partial alignments potentially be produced at node v_1 and v_2 are represented in Table 6.1 and Table 6.2 respectively. The differences comparing the solutions are shown in Figure 6.2. The differences comparing the solutions are shown in Figure 6.2.

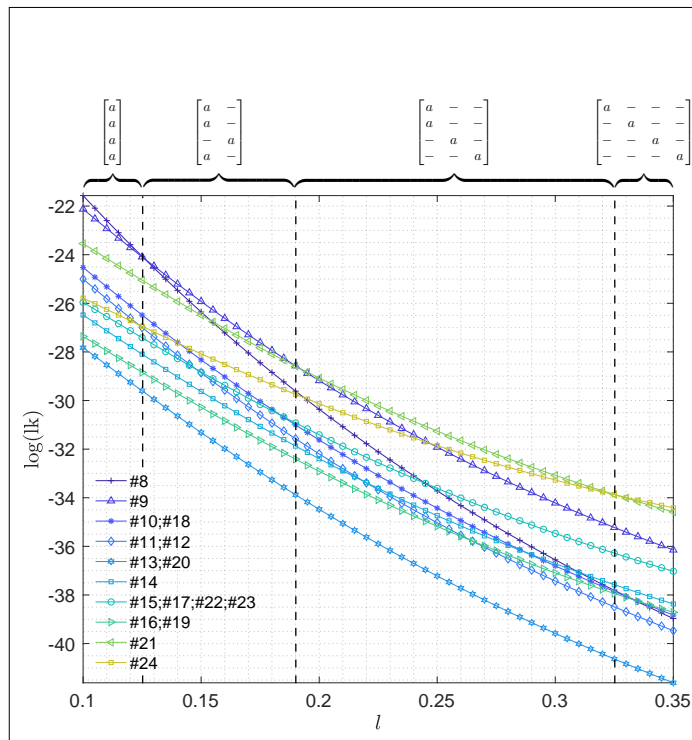


Figure 6.2: Marginal likelihood curves computed at the root node Ω of the alignments depicted in Table 6.3 as function of l . The curves refer to the ‘global’ approach, no ‘progressive’ approximation is introduced and at the root the information is entire available. The alignments having highest likelihood in the different regions are represented on top of the figure. For $l < 0.125$ the optimal alignment is MSA_{#8}, for $0.125 < l < 0.19$ MSA_{#18}, for $0.19 < l < 0.325$ MSA_{#21} and for $l > 0.325$ MSA_{#24}.

6.2.2 Progressive alignment

At each internal node, namely v_1 and v_2 , we have computed the likelihood of all possible alignments potentially be generated at the respective nodes where at the leaves are associated sequences of only a single character. The possible pairwise MSAs potentially be generated at node v_1 are shown in Table 6.1

marginal likelihood depends on l , the optimal MSA changes correspondingly.

and their corresponding likelihood, as function of l , are plotted in Figure 6.4. The alignments attainable at node v_2 are represented in Table 6.2 and their likelihood as function of the parameter l is plotted in Figure 6.3.

6.2.3 Progressive bias with unbalanced tree

Bias at node v_2

At node v_2 the optimal alignments are: for $l < 0.17$ $\text{MSA}_{\#3} = [\text{a a a}]^\top$, for $0.17 < l < 0.29$ $\text{MSA}_{\#4} = \begin{bmatrix} \text{a} & \text{a} & - \\ - & - & \text{a} \end{bmatrix}^\top$ and for $l > 0.29$ $\text{MSA}_{\#7} = \begin{bmatrix} \text{a} & - & - \\ - & \text{a} & - \\ - & - & \text{a} \end{bmatrix}^\top$. The boundaries at which a change of optimal MSA appear are shown in Figure 6.3 with a dotted line ‘.....’. In order to generate at the root node the MSA column $c(\Omega) = [\text{a a a a}]^\top$ it is necessary at node v_2 to obtain an MSA column $c(v_2) = [\text{a a a}]^\top$. This is the case, at v_2 , for $l < 0.17$ but considering the global alignment this boundary should be instead at $l < 0.125$ (boundary represented in Figure 6.3 with a dashed line ‘----’). This means that for $0.125 < l < 0.17$ the algorithm produces an MSA column that, at the next layer, will constraint the algorithm to a sub-optimal alignment. In the region $0.17 < l < 0.29$ the algorithm generates at node v_2 the alignment $\text{MSA} = \begin{bmatrix} \text{a} & \text{a} & - \\ - & - & \text{a} \end{bmatrix}^\top$ which is the correct basis to obtain at the root both the alignments $\text{MSA} = \begin{bmatrix} \text{a} & \text{a} & - & \text{a} \end{bmatrix}^\top$ and $\text{MSA} = \begin{bmatrix} \text{a} & \text{a} & - & - \\ - & - & \text{a} & - \\ - & - & - & \text{a} \end{bmatrix}^\top$. To generate at the root the alignment $\text{MSA} = \begin{bmatrix} \text{a} & - & - & - \\ - & \text{a} & - & - \\ - & - & \text{a} & - \\ - & - & - & \text{a} \end{bmatrix}^\top$ is mandatory to generate at node v_2 the alignment $\text{MSA} = \begin{bmatrix} \text{a} & - & - \\ - & \text{a} & - \\ - & - & \text{a} \end{bmatrix}^\top$. This alignment is generated at node v_2 starting from $l > 0.29$ rather than for $l > 0.325$ as demanded by the global optimum alignment. Therefore for $0.29 < l < 0.325$ the algorithm produce at v_2 an alignment that will force the algorithm to a sub-optimal MSA at the root. To summarize: at short l the algorithm tends to “over-align” whereas for long branch length l the algorithm loses some homologies.

Bias at node v_1

At node v_1 the DP produces the alignment $\text{MSA}_{\#1} = \begin{bmatrix} \text{a} \\ \text{a} \end{bmatrix}$ for $l < 0.255$ and $\text{MSA}_{\#2} = \begin{bmatrix} \text{a} & - \\ - & \text{a} \end{bmatrix}$ for $l > 0.255$. Clearly, under an i.i.d.model the likelihood of an alignment is not affected by the permutation of the column order, hence an alignment like $\text{MSA} = \begin{bmatrix} - & \text{a} \\ \text{a} & - \end{bmatrix}$ has the same marginal likelihood of $\text{MSA} = \begin{bmatrix} \text{a} & - \\ - & \text{a} \end{bmatrix}$. In order to produce at the root node either $[\text{a a a a}]^\top$, $[\text{a a} \text{ - a}]^\top$ or $\begin{bmatrix} \text{a} & \text{a} & - & - \\ - & - & \text{a} & - \\ - & - & - & \text{a} \end{bmatrix}^\top$

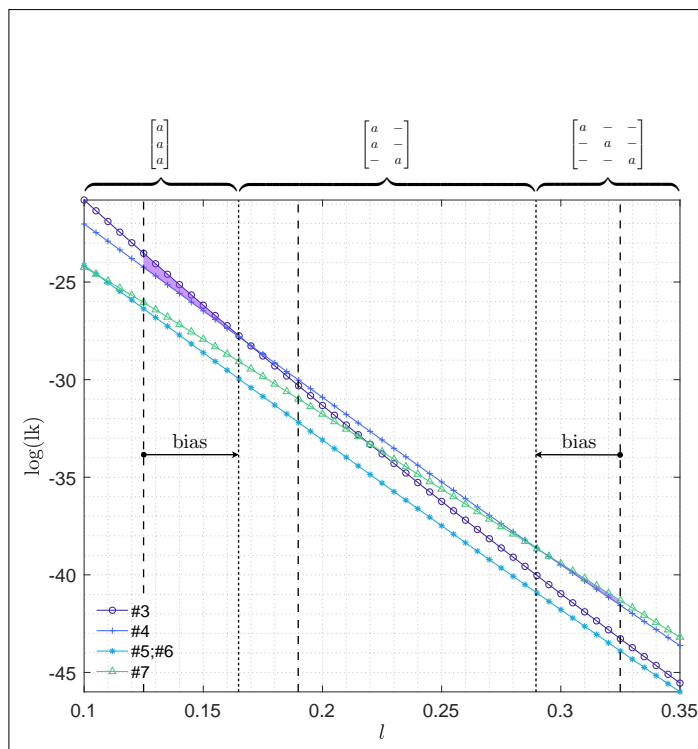


Figure 6.3: Marginal likelihood curves at the root node v_2 of the alignments depicted in Table 6.1 as function of l . The curves refer to the ‘progressive’ approximation. The alignments having highest likelihood in the different regions are represented on top of the figure. For $l < 0.255$ the optimal alignment is $MSA_{\#4}$, and for $l > 0.255$ $MSA_{\#7}$.

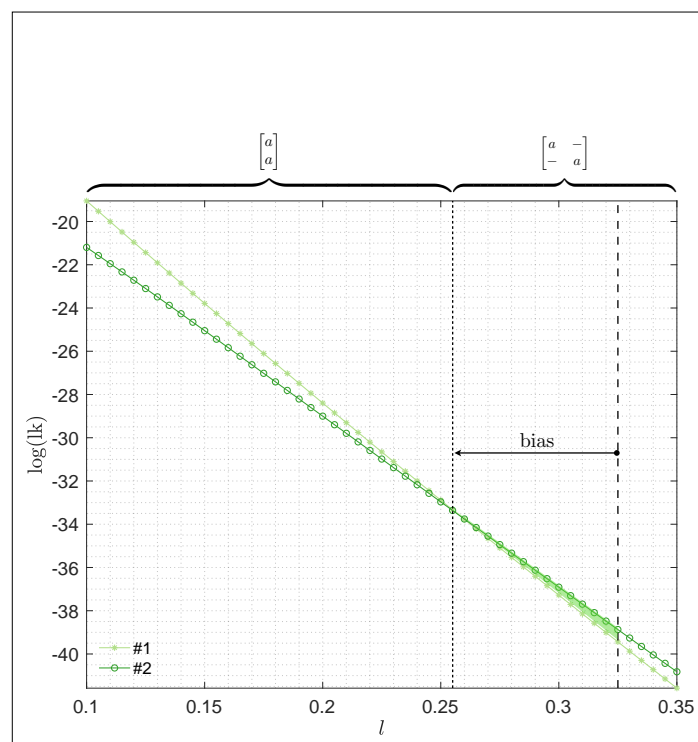


Figure 6.4: Marginal likelihood curves computed at the root node v_1 of the alignments depicted in Table 6.1 as function of l . The curves refer to the ‘progressive’ approximation. The alignments having highest likelihood in the different regions are represented on top of the figure. For $l < 0.255$ the optimal alignment is $MSA_{\#1}$, and for $l > 0.255$ $MSA_{\#2}$.

the algorithm, for $l < 0.325$, shall generate at v_1 an alignment $\text{MSA} = \begin{bmatrix} a \\ a \end{bmatrix}$. The progressive procedure already for $l > 0.255$ produces $\begin{bmatrix} a \\ a \end{bmatrix}$ and therefore in the region $0.255 < l < 0.325$ is biased. The MSA $\begin{bmatrix} a & - \\ - & a \end{bmatrix}$ is the basis to produce at v_2 the MSA $\begin{bmatrix} a & - & - \\ - & a & - \\ - & - & a \end{bmatrix}$ and at the root the MSA $\begin{bmatrix} a & - & - & - \\ - & a & - & - \\ - & - & a & - \\ - & - & - & a \end{bmatrix}$. This alignment is produced at v_1 for $l > 0.325$ instead then for $l < 0.255$ and at v_2 in the range $0.17 < l < 0.29$ instead than in the range $0.125 < l < 0.325$.

6.3 Alignment with balanced tree

6.3.1 Global alignment

For $l < 0.24$ the maximum likelihood alignment is $\text{MSA}_{\#8} = [a \ a \ a \ a]^\top$. For $0.24 < l < 0.285$ the maximum likelihood alignments are $\text{MSA}_{\#10} = \begin{bmatrix} a & a & - & - \\ - & - & a & a \end{bmatrix}^\top$ and $\text{MSA}_{\#18} = \begin{bmatrix} a & a & - & a \\ - & - & a & - \end{bmatrix}^\top$. For $0.285 < l < 0.325$ the alignments with highest likelihood are $\text{MSA}_{\#14} = \begin{bmatrix} a & - & - & - \\ - & a & - & - \\ - & - & a & a \end{bmatrix}^\top$ and $\text{MSA}_{\#21} = \begin{bmatrix} a & a & - & - \\ - & - & a & - \\ - & - & - & a \end{bmatrix}^\top$ and for $l > 0.325$ the alignment with highest likelihood is $\text{MSA}_{\#24} = \begin{bmatrix} a & - & - & - \\ - & a & - & - \\ - & - & a & - \\ - & - & - & a \end{bmatrix}^\top$. At $l = 0.24$, $l = 0.285$ and $l = 0.325$ a random choice is made to select between the MSAs with exact same likelihood.

6.3.2 Progressive alignment

The DP generates the maximum likelihood pairwise alignment conditioned on the sub-MSAs at the children nodes. That means that the algorithm selects the highest pairwise MSA that can be generated on the base of the available input. At node v_1 (and in the same way at its sister node on the right sub-tree) the progressive DP algorithm generates either on both sides $\text{MSA}_{\#1} = [a \ a]^\top$ when $l < 0.26$ and $\text{MSA}_{\#2} = \begin{bmatrix} a & - \\ - & a \end{bmatrix}^\top$ when $l > 0.26$.

At the root Ω , in the range $0.24 < l < 0.26$, the progressive algorithm is able to generate only $\text{MSA}_{\#10}$ but not $\text{MSA}_{\#18}$. Only for $l = 0.26$ the two MSAs, namely $\text{MSA}_{\#1}$ and $\text{MSA}_{\#2}$, have the same probability to be produced and therefore $\text{MSA}_{\#18}$ could be eventually be produced. When $l > 0.26$ the progressive DP generates on both children nodes of the root the alignment $\text{MSA}_{\#2}$. That translates on the fact that at the root $\text{MSA}_{\#14}$ and $\text{MSA}_{\#21}$ cannot be generated starting from two alignments of type #2.

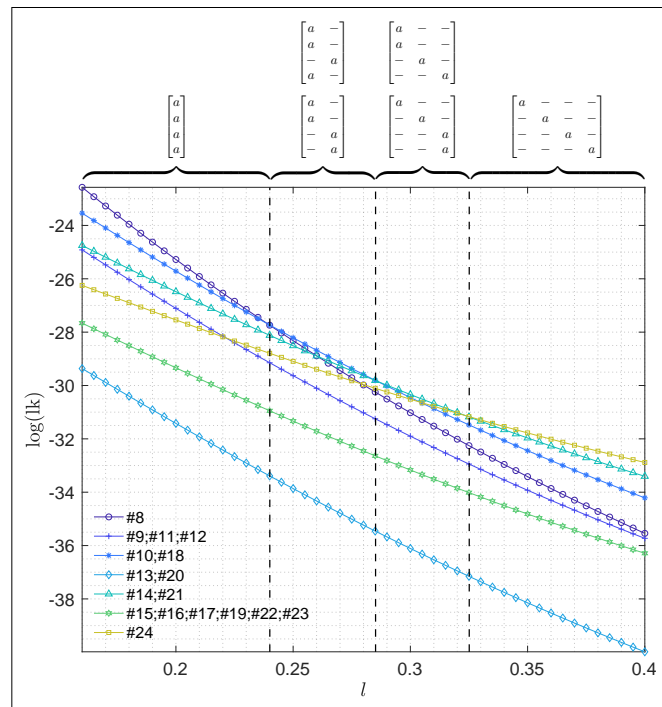


Figure 6.5: Marginal likelihood of the alignments depicted in Table 6.1 as function of l computed at node v_1 . The alignment having highest likelihood is represented on top of the figure. At $l = 0.26$ there is a crossover of the two marginal likelihood functions. Before that point the DP produces always $MSA_{\#1}$ as solution, after that point it generates alignment $MSA_{\#2}$.

At the root, the curve with the highest likelihood, after having discarded all the alignments that cannot be produced at that node, is the MSA relative to the curve #24. In Figure 6.7 the region highlighted in yellow represents the region where instead of taking the curve corresponding to the highest likelihood the algorithm selects the highest achievable being conditioned on the two sub-alignments present at the children nodes.

The solution generated at v_1 is the same as the one generated at the right sub-tree rooted at the root since the branch lengths are all equal.

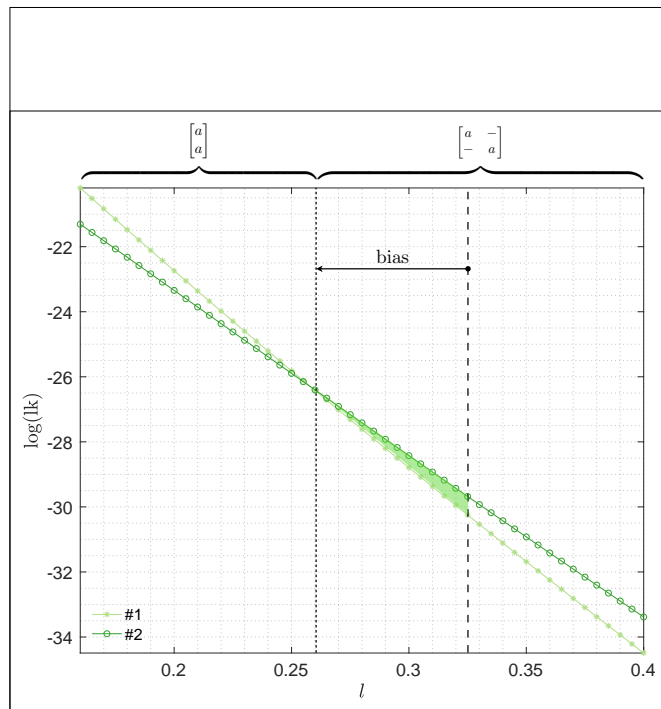


Figure 6. Marginal likelihood curves of l computed with a progressive approach (dotted line) and with a full search (dashed line).

Figure 6. Marginal likelihood curves of l computed with a progressive approach (dotted line) and with a full search (dashed line).

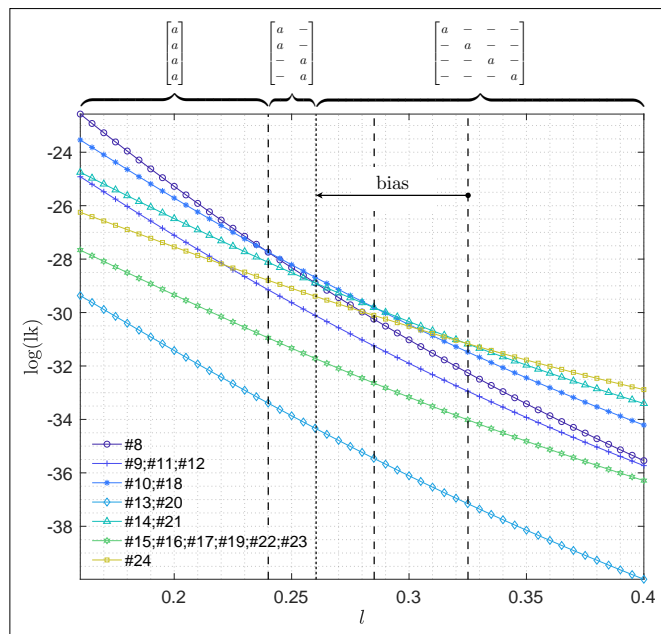


Figure 6.7: Top left: Marginal likelihood curves obtained at node v_1 in a progressive framework. Bottom left: Marginal likelihood curves of all possible alignments. Bottom right: The progressive approach is conditioning on the sub-MSAs at the children nodes and sometimes it produces sub-optimal solutions. Since our algorithm is a progressive approach, at the root node it aligns by ML the pairwise alignments present at the children nodes. There are potential alignments that have better likelihood than the one that the algorithm can produce but are not achievable. The region highlighted in yellow depicts the region where the algorithm is constrained on sub-optimal solutions.

Discussion & Conclusions

“We are just an advanced breed of monkeys on a minor planet of a very average star. But we can understand the Universe. That makes us something very special.”

– Stephen Hawking

Discussion

The fast improvement of modern sequencing platforms has contributed in the population of several protein families databases. Though, the abundance of sequenced genomes made available by these modern sequencing technologies constitute an incredible challenge for multiple sequence alignment algorithms. The inferred multiple sequence alignments are thereafter widely employed for an extensive range of purposes including, to name a few, the estimation of divergence patterns, for selection studies, for the analysis of the evolutionary changes, to identify key functional residues, to detect conserved regions and for building phylogenies. The reconstruction of evolutionary multiple sequence alignments is an essential, and therefore required, step for most – though not all – homology-based sequence analyses. Even though homology is an unobservable property shared by related sequences, it can be inferred by means of statistical tools exploiting the similarities between them under suitable conditions [52]. The homology inference started in 1981 with the contribution of Smith-Waterman [132] which allowed to efficiently calculate the minimal number of modifications to convert an input ancestral sequence to its descendant. Starting from this contribution many other probabilistic inference tools have been proposed. However, the evolutionary aligners are becoming overwhelmed by the computational effort as the data produced by the new technologies increases. Moreover, the evolutionary alignment inference, which is actually the topic of this thesis, concerns the hypothesis of site homology which clearly becomes increasingly complex to deduce as the distance among the input sequences grows [2, 15, 122, 126, 143].

It is also worth mentioning that aligner tools very often disagree with each other with regards to the inferred results ending in pervasive incongruences hampering in this way any achievement in evolutionary research [22, 59]. In particular, the generalized incidence of incongruencies among evolutionary multiple sequence aligners obstruct the phylogenetic analysis and therefore inhibit an effective revealing of evolutionary relationships [134]. The main source of error is primarily accounted for by an inappropriate mathematical modeling of the homology which inexorably affects the quality of multiple sequence alignments and hence the derived hypotheses of homologies. Although, however, the consequences of inaccurate inferred homologies on downstream analysis are still not fully explored and comprehended [17, 39, 57, 68, 69, 89, 95, 147, 165].

A second source of error can be imputed to the approximations and in general the heuristics applied to infer alignments of several sequences in reasonable timeframe. It has been shown that using for instance a sum-of-pairs scoring method the problem of finding the optimal multiple sequence alignment is NP-complete [159]. Presently, there is a great effort undertaken to speed-up the algorithms decomposing the different tasks to achieve parallelism. Often, although, a complete re-design of the algorithms is needed to avoid dependencies and to properly recycle the computations as much as possible. Very frequently this task is under-evaluated for the competencies required to achieve a satisfactory result, and approaches taken by neophytes in this domain are commonly ineffective. The most used technique to speed-up the computations, that we actually also pursued, is to employ progressive heuristics which are breaking the original problem into a series of pairwise alignments guided by a phylogeny. Whilst, without this greedy heuristic approach would not be possible to align more than few sequences together, on the other side it may introduces biases. These biases are very difficult to measure and typically are hereby accepted as inevitable price to pay for accelerating the alignment procedure. In Section [Progressive bias analysis](#) we are showing, albeit using a very small, that the progressive procedure, indeed, sometime introduces biases. In our test case, the bias depends on the choice of the system parameters, namely the branch lengths and the topology. In Section [Stochastic backtracking DP algorithm](#) we have proposed a possible route to deal with this issue. We have modified the Stochastic Backtracking algorithm (*Muekenstein et al. 2002* [100]) including the PIP evolutionary model. The main idea is to generate intermediate sub-optimal solutions which later may produce better alignments. Indeed, the results of the progressive bias (Section 6) has shown that the combination of greedy solutions is not necessarily always the best strategy. Although, the efficacy of this approach has not been quantified to date. In order to

save computational time we have implemented a short-time Fourier transform based homologous blocks detector (Section [Multi-scale STFT based homologous blocks detection](#)). The purpose of predicting candidate homologous segments is the successive elimination of the non-promising regions in the DP matrix prior to the effective alignment process. This new algorithm has been inspired by MAFFT [62] but differently from the original idea we have implemented a multi-scale time-frequency transform for the automatic detection of candidate homologous regions providing simultaneously the positional shift lag and the relative position of similar patterns inside the two sequences. Our proposed algorithm offers three different parallelisation levels: i) at the level of tree nodes, ii) at the level of homologous/linking blocks and iii) at the level of entries in one single layer of the DP matrix. This feature allows to adapt the parallelisation strategy at the particular problem at hand and to dynamically change strategy during the execution.

While changes between homologous characters are typically modeled by a Markov substitution process, the dynamics of indels, with a few notable exceptions, is never modeled explicitly. The reason lies in the fact that the computation of the marginal likelihood under such models has exponential time complexity in the number of taxa. Unfortunately though, the failure to adequately model indel evolution may lead to misleading inferred homologies artificially and hence typically short alignments. In turn, due to biased indel placement one might obtain inconsistent phylogenetic relationship. Recently, the classical indel model TKF91 has been modified describing indel evolution on a phylogeny by means of a Poisson process rather than with a birth-death process. This model, named PIP, allows to compute the joint marginal probability of an MSA and a tree in linear time. Based on PIP evolutionary model, we have developed and implemented – for the first time in the frequentist framework – a progressive MSA algorithm with an explicit evolutionary model of substitutions, insertions and deletions. The formulation of the algorithm is presented in Section [Progressive Dynamic Programming under PIP model](#) and to make it more clear we are showing a detailed example of an alignment inference in Section [3D Dynamic Programming under PIP](#). At the core of our method we have designed a new DP algorithm for the alignment of two homology paths represented by their corresponding MSAs. Our DP algorithm returns the maximum likelihood pairwise alignment under the PIP model exploiting its linear time complexity for the computation of marginal likelihood. Further details on our proposed approach can be found in the published manuscript attached. In our opinion, this represents a major achievement. With the same evolutionary model we are able to infer either

tree and multiple sequence alignment laying the groundwork for a sound joint inference of tree and alignment (Progressive multiple sequence alignment with indel evolution, Gatti, Maiolo, Gil, Anisimova, paper in preparation). Our MSA method is the first polynomial time progressive aligner with a rigorous mathematical formulation of indel evolution. The new method has shown to infer phylogenetically meaningful gap patterns alternative to the popular PRANK, while producing alignments of similar length. Moreover, the inferred gap patterns agree with what was predicted qualitatively by previous studies.

The explicit model of indel evolution comes at a price, in fact instead of requiring a bidimensional DP approach our method requires a third dimension to take into account for the MSA length. Therefore, the overall complexity of our progressive algorithm becomes $\mathcal{O}(N \cdot L^3)$, where N is number of taxa and L is the maximum sequence length. The cubic factor stems from the fact that the likelihood is not monotonically increasing in the MSA length, so that the length has to be incorporated as an extra dimension in the DP. The $\mathcal{O}(L^2)$ entries in a specific matrix layer along that dimension (i.e. corresponding to one particular alignment length) depend only on the layer above (and not on each other). Therefore, assuming $\mathcal{O}(L^2)$ processors, the computation can be parallelized taking down theoretically the running time to $\mathcal{O}(N \cdot L)$. Moreover, our empirical findings show that the likelihood has exactly one maximum, suggesting an early stop condition to the DP, this has been discussed in Section [Early stop condition](#). In addition, significant speed up is achieved by filtering out non-promising regions in the 3D-DP prior to align (see Section [Multi-scale STFT based homologous blocks detection](#)). In the manner of MAFFT, amino acids sequences are converted to sequences of their physicochemical properties and homologous regions are identified by short-time Fourier transform. Three 3D dynamic programming matrices are then created under PIP with homologous blocks defining sparse structures in which most part of the DP matrices is excluded from the calculations. A logically sound path to connect the homologous blocks through intermediate “linking blocks” is identified (see Section [Homology matrix and optimal path](#) and Appendix [Homologous blocks overlap resolution](#)). Homologous and linking blocks are aligned under PIP as independent DP sub-matrices and tracebacks merged to yield the final alignment. The new algorithm can therefore largely profit from parallel computing. We estimate a theoretical speed up proportional to the cubic power of the number of blocks (DP sub-matrices) and to the square of the average sequence length (proportional to the mean number of cells per layer). Furthermore, in Section [5.5: STFT vs. FT approach for block detection](#), we have compared the efficacy of the STFT against the FT approach regarding the ability of detect-

ing patterns (i.e., homologous regions) within the signals even when their are relative short and in presence of noise. Here, the noise is mimicking the substitution process which perturbs similar region and makes their identification more difficult to assess.

A know limitation of PIP and therefore also of our progressive aligner derives from the single character indel process. The reconciliation of modeling long indels and at the same time to obtain a polynomial algorithm that is able to exactly marginalize a continuous time stochastic process has unfortunately not been resolved, yet. At the moment have been proposed only complex approximations to the marginalization [70, 91] or models that allow for either only long insertions or only long deletions [92] or methods constraint to only a pair of sequences [146]. Regardless of this constraint, our method appears to perform surprisingly well compared to other state-of-the-art popular alignment tools such as PRANK and MAFFT (Progressive multiple sequence alignment with indel evolution, Gatti, Maiolo, Gil, Anisimova, paper in preparation). In contrast to traditional aligners that do not utilise phylogenetic information to distinguish insertions and deletions, our method produces longer alignments, avoiding the artificial compression of MSAs and inferring more indels, again similar to PRANK. According to the underlying indel model, our method appears to infer more shorter indels (e.g., compared to PRANK and MAFFT), while longer indels are described by several subsequent indel events. Including longer indels is considered desirable, however it has not been studied whether modeling one residue indels at a time may also work well. For example, for simplicity models of codon substitution typically allow only one-nucleotide mutations. Despite this gross simplification codon models have been demonstrated to perform extremely well for practical analyses of protein-coding genes. As can be seen in our example of an HIV protein gp120, it is unclear what inferred indel pattern is more realistic (given that alignments inferred by our methods and by PRANK have very similar length). Considering the nature of HIV mutations, it is quite plausible that indel evolution of gp120 is dominated by short indel events [1]. Arguably, in our example, indel penalisation of PRANK and MAFFT (affine penalty schemes allowing for long indels) might make these tools too restrictive to single-residue indels, leading to aesthetically more pleasing alignments. PIP might be more restrictive to long indels but also more realistic for sequence data dominated by short indel events. Both alignment benchmarking and the parameter optimisation of gap penalties are extremely difficult due to the absence of sufficiently challenging datasets where true alignments are known.

Conclusions

Presently, many existing molecular evolution models and methods have been proposed for the inference of multiple sequence alignments (MSAs) and phylogenies however, typically as independent steps and even worse based on different mathematical descriptions. Yet, this model inconsistency has an impact on the accuracy of estimations especially when the result of one inference is used as input for the second. Therefore, ideally MSAs and trees should be inferred jointly under the same mathematical framework. Some joint MSA-tree estimation algorithms have been implemented in the Bayesian framework, relying on the classic evolutionary birth-death model TKF91 that describes both substitutions and indels. However, such implementations are not suitable for large datasets, because of their intrinsic computational complexity. With this work, together with the development of the tree inference under PIP described in the manuscript [Phylogenetic inference under indel-aware evolutionary models increases tree reliability and robustness](#), we have laid the foundations for a sound joint inference of tree and alignment in the frequentist framework.

II

Appendices

A

Some technicalities

“Somewhere, something incredible is waiting to be known.”

– Carl Sagan

A.1 Detailed derivation of the marginal likelihood function $\varphi(v)$

In this section we are showing the mathematical steps to obtain a closed form expression for the marginal likelihood for all empty columns as defined in the original paper [13].

The marginal likelihood function under PIP is defined as:

$$p_\tau(m) = \mathbb{E} [\mathbb{P}(M = m) \mid |\mathbf{X}|] \quad (\text{A.1})$$

where the phylogeny τ is given and m represents the realization of the random variable M – the alignment function – obtained from a set of input sequences \mathbf{X} . The likelihood equation (A.1) is composed of two parts, the product of columns likelihood $p(c)$ – of the observed columns c – and the marginal likelihood of all non-observable empty columns computed by means of the function $\varphi(\cdot)$. Equation (A.1) rewrites therefore as follow

$$p_\tau(m) = \varphi(p(c_\emptyset), |m|) \prod_{c \in m} p(c) \quad (\text{A.2})$$

where $p(c_\emptyset)$ is the likelihood of a single column full of gaps and $|m|$ indicates the MSA length. If not otherwise specified $p(c_\emptyset) = p_v(c_\emptyset)$ where $v = \Omega$.

The marginal likelihood for the empty columns $\varphi(p(c_\emptyset), |m|)$ is computed starting from the likelihood of a single column full of gaps c_\emptyset and is calculated by means of the following equation

$$\varphi(p(c_\emptyset), |m|) = \sum_{n=|m|}^{\infty} \mathbb{P}(|\mathbf{X}| = n) \cdot \binom{n}{|m|} \cdot p(c_\emptyset)^{n-|m|} \quad (\text{A.3})$$

where $\mathbb{P}(|\mathbf{X}| = n)$ is the Poisson probability of observing an alignment of length n given a topology of total length¹ $\|\tau\|$ and $p(c_\emptyset)$ is the likelihood of a single empty column. The function $\varphi(\cdot)$ marginalizes over an unknown and infinite number of empty columns.

Let be $\|\nu\|$ the normalizing measure of the Poisson intensity (see the original paper for its derivation), which is equal to

$$\|\nu\| = \lambda \left(\|\tau\| + \frac{1}{\mu} \right) \quad (\text{A.4})$$

and where λ is the insertion (birth) rate and μ is the deletion (death) rate of a single character. The rates λ and μ are kept constant during the entire evolution process and do not depend on the type of character inserted respectively deleted.

The Poisson probability of observing an alignment of a fixed length given the model parameters can be written as

$$\mathbb{P}(|\mathbf{X}| = n) = \frac{\|\nu\|^n}{n!} e^{-\|\nu\|} \quad (\text{A.5})$$

where $\|\nu\|$ corresponds to the expected MSA length generated by the Poisson process. The total number of different alignments of total length n considering a number m of indistinguishable and unobservable empty columns is calculated by the binomial coefficient

$$\binom{n}{|m|} = \frac{n!}{|m|!(n-|m|)!} \quad (\text{A.6})$$

where $n - |m|$ represents the number of columns containing at least a character per column. Putting together all the elements (equations A.3-A.6) and marginalizing over all possible MSA lengths (considering observable and non-observable columns) we obtain

$$\varphi(p(c_\emptyset), |m|) = \sum_{n=|m|}^{\infty} \frac{\|\nu\|^n}{n!} e^{-\|\nu\|} \cdot \frac{n!}{|m|!(n-|m|)!} \cdot p(c_\emptyset)^{n-|m|}. \quad (\text{A.7})$$

¹The total tree length is computed as $\|\tau\| = \sum_{v \in \mathcal{V}} b(v)$.

At this point we may write $p(c_\emptyset)^{n-|m|}$ as

$$p(c_\emptyset)^{n-|m|} = \frac{p(c_\emptyset)^n}{p(c_\emptyset)^{|m|}} \quad (\text{A.8})$$

and by replacing equation A.8 into equation A.3 and pulling the constants out of the summation we get

$$\varphi(p(c_\emptyset), |m|) = \frac{e^{-\|\nu\|}}{|m|! p(c_\emptyset)^{|m|}} \sum_{n=|m|}^{\infty} \frac{\|\nu\|^n p(c_\emptyset)^n}{(n-|m|)!}. \quad (\text{A.9})$$

At this point it is convenient to define

$$a = \|\nu\| p(c_\emptyset), \quad (\text{A.10})$$

so that the infinite summation in equation A.9, using equation A.10, can be rewritten in the following way

$$\begin{aligned} \sum_{n=k}^{\infty} \frac{a^n}{(n-k)!} &= \\ &= \frac{a^k}{0!} + \frac{a^{k+1}}{1!} \frac{a^{k+2}}{2!} + \dots \\ &= \left(\frac{a^0 + a^1 + a^2 + \dots}{a^0 + a^1 + a^2 + \dots} \right) \frac{a^k}{0!} + \left(\frac{a^0 + a^1 + a^2 + \dots}{a^0 + a^1 + a^2 + \dots} \right) \frac{a^{k+1}}{1!} + \dots \\ &= \left(\frac{a^k + a^{k+1} + a^{k+2} + \dots}{a^0 + a^1 + a^2 + \dots} \right) \frac{a^0}{0!} + \left(\frac{a^k + a^{k+1} + a^{k+2} + \dots}{a^0 + a^1 + a^2 + \dots} \right) \frac{a^1}{1!} + \dots \\ &= \left(\frac{a^k + a^{k+1} + a^{k+2} + \dots}{a^0 + a^1 + a^2 + \dots} \right) \left(\frac{a^0}{0!} + \frac{a^1}{1!} + \frac{a^2}{2!} + \dots \right) \\ &= \frac{\sum_{n=k}^{\infty} a^n}{\sum_{n=0}^{\infty} a^n} \cdot \sum_{n=0}^{\infty} \frac{a^n}{n!}. \end{aligned} \quad (\text{A.11})$$

Now, using the result of equation A.11 into equation A.9 and substituting back the variable a , defined in equation A.10, we obtain

$$\varphi(p(c_\emptyset), |m|) = \frac{e^{-\|\nu\|}}{|m|! p(c_\emptyset)^{|m|}} \cdot \frac{\sum_{n=|m|}^{\infty} (\|\nu\| p(c_\emptyset))^n}{\sum_{n=0}^{\infty} (\|\nu\| p(c_\emptyset))^n} \cdot \sum_{k=0}^{\infty} \frac{\|\nu\|^k p(c_\emptyset)^k}{k!}. \quad (\text{A.12})$$

At this point we may recall some remarkable results from the sum of powers, that are

$$\sum_{n=0}^{\infty} a^n = \frac{1}{1-a} \quad \text{if } |a| < 1 \quad (\text{A.13})$$

and

$$\sum_{n=m}^k a^n = \frac{a^m - a^{k+1}}{1-a} \quad \text{for } a \neq 1 \quad (\text{A.14})$$

as well as

$$\sum_{n=0}^{k-1} a^n = \frac{1 - a^k}{1-a}. \quad (\text{A.15})$$

Therefore, the second factor in equation A.12, that is

$$\frac{\sum_{n=|m|}^{\infty} (\|\nu\|p(c_\emptyset))^n}{\sum_{n=0}^{\infty} (\|\nu\|p(c_\emptyset))^n} \quad (\text{A.16})$$

can be simplified in the following way. First note that equation

$$\frac{\sum_{n=k}^{\infty} a^n}{\sum_{n=0}^{\infty} a^n} = \frac{\sum_{n=0}^{\infty} a^n - \sum_{n=0}^{k-1} a^n}{\sum_{n=0}^{\infty} a^n} \quad (\text{A.17})$$

can be rewritten using the result of equations A.13-A.15 as

$$\frac{\frac{1}{1-a} - \frac{1-a^{|m|}}{1-a}}{\frac{1}{1-a}} = a^{|m|} \quad (\text{A.18})$$

and hence applying the result of equation A.18 and equation A.10 into equation A.12 we obtain

$$\frac{\sum_{n=|m|}^{\infty} (\|\nu\|p(c_\emptyset))^n}{\sum_{n=0}^{\infty} (\|\nu\|p(c_\emptyset))^n} = (\|\nu\|p(c_\emptyset))^m. \quad (\text{A.19})$$

Since

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \quad (\text{A.20})$$

we can simplify equation A.12 using equation A.19 and equation A.20 as follow

$$\varphi(p(c_\emptyset), |m|) = \frac{e^{-\|\nu\|}}{|m|! \cdot p(c_\emptyset)^{|m|}} \cdot \|\nu\|^{|m|} \cdot p(c_\emptyset)^{|m|} \cdot e^{\|\nu\|p(c_\emptyset)} \quad (\text{A.21})$$

$$= \frac{e^{-\|\nu\|}}{|m|!} \cdot \|\nu\|^{|m|} \cdot e^{\|\nu\|p(c_\emptyset)} \quad (\text{A.22})$$

and finally

$$\varphi(p(c_\emptyset), |m|) = \frac{1}{|m|!} \cdot \|\nu\|^{|m|} \cdot e^{-\|\nu\|} e^{\|\nu\|p(c_\emptyset)} \quad (\text{A.23})$$

$$= \frac{1}{|m|!} \cdot \|\nu\|^{|m|} \cdot e^{\|\nu\|(p(c_\emptyset)-1)}. \quad (\text{A.24})$$

A.2 Detailed derivation of the survival probability function $\beta(v)$

In this section we are showing in detail the mathematical derivation of the survival probability $\beta(v)$ associated to a node v . For the original introduction please refer to the original paper [13].

Let U be a random variable drawn from a uniform distribution $U \sim \mathcal{U}(0, t)$ and W be a random variable drawn from an exponential distribution $W \sim \mathcal{W}(\mu)$ and let U, W be independent among them, For fixed $t, \mu > 0$ then

$$\mathbb{P}(U + W > t) = \frac{1 - \exp(-t\mu)}{t\mu}. \quad (\text{A.25})$$

The stochastic variable U represents an insertion of a character in a random

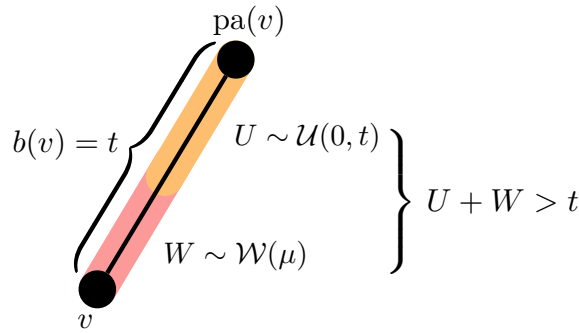


Figure A.1: The stochastic variable U expresses the random insertion location of a character drawn from a continuous uniform distribution $\mathcal{U}(0, t)$ on the edge $v \rightarrow \text{pa}(v)$ of length $b(v) = t$. The stochastic variable W denotes the waiting time between consecutive events in a Poisson process, in this case until the next deletion event. W is drawn from an exponential distribution $\mathcal{W}(\mu)$ of intensity μ (deletion rate) and describes the survival probability of the inserted character until the first node below the insertion point (node v).

of equation A.25, let be X and Y two independent random variables and let

denote by $f_U(U)$ and $f_W(W)$ their probability distribution functions (PDF's):

$$f_U(U = u) = \begin{cases} \frac{1}{t} & \text{if } 0 < u < t \\ 0 & \text{otherwise} \end{cases} \quad (\text{A.26})$$

and

$$f_W(W = w) = \begin{cases} \mu \exp(-\mu y) & \text{if } w > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.27})$$

We denote with Z the sum of the two stochastic variables, i.e. $Z = U + W$ and with $f_Z(Z)$ the probability distribution function of Z . The probability distribution of the sum of two stochastic variables can be obtained by convolving their probability distribution functions. Therefore, $f_Z(Z)$, the pdf of $Z = U + W$ reads

$$\begin{aligned} f_Z(Z = z) &= \mathbb{E}(f_U(z - y)) \\ &= \int_{-\infty}^{\infty} f_U(z - y) \cdot f_W(y) dy \\ &= \int_{-\infty}^{\infty} f_U(z - y) \cdot \mu \exp(-\mu y) dy \\ &= \int_{-\infty}^{\infty} \frac{1}{t} \cdot \mu \exp(-\mu y) dy && \text{where } 0 \leq z - y \leq t \\ &= \int_{-\infty}^{\infty} \frac{1}{t} \cdot \mu \exp(-\mu y) dy && \text{where } -t \leq y - z \leq 0 \\ &= \int_{-\infty}^{\infty} \frac{1}{t} \cdot \mu \exp(-\mu y) dy && \text{where } z - t \leq y \leq z \\ &= \int_{z-t}^z \frac{1}{t} \cdot \mu \exp(-\mu y) dy \\ &= -\frac{\exp(-\mu y)}{t} \Big|_{y=z-t}^{y=z} = -\frac{\exp(-\mu z)(\exp(\mu t) - 1)}{t}. \end{aligned} \quad (\text{A.28})$$

Once the character has been inserted on the edge e , the character must also survive until the node v . This condition is obtained by requiring that the stochastic variable Z is at least $b(v) = t$. Hence, to obtain the probability that $Z > t$ we must compute the tail of the cumulative distribution $\mathbb{P}(Z > t)$,

which gets

$$\begin{aligned}
 \mathbb{P}(W + U > t) &= \mathbb{P}(Z > t) \\
 &= \int_t^\infty \frac{\exp(-\mu z)(\exp(\mu t) - 1)}{t} dz \\
 &= - \left. \frac{(\exp(\mu t) - 1) \exp(-\mu z)}{\mu t} \right|_{z=t}^{z=\infty} \\
 &= \frac{1 - \exp(-\mu t)}{\mu t}. \tag{A.29}
 \end{aligned}$$

Equation [A.29](#) is therefore the survival probability $\beta(v)$ associated to a node v of a character inserted along the edge $v \rightarrow \text{pa}(v)$ as defined in the original PIP paper [\[13\]](#).

A.3 Overview of PIP equations

The likelihood of a single column c containing at least a character, that is $c \neq c_\emptyset$ can be computed by means of the following equation (see original PIP paper [13] for more details):

$$\begin{aligned}
 p(c) &= \mathbb{P}(C = c) = \sum_{v \in \mathcal{V}} \mathbb{P}(V = v) \times \underbrace{\mathbb{P}(C = c \mid V = v)}_{f_v} \\
 &= \sum_{v \in \mathcal{V}} \underbrace{\begin{cases} \bar{\nu}(e \setminus \{\Omega\}) & \text{if } v \neq \Omega \\ \bar{\nu}(\{\Omega\}) & \text{otherwise} \end{cases}}_{\mathbb{P}(V = v)} \times \underbrace{\begin{cases} c \neq c_\emptyset : \begin{cases} \tilde{f}_v & \text{if } v = \Omega \\ \mathbb{1}[v \in \mathcal{I}] \beta(v) \tilde{f}_v & \text{otherwise} \end{cases} \\ c = c_\emptyset : \begin{cases} \tilde{f}_v & \text{if } v = \Omega \\ 1 + \beta(v) (-1 + \tilde{f}_v) & \text{otherwise} \end{cases} \end{cases}}_{f_v} \\
 &= \sum_{v \in \mathcal{V}} \begin{cases} \frac{b(v)}{\|\tau\| + \frac{1}{\mu}} & \text{if } v \neq \Omega \\ \frac{1/\mu}{\|\tau\| + \frac{1}{\mu}} & \text{otherwise} \end{cases} \times \begin{cases} c \neq c_\emptyset : \begin{cases} \sum_{\sigma \in \mathcal{A}} \pi_\epsilon(\sigma) \tilde{f}_v(\sigma) & \text{if } v = \Omega \\ \underbrace{\mathbb{1}[v \in \mathcal{I}] \frac{1}{\mu b(v)} (1 - e^{-\mu b(v)})}_{\beta(v)} \underbrace{\sum_{\sigma \in \mathcal{A}} \pi_\epsilon(\sigma) \tilde{f}_v(\sigma)}_{\tilde{f}_v} & \text{otherwise} \end{cases} \\ c = c_\emptyset : \begin{cases} \sum_{\sigma \in \mathcal{A}} \pi_\epsilon(\sigma) \tilde{f}_v(\sigma) & \text{if } v = \Omega \\ 1 + \frac{1}{\mu b(v)} (1 - e^{-\mu b(v)}) \left(-1 + \underbrace{\sum_{\sigma \in \mathcal{A}} \pi_\epsilon(\sigma) \tilde{f}_v(\sigma)}_{\tilde{f}_v} \right) & \text{otherwise} \end{cases} \end{cases}
 \end{aligned}$$

where $\bar{\nu} = \nu / \|\nu\|$, $\mathbb{1}(\cdot)$ is the Indicator function, $\tilde{f}_v = \mathbb{P}(C = c \mid V = v, H(v) \neq \epsilon)$, and

$$\tilde{f}_v(\sigma) = \begin{cases} \mathbb{1}(c(v) = \sigma) & \text{if } v \in \mathcal{L} \\ \prod_{w \in \text{child}(v)} \sum_{\sigma' \in \mathcal{A}_\epsilon} \exp(b(w) \mathbf{Q}_\epsilon)_{\sigma, \sigma'} \tilde{f}_w(\sigma') & \text{otherwise} \end{cases}$$

B

Reversibility of TKF91 and PIP

“An expert is a person who has made all the mistakes that can be made in a very narrow field.”

– Niels Bohr

B.1 Introduction to TKF91

TKF91 [145] is a string-value stochastic process defined on the branches of a phylogenetic tree mathematically modeled by means of a continuous-time Markov chain (CTMC). In the seminal paper of Thorne, Kishino and Felsenstein [145] a infinite state-space birth-death process let evolve a string of characters via insertion and deletion events. This insertion-deletion process is described in terms of links between consecutive sequence residues rather than directly on the characters themselves. Hence, a sequence of N bases is transformed into a sequence of links where the first is an *immortal* link followed by N *mortal* links. The immortal link – as the name indicates – cannot be deleted. This construction prevents the sequences from vanishing over time and – under some constraints¹ – produces a realistic equilibrium distribution of sequence lengths.

The birth-death process applies to each link independently and, consequently, an event that happen to a link does not affect the birth-death probability of neighbors links. Mortal and immortal links are both associated with

¹Let be λ the insertion rate and μ the deletion rate, than by setting $\lambda < \mu$ the sequence lengths are geometrical distributed with parameter $\frac{\lambda}{\mu}$.

births with rate per link equal to λ . Links can give births only to mortal links which are placed immediately to the right of its parent link. The birth of a new link implies the insertion of a new character associated to it which is then arranged immediately to the left of the newborn link.

Mortal links are subject to death at a rate per link equal to μ producing the deletion of the character associated (to the left). The chance of more than a single event on a link at same instant is negligible hence a sequence increases or decreases in length by a single unit during a given instant.

Considering the birth of new links at a rate λ per link and the death at rate μ per mortal link, a sequence of N bases increases in length to $N + 1$ characters at rate $(N + 1)\lambda$ and decrease in length to $N - 1$ bases (assuming $N > 0$) at rate $N\mu$. Denoting γ_N the equilibrium probability of a sequences consisting of N characters, then the distribution of γ_N obtained under that birth-death model is the geometric distribution $\gamma_N = (1 - \lambda/\mu)(\lambda/\mu)^N$, subject to the condition $0 < \lambda < \mu$.

The birth-death model – just summarized – allows to compute the probability, given two related sequences A and B, that after an interval on time t sequence A evolves into B. The calculation of this transition probability can be separated into its two components representing two superimposed stochastic processes acting on the sequence: the *substitution process* and the *insertion-deletion process*. Chosen a reversible substitution models, the substitution probability $f_{i,j}(t)$ that a character of type i is replaced by one of type j in a period of time t gets

$$f_{i,j}(t) = \begin{cases} \exp(-s_{i,j}t) + \pi_j(1 - \exp(-s_{i,j}t)) & i = j \\ \pi_j(1 - \exp(-s_{i,j}t)) & i \neq j \end{cases} \quad (\text{B.1})$$

where $s_{i,j}$ represents the rate for such substitution and π_i is the steady state probability that the new character is of type i .

By considering sequence A as the ancestor of sequence B and putting them into an alignment ϱ the transition probability that describe the evolution from A to B is obtained by computing

$$\mathbb{P}(\varrho \mid \theta) = \mathbb{P}(\varrho, \varrho' \mid \theta) = \mathbb{P}(\varrho \mid \varrho', \theta)\mathbb{P}(\varrho' \mid \theta) \quad (\text{B.2})$$

where θ represents the collection of model parameters. ϱ' denotes the information on the presence or absence of bases in the given alignment ϱ and $\mathbb{P}(\varrho' \mid \theta)$ denotes this probability. ϱ' is therefore the alignment obtained from ϱ by substituting each character with a gap/non-gap symbol. The substitutions

probability is computed by $\mathbb{P}(\varrho \mid \varrho', \theta)$. Therefore, the probability $\mathbb{P}(\varrho \mid \theta)$ of a specific homology path represented by alignment ϱ is obtained from the insertion-deletion transition probability $\mathbb{P}(\varrho' \mid \theta)$ together with the substitution probability $\mathbb{P}(\varrho \mid \varrho', \theta)$.

With an ancestral sequence consisting of N bases, the probability $\mathbb{P}(\varrho' \mid \theta)$ is obtained as the product of $N + 2$ terms that are:

- (i) γ_N : the *geometric equilibrium probability* of an ancestral sequence of length N ;
- (ii) $p_N''(t)$: the transition probability for the *immortal link*;
- (iii) $p_N(t)$ or $p_N'(t)$: the N terms for the transition probabilities for *normal links*.

These probabilities refer to

- (i) $p_N(t)$: the probability after a timespan t that N links are descended from a normal link (including itself);
- (ii) $p_N'(t)$: the probability that N links are descended from a normal link and the original dies and
- (iii) $p_N''(t)$: the probability that the immortal link has N descendants (including itself).

The corresponding birth-death differential equations are:

$$\frac{\partial p_N(t)}{\partial t} = \lambda(N-1)p_{N-1}(t) - (\lambda + \mu)Np_N(t) + \mu Np_{N+1}(t) \quad (\text{B.3})$$

$$\frac{\partial p_N'(t)}{\partial t} = \lambda(N-1)p_{N-1}'(t) + \mu(N+1)p_{N+1}'(t) + \mu p_{N+1}(t) \quad (\text{B.4})$$

$$\frac{\partial p_0'(t)}{\partial t} = \mu p_1'(t) + \mu p_1(t) \quad (\text{B.5})$$

$$\frac{\partial p_0''(t)}{\partial t} = \lambda(N-1)p_{N-1}''(t) - (\lambda N + \mu(N-1))p_N''(t) + \mu Np_{N+1}''(t) \quad (\text{B.6})$$

with $N > 0$ and initial conditions

$$\begin{aligned} p_1(0) &= 1 & p_N'(0) &= 0 & N &= 0, 1, \dots \\ p_N(0) &= 0 & p_N''(0) &= 0 & N &= 2, 3, \dots \\ p_1''(0) &= 1, \end{aligned}$$

and by definition:

$$p_0(t) = 0 \quad \text{and} \quad p_0''(t) = 0. \quad (\text{B.7})$$

The solutions of the differential equations (eqs. B.3-B.6) are

$$p_N(t) = \exp(-\mu t)(1 - \lambda\psi(t))(\lambda\psi(t))^{N-1} \quad N > 0 \quad (\text{B.8})$$

$$p_N'(t) = (1 - \exp(-\mu t) - \mu\psi(t))(1 - \lambda\psi(t))(\lambda\psi(t))^{N-1} \quad N > 0 \quad (\text{B.9})$$

$$p_0'(t) = \mu\psi(t) \quad (\text{B.10})$$

$$p_N''(t) = (1 - \lambda\psi(t))(\lambda\psi(t))^{N-1} \quad N > 0 \quad (\text{B.11})$$

with

$$\psi(t) = \frac{1 - \exp((\lambda - \mu)t)}{\mu - \lambda \exp((\lambda - \mu)t)} \quad (\text{B.12})$$

In Section B.3 we will show, by means of some examples, how to compute the probability under TKF91 of an alignment along a generic branch of a tree. We are also testing in Section B.3 and Section B.4 whether TKF91, respectively PIP, are both reversible evolutionary processes. They, however, should not be intended as rigorous mathematical demonstrations. But before showing the likelihood formulation under TKF91, it will be useful to recall (Section B.2) the main aspects regarding the time reversibility of an evolutionary process that will be useful later in Sections B.3 and B.4.

B.2 Time-reversible evolutionary process

In this section we recall briefly the equations implied by a time-reversible Markov chains that are the basis for the model reversibility analysis performed respectively for TKF91 (in Section B.3) and PIP (in Section B.4).

The likelihood of a pair of related sequences A and B descendant from a common ancestral sequence C by a evolutionary time t is

$$\mathbb{P}_t(\text{A}, \text{B}) = \sum_{\text{C}} \mathbb{P}_{\infty}(\text{C}) \mathbb{P}_t(\text{A} \mid \text{C}) \mathbb{P}_t(\text{B} \mid \text{C})$$

graphically depicted in Figure B.1(i).

With a **reversible** (evolutionary) process one can write

$$\mathbb{P}_\infty(C)\mathbb{P}_t(A | C) = \mathbb{P}_\infty(A)\mathbb{P}_t(C | A) \quad \text{for every } A, C \text{ and } t > 0 \quad (\text{B.13})$$

where $\mathbb{P}_\infty(C) = \mathbb{P}_t(C)$ when $t \rightarrow \infty$ and refers to the probability of ending in state C after an indefinite long interval of time t . Equation B.13 leads to

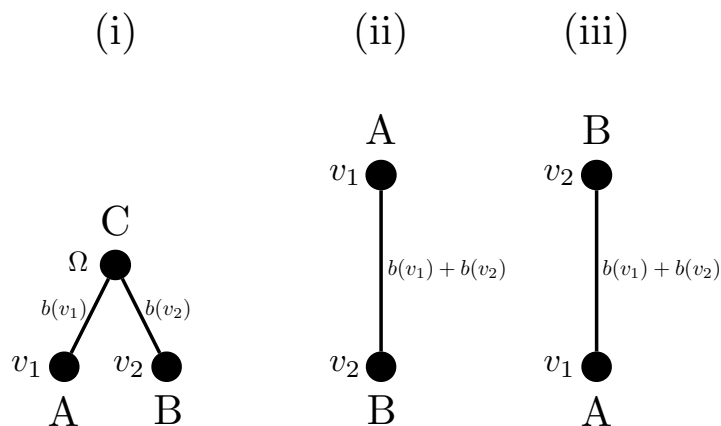


Figure B.1: (i) Pair of modern sequences, A and B, separated from a common ancestral sequence C. (ii) Degenerate topology with sequence A ancestor of sequence B separated by an evolutionary distance equal to the sum of the two branches $b(v_1)$ and $b(v_2)$. (iii) Degenerate topology with sequence B ancestor of sequence A separated by an evolutionary distance equal to the sum of the two branches $b(v_1)$ and $b(v_2)$.

B.3 Reversibility of TKF91

In this section we are testing whether TKF91 is a reversible model through two examples. To test the reversibility we have computed the likelihood of aligning two sequences A and B in four different configurations: (i) by taking A as ancestor of B; (ii) by taking B as ancestor of A; (iii) as in i) but reverting the columns order of the two sequences; (iv) as in ii) also reverting the columns order of the two sequences.

For the purpose of testing the reversibility of TKF91 model it is sufficient to compute the probability term $P(\varrho' | \theta)$ because this model is built on a reversible substitution process and the probability term $\mathbb{P}(\varrho | \varrho', \theta)$ is therefore reversible by definition.

Case 1

Using the pairwise alignments shown in Figure B.2a and assuming that the sequences are evolving on the topology show in Figure B.1(ii) the likelihood of the implied alignments reads

$$p_a = \left[\left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\mu}\right)^4 \right] \cdot \left[\left(1 - \lambda \cdot \psi(t)\right) \left(\lambda \cdot \psi(t)\right) \right] \cdot \left[\mu \cdot \psi(t) \right] \cdot \left[\exp(-\mu t) \left(1 - \lambda \cdot \psi(t)\right) \cdot \left(\lambda \cdot \psi(t)\right)^0 \right]. \quad (\text{B.14})$$

$$\cdot \left[\left(1 - \exp(-\mu t) - \mu \cdot \psi(t)\right) \left(1 - \lambda \cdot \psi(t)\right) \left(\lambda \cdot \psi(t)\right)^0 \right] \cdot \left[\exp(-\mu t) \cdot \left(1 - \lambda \cdot \psi(t)\right) \left(\lambda \cdot \psi(t)\right) \right]. \quad (\text{B.15})$$

The likelihood $P(\varrho' | \theta)$ obtained for the alignment in Figure B.2c – obtained by reverting the columns order of both sequences – is equal to the likelihood computed in equation B.14 for the original alignment of Figure B.2a.

In order to check whether the model is time-reversible, we have computed the likelihood of the alignment obtained by setting B as the ancestor of sequence A. The probability $P(\varrho' | \theta)$ of the resulting alignments, shown in Figure B.2c, gets

$$p_b = \left[\left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\mu}\right)^5 \right] \cdot \left[\left(1 - \lambda \cdot \psi(t)\right) \left(\lambda \cdot \psi(t)\right)^0 \right] \cdot \left[\mu \cdot \psi(t) \right] \cdot \left[\exp(-\mu t) \left(1 - \lambda \cdot \psi(t)\right) \cdot \left(\lambda \cdot \psi(t)\right)^0 \right] \cdot \left[\left(1 - \exp(-\mu t) - \mu \cdot \psi(t)\right) \left(1 - \lambda \cdot \psi(t)\right) \left(\lambda \cdot \psi(t)\right)^0 \right] \cdot \left[\exp(-\mu t) \cdot \left(1 - \lambda \cdot \psi(t)\right) \left(\lambda \cdot \psi(t)\right) \right] \cdot \left[\mu \cdot \psi(t) \right]. \quad (\text{B.16})$$

The probability $P(\varrho' | \theta)$ of the alignment in Figure B.2d (equation B.16), obtained by reversing the evolutionary time and the column order, is equal to

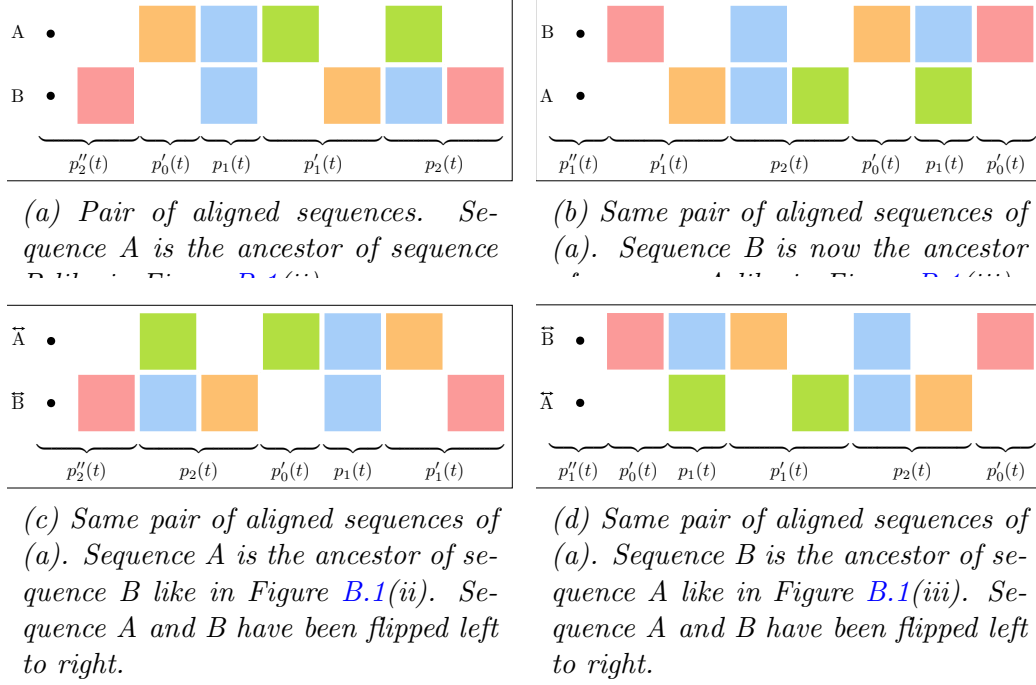


Figure B.2: Four pairwise alignments of the same two sequences A and B involving four different homologous histories. (a): original alignment; (b) alignment of (a) inverted top-down; (c) alignment of (a) where the sequences have been flipped left to right; (d) alignment of (a) inverted top-down and flipped left to right. Dots on the left side of the sequences represent the immortal links.

the probability of the original alignment (equations B.14 and B.16). Indeed, by computing the ratio between $p_{a,c}$ (equation B.14, where $p_a = p_c$) and $p_{b,d}$ (equation B.16, where $p_b = p_d$) we get

$$\frac{p_{b,d}}{p_{a,c}} = 1. \quad (\text{B.17})$$

Moreover, since TKF91 is using a reversible substitution model the probability $P(\varrho \mid \varrho', \theta)$ of evolving from A to B or evolving from B to A (see Figures B.1(ii) and (iii)) is by construction exactly the same. Therefore, in this example, TKF91 is a time-reversible model.

Case 2

Let us analyze a second alignment (depicted in Fig. B.3). The probability

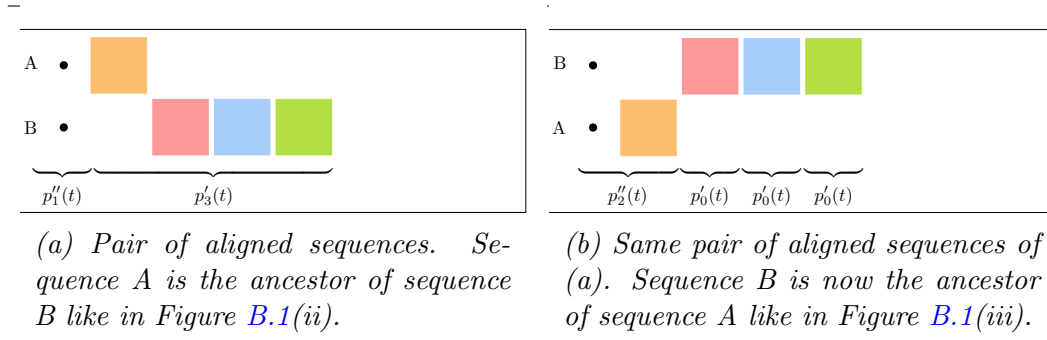


Figure B.3: Two pairwise alignments of the same two sequences A and B involving different homologous histories. (a): original alignment; (b) alignment of (a) inverted top-down. Dots on the left side of the sequences represent the immortal links.

$P(\varrho' | \theta)$ of the pairwise alignments shown in Figure B.3a is obtained by computing

$$p_a = \left[\left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\mu}\right) \right] \cdot \left[(1 - \lambda\psi(t)) (\lambda\psi(t))^0 \right] \cdot \left[(1 - \exp(-\mu t) - \mu\psi(t)) \cdot (1 - \lambda\psi(t)) (\lambda\psi(t))^2 \right] \quad (\text{B.18})$$

whereas the probability $P(\varrho' | \theta)$ of the alignment of Figure B.3b is

$$p_b = \left[\left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\mu}\right)^3 \right] \cdot \left[(1 - \lambda\psi(t)) (\lambda\psi(t)) \right] \cdot \left[(\mu\psi(t))^3 \right] \quad (\text{B.19})$$

Already at this point is evident that by reverting the time evolution the likelihood of the two implied alignments differs. In fact, when we compute the ratio between p_a (equation B.18) and p_b (equation B.19) we get

$$\frac{p_b}{p_a} = \frac{\lambda^4 \psi(t)^3 \mu^{-1}}{1 - \exp(-\mu t) - \mu\psi(t)}. \quad (\text{B.20})$$

The second example shows that TKF91 model in principle is NOT a time-reversible model. By reverting the time, the likelihood of the two corresponding

B.4. f

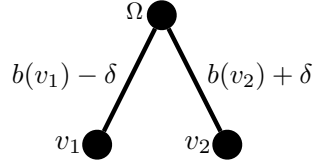


Figure B.4: The branches change their length from 0 to $b(v_1) + b(v_2)$ by setting $\delta = -b(v_2), \dots, b(v_1)$.

homologous paths is different which is confirmed by the fact that their ratio $\neq 1$. The reason lies in the link construction, by reverting the evolutionary direction (or by reading from right to left the sequences) the interpretation of the events might change.

B.4 Reversibility of PIP

In Section B.3, we have seen that TKF91 is in general not a time-reversible model. Let us now analyze the model reversibility under PIP. To simplify this analysis we have used the topology shown in Figure B.4 and we have computed the likelihood of an MSA column by moving the root Ω from v_1 to v_2 setting $\delta = -b(v_2), \dots, b(v_1)$. Like for the TKF91 analysis, let us compute the likelihood $P(\alpha' | \theta)$ of an MSA column $c = \begin{bmatrix} \# \\ \# \end{bmatrix}$, where the symbol $\#$ stays for any characters in the alphabet but the gap.

The likelihood of the column c is

$$p(c) = \iota(v_0)\beta(v_0) \cdot \left\{ \pi_\epsilon \circ \left[\exp\left(\left(b(v_1) - \delta\right)\mathbf{Q}_\epsilon\right)\mathbf{f}(v_1) * \right. \right. \\ \left. \left. * \exp\left(\left(b(v_2) + \delta\right)\mathbf{Q}_\epsilon\right)\mathbf{f}(v_2) \right] \right\}. \quad (\text{B.21})$$

The prior insertion probability and survival probability associated at the root node remains unchanged by moving the root node position along the phylogeny as long as the total tree length $\|\tau\| = b(v_1) + b(v_2)$ is kept constant. Therefore the likelihood of matching the left character with the right one will not be affected by moving the root.

The Chapman-Kolmogorov theorem, briefly outlined in Section B.2, assures that in *match case* – when there is a character $\sigma \in \mathcal{A}$ on both ends of the edge connecting them – the substitution probability moving the root on

one of the two sides remains constant. Thence, the PIP model in *match case* is always *reversible*.

To check the reversibility also when an insertion or deletion event happens, we have computed the likelihood of a general column $c = \begin{bmatrix} \# \\ - \end{bmatrix}$. This likelihood can be written as

$$p_1(c) = \iota(v_0)\beta(v_0) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_0)) \quad (\text{B.22})$$

$$p_2(c) = \iota(v_1)\beta(v_1) \cdot (\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_1)) \quad (\text{B.23})$$

$$p(c) = p_1(c) + p_2(c) \quad (\text{B.24})$$

where

$$\mathbf{f}(v_0) = \left\{ \boldsymbol{\pi}_\epsilon \circ \left[\exp\left((b(v_1) - \delta)\mathbf{Q}_\epsilon\right)\mathbf{f}(v_1) * \left((b(v_2) + \delta)\mathbf{Q}_\epsilon\right)\mathbf{f}(v_2) \right] \right\} \quad (\text{B.25})$$

with

$$f(v_1) = [1, 1, 1, 1, 0]^\top \quad (\text{B.26})$$

(any characters but the gap) and

$$\mathbf{f}(v_2) = [0, 0, 0, 0, 1]^\top. \quad (\text{B.27})$$

Equation B.25 can be decomposed in this way

$$\mathbf{f}(v_0) = f_L \cdot f_R \quad (\text{B.28})$$

From equations B.24, B.26 and B.28 we get

$$f_L = \boldsymbol{\pi}_\epsilon \circ \left[\exp\left((b(v_1) - \delta)\mathbf{Q}_\epsilon\right)\mathbf{f}(v_1) \right] = \exp\left(-\mu(b(v_1) - \delta)\right) \quad (\text{B.29})$$

and from equations B.24, B.27 and B.28

$$f_R = \exp\left((b(v_2) + \delta)\mathbf{Q}_\epsilon\right)\mathbf{f}(v_2) = 1 - \exp\left(-\mu(b(v_2) + \delta)\right). \quad (\text{B.30})$$

Hence, equation B.24 becomes

$$\begin{aligned} p(c) &= \iota(v_0)\beta(v_0) \cdot \exp\left(-\mu(b(v_1) - \delta)\right) \cdot \left(1 - \exp\left(-\mu(b(v_2) + \delta)\right)\right) + \\ &\quad + \iota(v_1)\beta(v_1) \cdot 1 \\ &= \frac{1/\mu}{\|\tau\| + 1/\mu} \cdot \exp\left(-\mu(b(v_1) - \delta)\right) \cdot \left(1 - \exp\left(-\mu(b(v_2) + \delta)\right)\right) + \\ &\quad + \frac{b(v_1) - \delta}{\|\tau\| + 1/\mu} \cdot \frac{1 - \exp\left(-\mu(b(v_1) - \delta)\right)}{\mu(b(v_1) - \delta)}. \end{aligned} \quad (\text{B.31})$$

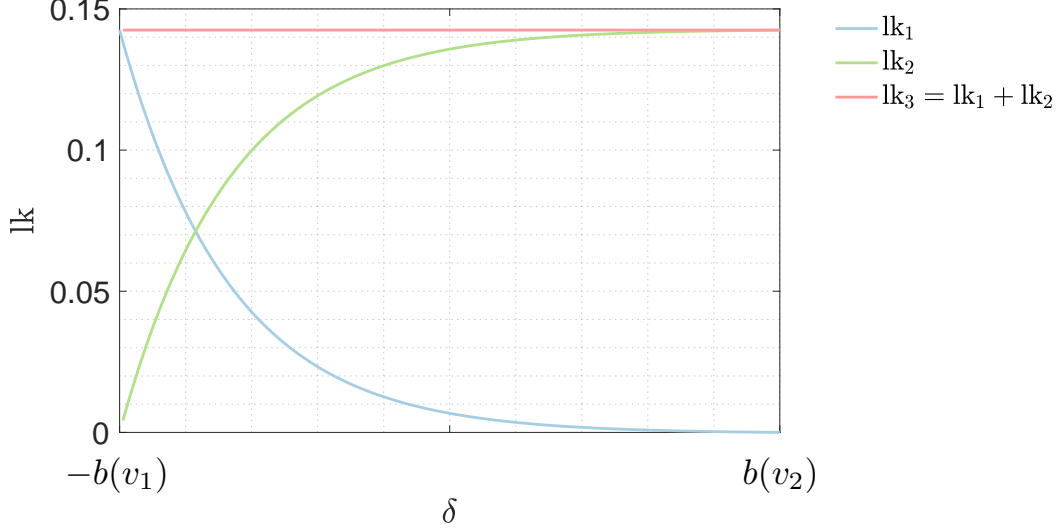


Figure B.5: Likelihood of an MSA column $c = [\#, -]^T$ as function of the root location. The position of the root Ω changes position by varying δ in the domain $[-b(v_1), \dots, b(v_2)]$. lk_1 represents the likelihood of equation B.22, lk_2 represents the likelihood of equation B.23 and lk_3 represents the likelihood of equation B.24.

In eq. B.31 we have used the property

$$\boldsymbol{\pi}_\epsilon \circ \mathbf{f}(v_1) = \boldsymbol{\pi}_\epsilon \circ [1, \dots, 1, 0]^T = 1. \quad (\text{B.32})$$

Equation B.31 can be simplified as

$$\begin{aligned} p(c) &= \frac{1}{\|\tau\| + 1/\mu} \cdot \frac{1}{\mu} \cdot \left[\exp\left(-\mu(b(v_1) - \delta)\right) - \exp\left(-\mu(b(v_1) + b(v_2))\right) \right] \\ &\quad + 1 - \exp\left(-\mu(b(v_1) - \delta)\right) \\ &= \frac{1/\mu}{\|\tau\| + 1/\mu} \cdot \left(1 - \exp\left(-\mu(b(v_1) + b(v_2))\right) \right). \end{aligned} \quad (\text{B.33})$$

From equation B.33 we can notice that the location of the root Ω – which is moving from v_1 to v_2 – does not affect the value of the likelihood as long as the sum of the two branches $b(v_1)$ and $b(v_2)$ remains constant. Therefore, PIP is a time-reversible model also in case of insertions and deletions.

Figure B.5 depicts the likelihood components $\mathbf{f}(v_0)$, f_L and f_R as function of δ . As shown analytically, the likelihood $\mathbf{f}(v_0)$ remains constant while changing the root location.

C

Characterization of Indel rates

*“Student: Dr. Einstein, Aren’t these the same questions as last year’s [physics] final exam?
Dr. Einstein: Yes; But this year the answers are different.”*

– Albert Einstein

In this Appendix we have re-elaborated some equations from the original PIP paper [13] in order to characterize the influence of the model parameters on the alignment.

C.1 Introduction

Under the PIP model, characters evolve along the branches of an evolutionary tree. Considering the insertion locations and their lifespan, one can classify the different fates to belong to one of the four classes, namely $N_1 - N_4$, which are graphically represented in Figure C.1. The four sets, $N_1 - N_4$, characterize the history of the characters depending on whether they were already present at the root of the tree and according to whether they survive till a fixed time point on the phylogeny, noted as v , after an evolutionary interval of time t . For instance, characters belonging to the set N_2 and N_4 are not observable at the instant v , while the ones included in the set N_1 and N_3 are. As demonstrated in the original paper ([13]), the total number of characters in the four sets at the root and at any point v is distributed according to $\text{Poi}(\lambda/\mu)$. In other words, under the PIP model the number of characters remains constant during the evolution, what is changing is rather the relative number in each set. The

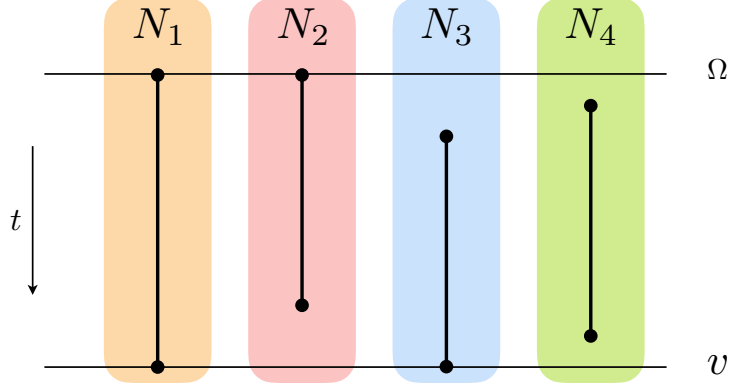


Figure C.1: Figure adapted from [13]-Supporting Information. The two horizontal lines denote the sequences at the root Ω and at a given node v , separated by an evolutionary time t . The horizontal lines denote the times where each character is present in the sequence. N_1 is the set of characters present both at the root and the node v , N_2 is the set of characters present only at the root. Set N_3 contains the characters inserted along the phylogeny not present at the root. N_4 is the set of character inserted below the root and died before reaching the time point v .

number of characters belonging to the different sets are distributed according to the following distributions [13]

$$N_1 \sim \text{Poi}\left(\frac{\lambda}{\mu}(1 - \exp(-\mu))\right)$$

$$N_2 \sim \text{Poi}\left(\frac{\lambda}{\mu} \exp(-\mu)\right)$$

$$N_3 \sim \text{Poi}\left(\frac{\lambda}{\mu}(1 - \exp(-\mu))\right)$$

where, as already stated above, λ corresponds to the insertion rate, μ represents the deletion rate. Therefore, given a topology τ of total length $\|\tau\|$ we can expect statistically

$$N_1(\|\tau\|) \simeq \frac{\lambda}{\mu}(1 - \exp(-\|\tau\|\mu)) \quad (\text{C.1})$$

$$N_2(\|\tau\|) \simeq \frac{\lambda}{\mu} \exp(-\|\tau\|\mu) \quad (\text{C.2})$$

$$N_3(\|\tau\|) \simeq \frac{\lambda}{\mu}(1 - \exp(-\|\tau\|\mu)) \quad (\text{C.3})$$

$$N_4(\|\tau\|) \simeq \frac{\lambda}{\mu} \exp(-\|\tau\|\mu) + \lambda\|\tau\| - \frac{\lambda}{\mu}. \quad (\text{C.4})$$

number of events.

Clearly, the following property holds

$$\begin{aligned} (N_1 + N_2 + N_3 + N_4)(\|\tau\|) &= 2\frac{\lambda}{\mu} \left(1 - \exp(-\|\tau\| \cdot \mu)\right) + 2\frac{\lambda}{\mu} \exp(-\|\tau\| \cdot \mu) + \\ &\quad + \lambda\|\tau\| - \frac{\lambda}{\mu} = \lambda\|\tau\| + \frac{\lambda}{\mu} \\ &= \lambda \left(\|\tau\| + \frac{1}{\mu}\right) = \|\nu\|. \end{aligned}$$

At the root we can expect to find in total

$$N_1(\|\tau\|) + N_2(\|\tau\|) \simeq \frac{\lambda}{\mu} \left(1 - \exp(-\|\tau\| \cdot \mu)\right) + \frac{\lambda}{\mu} \exp(-\|\tau\| \cdot \mu) = \frac{\lambda}{\mu}$$

number of characters and the same number can be expected at a given point v on the topology, after an interval of time t

$$N_2(\|\tau\|) + N_3(\|\tau\|) \simeq \frac{\lambda}{\mu} \exp(-\|\tau\| \cdot \mu) + \frac{\lambda}{\mu} \left(1 - \exp(-\|\tau\| \cdot \mu)\right) = \frac{\lambda}{\mu}.$$

Let us analyze the behavior of the four functions C.1-C.4 by taking the limits for an infinitely short and infinitely long interval of evolutionary time, that is

$$\lim_{\|\tau\| \rightarrow 0} N_1(\|\tau\|) = \lim_{\|\tau\| \rightarrow 0} N_3(\|\tau\|) = \lim_{\|\tau\| \rightarrow \infty} N_2(\|\tau\|) = \lim_{\|\tau\| \rightarrow 0} N_4(\|\tau\|) = 0 \quad (\text{C.5})$$

$$\lim_{\|\tau\| \rightarrow \infty} (N_1 + N_2)(\|\tau\|) = \lim_{\|\tau\| \rightarrow \infty} N_1(\|\tau\|) = \lim_{\|\tau\| \rightarrow \infty} N_3(\|\tau\|) = \frac{\lambda}{\mu} \quad (\text{C.6})$$

$$\lim_{\|\tau\| \rightarrow \infty} (N_2 + N_3)(\|\tau\|) = \lim_{\|\tau\| \rightarrow 0} N_2(\|\tau\|) = \frac{\lambda}{\mu} \quad (\text{C.7})$$

$$\lim_{\|\tau\| \rightarrow \infty} N_4(\|\tau\|) = \lim_{\|\tau\| \rightarrow \infty} (N_1 + N_2 + N_3 + N_4)(\|\tau\|) = \infty \quad (\text{C.8})$$

which are also represented graphically in Figure C.2.

C.2 Inferring indel rates from a given MSA

Suppose that we have an alignment m and that we want to infer the indel rates under PIP that could have generated it. Let be \bar{n} the expected sequence

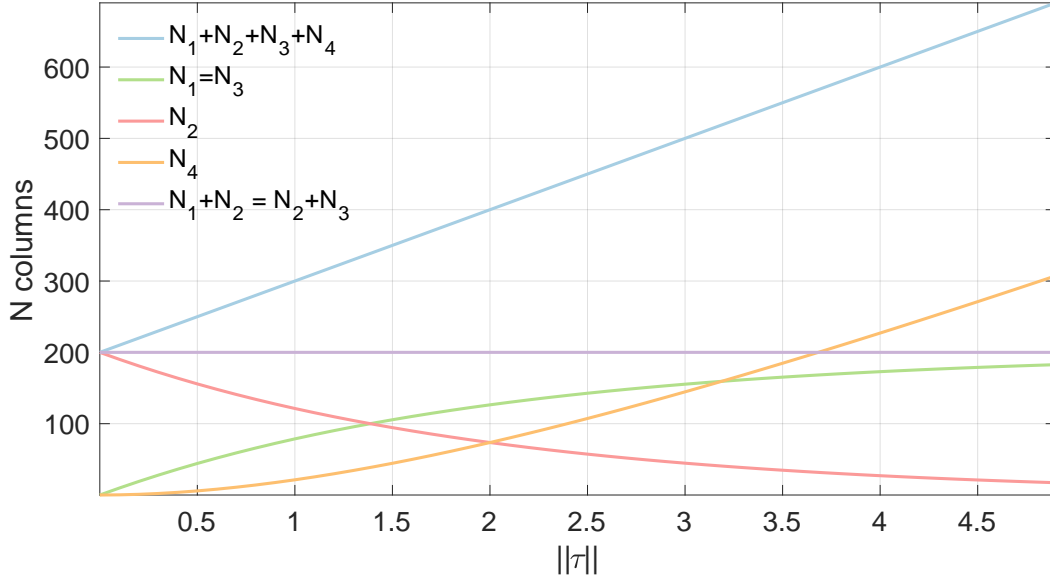


Figure C.2: Graphic representation of equations C.5-C.8 with $\lambda = 100$, $\mu = 1/2$ and $\|\tau\| = 0, \dots, 4.9$.

length computed as the average number of characters per sequence, and $|m|$ the length of the MSA m comprehending therefore the stretches introduced by the gaps.

Given that

$$\begin{cases} \bar{n} = \lambda/\mu \\ |m| = \lambda \left(\|\tau\| + \frac{1}{\mu} \right) = \|\nu\| \end{cases} \quad (\text{C.9})$$

we can estimate the indel rates of the PIP model solving equations C.9 with respect to λ and μ obtaining

$$\begin{cases} \lambda = \frac{|m| - \bar{n}}{\|\tau\|} \\ \mu = \frac{|m| - \bar{n}}{\|\tau\| \cdot \bar{n}}. \end{cases} \quad (\text{C.10})$$

In a PIP based MSA, we can expect to find a number of columns without gaps, denoted n_H , distributed according to $n_H \sim \text{Poi}\left(\frac{\lambda}{\mu} \exp(-\mu)\right)$ which, given

a topology τ with total length $\|\tau\|$ gives

$$n_{\text{H}} \simeq \frac{\lambda}{\mu} \exp(-\|\tau\|\mu). \quad (\text{C.11})$$

The expected number of columns containing gaps, denoted n_{G} (although not completely filled in with gaps), is given by

$$n_{\text{G}} \simeq \lambda\|\tau\| + \frac{\lambda}{\mu} \left[1 - \exp(-\|\tau\|\mu) \right]. \quad (\text{C.12})$$

The behavior of the variables n_{H} , n_{G} and their sum (total number of columns) as function of different model parameters is depicted graphically in Figure C.3 for the intensity parameter $I = \lambda \cdot \mu$, in Figure C.4 for the total tree length τ , in Figure C.5 for the insertion rate λ and in Figure C.6 for the deletion rate μ .

Alternatively, we can infer the indel rate parameters by considering the structure of the given MSA m which translates in counting the number of columns of the sets n_{H} and n_{G} . This gives the system of equations

$$\begin{cases} n_{\text{H}} = \frac{\lambda}{\mu} \exp(-\|\tau\|\mu) \\ n_{\text{G}} = \lambda\mu + \frac{\lambda}{\mu} (1 - \exp(-\|\tau\|\mu)) \end{cases} \quad (\text{C.13})$$

that can be solved with respect of λ and μ , as follow

$$\begin{cases} \lambda = \frac{n_{\text{H}} \cdot \exp(W(n_{\text{G}}, n_{\text{H}})) \cdot W(n_{\text{G}}, n_{\text{H}})}{\|\tau\|} \\ \mu = \frac{W(n_{\text{G}}, n_{\text{H}})}{\|\tau\|}. \end{cases} \quad (\text{C.14})$$

The function W in equation C.14 is defined as

$$W(\alpha, \beta) = \mathcal{W}_0^{\text{L}} \left(\exp(1) \cdot \frac{\alpha}{\beta} + 1 \right) - 1 \quad (\text{C.15})$$

where \mathcal{W}_0^{L} denotes the *0th* branch of the Lambert-W function¹. The Lambert-W function is a set of functions (called branches) of the function $f(z) = z \exp(z)$ where z is a complex number. Hence

$$z = f^{-1}(z \exp(z)) = \mathcal{W}^{\text{L}}(z \exp(z)). \quad (\text{C.16})$$

¹For all real $x \geq 0$, the equation has exactly one real solution $y = \mathcal{W}^{\text{L}}(x) = \mathcal{W}_0^{\text{L}}(x)$.

For any complex number $z_0 = z \exp(z)$ we get

$$z_0 = \mathcal{W}^L(z_0) \exp(\mathcal{W}^L(z_0)) . \quad (\text{C.17})$$

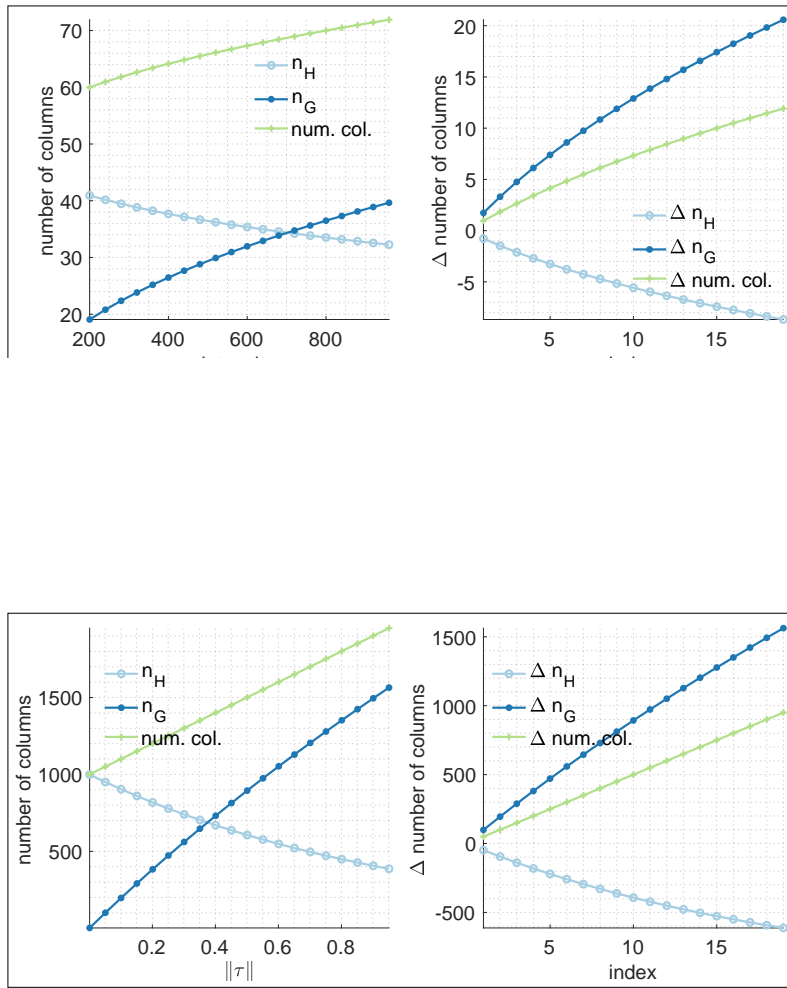


Figure C.4: Left: Graphical representation of n_H , n_G and their sum $\text{num.col} = n_H + n_G$ as function of τ . Right: Graphical representation of Δn_H , Δn_G and $\Delta \text{num.col}$ obtained, for instance, as $\Delta n_H[i] = n_H[i] - n_H[0]$ for $i = 1, \dots, 19$. The other parameters are: $\lambda = 100$, $\mu = 2$ and $\|\tau\| = 0.1$.

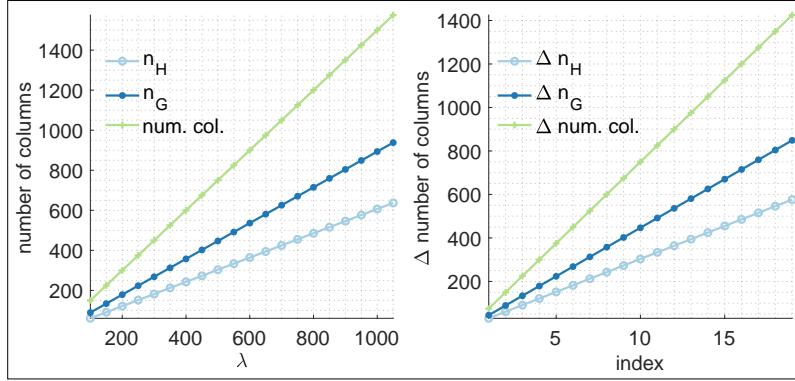


Figure C.5: Left: Graphical representation of n_H , n_G and their sum $\text{num. col} = n_H + n_G$ as function of τ . Right: Graphical representation of Δn_H , Δn_G and $\Delta \text{num. col}$ obtained, for instance, as $\Delta n_H[i] = n_H[i] - n_H[0]$ for $i = 1, \dots, 19$. The

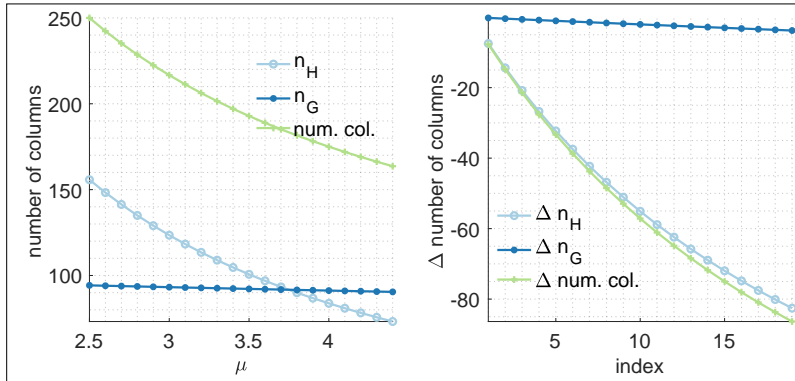


Figure C.6: Left: Graphical representation of n_H , n_G and their sum $\text{num. col} = n_H + n_G$ as function of τ . Right: Graphical representation of Δn_H , Δn_G and $\Delta \text{num. col}$ obtained, for instance, as $\Delta n_H[i] = n_H[i] - n_H[0]$ for $i = 1, \dots, 19$. The other parameters are: $\lambda = 100$, $\mu = 2$ and $\|\tau\| = 0.1$.

D

Doob-Gillespie method and PIP description

“If we knew what it was we were doing, it would not be called research, would it?”

– Albert Einstein

In this Appendix we aim to illustrate the connection between PIP and the Doob-Gillespie method (as mentioned in the original paper [13]) by showing how homologous sequences are created under the PIP model.

D.1 Doob-Gillespie method

The method proposed by *David Kendall*, in 1950, for the simulation of birth-death processes [67] has been extended to non pure-jump processes by *Joseph Leo Doob* (1942, 1945) and made popular by *Daniel Gillespie* (1976) for simulating chemical reactions. The *Doob-Gillespie algorithm* [9, 26, 27, 41, 42] – called also Gillespie’s Stochastic Simulation Algorithm – is a fast method to generate statistically correct trajectories of stochastic equations [37, 151]. This method, which mathematically is a variant of dynamic Monte Carlo method, avoid any rejection during the sampling phase and consequently it is very fast.

In the PIP article [13], the authors defined the PIP model using two different descriptions, namely the *local* and the *global* description. Moreover, the authors established the condition for which the two model descriptions coincide, that is by defining the Poisson insertion intensity as $\nu(dt) = \lambda \cdot (\tau(dt) + \frac{1}{\mu} \cdot \delta_{\Omega}(dt))$. In the following, we have looked at *local* description which can be useful to synthesize data under the PIP model, and with minor

changes also under TKF91.

To generate trajectory samples according to the underlying dynamical model and construct the correct statistical distribution of events, let us consider n_r event types labeled r , with their respective activity $\alpha_r = \gamma_r \cdot n_r$, where γ denotes the respective rate at which the event of type r occurs and n_r the number of “particles” (reads characters) subject to that event. Under the PIP model, the event types r are restricted to only three:

- (i) *insertions* with activity $\alpha_{\text{ins}} = \lambda \cdot 1$. PIP, differently from TKF91, uses only a single stochastic variable for the entire system;
- (ii) *deletions* with activity $\alpha_{\text{del}} = \mu \cdot n$, where n is the number of characters, and
- (iii) *substitutions* with α_{subs} according to the substitution rate matrix Θ , let say $\alpha_{\text{subs}} = \theta \cdot n$.

In order to stochastically simulate the evolution of a given sequence of N residues we need to be able to answer two questions:

1. *when* the next event happens (Sec. D.1.1) and
2. *what* kind of event it is (Sec. D.1.2).

Knowing at which time point (*when*) occurs an event of a given type (*what*) we will be able to simulate a statistically correct evolutionary trajectory under the PIP model.

D.1.1 When does the next event happen?

Suppose that the events occur in time according to a Poisson process with parameter α_r (where $r = \textit{insertion}, \textit{deletion}, \textit{substitution}$) and let t denote the interval of time until the first event. Then T is a continuous random variable with cumulative distribution as follow

$$P(T \leq t) = 1 - P(T > t) = 1 - P(\text{no events until time } t). \quad (\text{D.1})$$

The probability in equation D.1 that the first event happens after time t corresponds to the probability that there are no events until time t . From the Poisson distribution we can compute the probability distribution (CDF) of no events in a time interval t for an event type r which gets

$$P_r(t) = 1 - \frac{(\alpha_r t)^0 e^{-\alpha_r t}}{0!} = 1 - e^{-\alpha_r t}. \quad (\text{D.2})$$

The probability distribution function (PDF) computed deriving [D.2](#) with respect to t is

$$p(t_r) = \alpha_r e^{-\alpha_r t_r} \quad r = 1, \dots, n_r. \quad (\text{D.3})$$

In equation [D.3](#), $p(t_r)$ is the distribution of times till the first event. Assuming that event r occurred at time $t_r = t$, what is the probability that this event occurred before all others, given that it occurred at t ? The probability that t precedes t_2, t_3, t_4, \dots , indicated as $\mathbb{P}(r \text{ is first} \mid t_r = t)$ is

$$\begin{aligned} \mathbb{P}(r \text{ is first} \mid t_r = t) &= p(t < t_1) \cdot p(t < t_2) \cdot \dots \cdot p(t < t_{r-1}) \cdot p(t < t_{r+1}) \cdot \dots \\ &\quad \dots \cdot p(t < t_{n_r}) \\ &= \prod_{j \neq r} p(t < t_j) = \prod_{j \neq r} p(t_j > t). \end{aligned} \quad (\text{D.4})$$

From the result in equation [D.3](#), equation [D.4](#) can be re-written as

$$\begin{aligned} \mathbb{P}(r \text{ is first} \mid t_r = t) &= \prod_{j \neq r} p(t_j > t) = \prod_{j \neq r} \int_t^\infty \alpha_j e^{-\alpha_j t_j} dt_j \\ &= \prod_{j \neq r} e^{-\alpha_j t} \\ &= e^{\alpha_r t} \prod_j e^{-\alpha_j t} \\ &= e^{\alpha_r t} e^{-\sum_j \alpha_j t} \\ &= e^{\alpha_r t} e^{-\phi t}. \end{aligned} \quad (\text{D.5})$$

where

$$\phi = \sum_{j=1}^{n_r} \alpha_j \quad (\text{D.6})$$

represents the sum rates over the event types.

What is the probability that event r is the first event and that it occurs at time $t_r = t$? This joint probability can be computed from eq. [D.5](#)

$$\begin{aligned} \mathbb{P}(r \text{ is first} \wedge t_r = t) &= \mathbb{P}(r \text{ is first} \mid t_r = t) \mathbb{P}(t_r = t) \\ &= e^{\alpha_r t} e^{-\phi t} \alpha_r e^{-\alpha_r t} \\ &= \alpha_r e^{-\phi t}. \end{aligned} \quad (\text{D.7})$$

Since there are n_r competing events each of which could be the first one to occur, we have to sum together all their probabilities. The probability distribution $p(t)$ that the first event occurs at time t , given n_r possible events r , then becomes

$$\begin{aligned}
 p(t) &= \mathbb{P}(\text{first event occurs at } t) \\
 &= \sum_r^{n_r} \mathbb{P}(r \text{ is first} \wedge t_r = t) \\
 &= \sum_r^{n_r} \alpha_r e^{-\phi t} \\
 &= \phi e^{-\phi t}.
 \end{aligned} \tag{D.8}$$

The result obtained in D.8 for n_r competing events is similar to the result obtained in D.3 considering only one type of event. The combination of parallel memoryless processes leads to an exponential probability density with parameter $\phi = \sum_j \alpha_j$.

D.1.2 What kind of event happens next?

To simulate events under the PIP model we need to compute what is the probability p_r that the next event is r . This probability is obtained from the joint probability in equation D.7. In equation D.8 $p(t)$ has been obtained by marginalizing over all events r (D.7), now we have to integrate over all times, in this way

$$\begin{aligned}
 p_r &= \mathbb{P}(\text{the first event is } r) \\
 &= \int_0^\infty \mathbb{P}(r \text{ is first} \wedge t_r = t) dt = \int_0^\infty \alpha_r e^{-\phi t} dt \\
 &= \frac{\alpha_r}{\phi}.
 \end{aligned} \tag{D.9}$$

Equation D.8 and D.9 yield the basis for an algorithm that simulate different events competing among them.

D.2 Local PIP description

Under the PIP model substitutions and deletions are modeled by means of a continuous-time Markov chain with state space $\mathcal{A}_\epsilon = \mathcal{A} \cup \epsilon$. Insertions

of new characters are Poisson events defined on the topology. Considering that continuous-time Markov chains (CTMC's) can be seen as discrete-time Markov chains where the transition times occur according to a Poisson process and Poisson processes are themselves CTMC, we are allowed to treat insertion, deletions and substitutions in the same manner. Indeed, the waiting times between consecutive events (of any kind) are distributed according to an exponential distribution and each event occurs with a probability proportional to its rate. This property allows us to adapt Doob-Gillespie method also to the PIP model.

Pseudo-code

The algorithm shown below (Algorithm 1) along with Figures D.1-D.4 describe with pseudo-code and pictures how homologous sequences are generated according the PIP model using the Doob-Gillespie method briefly outlined above.

Algorithm 1 Doob-Gillespie-PIP procedure

```

1: procedure DOOB_GILLESPIE_PIP
2: init:
3:   for  $r \leftarrow 1, n_r$  do
4:      $\alpha_r \leftarrow \gamma_r \cdot n$ 
5:   end for
6:    $\phi = \sum_i^{n_r} \alpha_r$ 
7:    $T = 0$  ▷ reset the time
8: loop:
9:   for  $i \leftarrow 1, N$  do
10:    draw  $t$  according to:
11:     $p(t) = \mathbb{P}(\text{first event occurs at } t) = \phi e^{-\phi t}$ 
12:    advance the simulated time:  $T \leftarrow T + t$ 
13:    draw the next event  $r$  according to:
14:     $p_r = \mathbb{P}(\text{the first event is } r) = \frac{\alpha_r}{\phi}$ 
15:    update  $\alpha_r$  ▷ according to the ev. new number of characters
16:    update  $\phi$ 
17:   end for
18: end procedure

```

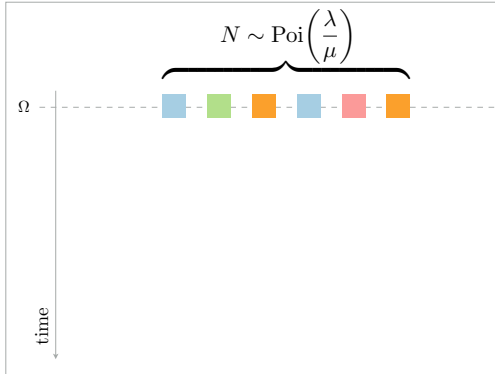


Figure D.1: Step 1) The process starts at the root. The Poisson process intensity at any point on the topology is given by $\nu(dt) = \lambda \cdot (\tau(dt) + \frac{1}{\mu} \cdot \delta_{\Omega}(dt))$ which at the root (at time 0) gives $\nu(0) = \lambda \cdot (\tau(0) + \frac{1}{\mu} \cdot \delta_{\Omega}(0)) = \frac{\lambda}{\mu}$. Hence, at the root, we can expect a number of character N distributed according to a Poisson distribution $N \sim \text{Poi}(\frac{\lambda}{\mu})$. At

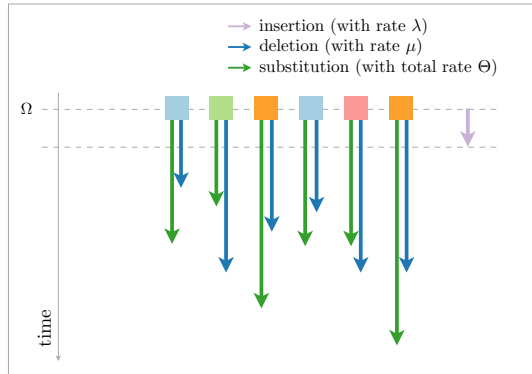


Figure D.2: Step 2) Each character draws an exponentially distributed random variable (waiting time of an exponential Poisson process) for the deletion (with rate μ), a second exponentially distributed random variable for the substitution (with total rate Θ , according to the substitution matrix), and then if selected a multinomial random

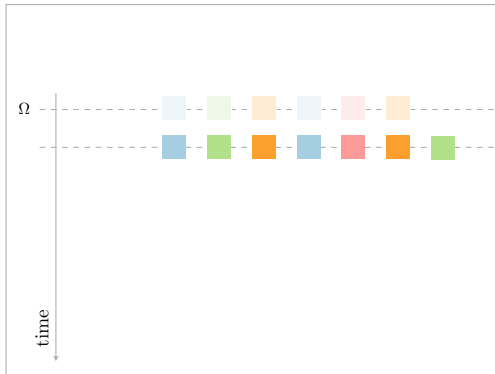


Figure D.3: Step 3) At step 2 the algorithm selects the smaller random variable, in this case the “winner” is the insertion random variable. To select the location of the insertion a uniformly distributed random variable is drawn and another multinomial random variable (distributed according to π_{ϵ}) is drawn to select the state of the new inserted character.

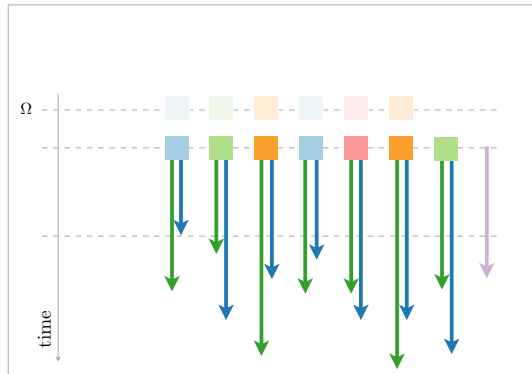


Figure D.4: Step 4) The “winner” random variable does not only select the next event but also the time point when the event happens. The Doob-Gillespie algorithm then continues by drawing for each character present at this time point 2 random variables for the substitution/deletion process and 1 variable for the entire system for the insertion.

2N+1 random variables (exponential distributed)

- insertion (with birth rate λ)
- deletion (with death rate μ)
- substitution (total rate Θ then multinomial with rates θ)

E

Grantham's distance

“There are in fact two things, science and opinion; the former begets knowledge, the latter ignorance.”

– Hippocrates

E.1 Introduction

The formulation described here exploits a mathematically sound Grantham's distance based metric defined on the space of the physicochemical properties of amino acids [48, 62]. The Grantham's distance is used as a measure of the physicochemical similarity of two molecules, which is used as “surrogate” for the likelihood that the two molecules might undergo a substitution in an evolutionary process. Substitutions between physico-chemically similar amino acids tend indeed to preserve the structure of proteins, and they are therefore more likely to occur during evolution. In the approach described by Katoh et al. [62], homologies between sequences are detected using a cross-correlation-based analysis of the physicochemical properties of the sequences residues. The computational complexity of the cross-correlation is reduced from N^2 to $N \log N$, where N is the average sequence length, by means of a Fast Fourier Transform (FFT) based algorithm.

However, the Grantham's distance implemented in our algorithm provides a more sound metric to identify homologous regions in the sequences to be aligned. Indeed, the cross-correlation can significantly underestimate the similarity of two amino acids when their physicochemical properties have small absolute values, yielding a small correlation coefficient, with respect to two dissimilar amino acids with properties characterized instead by large absolute

values, yielding therefore a large correlation.

Moreover, in our proposed approach, the space of physicochemical properties was extended to include also the chemical composition as suggested by Grantham in 1974 [48], besides the volume and polarity features already used in MAFFT. Because of the way the Grantham's distance is defined, our method can rely on a FT based approach for the identification of homologous sequences exactly as the MAFFT method, thus leading to the same computational complexity.

E.2 Grantham's distance computation

The Grantham's distance D_k can then be represented as a function of the positional lag k between two input sequences. Peaks in the distribution D_k identify lags k for which the columns of the two mutually shifted sequences present a high degree of physicochemical similarity. Since Fourier analysis cannot localize signals simultaneously in both time and frequency domain (see Section 5), the function D_k defines the shifts k where homologies can be found, without providing information on the location of the actual homologous regions within the sequences. In Section 5 we have addressed this issue by applying the short-time Fourier Transform (STFT). The use of the STFT, which is a time-frequency transform, localize simultaneously the shift and the position of putative homologous blocks.

An efficient way to quantify the similarity between amino acids consists in calculating a distance based on their physicochemical properties. Substitution processes between amino acids with similar physicochemical properties are more likely to occur during evolution since they tend to preserve the structural and functional properties of the proteins that they form. Therefore, the physicochemical distance between two amino acids can be interpreted as a surrogate measure of the likelihood of a substitution event during an evolutionary process.

The Grantham's distance [48] depends on three fundamental properties, that is, chemical composition c , polarity p and molecular volume v . In Section 5.4.4, Table 5.1, we listed the physicochemical properties, as defined by Grantham [48], and which have been used in our algorithm.

The Grantham's distance between a pair of amino acids i and j , is defined as,

$$D_{ij} = \sqrt{\alpha [c(i) - c(j)]^2 + \beta [p(i) - p(j)]^2 + \gamma [v(i) - v(j)]^2}, \quad (\text{E.1})$$

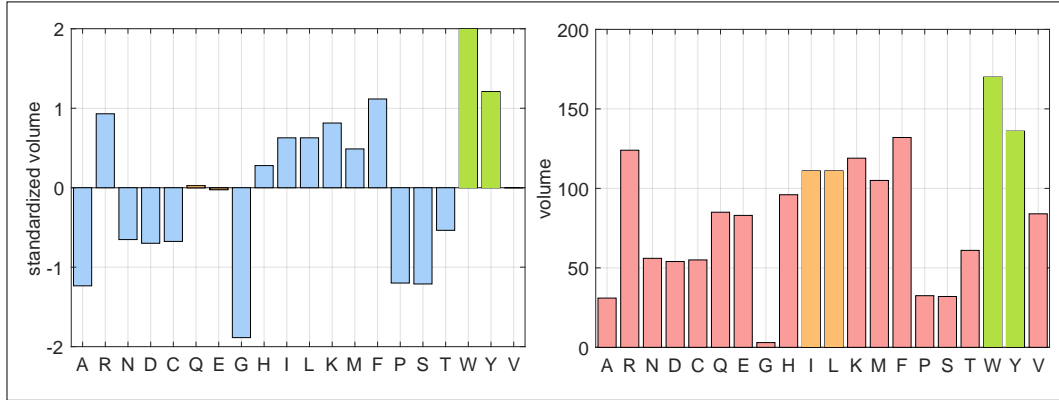


Figure E.1: Standardized and non standardized volumes. Left: standardized volumes as used in MAFFT for the cross correlation computation. Right: original volumes as described by Grantham [48] and used in our approach based on Grantham’s distance. From the left plot we can notice that two amino acids with similar volume (for instance Glutamine (Q) and Glutamic acid (E)) but at the same time a volume close to the average volume of the amino acids table will have a smaller correlation compared to two not so similar amino acids, for the point of view of the volumes, like Tryptophan (W) and Tyrosine (Y) but with a volume much different from the average volume. By taking the absolute volume difference, on the right side, two similar amino acids, like for instance Isoleucine (I) and Leucine (L) will have a lower distance (which would correspond to high correlation) than Tryptophan (W) and Tyrosine (Y) even if the latter have bigger volumes.

where α , β and γ are weights defined in such a way that the average value over the entire table is 100 (see [48] for more details). The larger the distance, the less similar the amino acids are, and therefore the less exchangeable they become during evolution.

The Grantham’s distance provides a more reliable measure of the similarity between amino acids than the cross-correlations c_v and c_p used by the authors of MAFFT [62]. The reason lies in the fact that the cross-correlations can underestimate the similarity of two amino acids when the absolute values of their physicochemical properties are small and, in the same way, overestimate it when the absolute values are large (see Figure E.1). So, for example, for both Isoleucine and Leucine one has $\hat{v} = 0.63$, where volumes are measured in \AA^3 and the symbol ‘ $\hat{\cdot}$ ’ represents a standardized volume¹. The cross-correlation

¹A standardized signal $\hat{\mathbf{s}}$ is defined in terms of its standardized components, i.e. $\hat{\mathbf{v}}$ and $\hat{\mathbf{p}}$. A standardized component is computed as $\hat{v} = (\mathbf{v} - \bar{v}) / \sigma_v$, where \bar{v} and σ_v are respectively the average and standard deviation computed over the input data rather than the 20 amino-acids in Table 5.1. Analogous definitions hold for $\hat{\mathbf{p}}$ and $\hat{\mathbf{c}}$.

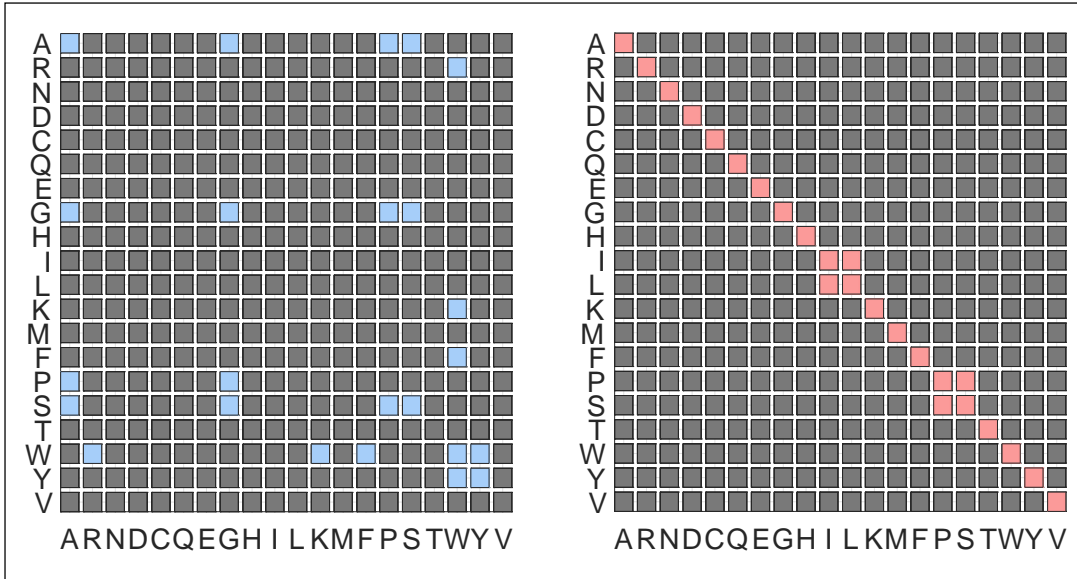


Figure E.2: Cross-correlation approach vs. Grantham's distance approach. Left: cross correlation table of the 20 amino acids volumes. Right: Grantham's distance calculation table for the volume component. The highest values should be placed in the table diagonal where the amino acids are the same and therefore there is no difference in the physicochemical component. In the right table this observation is respected whereas on the left table not. Apart the diagonal the next two couples with highest similarity of volumes are Isoleucine (I) with Leucine (L) and Proline (P) with Serine (S), as one would expect. This pattern is not respected under the cross-correlation approach proposed by the authors of MAFFT. In the latter approach the highest values are biased towards the biggest amino acids and not the most similar.

thus yields $c_v = \hat{v}_{\text{ILE}} \cdot \hat{v}_{\text{LEU}} = 0.39$. On the other hand, for the amino acids Tryptophan and Tyrosine one has $\hat{v}_{\text{TRP}} = 2.00$ and $\hat{v}_{\text{TYR}} = 1.21$, and thus $c_v = 2.42$. As clearly shown graphically in Figure E.1 amino acids Isoleucine and Leucine are physico-chemically more similar than Tryptophan and Tyrosine, whereas the cross-correlation c_v calculated as outlined in MAFFT [62] leads to the opposite inconsistent result. This simple calculation is presented here only for volumes for the sake of simplicity. However, the same argument holds for all other physicochemical properties. The Grantham's distance instead, considering only the volume property, yields $D_{\text{ILE,LEU}} = 0.01$ and $D_{\text{TRP,TYR}} = 0.68$, leading as expected to a smaller distance, and therefore a greater similarity, between Isoleucine and Leucine than between Tryptophan and Tyrosine.

Moreover, Figure E.2 clearly shows that the Grantham's distance (on the right panel) yields the largest physicochemical similarities along the diagonal, that is, between identical amino acids, as one would expect, unlike the cross-

correlations of MAFFT [62] (on the left panel) which depicts a different and misleading pattern. It should also be noticed that we use all three physicochemical properties c , p and v , whereas Katoh et al. [62] calculate only the cross-correlations c_p and c_v for the polarity and volume components, with a consequent loss of information.

For the Grantham's distance calculation, an amino acid i is replaced by a 3-dimensional vector of its physicochemical properties, $[c(i), p(i), v(i)]$, and a sequence of amino acids is thus transformed into a sequence of such vectors. By relating the lag j with respect to the position i , that is $j = i + k$, equation E.1 can be rewritten as

$$D_{i,i+k} = \sqrt{\alpha [c_1(i) - c_2(i+k)]^2 + \beta [p_1(i) - p_2(i+k)]^2 + \gamma [v_1(i) - v_2(i+k)]^2} \quad (\text{E.2})$$

with the subscripts 1 and 2 referring to sequences S_1 and S_2 , respectively. The squared Grantham's distance between the two sequences S_1 and S_2 , computed for site i from S_1 and applying a positional lag of k sites, at $i + k$ from S_2 , reads

$$D_{i,i+k}^2 = \alpha [c_1(i) - c_2(i+k)]^2 + \beta [p_1(i) - p_2(i+k)]^2 + \gamma [v_1(i) - v_2(i+k)]^2 . \quad (\text{E.3})$$

The squared Grantham's distance computed by means of equation E.3 over the entire length of the sequences, as function of the lags k , gives

$$\begin{aligned} \mathbf{D}_k^2 &= \sum_i D_{i,i+k}^2 \\ &= \sum_i \{ \alpha [c_1(i) - c_2(i+k)]^2 + \beta [p_1(i) - p_2(i+k)]^2 + \\ &\quad + \gamma [v_1(i) - v_2(i+k)]^2 \} \end{aligned} \quad (\text{E.4})$$

where \mathbf{D}_k is an array of length equal to the number of the applied lags. The coefficients α , β and γ are constants, hence equation E.4 can be rewritten as

$$\begin{aligned} \mathbf{D}_k^2 &= \alpha \sum_i [c_1(i) - c_2(i+k)]^2 + \beta \sum_i [p_1(i) - p_2(i+k)]^2 + \\ &\quad + \gamma \sum_i [v_1(i) - v_2(i+k)]^2 . \end{aligned} \quad (\text{E.5})$$

Expanding the square product in equation E.5 we get

$$\begin{aligned} \mathbf{D}_k^2 &= \alpha \sum_i [c_1(i)^2 + c_2(i+k)^2 - 2c_1(i)c_2(i+k)] + \\ &\quad + \beta \sum_i [p_1(i)^2 + p_2(i+k)^2 - 2p_1(i)p_2(i+k)] + \\ &\quad + \gamma \sum_i [v_1(i)^2 + v_2(i+k)^2 - 2v_1(i)v_2(i+k)] \\ &= \alpha \cdot \mathbf{C}_k^2 + \beta \cdot \mathbf{P}_k^2 + \gamma \cdot \mathbf{V}_k^2. \end{aligned} \quad (\text{E.6})$$

In equation E.6, \mathbf{C}_k^2 , \mathbf{P}_k^2 and \mathbf{V}_k^2 refer to the chemical, polarity and volume component, respectively. Since \mathbf{C}_k , \mathbf{P}_k , \mathbf{V}_k , in equation E.6, have the same structure let us proceed the analysis, for a moment, only with \mathbf{V}_k . Splitting the summation, equation E.6, we can rewrite the volume component as

$$\begin{aligned} \mathbf{V}_k^2 &= \gamma \sum_i [v_1^2(i) + v_2^2(i+k) - 2v_1(i)v_2(i+k)] \\ &= \gamma \left[\sum_i v_1^2(i) + \sum_i v_2^2(i+k) - 2 \sum_i v_1(i)v_2(i+k) \right]. \end{aligned} \quad (\text{E.7})$$

The last term of equation E.7 scales as $\mathcal{O}(N^2)$, where N is the average sequence length. However, using the well-known convolution theorem, in the same manner as Katoh et al. [62], the FFT based procedure reduces the complexity of that calculation to $\mathcal{O}(N \log N)$. Hence, the cross correlation in equation E.7 using the Fourier Transform, becomes

$$\mathbf{V}_k^2 = \gamma \left[\sum_i v_1^2(i) + \sum_i v_2^2(i+k) - 2 \cdot \mathcal{F}^{-1}\{\mathcal{F}\{\mathbf{v}_1\} \cdot \mathcal{F}\{\mathbf{v}_2\}\} \right] \quad (\text{E.8})$$

where \mathcal{F} denotes the Fourier transform and \mathcal{F}^{-1} its inverse. Applying the same procedure also to \mathbf{C}_k and \mathbf{P}_k , equation E.6 turns to

$$\mathbf{D}_k^2 = \alpha \left[\sum_i c_1^2(i) + \sum_i c_2^2(i+k) - 2 \cdot \mathcal{F}^{-1}\{\mathcal{F}\{\mathbf{c}_1\} \cdot \mathcal{F}\{\mathbf{c}_2\}\} \right] + \quad (\text{E.9})$$

$$+ \beta \left[\sum_i p_1^2(i) + \sum_i p_2^2(i+k) - 2 \cdot \mathcal{F}^{-1}\{\mathcal{F}\{\mathbf{p}_1\} \cdot \mathcal{F}\{\mathbf{p}_2\}\} \right] + \quad (\text{E.10})$$

$$+ \gamma \left[\sum_i v_1^2(i) + \sum_i v_2^2(i+k) - 2 \cdot \mathcal{F}^{-1}\{\mathcal{F}\{\mathbf{v}_1\} \cdot \mathcal{F}\{\mathbf{v}_2\}\} \right]. \quad (\text{E.11})$$

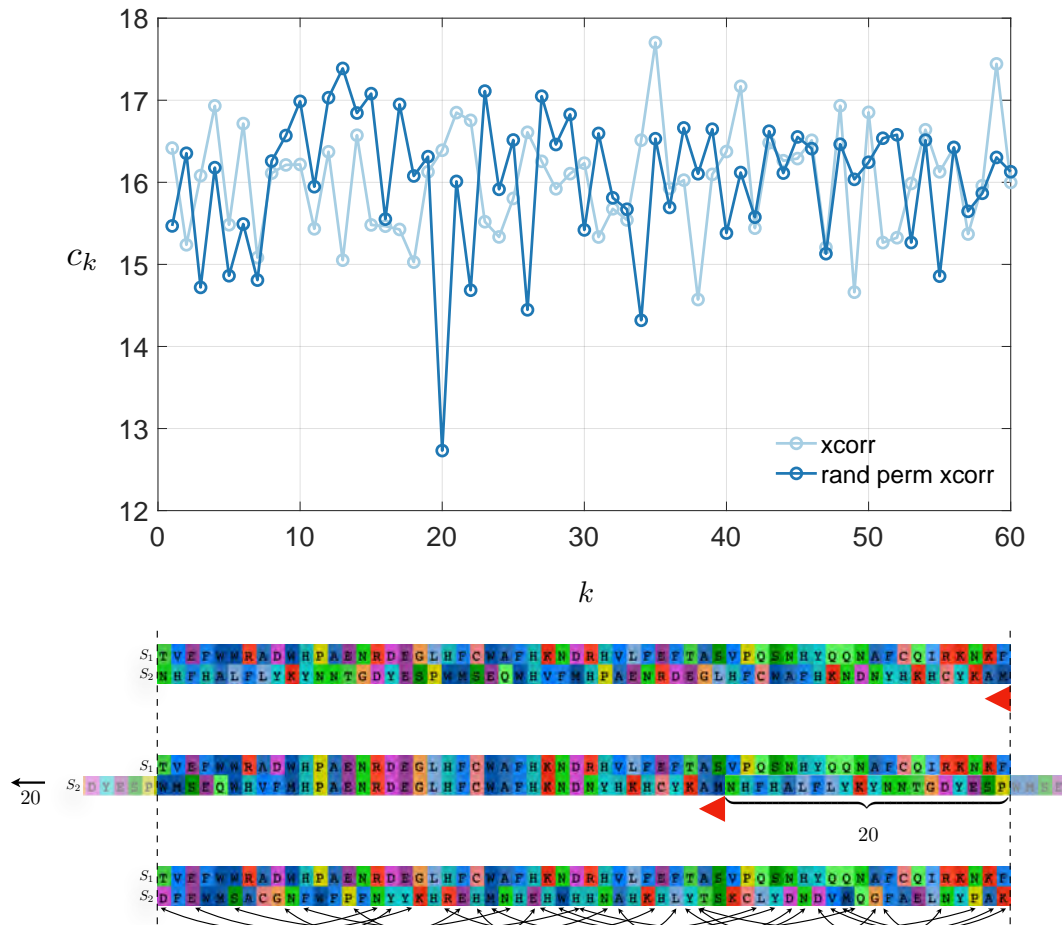


Figure E.3: Grantham's distance coefficients for a small pairs of sequences. Top: Grantham's distance coefficients for the two sequences S_1 and S_2 , shown below in sub-figure ii). Sequence S_2 shows the same pattern present in S_1 when shifted on the left direction by 20 positions, as depicted in sub-figure i) and iii). Indeed, the peak, minimum value, in sub-figure i) is relative to the positional lag to be applied in order to align the two patterns. The 'noise' threshold that segregates the Grantham's coefficients related to the presence of similar pattern in the two signals is computed by destroying statistically all the pattern contained. This is achieved by permuting randomly the order of the columns in S_2 and recomputing the Grantham's coefficients. In this way any patterns eventually present is destroyed and the coefficients refer to two random sequences. Our proposed approach has the advantage of reusing the same property values, only in another order, and therefore the so obtained threshold depends on the data at hand.

It should be noted that S_1 and S_2 can represent alignments rather than simple sequences. In that case the properties $c_1(i)$ and $c_2(i+k)$ of equation E.2 represents the column-wise average of the corresponding property at positions i and $i+k$, respectively. Moreover, for a gap in a column at a given position j we assume $c(j) = p(j) = v(j) = 0$.

Furthermore, we observe that the distance $D_k = \sqrt{D_k^2}$ between two sequences or alignments can be interpreted as the \mathbf{L}^2 -norm defined on the space of the physicochemical properties of the amino acids. For two identical signals, i.e., sequences or alignments, one has $D_k = 0$.

Finally, it should be mentioned that a similar distance-based approach can be extended to nucleotides. In that case, the alphabet includes 4 nucleotides and a gap character, that is, $\{A, C, G, T, -\}$. Each residue in a nucleic acid sequence is then converted into a 5-dimensional vector with 1 at the position corresponding to that character, and 0 elsewhere. So, for example, a Cytosine would be represented as $[0, 1, 0, 0, 0]$ [35].

F

Homologous blocks overlap resolution

“Mathematics reveals its secrets only to those who approach it with pure love, for its own beauty.”

– Archimedes

The short-time Fourier Transform detects candidate homologous blocks which corresponds to regions in the two sub-alignments or sequences with high physicochemical similarity. These candidate homologous regions define a set \mathcal{B} of blocks in a MSA homology matrix. The homologous blocks need to be selected, arranged and connected in a consistent way for designing a meaningful final alignment.

Prior to independently align the single blocks an algorithm has to connect them logically in a path that respect the order of the columns and avoid any duplication. This is achieved by first identifying within the set \mathcal{B} the set \mathcal{P} of all possible paths that connect the candidate blocks potentially yielding a final meaningful alignment. Firstly, one needs to identify the set \mathcal{P} of all possible consistent pathways through those blocks. A generic block $b_j \in \mathcal{B}$ is characterized by the coordinates of its upper-left (u) and bottom-right (v) vertexes, $\{[u_{j,1}, u_{j,2}], [v_{j,1}, v_{j,2}]\}$. The two subscripts 1, 2 refer to sequences S_1 and S_2 . Consider for example another block b_k adjacent to the first one with coordinates $\{[u_{k,1}, u_{k,2}], [v_{k,1}, v_{k,2}]\}$. The condition to connect the two blocks b_j and b_k to form a meaningful path is $(u_{k,1} \geq u_{j,1}) \wedge (u_{k,2} \geq u_{j,2})$. This procedure is iterated for all adjacent blocks finally yielding the sought pattern of possible paths \mathcal{P} .

The removal of the overlaps should alter in the least possible way the structure of homologous regions that was previously identified. Therefore, the

overlaps are removed by replacing the overlapping blocks by two re-sized blocks that retain the largest possible part of the diagonal elements of the original ones without exhibiting any more overlaps. The diagonal corresponds to matches in the DP alignment and represents therefore the expected traceback path that we want to preserve as much as possible.

Our procedure allows us to remove overlaps only between two consecutive blocks. Therefore given the set \mathcal{P} the algorithm iterates in each path p_i and for all pairs of consecutive blocks in this path check and if needed resolve the overlaps, until all overlaps are removed. The algorithm generates in this way the set $\tilde{\mathcal{P}}$ of overlap-free paths in which we choose the optimal path p^* that maximizes a given target condition. Finally, among the overlap-free paths $\tilde{\mathcal{P}}$ we select the optimal path p^* bearing the largest homologous regions, that is, the largest total number of columns.

The pseudocode and the structure of the algorithm that first resolve the potential overlaps, successively builds candidate paths and finally select the optimal one is sketched here below.

Algorithm 2 Overlap resolution procedure

```

1: procedure OVERLAPRESOLUTION(blocks)
2:   blockssorted  $\leftarrow$  SORTBLOCKS(blocks)  $\triangleright$  sort blocks by first coordinate
3:   treeblocks  $\leftarrow$  CREATEBLOCKTREE(blockssorted)
4:   pathsoverlapFree  $\leftarrow$  RESOLVEOVERLAPS(treeblocks, blockssorted)
5:    $p^* \leftarrow$  GETBESTPATH(pathsoverlapFree)  $\triangleright p^*$  is the best path
6:   return  $p^*$ 
7: end procedure

```

Algorithm 3 Create block tree procedure

```

1: procedure CREATEBLOCKTREE
2:   rootbtree  $\leftarrow$  createNode()
3:    $n \leftarrow |\mathcal{B}|$ 
4:   for doi  $\leftarrow n$ 
5:     nodei  $\leftarrow$  createNode()
6:     insertChildNode(root, nodei)
7:     EXPANDNODE(nodei)
8:   end for
9: end procedure

```

Algorithm 4 Expand node procedure

```

1: procedure EXPANDNODE( $node_i$ )
2:    $n \leftarrow |\mathcal{B}|$ 
3:   for  $i \leftarrow 1, n - 1$  do
4:      $(a_1, a_2) \leftarrow coord(\mathcal{B}_i)$ 
5:     for  $j \leftarrow i + 1, n$  do
6:        $(b_1, b_2) \leftarrow coord(\mathcal{B}_j)$ 
7:       if  $b_1 \geq a_1 \wedge b_2 \geq a_2$  then
8:          $addNewNode(node)$ 
9:       end if
10:    end for
11:  end for
12:  for  $i \leftarrow 1, n$  do CREATEBLOCKTREE( $node$ )
13:  end for
14: end procedure

```

Algorithm 5 Resolve all overlaps procedure

```

1: procedure RESOLVEOVERLAPS( $tree_{blocks}, blocks_{sorted}$ )
2:    $\mathcal{P} \leftarrow getPathS(tree_{blocks}, blocks_{sorted})$ 
3:   repeat
4:      $numPaths \leftarrow |\mathcal{P}|$ 
5:     for  $i \leftarrow 1, numPaths$  do
6:        $\tilde{p}_i \leftarrow RESOLVESINGLEPATH(p_i)$ 
7:     end for
8:   until not all overlaps are resolved
9:   return  $\tilde{p}$  ▷ overlap-free paths
10: end procedure

```

Algorithm 6 Resolve single path procedure

```

1: procedure RESOLVESINGLEPATH( $p_i$ )
2:    $pairs \leftarrow getBlockPairs(p_i)$  ▷ split path into block pairs
3:    $pairs_{resolved} \leftarrow RESOLVEPAIRS(pairs)$ 
4:    $path_{resolved} \leftarrow connectPairs(pairs_{resolved})$  ▷ connect pairs to rebuild path
5:   return  $path_{resolved}$  ▷ overlap-free path
6: end procedure

```

Algorithm 7 Resolve pairs procedure

```

1: procedure RESOLVEPAIRS(pairs)
2:   for  $i \leftarrow 1, size(pairs) - 1$  do
3:      $(\tilde{p}_i, \tilde{p}_{i+1}) \leftarrow \text{RESOLVEOVERLAPPAIR}(p_i, p_{i+1})$ 
4:      $p_{i+1} \leftarrow \tilde{p}_{i+1}$ 
5:   end for
6:   return  $\tilde{p}$  ▷ overlap-free pairs
7: end procedure

```

Algorithm 8 Resolve overlap pairs procedure

```

1: procedure RESOLVEOVERLAPPAIR( $p_i, p_{i+1}$ )
2:    $type_{\text{overlap}} \leftarrow \text{getOverlapType}(p_i, p_{i+1})$ 
3:    $(\tilde{p}_i, \tilde{p}_{i+1}) \leftarrow \text{resolveOverlapType}(pairs_i, pairs_{i+1}, type_{\text{overlap}})$ 
4:   return  $(\tilde{p}_i, \tilde{p}_{i+1})$ 
5: end procedure

```

Algorithm 9 Search the best path procedure

```

1: procedure GETBESTPATH(p)
2:    $score_{\text{max}} \leftarrow -\infty$ 
3:    $p^* \leftarrow \emptyset$ 
4:    $numPaths \leftarrow sizeof(p)$ 
5:   for  $i \leftarrow 1, numPaths$  do
6:      $s \leftarrow 0$ 
7:      $numBlocksInPath \leftarrow sizeof(p_i)$ 
8:     for  $j \leftarrow 1, numBlocksInPath$  do
9:        $block_{\text{size}} \leftarrow p_{i,j}$ 
10:       $s \leftarrow s + block_{\text{size}}$ 
11:    end for
12:    if  $s > score_{\text{max}}$  then
13:       $score_{\text{max}} \leftarrow s$ 
14:       $p^* \leftarrow p_i$ 
15:    end if
16:  end for
17:  return  $(p^*, s)$ 
18: end procedure

```

		1	2	3	4	5	6	7	8	9	10	11	12	13
A														
B														
C														
D														
E														
F														
G														
H														
I														
L														
M														
N														
O														

Figure F.1: Overlaps table. A generic block $b_j \in \mathcal{B}$ is characterized by the coordinates of its upper-left (u) and bottom-right (v) vertexes, $\{[u_{j,1}, u_{j,2}], [v_{j,1}, v_{j,2}]\}$. The two subscripts 1,2 refer to sequences S_1 and S_2 . This Table shows the different type of overlap dictated by the relative position of two blocks b_j, b_k . The blue color refers to S_1 while pink color to S_2 . On the vertical axis are depicted the relative position of the box edge $[u_{j,1}, v_{j,1}]$ in blue and the box edge $[u_{k,1}, v_{k,1}]$. On the horizontal axis are depicted the relative position of the box edge $[u_{j,2}, v_{j,2}]$ in blue and the box edge $[u_{k,2}, v_{k,2}]$. From each type of overlap in the Table, for instance G2, or D8, ..., we have defined a general strategy for its resolution.

G

Multiple sequence alignment evaluation

“But in my opinion, all things in nature occur mathematically.”

– René Descartes

G.1 Overview

The major goals of evaluating multiple sequence alignment software packages are typically:

- (i) to demonstrate the accuracy of a new heuristic strategy in achieving better sum-of-pairs(SP) scores;
- (ii) to exhibit the results obtained with a tools regarding the accuracy against benchmarks [103], real or simulated proteins [132]
- (iii) or to quantify the alignment accuracy vis-à-vis real data in a reference-independent manner [25, 45, 106].

The most frequently adopted method to assess the accuracy of multiple sequence alignment algorithms is by using reference alignments from benchmark test sets. Example of such benchmark datasets include HomFam [12, 129], BALiBASE [142], 10AA collection [121], Prefab [30] and The Comparative Ribosomal Website (for RNA alignments) [18]. The BALiBASE benchmark database, for instance, contains multiple sequence alignments organized in different reference datasets [143] comprehending specific problems such as (i) small numbers

of sequences; (ii) unequal phylogenetic distributions; (iii) large N/C-terminal extensions; (iv) internal insertions; (v) repeats; (vi) inverted domains and (vii) transmembrane regions.

It is worth to note, however, that *structural alignments* of real protein sequences are in general very different from *evolutionary alignments* where attempts are made to infer the most plausible homology paths. Regrettably, structural alignments of well known proteins are readily accessible and therefore are in practice the standards against which alignment tools are often measured. However, benchmarking an evolutionary based alignment against BALiBASE can be criticized on the following grounds [58] (i) inconsistencies with known annotations from external sources; (ii) debatable choices of parameters in automatic structural alignment procedures; (iii) Use of primary sequence-alignment procedures like BLAST, employing substitution matrices and gap penalty scores, and, thus, making modelling assumptions (iv) subjective biases through manual curation of primary sequence and structural alignments.

Moreover, BALiBASE (v2.0) [7, 141] and HOMSTRAD [93] are – among others – two well-know benchmark databases of *hand-curated* structural alignments. Since hand curation is not easily reproducible, is highly subjective and lacks standard protocols other databases have been created relying on automated structural alignment rules. Among them we can list, for instance, SABmark [157] along with PREFAB [30], OxBench [124] and partially also BALiBASE (v3.0) [142]. To cope with ambiguous alignment regions, in several databases the portions annotated as *reliable* are retained while the remaining are “filtered out” so that only the selected regions are then measured with respect to the accuracy. Nonetheless, none of these databases is actually appropriate for benchmarking evolutionary based alignments.

For alignments based on the evolution, there are programs that simulate substitutions, as well as insertions and deletions events at the molecular level providing synthetic datasets useful for detailed performance studies of aligners. Among others SIMPROT [109, 113] or Rose [135]. Even though in simulated datasets the true homology histories as well as the relationships between the input sequences are known, therefore the obtained performances in term of accuracy in reconstructing the synthetic dataset are not generalizable for true data. Another important consideration must be borne in mind when evaluating alignments accuracy, i.e. often commonly used reference alignments are derived from biased samples of proteins and RNA[31, 66, 98, 118, 162]. Therefore multiple sequence alignment tools calibrated according to them might be also biased [3, 58, 98, 107]. Unfortunately, apart from small empirical datasets,

there are no gold-standard benchmarks specific to homology alignments [98]. An alternative approach to the comparison against standard benchmarks are synthesized data coming from simulations [127], however, such datasets often contradict those based on the gold standards [66, 73, 74, 85] suggesting that such approach may be inadequate [58]. Furthermore, it should not be forgotten that the more the model implemented in the software package is similar to the model coded in the simulator the better would be the results, but both might be inconsistent with real biological phenomena.

Finally, functional and structural concordance does not imply evolutionary homology, as it may have arisen independently by evolutionary convergence. Therefore, structural benchmarks are particularly inappropriate to evaluate a phylogenetic aligner, whose objective is evolutionary homology.

G.2 Benchmarks

Even though recent literature shows that phylogenetic aligners are outperformed in structural benchmarks, but perform well in phylogenetic tests (an indeed our tool as well), a comparison against BALiBASE benchmarks is often expected or required. In Table G.1 we have thence, summarized the results of our benchmarks.

In Figures G.2-G.8 we are also showing an example of alignment under PIP compared with the reference alignment of BALiBASE. The phylogenetic tree in Figure G.1 and the alignments referring both to *RV911/Box115* dataset . This datasets contains 7 amino acids sequences from

1. *Xenopus tropicalis* (Western clawed frog): A1A5F0;
2. *Danio rerio* (Zebrafish): Q52KB5;
3. *Homo sapiens* (Human): O15060;
4. *Drosophila melanogaster* (Fruit fly): P17789;
5. *Drosophila melanogaster*: Q24174;
6. *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4;
7. *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

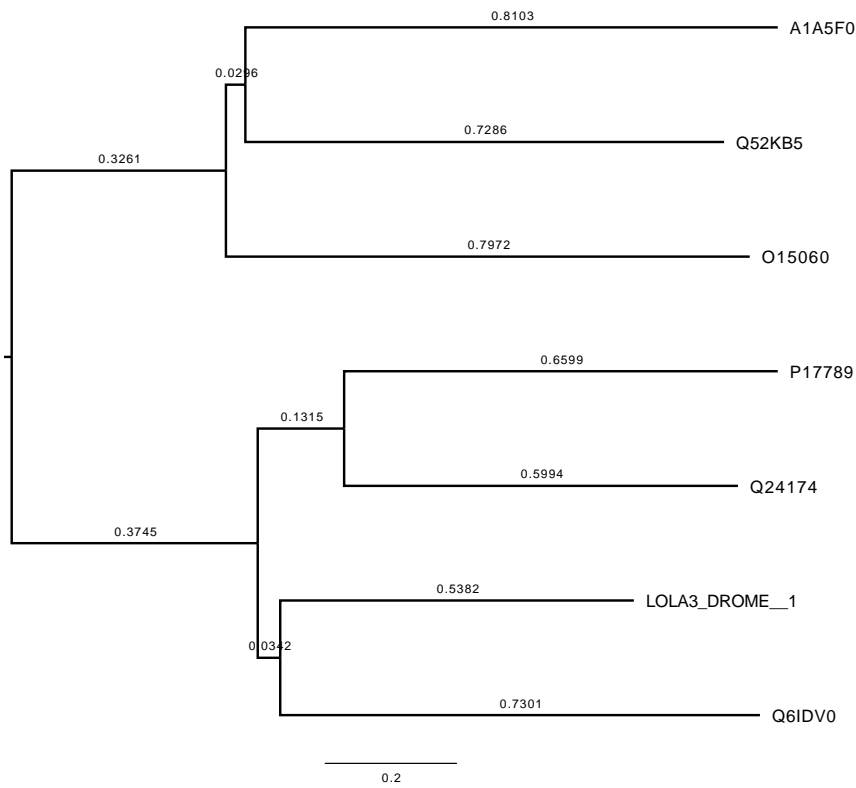


Figure G.2: Evolutionary vs Structural alignment. Columns (1-200). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4, *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

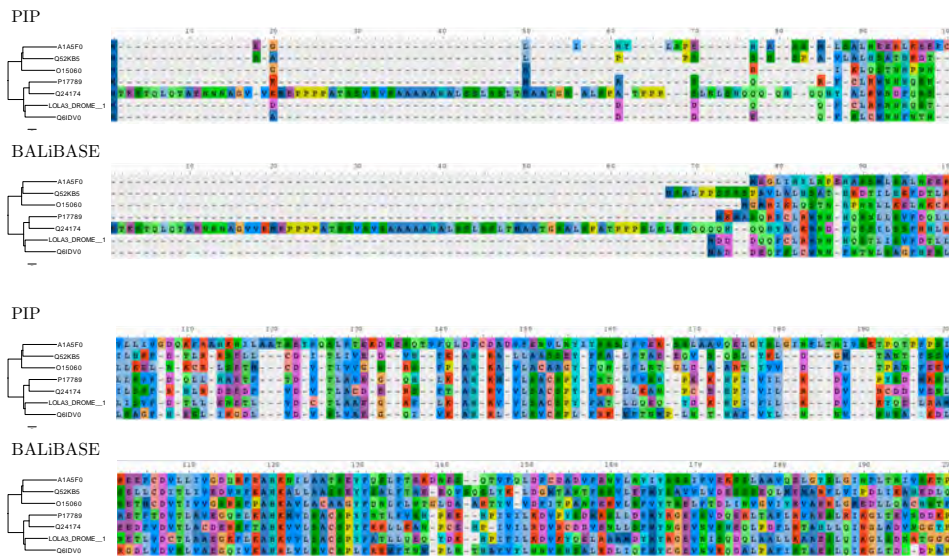


Figure G.2: Evolutionary vs Structural alignment. Columns (1-200). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4, *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

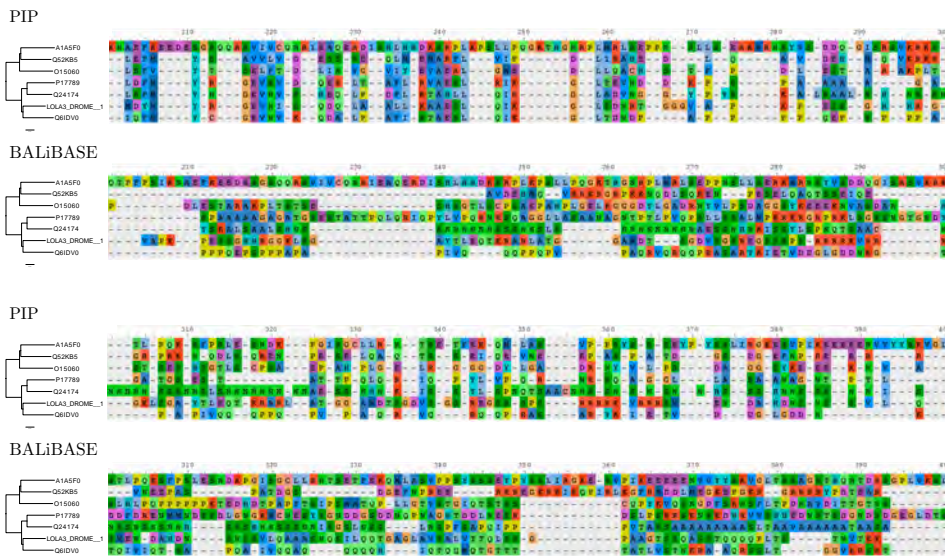


Figure G.3: Evolutionary vs Structural alignment. Columns (201-400). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7K

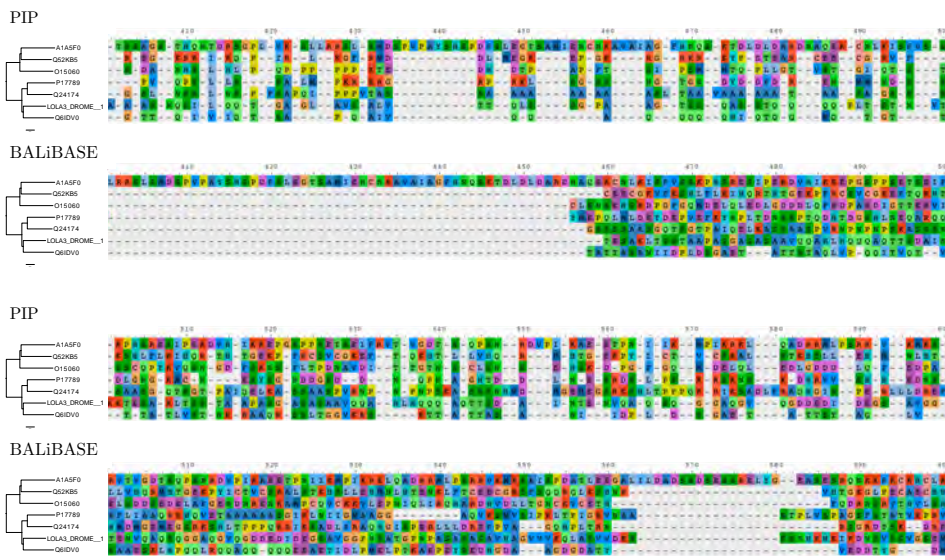


Figure G.4: Evolutionary vs Structural alignment. Columns (401-600). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4, *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

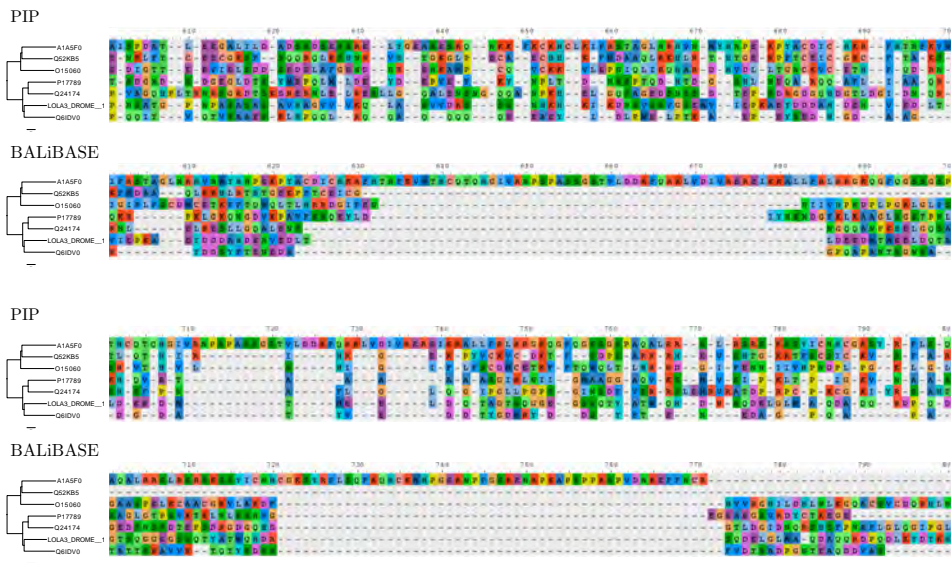


Figure G.5: Evolutionary vs Structural alignment. Columns (601-800). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4, *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

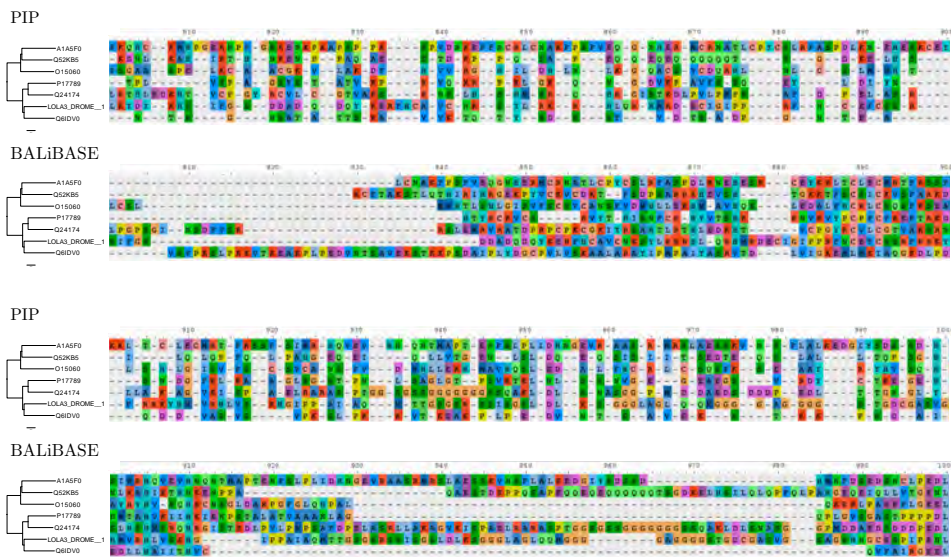


Figure G.6: Evolutionary vs Structural alignment. Columns (801-1000). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4, *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

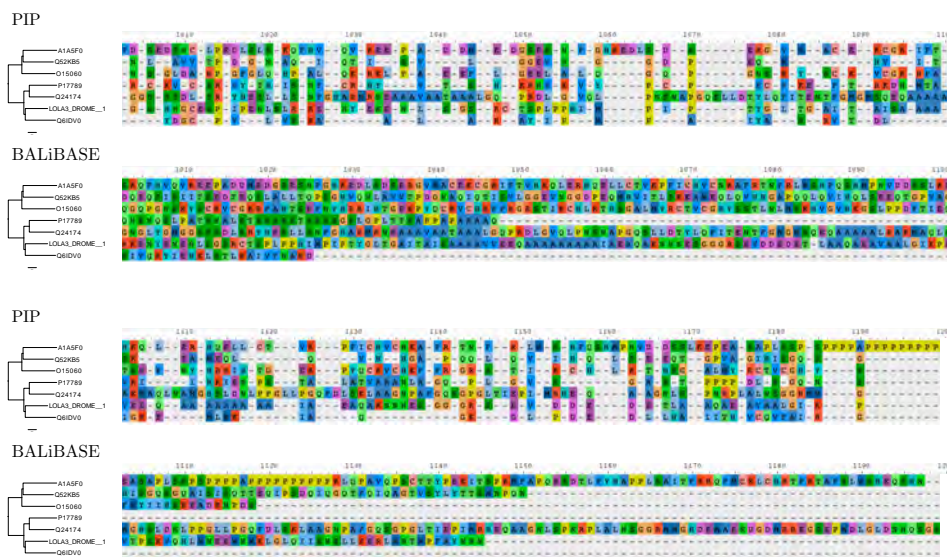


Figure G.7: Evolutionary vs Structural alignment. Columns (1001-1200). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4, *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

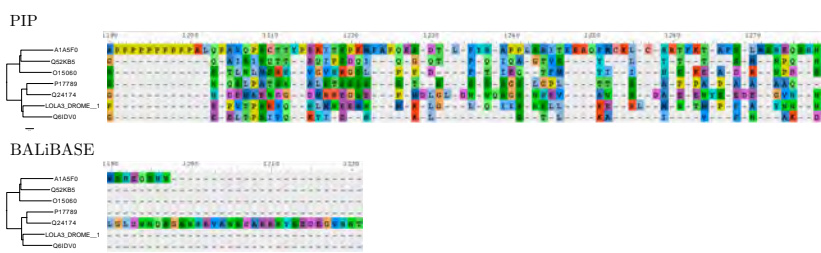


Figure G.8: Evolutionary vs Structural alignment. Columns (1190-...). BALiBASE benchmark datasets RV911-115 containing 7 amino acids sequences from *Xenopus tropicalis* (Western clawed frog): A1A5F0, *Danio rerio* (Zebrafish): Q52KB5, *Homo sapiens* (Human): O15060, *Drosophila melanogaster* (Fruit fly): P17789, *Drosophila melanogaster*: Q24174, *Drosophila melanogaster* (LOLA3_DROME): Q7KQZ4, *Drosophila pseudoobscura pseudoobscura* (Fruit fly): Q6IDV0.

	RV911				RV912				RV913					
#10	0.0868	0.0000	0.0000	0.0881	#11	0.6090	0.4020	0.6170	0.5840	#12	0.8490	0.6800	0.8670	0.8450
#22	0.2160	0.0090	0.1650	0.2390	#32	0.7640	0.4960	0.7850	0.7550	#17	0.6820	0.5140	0.7070	0.6850
#32	0.3560	0.1270	0.4070	0.3800	#45	0.4700	0.2010	0.4680	0.4760	#32	0.9020	0.7920	0.9080	0.9020
#34	0.2060	0.0105	0.1400	0.2030	#47	0.7260	0.4620	0.7270	0.7210	#34	0.9000	0.7690	0.9130	0.9050
#35	0.1700	0.0007	0.1010	0.1560	#49	0.7660	0.3950	0.8160	0.7870	#36	0.9100	0.7870	0.9200	0.9170
#46	0.0272	0.0000	0.0000	0.0236	#50	0.6000	0.2710	0.6330	0.6100	#43	0.9430	0.8440	0.9530	0.9450
#60	0.1390	0.0301	0.1360	0.1980	#54	0.5220	0.3370	0.5030	0.5140	#49	0.9240	0.7940	0.9430	0.9330
#63	0.4730	0.2600	0.4940	0.4610	#60	0.5820	0.2080	0.6360	0.6520	#63	0.8770	0.7130	0.8960	0.8720
#76	0.1520	0.0037	0.0915	0.1610	#75	0.6650	0.2390	0.7100	0.6840	#75	0.9690	0.9150	0.9710	0.9680
#96	0.0748	0.0133	0.0000	0.0708	#76	0.7990	0.5410	0.8360	0.8050	#76	0.9370	0.8640	0.9510	0.9380
#115	0.1650	0.0699	0.0880	0.1560	#96	0.7260	0.5180	0.7460	0.7280	#79	0.9260	0.8220	0.9300	0.9260
#121	0.2590	0.0127	0.2220	0.2470	#121	0.6940	0.3940	0.7120	0.6880	#82	0.9590	0.9030	0.9670	0.9600
#122	0.5660	0.2380	0.5990	0.5530	#122	0.7680	0.4850	0.8020	0.7730	#121	0.8300	0.6190	0.8570	0.8400
#123	0.2440	0.1130	0.2480	0.2510	#142	0.7110	0.4900	0.7730	0.7110	#122	0.9390	0.8650	0.9520	0.9400
#142	0.0790	0.0036	0.0147	0.0796	#149	0.6770	0.3160	0.7090	0.6880	#126	0.9430	0.8700	0.9580	0.9450
#172	0.1370	0.0092	0.0869	0.1380	#154	0.4800	0.2600	0.4200	0.4510	#132	0.8630	0.6500	0.8770	0.8730
#175	0.2380	0.1090	0.1370	0.2140	#175	0.4720	0.3030	0.4590	0.4520	#133	0.8650	0.7270	0.8830	0.8680
#177	0.2540	0.0549	0.2420	0.2430	#177	0.6880	0.4560	0.7190	0.6970	#142	0.9680	0.9270	0.9750	0.9690
#180	0.3260	0.0290	0.3220	0.3230	#187	0.6350	0.4990	0.6500	0.6240	#146	0.8940	0.7510	0.8890	0.8750
#181	0.3050	0.1970	0.3100	0.3150	#192	0.3130	0.1640	0.3170	0.3410	#158	0.9490	0.6650	0.9560	0.9510
#192	0.2400	0.0566	0.2690	0.3240	#202	0.4580	0.1790	0.4430	0.4800	#180	0.8760	0.7710	0.9050	0.8810
#212	0.1520	0.0202	0.1160	0.1470	#214	0.6390	0.2490	0.6970	0.6670	#183	0.8560	0.6570	0.8770	0.8590
#240	0.1240	0.0000	0.0000	0.1290	#240	0.5500	0.3480	0.5430	0.5510	#214	1.0000	1.0000	1.0000	1.0000
#246	0.0518	0.0000	0.0000	0.0489	#246	0.5570	0.2670	0.5620	0.5690	#222	0.8320	0.6580	0.8370	0.8300
#258	0.4340	0.1800	0.4510	0.4330	#258	0.5470	0.1610	0.5410	0.5520	#246	0.7500	0.5390	0.7360	0.7600
#284	0.2090	0.0191	0.2150	0.2150	#259	0.7680	0.5490	0.7900	0.7730	#258	0.7670	0.5360	0.7660	0.7620
					#281	0.5170	0.2610	0.5290	0.5310	#290	0.9660	0.9200	0.9750	0.9660
					#290	0.7410	0.5820	0.7630	0.7400					
mean	0.2186	0.0602	0.1867	0.2230		0.6230	0.3583	0.6395	0.6287		0.8917	0.7612	0.9026	0.8931
std	0.1315	0.0790	0.1655	0.1316		0.1217	0.1312	0.1396	0.1217		0.0737	0.1293	0.0731	0.0732

Table G.1: PIP based MSA from three BAliBASE datasets, namely RV911, RV912 and RV913.

Bibliography

- [1] M. E. Abram, A. L. Ferris, W. Shao, W. G. Alvord, and S. H. Hughes. Nature, position, and frequency of mutations made in a single cycle of HIV-1 replication. *Journal of Virology*, 84(19):9864–9878, jul 2010.
- [2] Stephen F. Altschul and Warren Gish. [27] local alignment statistics. In *Computer Methods for Macromolecular Sequence Analysis*, volume 266 of *Methods in Enzymology*, pages 460 – 480. Academic Press, 1996.
- [3] M. R. Aniba, O. Poch, and J. D. Thompson. Issues in bioinformatics benchmarking: the case study of multiple sequence alignment. *Nucleic Acids Research*, 38(21):7353–7363, jul 2010.
- [4] Maria Anisimova, Gina Cannarozzi, and David A. Liberles. Finding the balance between the mathematical and biological optima in multiple sequence alignment. *Trends in Evolutionary Biology*, 2(1):7, nov 2010.
- [5] F. Armougom, S. Moretti, O. Poirot, S. Audic, P. Dumas, B. Schaeli, V. Keduas, and C. Notredame. Espresso: automatic incorporation of structural information in multiple sequence alignments using 3d-coffee. *Nucleic Acids Research*, 34(Web Server):W604–W608, jul 2006.
- [6] Leandro C. S. Assis. Are homology and synapomorphy the same or different? *Cladistics*, 29(1):7–9, sep 2012.
- [7] A. Bahr. BALiBASE (benchmark alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Research*, 29(1):323–326, jan 2001.
- [8] Y. Bao, P. Bolotov, D. Dernovoy, B. Kiryutin, L. Zaslavsky, T. Tatusova, J. Ostell, and D. Lipman. The influenza virus resource at the national center for biotechnology information. *Journal of Virology*, 82(2):596–601, oct 2007.
- [9] Maurice S. Bartlett. Stochastic processes or the statistics of change. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 2(1):44–64, 1953.

- [10] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–516, nov 1954.
- [11] B. P. Blackburne and S. Whelan. Class of multiple sequence alignment algorithm affects genomic analysis. *Molecular Biology and Evolution*, 30(3):642–653, nov 2012.
- [12] Gordon Blackshields, Fabian Sievers, Weifeng Shi, Andreas Wilm, and Desmond G Higgins. Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, 5(1):21, 2010.
- [13] A. Bouchard-Côté and M. I. Jordan. Evolutionary inference via the poisson indel process. *Proceedings of the National Academy of Sciences*, 110(4):1160–1166, 2013.
- [14] Robert K. Bradley, Adam Roberts, Michael Smoot, Sudeep Juvekar, Jaeyoung Do, Colin Dewey, Ian Holmes, and Lior Pachter. Fast statistical alignment. *PLoS Computational Biology*, 5(5):e1000392, may 2009.
- [15] P. Briffeuil, G. Baudoux, Christophe G. Lambert, Xavier De Bolle, C. Vinals, Ernest Feytmans, and Eric Depiereux. Comparative analysis of seven multiple protein sequence alignment servers: clues to enhance reliability of predictions. *Bioinformatics*, 14 4:357–66, 1998.
- [16] Patterson C. Morphological characters and homology. In AE Friday KA Joysey, editor, *Problems of Phylogenetic Reconstruction*, page 2174. Academic Press: London, 1983.
- [17] Piero Cammarano, Roberta Creti, Anna Maria Sanangelantoni, and P. Palm. The archaea monophyly issue: A phylogeny of translational elongation factor g(2) sequences inferred from an optimized selection of alignment positions. *Journal of Molecular Evolution*, 49:524–537, 1999.
- [18] Jamie J Cannone, Sankar Subramanian, Murray N Schnare, James R Collett, Lisa M D Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V Madabusi, Kirsten M Mueller, Nupur Pande, Zhidi Shang, Nan Yu, and Robin R Gutell. The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas. *BMC Bioinformatics*, 3(1):2, 2002.

- [19] Salvador Capella-Gutiérrez and Toni Gabaldón. Measuring guide-tree dependency of inferred gaps in progressive aligners. *Bioinformatics*, 29(8):1011–1017, feb 2013.
- [20] Jiří Černý, Barbora Černá Bolfíková, James J. Valdés, Libor Grubhofer, and Daniel Růžek. Evolution of tertiary structure of viral RNA dependent polymerases. *PLoS ONE*, 9(5):e96070, may 2014.
- [21] Jurate Daugelaite, Aisling O' Driscoll, and Roy D. Sleator. An overview of multiple sequence alignments and cloud computing in bioinformatics. *ISRN Biomathematics*, 2013:1–14, 2013.
- [22] James H. Degnan and Noah A. Rosenberg. Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends in Ecology & Evolution*, 24(6):332–340, jun 2009.
- [23] C. B. Do. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340, jan 2005.
- [24] Chuong B. Do, Samuel S. Gross, and Serafim Batzoglou. CONTRAlign: Discriminative training for protein sequence alignment. In *Lecture Notes in Computer Science*, pages 160–174. Springer Berlin Heidelberg, 2006.
- [25] Chuong B. Do and Kazutaka Katoh. *Protein Multiple Sequence Alignment*, pages 379–413. Humana Press, Totowa, NJ, 2008.
- [26] J. L. Doob. Topics in the theory of markoff chains. *Transactions of the American Mathematical Society*, 52(1):37–37, jan 1942.
- [27] J. L. Doob. Markoff chains—denumerable case. *Transactions of the American Mathematical Society*, 58(3):455, nov 1945.
- [28] Casey W. Dunn, Andreas Hejnol, David Q. Matus, Kevin Pang, William E. Browne, Stephen A. Smith, Elaine Seaver, Greg W. Rouse, Matthias Obst, Gregory D. Edgecombe, Martin V. Sørensen, Steven H. D. Haddock, Andreas Schmidt-Rhaesa, Akiko Okusu, Reinhardt Møbjerg Kristensen, Ward C. Wheeler, Mark Q. Martindale, and Gonzalo Giribet. Broad phylogenomic sampling improves resolution of the animal tree of life. *Nature*, 452(7188):745–749, mar 2008.
- [29] Mitchell J. Eaton, Andrew Martin, John Thorbjarnarson, and George Amato. Species-level diversification of african dwarf crocodiles (genus *osteolaemus*): A geographic and phylogenetic perspective. *Molecular Phylogenetics and Evolution*, 50(3):496–506, mar 2009.

- [30] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, mar 2004.
- [31] Robert C. Edgar. Quality measures for protein alignment benchmarks. *Nucleic Acids Research*, 38(7):2145–2153, jan 2010.
- [32] Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current Opinion in Structural Biology*, 16(3):368–373, jun 2006.
- [33] Isaac Elias. Settling the intractability of multiple alignment. *Journal of Computational Biology*, 13(7):1323–1339, sep 2006.
- [34] John Ellis and David Morrison. Effects of sequence alignment on the phylogeny of *Sarcocystis* deduced from 18s rDNA sequences. *Parasitology Research*, 81(8):696–699, 1995.
- [35] Joseph Felsenstein, Stanley Sawyer, and Rochelle Kochin. An efficient method for matching nucleic acid sequences. *Nucleic Acids Research*, 10(1):133–139, 1982.
- [36] Da-Fei Feng and Russell F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, aug 1987.
- [37] Walter Fontana. Continuous-time monte-carlo of reaction systems. Lecture Notes, Systems Biology, Harvard Medical School.
- [38] John V. Freudenstein. Characters, states and homology. *Systematic Biology*, 54(6):965–973, dec 2005.
- [39] M. C. Frith. Finding functional sequence elements by multiple local alignment. *Nucleic Acids Research*, 32(1):189–200, jan 2004.
- [40] Martin C. Frith, Neil F. W. Saunders, Bostjan Kobe, and Timothy L. Bailey. Discovering sequence motifs with arbitrary insertions and deletions. *PLoS Computational Biology*, 4(5):e1000071, may 2008.
- [41] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, dec 1976.

- [42] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, dec 1977.
- [43] G. B. Golding. Estimates of DNA and protein sequence divergence: an examination of some assumptions. *Molecular Biology and Evolution*, feb 1984.
- [44] O Gotoh. Consistency of optimal sequence alignments. *Bulletin of Mathematical Biology*, 52(4):509–525, 1990.
- [45] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, dec 1982.
- [46] Osamu Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Bioinformatics*, 9(3):361–370, 1993.
- [47] Sean W. Graham, Patrick A. Reeves, Analiese C. E. Burns, and Richard G. Olmstead. Microstructural changes in noncoding chloroplast DNA: Interpretation, evolution, and utility of indels and inversions in basal angiosperm phylogenetic inference. *International Journal of Plant Sciences*, 161(S6):S83–S96, nov 2000.
- [48] Richard Grantham. Amino acid difference formula to help explain protein evolution. *Science*, 185:862, 1974.
- [49] X. Gu and J. Zhang. A simple method for estimating the parameter of substitution rate variation among sites. *Molecular Biology and Evolution*, 14(11):1106–1113, nov 1997.
- [50] Li WH. Gu X, Fu YX. Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. *Molecular Biology and Evolution*, jul 1995.
- [51] S. Blair Hedges, Julie Marin, Michael Suleski, Madeline Paymer, and Sudhir Kumar. Tree of life reveals clock-like speciation and diversification. *Molecular Biology and Evolution*, 32(4):835–845, mar 2015.
- [52] S. Henikoff and J. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 89:10915–10919, 1992.

- [53] Robert E. Hickson, Chris Simon, and Soren W. Perrey. The performance of several multiple-sequence alignment programs in relation to secondary-structure features for an rRNA sequence. *Molecular Biology and Evolution*, 17(4):530–539, apr 2000.
- [54] Ian Holmes and William J Bruno. Evolutionary hmms: a bayesian approach to multiple alignment. *Bioinformatics*, 17(9):803–820, 2001.
- [55] HPG. Human genome project information. <http://web.ornl.gov/sci/techresources/HumanGenome/home.shtml>.
- [56] Melanie A. Huntley and Andrew G. Clark. Evolutionary analysis of amino acid repeats across the genomes of 12 drosophila species. *Molecular Biology and Evolution*, 24(12):2598–2609, jun 2007.
- [57] Ui W. Hwang, Won Kim, Diethard Tautz, and Markus Friedrich. Molecular phylogenetics at the felsenstein zone: Approaching the strepsiptera problem using 5.8s and 28s rDNA sequences. *Molecular Phylogenetics and Evolution*, 9(3):470–480, jun 1998.
- [58] S. Iantorno, K. Gori, N. Goldman, M. Gil, and C. Dessimoz. *Who Watches the Watchmen? An Appraisal of Benchmarks for Multiple Sequence Alignment*, pages 59–73. Humana Press, Totowa, NJ, 2014.
- [59] O JEFFROY, H BRINKMANN, F DELSUC, and H PHILIPPE. Phylogenomics: the beginning of incongruence? *Trends in Genetics*, 22(4):225–231, apr 2006.
- [60] THOMAS H. JUKES and CHARLES R. CANTOR. Evolution of protein molecules. In *Mammalian Protein Metabolism*, pages 21–132. Elsevier, 1969.
- [61] Winfried Just. Computational complexity of multiple sequence alignment with SP-score. *Journal of Computational Biology*, 8(6):615–623, nov 2001.
- [62] Kazutaka Katoh, Kazuharu Misawa, Kei ichi Kuma, and Takashi Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059, 2002.
- [63] Kazutaka Katoh and Daron M Standley. Mafft multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, 30(4):772–780, 2013.

- [64] Scot A Kelchner and Lynn G Clark. Molecular evolution and phylogenetic utility of the chloroplast rpl16 intron in *Chusquea* and the bambusoideae (poaceae). *Molecular Phylogenetics and Evolution*, 8(3):385–397, dec 1997.
- [65] Scot A. Kelchner and J. F. Wendel. Hairpins create minute inversions in non-coding regions of chloroplast DNA. *Current Genetics*, 30(3):259–262, aug 1996.
- [66] Carsten Kemena and Cedric Notredame. Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics*, 25(19):2455–2465, jul 2009.
- [67] David G. Kendall. An artificial realization of a simple "birth-and-death" process. *Journal of the Royal Statistical Society. Series B (Methodological)*, 12(1):116–119, 1950.
- [68] Karl M Kjer. Aligned 18s and insect phylogeny. *Systematic biology*, 53 3:506–14, 2004.
- [69] K.M. Kjer. Use of rRNA secondary structure in phylogenetic studies to identify homologous positions: An example of alignment and data presentation from the frogs. *Molecular Phylogenetics and Evolution*, 4(3):314–330, sep 1995.
- [70] Bjarne Knudsen and Michael M. Miyamoto. Sequence alignments and pair hidden markov models using evolutionary history. *Journal of Molecular Biology*, 333(2):453–460, oct 2003.
- [71] Remko K. Kuipers, Henk-Jan Joosten, Willem J. H. van Berkel, Nicole G. H. Leferink, Erik Rooijen, Erik Ittmann, Frank van Zimmeren, Helge Jochens, Uwe Bornscheuer, Gert Vriend, Vitor A. P. Martins dos Santos, and Peter J. Schaap. 3dm: Systematic analysis of heterogeneous superfamily data to discover protein functionalities. *Proteins: Structure, Function, and Bioinformatics*, pages NA–NA, 2010.
- [72] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25, 2009.
- [73] T. Lassmann. Automatic assessment of alignment quality. *Nucleic Acids Research*, 33(22):7120–7128, dec 2005.

- [74] Timo Lassmann and Erik L.L. Sonnhammer. Quality assessment of multiple alignment programs. *FEBS Letters*, 529(1):126–130, aug 2002.
- [75] Timo Lassmann and Erik LL Sonnhammer. Kalign - an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics*, 6(1):298, 2005.
- [76] Christian Ledergerber and Christophe Dessimoz. Alignments with non-overlapping moves, inversions and tandem duplications in $O(n^4)$ time. *Journal of Combinatorial Optimization*, 16(3):263–278, 2008.
- [77] Arthur M. Lesk and Cyrus Chothia. How different amino acid sequences determine similar protein structures: The structure and evolutionary dynamics of the globins. *Journal of Molecular Biology*, 136(3):225–270, jan 1980.
- [78] Anthony Levasseur, Pierre Pontarotti, Olivier Poch, and Julie D. Thompson. Strategies for reliable exploitation of evolutionary concepts in high throughput biology. *Evol Bioinform Online*, 4:121–137, May 2008.
- [79] H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473–483, may 2010.
- [80] H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, nov 2008.
- [81] R. Li, Y. Li, K. Kristiansen, and J. Wang. SOAP: short oligonucleotide alignment program. *Bioinformatics*, 24(5):713–714, jan 2008.
- [82] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science*, 324(5934):1561–1564, jun 2009.
- [83] Y. Liu, B. Schmidt, and D. L. Maskell. MSAProbs: multiple sequence alignment based on pair hidden markov models and partition function posterior probabilities. *Bioinformatics*, 26(16):1958–1964, jun 2010.
- [84] Juke S. Lolkema and Dirk-Jan Slotboom. Hydrophathy profile alignment: a tool to search for structural homologues of membrane proteins. *FEMS Microbiology Reviews*, 22(4):305–322, oct 1998.

- [85] A. Loytynoja and N. Goldman. Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science*, 320(5883):1632–1635, jun 2008.
- [86] Ari Löytynoja. Alignment methods: Strategies, challenges, benchmarking, and comparative overview. In *Methods in Molecular Biology*, pages 203–235. Humana Press, 2012.
- [87] Norman MacLeod. Understanding morphology in systematic contexts. In *The New Taxonomy*, pages 143–209. CRC Press, apr 2008.
- [88] Massimo Maiolo, Xiaolei Zhang, Manuel Gil, and Maria Anisimova. Progressive multiple sequence alignment with indel evolution. *BMC Bioinformatics*, 19(1), sep 2018.
- [89] E. H. Margulies. Identification and characterization of multi-species conserved sequences. *Genome Research*, 13(12):2507–2518, dec 2003.
- [90] P. W. Messer and P. F. Arndt. The majority of recent short DNA insertions in the human genome are tandem duplications. *Molecular Biology and Evolution*, 24(5):1190–1197, feb 2007.
- [91] I. Miklos. A "long indel" model for evolutionary sequence alignment. *Molecular Biology and Evolution*, 21(3):529–540, dec 2003.
- [92] István Miklós and Zoltán Toroczka. An improved model for statistical alignment. In *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2001.
- [93] Kenji Mizuguchi, Charlotte M. Deane, Tom L. Blundell, and John P. Overington. HOMSTRAD: A database of protein structure alignments for homologous families. *Protein Science*, 7(11):2469–2471, nov 1998.
- [94] B. Morgenstern. DIALIGN: multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Research*, 32(Web Server):W33–W36, jul 2004.
- [95] D. A. Morrison and J. T. Ellis. Effects of nucleotide sequence alignment on phylogeny estimation: a case study of 18s rDNAs of apicomplexa. *Molecular Biology and Evolution*, 14(4):428–441, apr 1997.
- [96] David A. Morrison. Why would phylogeneticists ignore computerized sequence alignment? *Systematic Biology*, 58(1):150–158, feb 2009.

- [97] David A. Morrison. Is sequence alignment an art or a science? *Systematic Botany*, 40(1):14–26, feb 2015.
- [98] David A. Morrison, Matthew J. Morgan, and Scot A. Kelchner. Molecular homology and multiple-sequence alignment: an analysis of concepts and practice. *Australian Systematic Botany*, 28(1):46, 2015.
- [99] D. W. Mount. Using iterative methods for global multiple sequence alignment. *Cold Spring Harbor Protocols*, 2009(7):pdb.top44–pdb.top44, jul 2009.
- [100] U. Mueckstein, I.L. Hofacker, and P.F. Stadler. Stochastic pairwise alignments. *Bioinformatics*, 18(Suppl 2):S153–S160, oct 2002.
- [101] M. Murata, J. S. Richardson, and J. L. Sussman. Simultaneous comparison of three protein sequences. *Proc Natl Acad Sci U S A*, 82(10):3073–3077, May 1985.
- [102] Eugene W. Myers and Webb Miller. Optimal alignments in linear space. *Computer Applications in the Biosciences*, 4(1):11–17, 1988.
- [103] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, mar 1970.
- [104] S. Nelesen, K. Liu, D. Zhao, C. R. Linder, and T. Warnow. The effect of the guide tree on multiple sequence alignments and subsequent phylogenetic analyses. In *Pacific Symposium on Biocomputing 2008, PSB 2008*, pages 25–36, 12 2008.
- [105] C. Notredame and D. G. Higgins. Saga: sequence alignment by genetic algorithm. *Nucleic Acids Res*, 24(8):1515–1524, Apr 1996.
- [106] Cédric Notredame. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144, jan 2002.
- [107] Cédric Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Computational Biology*, 3(8):e123, 2007.
- [108] Cédric Notredame, Desmond G Higgins, and Jaap Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment 1 ledited by j. thornton. *Journal of Molecular Biology*, 302(1):205–217, sep 2000.

- [109] Paulo AS Nuin, Zhouzhi Wang, and Elisabeth RM Tillier. The accuracy of several multiple sequence alignment programs for proteins. *BMC Bioinformatics*, 7(1):471, 2006.
- [110] Genome 10K Community of Scientists. Genome 10k: A proposal to obtain whole-genome sequence for 10 000 vertebrate species. *Journal of Heredity*, 100(6):659–674, 2009.
- [111] Francisco M. Ortuño, Olga Valenzuela, Hector Pomares, Fernando Rojas, Javier P. Florido, Jose M. Urquiza, and Ignacio Rojas. Predicting the accuracy of multiple sequence alignment algorithms by using computational intelligent techniques. *Nucleic Acids Research*, 41(1):e26–e26, oct 2012.
- [112] Orla O'Sullivan, Karsten Suhre, Chantal Abergel, Desmond G Higgins, and Cédric Notredame. 3dcoffee: Combining protein sequences and structures within multiple sequence alignments. *Journal of Molecular Biology*, 340(2):385–395, jul 2004.
- [113] Andy Pang, Andrew D Smith, Paulo AS Nuin, and Elisabeth RM Tillier. Simprot: Using an empirically determined indel distribution in simulations of protein evolution. *BMC Bioinformatics*, 6(1):236, 2005.
- [114] Georgios Joannis Pappas and Shankar Subramaniam. Analysis of the effects of multiple sequence alignments in protein secondary structure prediction. In João Carlos Setubal and Sergio Verjovski-Almeida, editors, *Advances in Bioinformatics and Computational Biology*, pages 128–140, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [115] J. Pei, R. Sadreyev, and N. V. Grishin. PCMA: fast and accurate multiple sequence alignment based on profile consistency. *Bioinformatics*, 19(3):427–428, feb 2003.
- [116] Jimin Pei. Multiple protein sequence alignment. *Current Opinion in Structural Biology*, 18(3):382–386, jun 2008.
- [117] Jimin Pei and Nick V. Grishin. MUMMALS: multiple sequence alignment improved by using hidden markov models with local structural information. *Nucleic Acids Research*, 34(16):4364–4374, aug 2006.
- [118] Jimin Pei and Nick V. Grishin. PROMALS: towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics*, 23(7):802–808, jan 2007.

- [119] O. Penn, E. Privman, G. Landan, D. Graur, and T. Pupko. An alignment confidence score capturing robustness to guide tree uncertainty. *Molecular Biology and Evolution*, 27(8):1759–1767, mar 2010.
- [120] Aloysius J. Phillips. Homology assessment and molecular sequence alignment. *Journal of Biomedical Informatics*, 39(1):18–33, feb 2006.
- [121] Nam phuong D. Nguyen, Siavash Mirarab, Keerthana Kumar, and Tandy Warnow. Ultra-large alignments using phylogeny-aware profiles. *Genome Biology*, 16(1), jun 2015.
- [122] Daniel A Pollard, Casey M Bergman, Jens Stoye, Susan E Celniker, and Michael B Eisen. Benchmarking tools for the alignment of functional noncoding dna. *BMC Bioinformatics*, 5(1):6, 2004.
- [123] Dietmar Quandt and Michael Stech. Molecular evolution of the trnLUAA intron in bryophytes. *Molecular Phylogenetics and Evolution*, 36(3):429–443, sep 2005.
- [124] GPS Raghava, Stephen MJ Searle, Patrick C Audley, Jonathan D Barber, and Geoffrey J Barton. Oxbench: A benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics*, 4(1):47, 2003.
- [125] Gerald R. Reeck, Christoph de Han, David C. Teller, Russell F. Doolittle, Walter M. Fitch, Richard E. Dickerson, Pierre Chambon, Andrew D. McLachlan, Emanuel Margoliash, Thomas H. Jukes, and Emile Zuckerkandl. “homology” in proteins and nucleic acids: A terminology muddle and a way out of it. *Cell*, 50(5):667, aug 1987.
- [126] Michael S Rosenberg. Evolutionary distance estimation and fidelity of pair wise sequence alignment. *BMC Bioinformatics*, 6(1):102, 2005.
- [127] Ogden TH Rosenberg MS. *Sequence Alignment: Methods, Models, Concepts, and Strategies*, chapter Simulation approaches to evaluating alignment error and methods for comparing alternate alignments. University of California Press, 1 edition, 2009.
- [128] Elke Schaper, Olivier Gascuel, and Maria Anisimova. Deep conservation of human protein tandem repeats within the eukaryotes. *Molecular Biology and Evolution*, 31(5):1132–1148, feb 2014.

- [129] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular Systems Biology*, 7(1):539–539, apr 2014.
- [130] Samer Singh, Robert Tokhunts, Valerie Baubet, John A. Goetz, Zhen Jane Huang, Neal S. Schilling, Kendall E. Black, Todd A. MacKenzie, Nadia Dahmane, and David J. Robbins. Sonic hedgehog mutations identified in holoprosencephaly patients can act in a dominant negative manner. *Human Genetics*, 125(1):95–103, dec 2008.
- [131] Jordan V. Smith, Edward L. Braun, and Rebecca T. Kimball. Ratite nonmonophyly: Independent evidence from 40 novel loci. *Systematic Biology*, 62(1):35–49, sep 2012.
- [132] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, mar 1981.
- [133] Robert R. Sokal. The comparative method in evolutionary biology. by paul h. harvey mark d. pagel. new york: Oxford university press. 1991. *American Journal of Physical Anthropology*, 88(3):405–406, jul 1992.
- [134] A. Som. Causes, consequences and solutions of phylogenetic incongruence. *Briefings in Bioinformatics*, 16(3):536–548, may 2014.
- [135] J. Stoye, D. Evers, and F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14(2):157–163, mar 1998.
- [136] Amarendran R Subramanian, Jan Weyer-Menkhoff, Michael Kaufmann, and Burkhard Morgenstern. Dialign-t: An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, 6(1):66, 2005.
- [137] Nei M. Tamura K. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution*, may 1993.
- [138] W. R. Taylor. Multiple sequence alignment by a pairwise algorithm. *Comput. Appl. Biosci.*, 3(2):81–87, Jun 1987.
- [139] William R. Taylor. A flexible method to align large numbers of biological sequences. *Journal of Molecular Evolution*, 28(1-2):161–169, dec 1988.

- [140] J. D. Thompson, D. G. Higgins, and T. J. 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(22):4673–4680, Nov 1994.
- [141] J. D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13):2682–2690, jul 1999.
- [142] Julie D. Thompson, Patrice Koehl, Raymond Ripp, and Olivier Poch. BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function, and Bioinformatics*, 61(1):127–136, jul 2005.
- [143] Julie D. Thompson, Benjamin Linard, Odile Lecompte, and Olivier Poch. A comprehensive benchmark study of multiple sequence alignment methods: Current challenges and future perspectives. *PLoS One*, 6(3):e18093, Mar 2011.
- [144] Julie D Thompson and Olivier Poch. *Sequence Alignment*, chapter Sequence Alignment, page 1. American Cancer Society, 2006.
- [145] J. L. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution*, 33(2):114–124, 1991.
- [146] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34(1):3–16, jan 1992.
- [147] Tom A. Titus and Darrel R. Frost. Molecular homology assessment and phylogeny in the lizard family opluridae (squamata: Iguania). *Molecular Phylogenetics and Evolution*, 6(1):49–62, aug 1996.
- [148] P. Di Tommaso, S. Moretti, I. Xenarios, M. Orobitg, A. Montanyola, J.-M. Chang, J.-F. Taly, and C. Notredame. T-coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension. *Nucleic Acids Research*, 39(suppl):W13–W17, may 2011.
- [149] Annamária Tóth, Anton Hausknecht, Irmgard Krisai-Greilhuber, Tamás Papp, Csaba Vágvolgyi, and László G. Nagy. Iteratively refined guide

- trees help improving alignment and phylogenetic inference in the mushroom family bolbitiaceae. *PLoS ONE*, 8(2):e56143, feb 2013.
- [150] T. Uzzell and K. W. Corbin. Fitting discrete probability distributions to evolutionary events. *Science*, 172(3988):1089–1096, jun 1971.
- [151] Christian L. Vestergaard and Mathieu G enois. Temporal gillespie algorithm: Fast simulation of contagion processes on time-varying networks. *PLoS Computational Biology*, 11(10):e1004579, oct 2015.
- [152] Martin Vingron and Patrick Argos. A fast and sensitive multiple sequence alignment algorithm. *Bioinformatics*, 5(2):115–121, 1989.
- [153] Martin Vingron and Patrick Argos. Determination of reliable regions in protein sequence alignments. *Protein Engineering, Design and Selection*, 3(7):565–569, 1990.
- [154] Martin Vingron and Patrick Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *Journal of Molecular Biology*, 218(1):33–43, mar 1991.
- [155] Iain M Wallace, Gordon Blackshields, and Desmond G Higgins. Multiple sequence alignments. *Current Opinion in Structural Biology*, 15(3):261–266, jun 2005.
- [156] I. Van Walle, I. Lasters, and L. Wyns. Align-m—a new algorithm for multiple alignment of highly divergent sequences. *Bioinformatics*, 20(9):1428–1435, feb 2004.
- [157] I. Van Walle, I. Lasters, and L. Wyns. SABmark—a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, 21(7):1267–1268, aug 2004.
- [158] Ivo Van Walle, Ignace Lasters, and Lode Wyns. Consistency matrices: Quantified structure alignments for sets of related proteins. *Proteins: Structure, Function, and Genetics*, 51(1):1–9, 2003.
- [159] Lusheng Wang and Tao Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, jan 1994.
- [160] N. Wang, E. L. Braun, and R. T. Kimball. Testing hypotheses about the sister group of the passeriformes using an independent 30-locus data set. *Molecular Biology and Evolution*, 29(2):737–750, sep 2011.

- [161] Tandy Warnow. Large-scale multiple sequence alignment and phylogeny estimation. In *Models and Algorithms for Genome Evolution*, pages 85–146. Springer London, 2013.
- [162] Andreas Wilm, Indra Mainz, and Gerhard Steger. An enhanced rna alignment benchmark for sequence alignment programs. *Algorithms for Molecular Biology*, 1(1):19, 2006.
- [163] K. M. Wong, M. A. Suchard, and J. P. Huelsenbeck. Alignment uncertainty and genomic analysis. *Science*, 319(5862):473–476, jan 2008.
- [164] Xuefeng Xia, Song Zhang, Yu Su, and Zhirong Sun. Micalign: a sequence-to-structure alignment tool integrating multiple sources of information in conditional random fields. *Bioinformatics*, 25(11):1433–1434, 2009.
- [165] Xuhua Xia, Zheng Xie, and Karl M. Kjer. 18s ribosomal rna and tetrapod phylogeny. *Systematic Biology*, 52(3):283–295, 2003.
- [166] Ziheng Yang. Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: Approximate methods. *Journal of Molecular Evolution*, 39(3):306–314, Sep 1994.
- [167] Ziheng Yang. Among-site rate variation and its impact on phylogenetic analyses. *Trends in Ecology & Evolution*, 11(9):367–372, sep 1996.
- [168] Jianzhi Zhang and Xun Gu. Correlation between the substitution rate and rate variation among sites in protein evolution. *Genetics*, 149(3):1615–1625, 1998.
- [169] Jin Zhang, Xiangling Chen, Michael S. Kent, Carlos O. Rodriguez, and Xinbin Chen. Establishment of a dog model for the p53 family pathway and identification of a novel isoform of p21 cyclin-dependent kinase inhibitor. *Molecular Cancer Research*, 7(1):67–78, jan 2009.
- [170] Z. Zhang, A. A. Schäffer, W. Miller, T. L. Madden, D. J. Lipman, E. V. Koonin, and S. F. Altschul. Protein sequence similarity searches using patterns as seeds. *Nucleic Acids Res*, 26(17):3986–3990, Sep 1998.

Journal article

METHODOLOGY ARTICLE

Open Access



Progressive multiple sequence alignment with indel evolution

Massimo Maiolo^{1,2,4}, Xiaolei Zhang³, Manuel Gil^{1,4†} and Maria Anisimova^{1,4*†}

Abstract

Background: Sequence alignment is crucial in genomics studies. However, optimal multiple sequence alignment (MSA) is NP-hard. Thus, modern MSA methods employ progressive heuristics, breaking the problem into a series of pairwise alignments guided by a phylogeny. Changes between homologous characters are typically modelled by a Markov substitution model. In contrast, the dynamics of indels are not modelled explicitly, because the computation of the marginal likelihood under such models has exponential time complexity in the number of taxa. But the failure to model indel evolution may lead to artificially short alignments due to biased indel placement, inconsistent with phylogenetic relationship.

Results: Recently, the classical indel model TKF91 was modified to describe indel evolution on a phylogeny via a Poisson process, termed PIP. PIP allows to compute the joint marginal probability of an MSA and a tree in linear time. We present a new dynamic programming algorithm to align two MSAs –represented by the underlying homology paths– by full maximum likelihood under PIP in polynomial time, and apply it progressively along a guide tree. We have corroborated the correctness of our method by simulation, and compared it with competitive methods on an illustrative real dataset.

Conclusions: Our MSA method is the first polynomial time progressive aligner with a rigorous mathematical formulation of indel evolution. The new method infers phylogenetically meaningful gap patterns alternative to the popular PRANK, while producing alignments of similar length. Moreover, the inferred gap patterns agree with what was predicted qualitatively by previous studies. The algorithm is implemented in a standalone C++ program: <https://github.com/acg-team/ProPIP>. Supplementary data are available at BMC Bioinformatics online.

Keywords: Sequence alignment, Indel, Phylogeny, Dynamic programming, Poisson process

Background

Multiple sequence alignments (MSAs) are routinely required in the early stages of comparative and evolutionary genomics studies. Not surprisingly, accuracy of MSA inference affects subsequent analyses that rely on MSA estimates [1]. MSA estimation is among the oldest bioinformatics problems, yet remains intensely studied due to its complexity (NP-hard [2–4]). The progressive alignment approach has allowed to reduce the overall

computational complexity to polynomial time by breaking the MSA problem into a series of pairwise alignments guided by a tree representing the evolutionary relationship of sequences. Today most popular alignment programs employ the progressive approach (e.g., ClustalW [5], MAFFT [6], MUSCLE [7], PRANK [8, 9] and T-Coffee [10] among others).

All state-of-the-art MSA programs nowadays use an evolutionary model to describe changes between homologous characters, providing a more realistic description of molecular data and thus more accurate inferences. However, a mathematical formulation of the insertion-deletion (indel) process still remains a critical issue. Describing the

*Correspondence: maria.anisimova@zhaw.ch

†Manuel Gil and Maria Anisimova contributed equally to this work.

¹Institute of Applied Simulation, School of Life Sciences and Facility Management, Zurich University of Applied Sciences (ZHAW), Grüentalstrasse 14, P.O. Box, CH-8820 Wädenswil, Switzerland

⁴Swiss Institute of Bioinformatics (SIB), Quartier Sorge - Bâtiment Géopode, CH-1015 Lausanne, Switzerland

Full list of author information is available at the end of the article



indel process in probabilistic terms is more challenging: unlike substitutions, indels often involve several sites, vary in length and may overlap obscuring the underlying mechanisms. Instead, the popular PRANK program adopts a pragmatic approach; it uses an outgroup to distinguish insertions from deletions during the progressive alignment procedure, so that insertions are not over-penalized [9]. As a result, PRANK produces exceptionally accurate alignments, notably with densely sampled data and given an accurate guide tree. Still, the method lacks a mathematical model describing the evolution of indels. Indeed, the computation of the marginal likelihood under the classical indel models TKF91 [11] and TKF92 [12] is exponential in the number of taxa due to the absence of site independence assumption.

A recent modification of TKF91 describes the evolution of indels on a phylogenetic tree as a Poisson process, thus dubbed the Poisson indel process or the PIP model [13]. The indels occur uniformly within a sequence. Standard mathematical results, particularly the Poisson thinning, allow achieving linear time complexity for computing the joint marginal probability of a tree and an MSA. This includes analytic marginalisation of unobservable homologous paths which occur whenever an ancestral character is inserted and subsequently deleted, and consequently cannot be detected in the extant sequences. For a given MSA and a tree, a likelihood score under PIP can be computed in linear time. This score can be used to find the maximum a posteriori tree-alignment solution. Remarkably, this breakthrough allows for a necessary rigorous way of combining models of substitutions and indels, and a tractable computation of the marginal likelihood function. At the moment the algorithm has only been applied in a Bayesian framework via tree-alignment space sampling.

Here we propose a new progressive algorithm to estimate an MSA under the explicit model of substitutions and indels. We have re-framed the original PIP equations into a dynamic programming (DP) approach. It aligns two MSAs –represented by their homology paths on the two corresponding subtrees– by maximum likelihood (ML) in polynomial time. The progressive algorithm traverses a guide tree in postorder; at each internal node the DP is applied to align the two sub-alignments at the child nodes. The procedure terminates at the root of the guide tree, with the complete MSA and the corresponding likelihood, which by construction is the likelihood under the PIP model. We have implemented the progressive MSA algorithm in a prototype program and verified its correctness by simulation. To our knowledge, this is the first progressive MSA algorithm with polynomial time complexity, using a mathematical formulation of an explicit indel process. Note that an equivalent formulation under TKF91 or TKF92 –i.e. using the full marginal likelihood along the subtrees in question– would have exponential time

complexity. Quadratic time complexity under the TKF models could be obtained [14] by representing sequences at internal nodes through probability profiles, and aligning those. However, this approach does not consider the evolutionary history in the subtrees.

The remainder of this manuscript is organized as follows. We first introduce notation and the PIP model. Then, we describe our DP algorithm and provide the simulation results. We conclude the paper with an illustrative real dataset, where we contrast our method with PRANK, as well as with MAFFT, representing a state of the art similarity based progressive method.

Methods

Preliminaries: the PIP model

Let $\tau = (\mathcal{V}, \mathcal{E}, b)$ represent a rooted binary phylogenetic tree with N leaves. τ is a directed, connected, labelled acyclic graph, with a finite set of branching points \mathcal{V} of cardinality $|\mathcal{V}| = 2N - 1$ and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Leaves $\mathcal{L} \subset \mathcal{V}$ denotes N observed taxa, represented by strings of characters from a finite alphabet Σ (nucleotides, amino acids or codons). There are $N - 1$ internal vertices $v \in \mathcal{V}$ whereof the root Ω is the most recent common ancestor of all leaves. Branch length $b(v)$ associated with node $v \in \mathcal{V}$ spans from v to its parent node $\text{pa}(v)$. The total tree length $\|\tau\|$ is a sum of all the branch lengths.

The PIP model describes a string-valued evolutionary process along the branches of τ . We denote the distance from the root to a given point on the tree, by the same symbol τ . Atomic insertions are Poisson events with rate measure $\nu(dt) = \lambda(\tau(dt) + \mu^{-1}\delta_{\Omega}(dt))$, where λ is the insertion rate, μ the deletion rate, and $\delta_{\Omega}(\cdot)$ Dirac's delta function. This formulation guarantees that the expected sequence length remains constant during the whole evolutionary process. Point substitutions and deletions are modelled by a continuous-time Markov process on $\Sigma_{\epsilon} = \Sigma \cup \{\epsilon\}$, where ϵ is the deletion symbol. Accordingly, the generator matrix \mathbf{Q}_{ϵ} of the combined substitution and indel process extends the instantaneous substitution rate matrix \mathbf{Q} by a row and a column to include ϵ , which is modelled as an absorbing state as there can be no substitutions after a deletion event. The quasi-stationary distribution of \mathbf{Q}_{ϵ} is denoted by π_{ϵ} . Root Ω has a virtual infinite length stem, reflecting the equilibrium steady state distribution at the root.

For an internal node v , the probability $\iota(v)$ of inserting a single character on branch $\text{pa}(v) \rightarrow v$, is proportional to branch length $b(v)$. For $v \neq \Omega$ it is given by $\iota(v) = b(v)/(\|\tau\| + \mu^{-1})$; at the root atomic mass point probability $\iota(\Omega) = \mu^{-1}/(\|\tau\| + \mu^{-1})$ so that $\sum_{v \in \mathcal{V}} \iota(v) = 1$. The survival probability $\beta(v)$ associated with an inserted character on branch $\text{pa}(v) \rightarrow v$ is given by $\beta(\Omega) = 1$ and $\beta(v) = (1 - \exp(-\mu b(v)))/(\mu b(v))$.

The marginal likelihood $p_\tau(m)$ of MSA m of length $|m|$ is computable in $O(N \cdot |m|)$ and can be expressed as

$$p_\tau(m) = \varphi(p(c_\emptyset), |m|) \prod_{c \in m} p(c), \tag{1}$$

where $p(c)$ is the likelihood of a single column c , and $p(c_\emptyset)$ is the likelihood of an unobservable character history, represented by a column c_\emptyset with a gap at every leaf. The factor in (1)

$$\varphi(p(c_\emptyset), |m|) = \|\nu\|^{||m||} \exp(\|\nu\| (p(c_\emptyset) - 1)) / |m|! \tag{2}$$

is the marginal likelihood over all unobservable character histories, where $\|\nu\|$ is the normalising Poisson intensity.

The column likelihood can be expressed as

$$p(c) = \sum_{\nu \in \mathcal{V}} \iota(\nu) f_\nu, \tag{3}$$

where f_ν denotes the probability of the homology path underlying column c , given that the corresponding character was inserted at ν . This probability can be computed in $O(N)$ using a variant of Felsenstein's peeling recursion [15]. Let \mathcal{S} be the set of leaves that do not have a gap in column c , and \mathcal{A} the set of nodes ancestral to \mathcal{S} . Then

$$f_\nu = \begin{cases} \mathbf{1}[\nu \in \mathcal{A}] \beta(\nu) \sum_{\sigma \in \Sigma} \pi_\epsilon(\sigma) \tilde{f}_\nu(\sigma) & \text{if } c \neq c_\emptyset \\ 1 - \beta(\nu) + \beta(\nu) \sum_{\sigma \in \Sigma} \pi_\epsilon(\sigma) \tilde{f}_\nu(\sigma) & \text{o.w.,} \end{cases} \tag{4}$$

where

$$\tilde{f}_\nu(\sigma) = \begin{cases} \mathbf{1}[c(\nu) = \sigma] & \text{if } \nu \in \mathcal{L} \\ \prod_{w \in \text{child}(\nu)} \left[\sum_{\sigma' \in \Sigma_\epsilon} \exp(b(w) \mathbf{Q}_\epsilon)_{\sigma, \sigma'} \tilde{f}_w(\sigma') \right] & \text{o.w.,} \end{cases} \tag{5}$$

and $\mathbf{1}[\cdot]$ is the indicator function. In Eq. 4, the term $1 - \beta(\nu)$ accounts for the probability that the inserted character does not survive till the first node below the insertion point. The recursive function \tilde{f}_ν computes the probability of the substitution-deletion process of a single character.

Dynamic programming algorithm under PIP

Given an internal node ν , our DP algorithm proceeds to align the two sub-alignments obtained in the left and right sub-trees maximizing the likelihood (Eq. 1) of the tree rooted at ν . Let \mathbf{X} and \mathbf{Y} denote these sub-alignments, respectively with N_X and N_Y sequences and alignment lengths $|\mathbf{X}|$ and $|\mathbf{Y}|$. If a sub-tree is a leaf then the sub-alignment, say \mathbf{X} , is reduced to an input sequence, i.e. $N_X = 1$ and $|\mathbf{X}|$ corresponds to the sequence length.

Note that the marginal likelihood function $p_\tau(m)$ (Eq. 1) is not monotonically increasing in the alignment length $|m|$. While the product of column likelihoods is monotonically increasing, the marginal likelihood of unobserved histories $\varphi(p(c_\emptyset), |m|)$ is non-monotonic (Fig. 1). This means that $p_\tau(m)$ cannot be maximised by means of

a standard two-dimensional DP approach (in particular, because the alignment length is not known a priori). Similarly to TKF91 [11], we need three DP matrices, one for each state (i.e. match, gapX and gapY), however to account for the dependence on alignment length we have extended the matrices with a third dimension.

The algorithm works with three three-dimensional sparse matrices \mathbf{S}^M , \mathbf{S}^X and \mathbf{S}^Y each of size $(|\mathbf{X}| + 1) \times (|\mathbf{Y}| + 1) \times (|\mathbf{X}| + |\mathbf{Y}| + 1)$ with entries defined as follows (Fig. 2b):

1. *match* cell $\mathbf{S}_{i,j,k}^M$ contains the likelihood of the partial optimal MSA of length k between $\mathbf{X}_1 \dots \mathbf{X}_i$ and $\mathbf{Y}_1 \dots \mathbf{Y}_j$ with the columns \mathbf{X}_i and \mathbf{Y}_j aligned. Consequently, all characters in the two columns are inferred to be homologous.
2. *gapX* cell $\mathbf{S}_{i,j,k}^X$ contains the likelihood of the partial optimal MSA of length k between $\mathbf{X}_1 \dots \mathbf{X}_i$ and $\mathbf{Y}_1 \dots \mathbf{Y}_j$ with the column \mathbf{X}_i aligned with a column of size N_Y containing gaps only. The characters in the two columns do not share a common history, either because the ancestor character had been deleted on the right subtree, or because it had been inserted on the left subtree, below the node ν .
3. similarly, *gapY* cell $\mathbf{S}_{i,j,k}^Y$ matches column \mathbf{Y}_j with a column of size N_X containing gaps only.

Forward phase

Each matrix \mathbf{S}^M , \mathbf{S}^X and \mathbf{S}^Y is initialized with $\varphi(p(c_\emptyset), 0)$ at position $(0, 0, 0)$ and a zero in every other position. The DP equations are:

$$\mathbf{S}_{i,j,k}^M = \frac{\|\nu\|}{k} \cdot p \left(\begin{bmatrix} \mathbf{X}_i \\ \mathbf{Y}_j \end{bmatrix} \right) \cdot \max \left\{ \begin{aligned} &\mathbf{S}_{i-1,j-1,k-1}^M, \\ &\mathbf{S}_{i-1,j-1,k-1}^X, \\ &\mathbf{S}_{i-1,j-1,k-1}^Y \end{aligned} \right\} \tag{6}$$

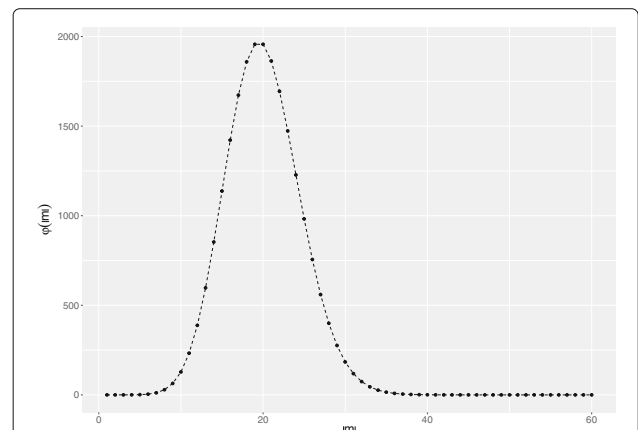


Fig. 1 An example of $\varphi(|m|)$ (Eq. 2), i.e. the marginal likelihood of all non-observable histories, as a function of MSA length $|m|$. The parameters are: $\tau = 1, \lambda = 10, \mu = 1, p(c_\emptyset) = 0.5$

$$S_{i,j,k}^X = \frac{\|v\|}{k} \cdot p \left(\begin{bmatrix} X_i \\ c_{\emptyset} \end{bmatrix} \right) \cdot \max \left\{ \begin{array}{l} S_{i-1,j,k-1}^M \\ S_{i-1,j,k-1}^X \\ S_{i-1,j,k-1}^Y \end{array} \right\} \quad (7)$$

$$S_{i,j,k}^Y = \frac{\|v\|}{k} \cdot p \left(\begin{bmatrix} c_{\emptyset} \\ Y_j \end{bmatrix} \right) \cdot \max \left\{ \begin{array}{l} S_{i,j-1,k-1}^M \\ S_{i,j-1,k-1}^X \\ S_{i,j-1,k-1}^Y \end{array} \right\} \quad (8)$$

for $i = 1, \dots, |X|, j = 1, \dots, |Y|$ and $k = 1, \dots, |X| + |Y|$.

The symbol c_{\emptyset} in Eqs. 7 and 8 represents a column with gaps, respectively of length N_Y and N_X . The factor $\|v\|/k$ successively constructs $\varphi(p(c_{\emptyset}), k)$ along the third dimension as columns are added into partial alignments.

As pointed out above, a column likelihood under PIP (Eq. 1) can be computed recursively in linear time in the number of input sequences. The recursion corresponds to a postorder tree traversal (Eq. 5), which coincides with the tree traversal of our progressive algorithm. As a consequence, during the progressive alignment a column likelihood for the DP ($p(\cdot)$ in Eqs. 6–8) at a particular node v can be computed in constant time by re-using appropriate summands (defined by Eq. 4) from the column likelihoods

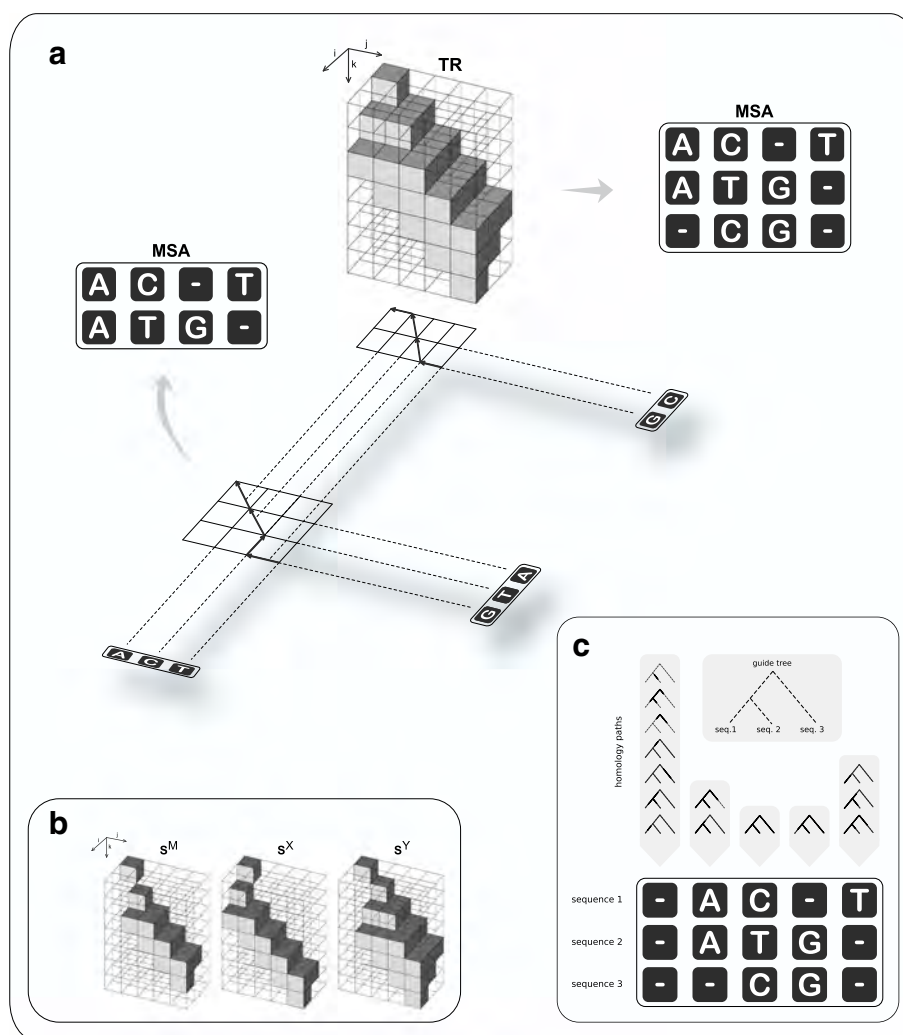


Fig. 2 Overview of the progressive algorithm. The algorithm traverses a guide tree (indicated by the shadow in Panel **a**) in postorder. At each internal node, the evolutionary paths from the two children down to the leaves (dotted lines in Panel **a**) are aligned by full maximum likelihood under the PIP model, using a dynamic programming approach (DP). Since the likelihood function does not increase monotonically in the MSA length (see Fig. 1), the DP accommodates the MSA length along a third dimension (indicated by k in Panels **a**, **b**); thus, it works with cubic matrices (in contrast to the traditional quadratic DP alignment). The forward phase of the DP stores likelihood values in three sparse matrices (Panel **b**: S^M for matching columns; S^X and S^Y to introduce new indel events). Further, matrix TR (Panel **a**) at position (i, j, k) records the name of the DP matrix (either “ S^M ”, “ S^X ”, or “ S^Y ”) with highest likelihood at (i, j, k) . An optimal alignment is determined by backtracking along TR (indicated in Panel **a** by the arrows in the projection of TR onto the plane). Note that the likelihood function marginalises over all indel scenarios compatible with putative homology (Panel **c**)

at the two children of v . In particular, the set \mathcal{A} can be constructed from the corresponding sets at the two children $\mathcal{A}_{\text{left}}$ and $\mathcal{A}_{\text{right}}$:

$$\mathcal{A} = \begin{cases} \{v\} & \text{for match state} \\ \mathcal{A}_{\text{left}} \cup \{v\} & \text{for gapX state} \\ \mathcal{A}_{\text{right}} \cup \{v\} & \text{for gapY state} \end{cases} \quad (9)$$

Consequently, the total asymptotic running time of the forward phase is $O(Nl^3)$, where l is bounded by the length of the longest input sequence. The independence structure of the DP along the dimension of the MSA length (i.e. index k) readily allows parallelisation; all the entries in the DP matrices for a fixed k can be computed in parallel from the entries at the layer $k - 1$, taking down the time to $O(Nl)$.

Backtracking

An optimal alignment is determined by backtracking along a trace-back matrix \mathbf{TR} of size $(|X| + 1) \times (|Y| + 1) \times (|X| + |Y| + 1)$. In the forward phase, \mathbf{TR} records at position (i, j, k) the name of the DP matrix (either “ \mathbf{S}^M ”, “ \mathbf{S}^X ”, or “ \mathbf{S}^Y ”) with highest likelihood at the same position (i, j, k) . If the maximum is not unique then a uniform random choice is made. The backtracking algorithm starts at $\mathbf{TR}(|X|, |Y|, k_0)$, where

$$k_0 = \arg \max_{k=\max(|X|,|Y|) \dots (|X|+|Y|)} s(k)$$

with

$$s(k) = \left\{ \mathbf{S}^M(|X|, |Y|, k), \mathbf{S}^X(|X|, |Y|, k), \mathbf{S}^Y(|X|, |Y|, k) \right\}$$

is the length of the best scoring alignment. If k_0 is not unique a random uniform choice is made. \mathbf{TR} is then traversed from $(|X|, |Y|, k_0)$ to $(0, 0, 0)$. Suppose the algorithm is at position (i, j, k) . If $\mathbf{TR}(i, j, k) = \mathbf{S}^M$ then the columns X_i and Y_j are matched and all the indices are decremented, i.e. $i \leftarrow i - 1, j \leftarrow j - 1, k \leftarrow k - 1$. If $\mathbf{TR}(i, j, k)$ is set to “ \mathbf{S}^X ” then the column X_i is matched with a column of gaps of size N_Y and the indices i and k are decremented, and, if $\mathbf{TR}(i, j, k)$ contains the value “ \mathbf{S}^Y ” then the column Y_j is matched with a column of gaps of size N_X and the indices j and k are decremented.

Results

Since the main goal of the article is to describe a new method, it is desirable to evaluate the correctness of the implementation (i.e., likelihood values and optimisation) and the accuracy of the estimator. Correctness can be evaluated by simulations under the true model or by comparison with existing implementations. The evaluation of alignment accuracy is more problematic ([16]), because the historical evolutionary events are not observable, so that we have no access to true alignments. Benchmarks like BALiBASE have attempted to provide sets of reference

alignments. Those, however, represent structural similarity, not necessarily reflecting homology, but also could be due to structural convergence. Moreover, benchmarks tend to represent alignments with highly compact and conserved cores offering little information about indel placement ([16]). Alternatively, synthetic data can be generated, where the true alignments are known. However, simulations rely on a generative model, which never perfectly correspond to the real process. The closer the generative model is to the assumed by the estimator, the better the estimator should perform.

Recently, it has been shown that the results obtained from structural benchmarks and from phylogenetic simulations have produced inconsistent results ([17–20]). Phylogeny-aware aligners like PRANK tend to perform well in simulations, while poorly on structural benchmarks. This can be explained by the fact that the objective of phylogenetic aligners is to infer evolutionary homology, rather than conserved structural features.

Below we provide results from some basic evaluations of our proposed method.

Empirical verification of correctness

To test the correctness of the algorithm and implementation, we generated data under PIP using a simulator provided by the authors of PIP. We chose relatively small trees and short sequences to be able to perform analytical tests during algorithm design and program debugging. Specifically, we simulated 120 datasets in total, on trees with 4, 5, 6 and 7 leaves, and using the following parameter combinations $(\lambda, \mu) \in \{(0.1, 0.1), (0.1, 1), (1, 0.1), (1, 1)\}$. The resulting sequence lengths varied between 5 and 8 nucleotides.

First, we confirmed the correctness of the likelihoods obtained with the DP algorithm, by scoring the resulting MSAs with an independent implementation provided by the authors of PIP. In all cases the likelihoods matched. In a second test, we verified that the DP generates optimal pairwise MSA alignments. To this end, all the possible pairwise alignments were generated at each internal node of the guide-trees and scored with the independent implementation. The DP algorithm always reconstructed an optimal MSA.

Aligning simulated data

To assess the quality of inferred alignments we have applied our method to simulated data that was previously used to evaluate PRANK [8]). These data sets were each 1000 nucleotides long and were generated under realistic evolutionary parameters on 16- 32- and 64-taxon trees and with different degrees of divergence. Note that indel lengths were drawn from a Poisson distribution with a mean of 1.7 bases. Inferred MSA lengths and four standard quality scores obtained with our method

were compared to those inferred with MAFFT v7.402 (with option `-auto`) and PRANK v.140603 (with options `-protein -termgap -nomissing -once`, with and without the `+F` option). The results of this comparison are shown in Additional file 1: Table S1 and Figure S1. No matter what evaluation score was considered, progressive alignment under PIP produced alignment quality similar to both PRANK and MAFFT. In terms of approaching the true MSA length, our method infers alignments of a similar length to PRANK but consistently outperform MAFFT. In many cases, our method also infers MSA lengths closer to the true compared to PRANK, albeit by a small margin. These results are encouraging, especially considering that the simulation scenario with long indels explicitly favours MAFFT and PRANK, both of which allow for long indels in their scoring schemes, although they are not explicitly modelled.

Aligning sequences from HIV/SIV envelope glycoprotein gp120

Using our new algorithm we inferred an MSA for a challenging dataset, 23 envelope glycoprotein gp120 sequences from HIV/SIV, previously analysed by Löytynoja and Goldman [8]. We compared the results of our algorithm with the MSAs inferred by MAFFT and PRANK. The resulting MSAs (Fig. 3) showed good agreement in the conserved regions. Indeed, the use of structural benchmarks [16], which are mainly restricted to such regions, has illustrated that it is difficult to distinguish state-of-the-art aligners. In contrast, variable regions display distinctly different indel patterns, which was reflected in the MSA lengths. Consistent with previous reports [8, 21] MAFFT over-aligns the sequences resulting in a short alignment (579 columns). The alignment inferred with our method had similar length (661 columns) to the one inferred by PRANK (669 columns).

The indel patterns reflected the underlying indel model or scoring function of the methods. Our algorithm favoured shorter indels, compared to PRANK and MAFFT, which reconstructed visually tidier gap regions. A phylogenetic interpretation of MAFFT's indel placement implies few insertions, followed by several subsequent deletions, leading to a short MSA. PRANK infers a longer alignment, with phylogenetically meaningful and balanced number of insertions and deletions. Note that similar to MAFFT, PRANK also tends to block long indels. Our method infers a phylogenetically meaningful MSA, with multiple single amino-acid insertions, which sometimes fuse to mimic long indels (e.g., 4 amino-acids from #501 to #504). Our method infers short indels, which allows for gap regions with higher conservation in terms of the substitution rates; we observe more conserved columns. To quantify this, we estimated tree-lengths (in expected substitutions per site), by fitting the branch-

lengths of the guide-tree topology based on the inferred MSAs using PhyML [22]. Consistent with the visual observation, our algorithm leads to the shortest tree (4.35), compared to PRANK (4.60) and MAFFT (4.90).

Discussion

Here for the first time in the frequentist framework, we have developed and implemented a progressive MSA algorithm with an explicit evolutionary model of substitutions, insertions and deletions. The evolution of indels was described as a Poisson process as part of a continuous-time Markov model known as PIP. At the core of our method we have designed a new DP algorithm for the alignment of two MSAs by ML, which exploits PIP's linear time complexity for the computation of marginal likelihoods. The overall complexity of the progressive algorithm is $O(Nl^3)$, where N is number of taxa and l is the maximum sequence length. The cubic factor stems from the fact that the likelihood is not monotonically increasing in the MSA length, so that the length has to be incorporated as an extra dimension in the DP. The $O(l^2)$ entries in a specific matrix layer along that dimension (i.e. corresponding to one particular alignment length) depend only on the layer above (and not on each other). Therefore, their computation can be parallelized, taking down the running time to $O(Nl)$, assuming $O(l^2)$ processors. Further, our empirical findings show that the likelihood has exactly one maximum, suggesting an early stop condition to the DP. We are currently optimising our implementation with respect to this and other time-critical aspects. To date inference of MSAs under an evolutionary indel model (TKF91 or TKF92) has only been implemented using a Bayesian framework. Such approaches are however computationally expensive with large datasets. Our method for MSA inference under PIP is the first step towards equivalent developments in the frequentist framework.

Despite only allowing single residue indels our method appears to fare surprisingly well compared to other state-of-the-art popular alignment tools such as PRANK and MAFFT. Indeed, our example above (as well as other preliminary data analyses, not shown) demonstrate that our new method allows inferring alignments with phylogenetically sensible gap patterns, similar to the phylogenetically-aware PRANK. In contrast to traditional aligners that do not utilise phylogenetic information to distinguish insertions and deletions, our method produces longer alignments, avoiding the artificial compression of MSAs and inferring more indels, again similar to PRANK. According to the underlying indel model, our method appears to infer more shorter indels (e.g., compared to PRANK and MAFFT), while longer indels are described by several subsequent indel events. Including longer indels is considered desirable, however it has not been studied whether modeling one residue indels



at a time may also work well. For example, for simplicity models of codon substitution typically allow only one-nucleotide mutations. Despite this gross simplification codon models have been demonstrated to perform extremely well for practical analyses of protein-coding genes. As can be seen in our example of an HIV protein gp120, it is unclear what inferred indel pattern is more realistic (given that alignments inferred by our methods and by PRANK have very similar length). Considering the nature of HIV mutations, it is quite plausible that indel evolution of gp120 is dominated by short

indel events [23]. Arguably, in our example, indel penalisation of PRANK and MAFFT (affine penalty schemes allowing for long indels) might make these tools too restrictive to single-residue indels, leading to aesthetically more pleasing alignments. PIP might be more restrictive to long indels but also more realistic for sequence data dominated by short indel events. Both alignment benchmarking and the parameter optimisation of gap penalties are extremely difficult due to the absence of sufficiently challenging datasets where true alignments are known.

Conclusion

Our new methods provides not only a first step towards the explicit modeling of indels in the frequentist framework but also enables to test a different hypothesis of indel evolution. In our follow up studies we intend to further scrutinise the various properties of our new method, its further development including less greedy algorithm versions, variation of indel rates across sites and the approximations to include longer indels.

Additional file

Additional file 1: Supplemental Materials. Qscores, MSA length and MSA magnifications. (PDF 3392 kb)

Abbreviations

DP: Dynamic programming; Indel: Insertion and deletion; ML: Maximum likelihood; MSA: Multiple sequence alignment; PIP: Poisson indel process

Acknowledgements

We thank Alexandre Bouchard-Côté for providing his code to simulate sequences and to compute marginal likelihoods under PIP.

Funding

This work was supported by the Swiss National Science Foundation (SNF) grants 31003A_157064 and 31003A_176316 to M. Anisimova. The funding body did not play any role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Availability of data and materials

The proposed algorithm has been coded using the code base of PrographMSA [24]. The code is written in C++ and tested on Linux. Our command-line tool, freely available on GitHub at <https://github.com/acg-team/ProPIP>, is distributed under the terms of the GNU GPLv3 license.

Authors' contributions

MM, MG, MA designed the study; MM, XZ, MG performed research; MM analyzed data; MM, MG, MA interpreted the results and wrote the paper. All authors read and approved the final version of the manuscript.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

¹Institute of Applied Simulation, School of Life Sciences and Facility Management, Zurich University of Applied Sciences (ZHAW), Grüentalstrasse 14, P.O. Box, CH-8820 Waedenswil, Switzerland. ²Institute of Molecular Life Sciences, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland. ³National Heart and Lung Institute, Imperial College London, South Kensington Campus, SW7 2AZ London, UK. ⁴Swiss Institute of Bioinformatics (SIB), Quartier Sorge - Bâtiment Génopode, CH-1015 Lausanne, Switzerland.

References

1. Wong KM, Suchard MA, Huelsenbeck JP. Alignment uncertainty and genomic analysis. *Science*. 2008;319(5862):473–6.
2. Just W. Computational complexity of multiple sequence alignment with sp-score. *J Comput Biol*. 2001;8(6):615–23.
3. Bonizzoni P, Della Vedova G. The complexity of multiple sequence alignment with sp-score that is a metric. *Theor Comput Sci*. 2001;259(1):63–79.
4. Wang L, Jiang T. On the complexity of multiple sequence alignment. *J Comput Biol*. 1994;1(4):337–48.
5. Thompson JD, Higgins DG, Gibson TJ. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*. 1994;22(22):4673–80.
6. Katoh K, Standley DM. MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Mol Biol Evol*. 2013;30(4):772.
7. Edgar RC. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*. 2004;5(1):113.
8. Löytynoja A, Goldman N. Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science*. 2008;320(5883):1632–5.
9. Löytynoja A, Goldman N. Proceedings of the National Academy of Sciences of the United States of America. 2005;102(30):10557–62.
10. Notredame C, Higgins DG, Heringa J. T-coffee: a novel method for fast and accurate multiple sequence alignment. *J Mol Biol*. 2000;302(1):205–17.
11. Thorne JL, Kishino H, Felsenstein J. An evolutionary model for maximum likelihood alignment of dna sequences. *J Mol Evol*. 1991;33(2):114–24.
12. Thorne JL, Kishino H, Felsenstein J. Inching toward reality: an improved likelihood model of sequence evolution. *J Mol Evol*. 1992;34(1):3–16.
13. Bouchard-Côté A, Jordan MI. Evolutionary inference via the poisson indel process. *Proc Natl Acad Sci*. 2013;110(4):1160–6.
14. Holmes I, Bruno WJ. Evolutionary hmms: a bayesian approach to multiple alignment. *Bioinformatics*. 2001;17(9):803–20.
15. Felsenstein J. Evolutionary trees from dna sequences: A maximum likelihood approach. *J Mol Evol*. 1981;17(6):368–76.
16. Iantorno S, Gori K, Goldman N, Gil M, Dessimoz C. Who Watches the Watchmen? An Appraisal of Benchmarks for Multiple Sequence Alignment. Totowa: Humana Press; 2014, pp. 59–73.
17. Tan G, Gil M, Löytynoja AP, Goldman N, Dessimoz C. Simple chained guide trees give poorer multiple sequence alignments than inferred trees in simulation and phylogenetic benchmarks. *Proc Natl Acad Sci*. 2015;112(2):99–100.
18. Boyce K, Sievers F, Higgins DG. Simple chained guide trees give high-quality protein multiple sequence alignments. *Proc Natl Acad Sci*. 2014;111(29):10556–61.
19. Boyce K, Sievers F, Higgins DG. Instability in progressive multiple sequence alignment algorithms. *Algorithm Mol Biol*. 2015;10(1):26.
20. Nute MG, Saleh E, Warnow T. Benchmarking Statistical Multiple Sequence Alignment. *bioRxiv*. 2018. Cold Spring Harbor Laboratory. <https://www.biorxiv.org/content/early/2018/04/20/304659>.
21. Szalkowski AM, Anisimova M. Graph-based modeling of tandem repeats improves global multiple sequence alignment. *Nucleic Acids Res*. 2013;41(17):162.
22. Guindon S, Dufayard J-F, Lefort V, Anisimova M, Hordijk W, Gascuel O. New algorithms and methods to estimate maximum-likelihood phylogenies: Assessing the performance of phylml 3.0. *Syst Biol*. 2010;59(3):307–21.
23. Abram ME, Ferris AL, Shao W, Alvord WG, Hughes SH. Nature, position, and frequency of mutations made in a single cycle of hiv-1 replication. *J Virol*. 2010;84(19):9864–78.
24. Szalkowski AM. Fast and robust multiple sequence alignment with phylogeny-aware gap placement. *BMC Bioinformatics*. 2012;13:129.

Received: 3 April 2018 Accepted: 3 September 2018

Published online: 21 September 2018

Supporting Information

Progressive Multiple Sequence Alignment with Indel Evolution

Massimo Maiolo,^{*,1,2,4} Xiaolei Zhang,³ Manuel Gil,^{1,4,†} and Maria
Anisimova^{*,1,4,†}

¹Institute of Applied Simulation, School of Life Sciences and Facility Management, Zurich University of Applied Sciences (ZHAW), CH-8820 Waedenswil

²Institute of Molecular Life Sciences, University of Zurich, CH-8057 Zuerich, Switzerland

³National Heart and Lung Institute, Imperial College London, London SW7 2AZ, United Kingdom

⁴Swiss Institute of Bioinformatics (SIB), CH-1015 Lausanne, Switzerland †share senior authorship

		2X	4X	close	intermediate	distant
PRNK	Q	0.9932 (0.0020)	0.9926 (0.0020)	0.9928 (0.0024)	0.9754 (0.0058)	0.9354 (0.0128)
	TC	0.9547 (0.0092)	0.9271 (0.0120)	0.9696 (0.0086)	0.9080 (0.0177)	0.7931 (0.0317)
	Cline	0.9944 (0.0016)	0.9940 (0.0015)	0.9941 (0.0019)	0.9802 (0.0046)	0.9488 (0.0102)
	Modeler	0.9933 (0.0020)	0.9928 (0.0018)	0.9928 (0.0024)	0.9756 (0.0057)	0.9366 (0.0124)
PRNK+F	Q	0.9931 (0.0019)	0.9926 (0.0019)	0.9928 (0.0024)	0.9755 (0.0059)	0.9354 (0.0129)
	TC	0.9546 (0.0095)	0.9271 (0.0115)	0.9693 (0.0087)	0.9081 (0.0178)	0.7929 (0.0320)
	Cline	0.9944 (0.0016)	0.9940 (0.0015)	0.9941 (0.0019)	0.9803 (0.0046)	0.9488 (0.0102)
	Modeler	0.9932 (0.0019)	0.9928 (0.0018)	0.9928 (0.0024)	0.9756 (0.0058)	0.9366 (0.0124)
MAFFT	Q	0.9896 (0.0037)	0.9895 (0.0028)	0.9889 (0.0043)	0.9606 (0.0099)	0.9010 (0.0181)
	TC	0.9355 (0.0154)	0.8975 (0.0182)	0.9551 (0.0144)	0.8623 (0.0270)	0.7033 (0.0398)
	Cline	0.9915 (0.0030)	0.9914 (0.0023)	0.9910 (0.0034)	0.9682 (0.0078)	0.9212 (0.0145)
	Modeler	0.9890 (0.0040)	0.9889 (0.0031)	0.9884 (0.0046)	0.9581 (0.0105)	0.8957 (0.0188)
P-PIP	Q	0.9891 (0.0028)	0.9892520 (0.0024)	0.9888 (0.0034)	0.9654 (0.0069)	0.9207 (0.0141)
	TC	0.9276 (0.0139)	0.88886800 (0.0153)	0.9525 (0.0124)	0.8686 (0.0215)	0.7386 (0.0329)
	Cline	0.9913 (0.00231)	0.99146400 (0.0019)	0.9911 (0.0027)	0.9728 (0.0055)	0.9380 (0.0113)
	Modeler	0.9891 (0.0029)	0.9891600 (0.0024)	0.9887 (0.0035)	0.9648 (0.0071)	0.9189 (0.0144)

Table S1 Quality scores. The alignments obtained with PRANK, PRANK+F, MAFFT and our Progressive-PIP have been compared using 5 different score metrics computed with Qscore [25]. The table reports the mean Q scores (SPS), TC (CS), Shift scores (Cline) and Modeler scores for the different evolutionary scenarios ('2X', '4X', 'close', 'intermediate', 'distant'), standard deviation in parentheses. The analyses were performed on the five scenarios simulated in [8]. The following type of MSAs were simulated: "close", "intermediate", "distant", "2X" and "4X", reflecting divergence levels. Each dataset contained 250 MSAs generated along a symmetric 16- (close, intermediate, distant), 32- (2X) and 64-taxon tree (4X). The branch lengths were 0.025 (close), 0.050 (intermediate), 0.075 (distant), 0.020 (2X) and 0.0167 (4X) expected substitutions per site. The sequences were simulated according to the JC69 substitution model, with root sequences of length 1000 nucleotides, and a rate of 1 indel per 25 substitutions. The indel length was drawn from the Poisson distribution with mean 1.7 nucleotide bases.

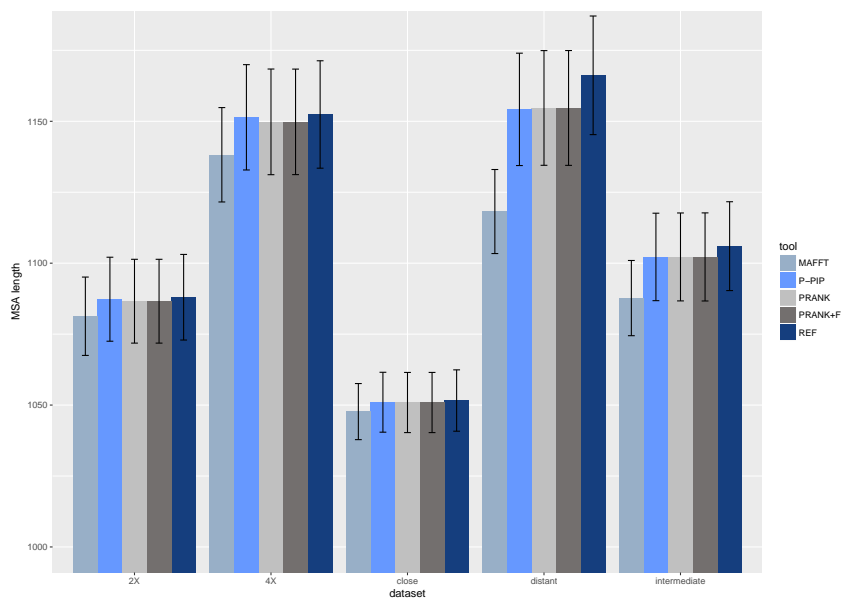


Figure S1 MSA length. The labels REF, PRNK, PRNK+F and P-PIP refer to REFERENCE alignments, PRANK, PRANK using the 'F' option and our Progressive-PIP algorithm, respectively. The values reported correspond to the mean MSA length over 250 simulations for each different evolutionary scenario, standard deviation in parentheses.

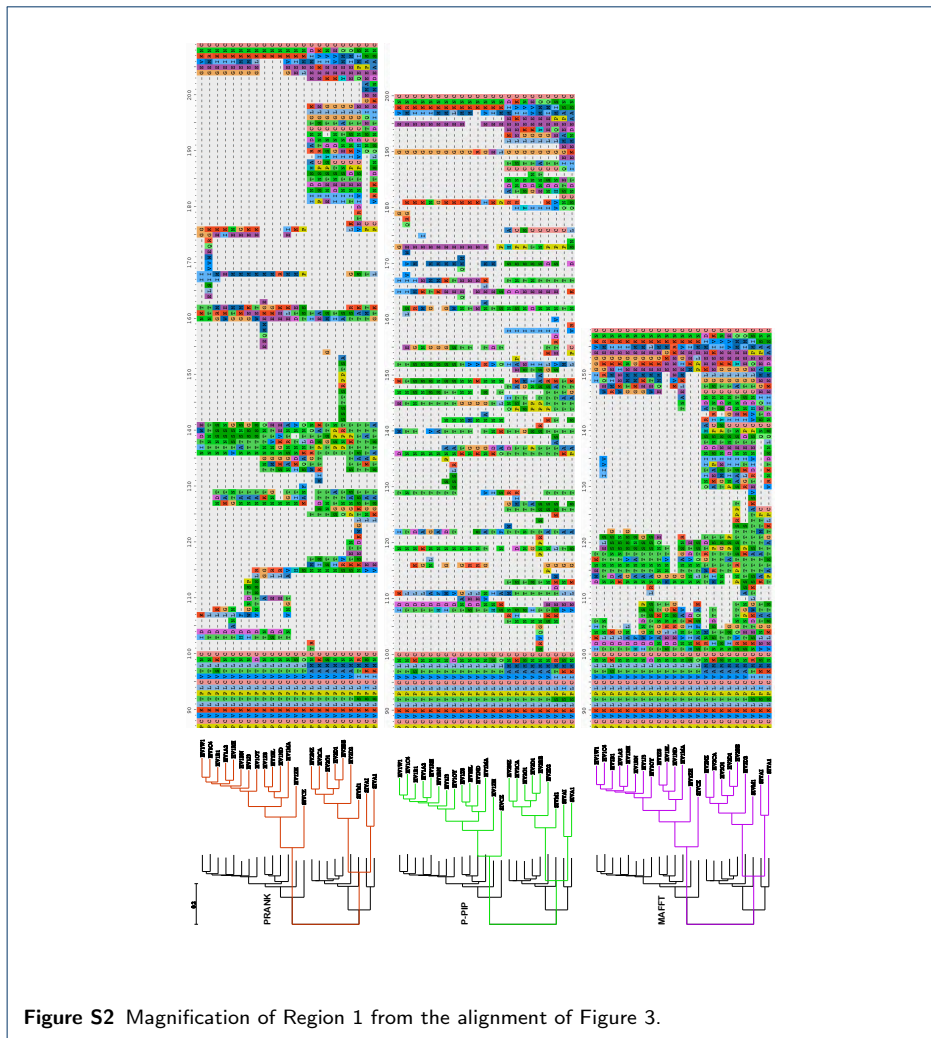


Figure S2 Magnification of Region 1 from the alignment of Figure 3.

