Selection of Models of Genomic Evolution in HPC Environments

Diego Darriba López

Departmento de Electrónica e Sistemas
Universidade da Coruña

Departmento de Bioquímica, Xenética e Inmunoloxía
Universidade de Vigo

Departmento de Electrónica e Sistemas
Universidade da Coruña

Departmento de Bioquímica, Xenética e Inmunoloxía
Universidade de Vigo

PhD Thesis

Selection of Models of Genomic Evolution in High Performance Computing Environments

Diego Darriba López

October 2015

PhD Advisors:
Guillermo López Taboada
Ramón Doallo Biempica
David Posada González

Dr. Guillermo López Taboada
Profesor Titular de Universidad
Dpto. de Electrónica y Sistemas
Universidade da Coruña

Dr. Ramón Doallo Biempica
Catedrático de Universidad
Dpto. de Electrónica y Sistemas
Universidade da Coruña

Dr. David Posada González
Catedrático de Universidad
Dpto. de Bioquímica, Genética e Inmunología
Universidade de Vigo

CERTIFICAN

Que la memoria titulada "*Selection of Models of Genomic Evolution in HPC Environments*" ha sido realizada por D. Diego Darriba López bajo nuestra dirección en el Departamento de Electrónica y Sistemas de la Universidad de A Coruña y en el Departamento de Bioquímica, Genética e Inmunología de la Universidade de Vigo en régimen de cotutela, concluye la tesis doctoral que presenta para optar al grado de Doctor en Ingeniería Informática con la Mención de Doctorado Internacional.

En A Coruña y Vigo, a 20 de Octubre de 2015

Fdo.:
Guillermo López Taboada
Director de la Tesis Doctoral

Fdo.:
Ramón Doallo Biempica
Director de la Tesis Doctoral

Fdo.:
David Posada González
Director de la Tesis Doctoral

Fdo.: Diego Darriba López
Autor de la Tesis Doctoral

Talvez desapareciches da miña vida,
pero nunca do meu corazón.

Acknowledgments

This research project would not have been possible without the support of many people. I wish to express my gratitude to my PhD advisors, Guillermo, Ramón and David who were abundantly helpful and offered invaluable assistance, support and dedicated guidance.

Special thanks also to all my collaborators and fellow group members; Adrián, Andión, Andrés, CH, Dyer, Iván, Jorge, Jurjo, Moisés, Porta, Raquel, Rober, Sabela and Toño from GAC; Álex, Andrés, Carlos, Diego, Klaus, Leo, María, Mateus, Merchi, Miguel A., Miguel F., Ramón and Sara from XB5/Phylogenomics; Alexey, Andre, Fernando, Jiajie, Kassian, Nikos, Paschalia, Pavlos, Solon and Tomas from H-ITS. Thank you all for being so collaborative and helpful. I would like to thank my parents, brothers and family for their unconditional support, as well as my friends from Lugo, A Coruña and Heidelberg, who have always stood by me.

I would like to sincerely thank Prof. Dr. Alexandros Stamatakis and the Scientific Computing Group for hosting me during my three-month research visit to the Heidelberg Institute for Theoretical Studies (Germany). Moreover, I am grateful to CESGA (Galicia Supercomputing Center) for providing access to the clusters and supercomputers that have been used as testbeds for this work.

Finally, I want to acknowledge the following institutions and projects for funding this work: the Computer Architecture Group and the Department of Electronics and Systems from the University of A Coruña; and the Phylogenomics Group and the Department of Biochemistry, Genetics and Immunology from the University of Vigo, for the human and material support, and for funding my attendance at several conferences; the Galician Government (Xunta de Galicia) for the Consolidation Program of Competitive Research Groups (Computer Architecture Group, refs. GRC2013/055 and 2010/6), the Galician Network of High Performance Computing (ref. 2010/53) and the Galician Network of Bioinformatics (ref.

2010/90); the European Research Council for the project ERC-2007-Stg 203161-PHYGENOM; the Spanish Ministry of Science and Innovation for the project TIN2010-16735 (Computer Architecture Group); and the Spanish Ministry of Science and Education for the project BFU2009-08611 (Phylogenomics Group).

A convincing demonstration of correctness being impossible as long as the mechanism is regarded as a black box, our only hope lies in not regarding the mechanism as a black box.

E.W.Dijkstra

Resumo

A introducción das tecnoloxías de sequenciación de nova xeración, ou "Next-Generation Sequencing" (NGS), representou un notable cambio no campo da filoxenética. A cantidade de información molecular dispoñible está a crecer cada vez máis rápido, propiciando o desenvolvemento de métodos e ferramentas de análise máis eficientes, así coma o uso de técnicas de computación de altas prestacións (HPC) para acelerar as análises. O campo está a cambiar velozmente da análise filoxenética (i.e., estudo dun ou un conxunto reducido de xens) á filoxenómica (i,e., estudo de centos ou millares de xens de xenomas completos ou incompletos). Moitos métodos filoxenéticos requiren o uso de modelos probabilísticos de evolución molecular, e é coñecido que o uso dun modelo ou outro pode derivar en diferentes estimacións filoxenéticas. Tanto modelos sub- como sobreparametrizados presentan desvantaxes en termos de precisión. Polo tanto, existen ferramentas populares que fan uso de marcos estadísticos para seleccionar o modelo que mellor se axusta aos datos, buscando o mellor compromiso entre likelihood (verosimilitude) e parametrización.

Esta tese doutoral presenta o deseño, implementación e evaluación de métodos HPC para seleccionar o modelo de evolución máis adecuado, en conxunto co desenvolvemento de novas funcións orientadas a facer máis sinxela a análise de datos filoxenéticos. En particular, extendemos e paralizamos as dúas ferramentas máis populares para a selección de modelos de ADN e proteínas, *jModelTest* e *ProtTest*. Ademáis, esta tese presenta o deseño, implementación e evaluación de algoritmos para a análise rápida e precisa de datos xenómicos. Creamos unha ferramenta incorporando todas estas técnicas, denominada *PartitionTest*, delegando a computación principal na librería de análise filoxenética PLL. Finalmente, fixemos un estudo de simulacións sobre a importancia do uso de técnicas de selección de modelos en datos xenómicos, e o seu impacto na precisión ao recuperar os modelos

xeradores e, máis importante, a árbore de evolución verdadeira.

Resumen

La introducción de las tecnologías de secuenciación de nueva generación, o "Next-Generation Sequencing" (NGS), ha representado un notable cambio en el campo de la filogenética. La cantidad de información molecular disponible está creciendo cada vez más rápido, propiciando el desarrollo de métodos y herramientas de análisis más eficientes, así como el uso de técnicas de computación de altas prestaciones (HPC) para acelerar los análisis. El campo está cambiando rápidamente del análisis filogenético (i.e., estudio de uno o un conjunto reducido de genes) al filogenómico (i,e., estudio de cientos o miles de genes de genomas completos o incompletos). Muchos métodos filogenéticos requieren utilizar modelos probabilísticos de evolución molecular, y es sabido que el uso de un modelo u otro puede derivar en diferentes estimaciones filogenéticas. Tanto modelos sub- como sobreparametrizados presentan desventajas en términos de precisión. Por lo tanto, existen herramientas populares que hacen uso de marcos estadísticos para seleccionar el modelo que mejor se ajuste a los datos, buscando el mejor compromiso entre likelihood (verosimilitud) y parametrización.

Esta tesis doctoral presenta el diseño, implementación y evaluación de métodos HPC para seleccionar el modelo de evolución más adecuado, conjuntamente con el desarrollo de nuevas funciones orientadas a facilitar el análisis de datos filogenéticos. En concreto, hemos extendido y paralizado las dos herramientas más populares para selección de modelos de ADN y proteínas, *jModelTest* y *ProtTest*. Además, esta tesis presenta el diseño, implementación y evaluación de algoritmos para el análisis rápido y preciso de datos genómicos. Hemos creado una herramienta incorporando todas estas técnicas, denominada *PartitionTest*, delegando la computación principal en la librería de análisis filogenético PLL. Finalmente, hemos hecho un estudio de simulaciones sobre la importancia del uso de técnicas de selección de modelos en datos genómicos, y su impacto en la precisión al recuperar

los modelos generadores y, más importante, el árbol de evolución verdadero.

Abstract

The irruption of Next-Generation Sequencing (NGS) technologies has changed dramatically the landscape of phylogenetics. The available molecular data keeps growing faster and faster, prompting the development of more efficient analytical methods and tools, as well as the use of High Performance Computing (HPC) techniques for speeding-up the analyses. The field is rapidly changing from phylogenetics (i.e., the study of a single or a few genes) to phylogenomics (i.e., the study of hundreds or thousands of genes from incomplete or complete genomes). Many phylogenetic methods require the use of probabilistic models of molecular evolution, and it is well known that the use of different models may lead to different phylogenetic estimates. Both under- and overparameterized models present disadvantages in terms of accuracy. Therefore, there are popular tools that employ statistical frameworks for selecting the most suitable model of evolution for the data, finding the best trade-off among likelihood and parameterization.

This PhD thesis presents the design, implementation and evaluation of HPC methods for selecting the best-fit model of evolution, together with improved features that facilitate the analysis of single-gene data. In particular, we extended and parallelized the two most popular tools for selecting the best-fit model of evolution for DNA and proteins, *jModelTest* and *ProtTest*. Furthermore, this thesis presents the design, implementation and evaluation of algorithms for fast and accurate analysis of multi-gene data. We created a tool incorporating all these techniques, called *PartitionTest*, delegating the core computations to the Phylogenetic Likelihood Library (PLL). Finally, we made a simulation study on the importance of using model selection techniques on multi-gene data, and its impact on the accuracy retrieving the true generating models and, most important, the

true phylogenies.

Contents

Preface **1**

1. Introduction **7**

1.1. Phylogenetic Inference 7

1.1.1. Models of Evolution 8

1.1.2. Phylogenetic Trees 13

1.1.3. Computing the Likelihood of a Tree 15

1.1.4. Selecting the Best-Fit Model of Evolution For Single-Gene Alignments 21

1.1.5. Selecting the Best-Fit Model of Evolution For Multigene Alignments: Partitioning 27

1.2. Software for Model Selection 31

1.2.1. ProtTest 2.x 31

1.2.2. ModelTest and jModelTest 1.x 32

1.3. Scope and Motivation 34

1.4. Main Objectives of the Thesis 34

2. ProtTest 3 - Protein Model Selection **37**

3. HPC Model Selection for Multicore Clusters **45**

4. DNA Model Selection on the Cloud **61**

5. jModelTest2 - Nature Methods **69**

6. PartitionTest - Multigene Model Selection **85**

7. Discussion **113**

8. Conclusions **117**

9. Future Work **119**

10. A Summary in Spanish **121**

10.1. Introducción 121

10.2. Organización de la Tesis 123

10.3. Medios 125

10.4. Discusión 127

10.5. Conclusiones 130

10.6. Trabajo Futuro 132

10.7. Principales Contribuciones 133

References **135**

List of Tables

1.1. Standard amino-acid abbreviations 8

1.2. Substitution models available in jModelTest. 33

List of Figures

1.1. Graphic representation of a rooted tree for 3 species. 14

1.2. Graphic representation of an unrooted tree for 4 species (in the middle). A root can be placed on any of the 5 branches, producing 5 different rooted topologies out of the same unrooted topology. . . 15

1.3. Conditional likelihood vectore entries are computed based on the CLVs of the children nodes, the model and the transition probability matrices $P(b_{qp})$ and $P(b_{rp})$. 18

1.4. Probability density function of the Γ distribution 21

1.5. Example of a forward hierarchy of Likelihood Ratio Tests (hLRT) for 6 candidate models 24

1.6. Example of dynamical Likelihood Ratio Tests (dLRT) 24

1.7. Partitioning schemes for $N = 4$. Numbers and colors represent the different partitions. In this example, there are $2^N - 1 = 15$ partitions and $B(N) = 15$ partitioning schemes. 30

4.1. jModelTest.org login screen 65

4.2. jModelTest.org MSA input 65

4.3. jModelTest.org execution options 66

4.4. jModelTest.org running status 67

7.1. ProtTest 3 shared memory strategy. 114

Preface

Work Methodology

This thesis follows a classical approach in scientific and technological research: analysis, design, implementation and evaluation. Thus, the thesis starts with the analysis of the importance of selecting the best-fit models of evolution, the options already available and the feasibility and impact analysis of providing High Performance Computing capabilities to this task. We first tackled the model selection for single-gene DNA and protein data, re-designing from scratch the two most popular tools, adding HPC and fault tolerance capabilities, as well as new methods and algorithms. We then evaluated the accuracy and performance in several representative architectures.

Next, we analysed the problem of model assignment for multigene data, a problem also known as *partitioning*. Among available options for Maximum-Likelihood parameter estimation and phylogenetic searches, the use of the Phylogenetic Likelihood Library (PLL) turned out to be the most flexible approach. We collaborated in the development of PLL, adding flexibility for managing partitions of data (i.e., subsets of the data where different models can be used), from which we devised and implemented heuristic algorithms in linear and polynomial time for selecting the best-fit partitioning scheme. Finally, we evaluated the partitioning and phylogenetic accuracy of our algorithms using simulated and real-world data sets.

Structure of the Thesis

In accordance with the current regulations of the University of A Coruña, this PhD dissertation has been structured as a compilation thesis (i.e., merging research articles). In particular, this thesis comprises three JCR-indexed journal scientific articles, presented in chapters 3, 4 and 5, together with two additional articles of great interest within the scope of the work. The thesis begins with the Introduction chapter, intended to give the reader a general overview of model selection in phylogenetics.

This chapter introduces the scope, main motivations and objectives of the thesis.

The scientific articles included in the compilation, each one presented as a separate chapter (Chapters 2-6) are the following:

- Chapter 2: **Darriba, D.**, Taboada, G. L., Doallo, R., & Posada, D. (2011). ProtTest 3: fast selection of best-fit models of protein evolution. *Bioinformatics, 27(8), 1164-1165.*

  Impact factor: **4.981**, **547** citations as of October 2015.

- Chapter 3: **Darriba, D.**, Taboada, G. L., Doallo, R., & Posada, D. (2013). High-performance computing selection of models of DNA substitution for multicore clusters. *International Journal of High Performance Computing Applications, 1094342013495095.*

  Impact factor: **1.477**, **4** citations as of October 2015.

- Chapter 4: Santorum, J. M., **Darriba, D.**, Taboada, G. L., & Posada, D. (2014). jmodeltest.org: selection of nucleotide substitution models on the cloud. *Bioinformatics, btu032.*

  Impact factor: **4.981**

- Chapter 5: **Darriba, D.**, Taboada, G. L., Doallo, R., & Posada, D. (2012). jModelTest 2: more models, new heuristics and parallel computing. *Nature methods, 9(8), 772-772.*

  Impact factor: **32.072**, **1,849** citations as of October 2015.

- Chapter 6: **Darriba, D.**, & Posada, D. (2015). The impact of partitioning on phylogenomic accuracy. *bioRxiv, 023978*.

In Chapter 7 we include a general discussion of the included research articles. Chapter 8 describes the conclusions of the thesis. Finally, in Chapter 9 we describe the future work.

Funding and Technical Means

- Working material, human and financial support primarily by the Computer Architecture Group of the University of A Coruña and the Phylogenomics Group of the University of Vigo.

- Access to bibliographical material through the libraries of the Universities of A Coruña and Vigo.

- Additional funding through the following research projects:

  - Regional funding by the Galician Government (Xunta de Galicia) under the Consolidation Program of Competitive Research Groups (Computer Architecture Group, refs. GRC2013/055 and 2010/6), Galician Network of High Performance Computing (ref. 2010/53), and Galician Network of Bioinformatics (ref. 2010/90).
  - European Research Council (Phylogenomics Group, ref. ERC-2007-Stg 203161-PHYGENOM).
  - The Ministry of Science and Innovation of Spain under Project TIN2010-16735 (Computer Architecture Group).
  - Amazon Web Services (AWS) research grant "EC2 in phylogenomics".
  - The Spanish Ministry of Science and Education under project BFU2009-08611 (Phylogenomics Group)

- Access to clusters, supercomputers and cloud computing platforms:

- *Pluton cluster* (Computer Architecture Group, University of A Coruña, Spain). Initially, 16 nodes with 2 Intel Xeon quad-core Nehalem-EP processors and up to 16 GB of memory, all nodes interconnected via InfiniBand DDR and 2 of them via 10 Gigabit Ethernet. Additionally, two nodes with one Intel Xeon quad-core Sandy Bridge-EP processor and 32 GB of memory, interconnected via InfiniBand FDR, RoCE and iWARP, and four nodes with one Intel Xeon hexa-core Westmere-EP processor, 12 GB of memory and 2 GPUs NVIDIA Tesla "Fermi" 2050 per node, interconnected via InfiniBand QDR. Later, 16 nodes have been added, each of them with 2 Intel Xeon octa-core Sandy Bridge-EP processors, 64 GB of memory and 2 GPUs NVIDIA Tesla "Kepler" K20m per node, interconnected via InfiniBand FDR.
- *Diploid cluster* (Phylogenomics Group, University of Vigo, Spain). (1) One fat node with 4 Intel Xeon ten-core Westmere-EX processors and 512 GB of memory, (2) 30 nodes with 2 Intel Xeon E5-420 quad-core Harpertown processors (a total of 8 cores) and 16 GB of memory, and (3) 44 nodes with 2 Intel Xeon X5675 hexa-core Westmere-EP (a total of 12 cores); 50 GB of memory and Hyperthreading disabled.
- *Magny cluster* (Heidelberg Institute for Theoretical Studies, Heidelberg, Germany). (1) 2 Sandy Bridge nodes with 2 Intel Xeon E5-2630 hexa-core Sandy Bridge processors (a total of 12 cores) and Hyperthreading disabled;32 GB of memory, and (2) Magny-Cours node with 4 AMD Opteron 6174 12-core processors (a total of 48 cores); 128 GB of memory.
- *Finis Terrae supercomputer* (Galicia Supercomputing Center, CESGA, Spain): 144 nodes with 8 Intel Itanium-2 dual-core Montvale processors and 128 GB of memory, interconnected via InfiniBand DDR. Additionally, we have used one Superdome system with 64 Intel Itanium-2 dual-core Montvale processors and 1 TB of memory.
- Amazon EC2 IaaS cloud platform (Amazon Web Services, AWS). Several instance types have been used: (1) CC1, 2 Intel Xeon quad-core Nehalem-EP processors, 23 GB of memory and 2 local storage disks per instance; (2) CC2, 2 Intel Xeon octa-core Sandy Bridge-EP processors,

60.5 GB of memory and 4 local storage disks per instance; (3) CG1, 2 Intel Xeon quad-core Nehalem-EP processors, 22 GB of memory, 2 GPUs NVIDIA Tesla "Fermi" 2050 and 2 local storage disks per instance; (4) HI1, 2 Intel Xeon quad-core Westmere-EP processors, 60.5 GB of memory and 2 SSD-based local storage disks per instance; (5) CR1, 2 Intel Xeon octa-core Sandy Bridge-EP processors, 244 GB of memory and 2 SSD-based local storage disks per instance; and (6) HS1, 1 Intel Xeon octa-core Sandy Bridge-EP processor, 117 GB of memory and 24 local storage disks per instance. All these instance types are interconnected via 10 Gigabit Ethernet.

- Pre-doctoral fellowship at the University of A Coruña, Spain.
- A six-month research visit, as well as regular short-time visits, to the Phylogenomics Group at the University of Vigo, Spain.
- A three-month research visit to the Heidelberg Institute for Theoretical Studies at Heidelberg, Germany, which has allowed to collaborate in the development of the Phylogenetic Likelihood Library, adding flexibility for partitioning management and making it suitable for *PartitionTest*. This research visit was funded by the University of A Coruña.
- Research associate contract at the Heidelberg Institute for Theoretical Studies, Germany.

Main Contributions

The main contributions of this Thesis are:

1. Design, implementation and evaluation of High Performance Computing algorithms for the statistical selection of best-fit empirical amino acid replacement models for protein data, incorporated into *ProtTest 3.0*.
2. Design, implementation and evaluation of High Performance Computing algorithms for the statistical selection of mechanistic best-fit substitution models for DNA, incorporated into *jModelTest 2.0*. This work facilitates the use

of HPC architectures for selecting the most suitable evolution model. It incorporates also fault tolerance through a *checkpointing* system.

3. Design of new visualization techniques for model selection.

4. Extended methods for DNA model selection, supporting all 203 substitution schemes of GTR submodels.

5. Extended analysis methods of the impact of DNA substitution models on particular data sets, such as HTML reports and topological summary on the different competing models.

6. Extension of the Phylogenetic Likelihood Library for supporting flexible data partitions.

7. Design, implementation and evaluation of High Performance Computing algorithms for correctly address heterogeneity through data partitioning, incorporated into *PartitionTest*.

8. Evaluation of the impact of partitioning schemes on phylogenetic inference.

Chapter 1

Introduction

This introductory chapter is intended to provide the reader with the appropriate conceptual background in order to better understand the research carried out in this thesis. The structure of this chapter is as follows: Section 1.1 introduces the underlying theoretical background, Section 1.2 introduces the base software for Chapters 2 to 5. Section 1.3 describes the scope and main motivations of the thesis. Finally, a clear description of the main objectives is included in Section 1.4.

1.1. Phylogenetic Inference

Bioinformatics is an interdisciplinary field whose goal is the development of methods and software tools for understanding biological data. It combines computer science, statistics, mathematics, and engineering to study and process biological data. Bioinformatics is a huge field, that involves many different activities such as genome annotation, molecular evolution, protein structure prediction, biological networks, systems biology, among others. In particular, this work focuses on phylogenetic analysis.

Phylogenetics is the study of evolutionary relatedness among groups of organisms of any taxonomic rank (usually species or populations) or among molecules (genes, proteins), broadly referred to as "taxa", descending from a common ancestor. This relationships are established upon similarities and differences in their

morphological or molecular characteristics. Morphological similarities are inferred from data describing, for example, traits shared among species. On the other hand, molecular sequences (e.g., DNA, RNA, or Amino Acid data) can be obtained through sequencing technologies. Nowadays, Next-Generation Sequencing (NGS) technologies are the most common source of molecular data [72, 58], due to their much higher throughput compared to previous approaches [71], and the continuous decreasing cost. NGS sequencing techniques present a triple trade-off between speed (data output), price and accuracy.

A molecular sequence (from now on, a "sequence") can be represented as a string of characters from a finite alphabet. For example, this alphabet comprises 4 nucleotides (a.k.a., nitrogen bases or bases) for DNA {A,C,G,T}, or RNA, {A,C,G,U}, and 20 amino acids for protein data (Table 1.1).

Table 1.1: Standard amino-acid abbreviations

1-Letter	**3-Letter**	**Amino-Acid**	**1-Letter**	**3-Letter**	**Amino-Acid**
A	Ala	Alanine	R	Arg	Arginine
N	Asn	Asparagine	D	Asp	Aspartic acid
C	Cys	Cysteine	E	Glu	Glutamic acid
Q	Gln	Glutamine	G	Gly	Glycine
H	His	Histidine	I	Ile	Isoleucine
L	Leu	Leucine	K	Lys	Lysine
M	Met	Methionine	F	Phe	Phenylalanine
P	Pro	Proline	S	Ser	Serine
T	Thr	Threonine	W	Trp	Tryptophan
Y	Tyr	Tyrosine	V	Val	Valine

It is possible to convert DNA data from coding regions into the amino acids that conform the proteins [17]. In a gene, the coding region or coding sequence is a portion of its DNA that codes for proteins.

1.1.1. Models of Evolution

In molecular evolution, a typical data set consists of an alignment of multiple sequences (DNA from coding or non-coding regions, polypeptide chains, ...), or MSA (Multiple Sequence Alignment) and the model, in the large sense, is the

phylogenetic tree that relates the sequences (see Section 1.1.2) plus the mechanism of molecular change. For clarity, we separate the two parts of the model, and call the phylogenetic tree part "the tree" and the mechanism part "the model", or *statistical model of substitution.*

The model, in keeping with the previous definition, describes how likely it is for a state to change into a different state within a certain evolutionary time t.

We describe here the mathematical background of substitution models. The models used in this thesis contain two main sets of parameters: a vector π describing the equilibrium state frequencies, and a substitution rate matrix $Q = \{q_{i,j}\}$ defining the probability that state i mutates into state j, for $i \neq j$, in an infinitely small amount of time dt. The diagonals of the Q matrix are chosen so that the rows sum to zero (Equation 1.1).

$$Q_{ii} = -\sum_{\{j|j\neq i\}} Q_{ij} \tag{1.1}$$

The number of substitutions can be estimated using a memory-less continuous-time Markov model. We assume that sites evolve independently from each other. For example, for DNA data the Markov chain has four possible states (A, C, G, T) and the next state depends only on the current one.

The Q matrix is computed out the stationary frequencies and a matrix of relative substitution rates, R. It defines the relative rate at which one state can change into another. If the R matrix is symmetrical, the Q matrix will be also symmetrical, and we call the model time-reversible. A symmetrical R matrix contains $\frac{1}{2}s(s-1)$ parameters, where s is the number of different states (e.g., 4 for nucleotides, 20 for amino acids).

$$R = \begin{pmatrix} - & \alpha & \beta & \gamma \\ \alpha & - & \delta & \epsilon \\ \beta & \delta & - & \zeta \\ \gamma & \epsilon & \zeta & - \end{pmatrix}$$

The Q matrix is then computed by multiplying columns in R by the π vector.

The diagonal in Q is set so that its rows (and columns) sum to 0, then it follows that $\pi Q = 0$.

$$Q = \begin{pmatrix} -(\pi_c\alpha + \pi_g\beta + \pi_t\gamma) & \pi_c\alpha & \pi_g\beta & \pi_t\gamma \\ \pi_a\alpha & -(\pi_a\alpha + \pi_g\delta + \pi_t\epsilon) & \pi_g\delta & \pi_t\epsilon \\ \pi_a\beta & \pi_c\delta & -(\pi_a\beta + \pi_c\delta + \pi_t\zeta) & \pi_t\zeta \\ \pi_a\gamma & \pi_c\epsilon & \pi_g\zeta & -(\pi_a\gamma + \pi_c\epsilon + \pi_g\zeta) \end{pmatrix}$$

To compute the probability of a state changing into another given time t, we need to calculate the transition-probability matrix as follows:

$$P(t) = e^{Qt} \tag{1.2}$$

The Markov chain is reversible if and only if $\pi_i q_{i,j} = \pi_j q_{j,i}$ for all $i \neq j$. Furthermore, $\lim_{t\to\infty} P_{ij}(t) = \pi_j$, or, in other words, as time goes to infinity the probability of finding base j at a position where originally was a base i goes to the equilibrium probability π_j, regardless of the original base. Furthermore, it follows that $\pi P(t) = \pi$ for all t.

The standard approach to calculate the transition probability matrix $P(t)$ is to compute numerically the eigenvalues and eigenvectors of the rate matrix Q. The decomposition is $Q = U\Lambda U^{-1}$, where Λ is a diagonal matrix containing the eigenvalues of Q and U is a square matrix whose i^{th} column is the eigenvector of Q. The transition probability matrix for time t is then computed as shown in Equation 1.3:

$$P(t) = e^{Qt} = Q = Ue^{\Lambda t}U^{-1} \tag{1.3}$$

Empirical vs Mechanistic Models

A main difference among evolutionary models is how many free parameters (i.e., degrees of freedom) are estimated every time for the data set under consideration and how many of them are estimated once on a large database. Mechanistic

models, typically used for DNA data, describe all substitution as a function of a number of parameters which are estimated for every data set analysed, preferably using Maximum Likelihood (see Section 1.1.3). This has the advantage that the model can be adjusted to the particularities of a specific data set (e.g. different composition biases in DNA). However, problems can arise when too many parameters are used, particularly if they can compensate for each other. Then it is often the case that the data set is too small to yield enough information to estimate all parameters accurately. For example, a mechanistic time-reversible model for protein data would need to estimate 190 substitution rate parameters (all possible transitions between the 20 amino acids), 19 equilibrium frequencies and also parameters for accounting rate heterogeneity. For this reason empirical models are usually preferred for the analysis of protein alignments, because either there is not enough data to estimate such amount of free parameters, or it is very computational expensive.

Empirical protein models are defined by estimating many parameters (typically all entries of the substitution rate matrix and the equilibrium frequencies) from a large database. These parameters are then fixed and reused for the data set at hand. This has the advantage that these parameters can be estimated more accurately. Normally, empirical models are used when it is not possible to estimate all entries of the substitution matrix from the current data set. On the downside, the estimated parameters might be too generic and not fit a particular data set well enough.

With the high throughput genome sequencing still producing very large amounts of DNA and protein sequences, there is enough data available to create empirical models with any number of parameters. However, because of the problems mentioned above, the two approaches are often combined, by estimating most of the parameters once on large-scale data, while only a few are then adjusted to the data set under consideration, as is usually the case of parameters for addressing rate heterogeneity (e.g., different per-site rates). Rate heterogeneity in models of evolution is explained in detail in Section 1.1.3.

Models of Nucleotide Substitution

For DNA data (4 different states), it is relatively easy to estimate the model free parameters, and therefore mechanistic models are used. The GTR (General Time Reversible) model [52, 87] is the most general reversible model. A detailed description of GTR and other nucleotide and amino acid evolution models can be found in Chapters 1 and 2 of [96].

$$Q = \begin{pmatrix} -\mu_A & \mu_{GA} & \mu_{CA} & \mu_{TA} \\ \mu_{AG} & -\mu_G & \mu_{CG} & \mu_{TG} \\ \mu_{AC} & \mu_{GC} & -\mu_C & \mu_{TC} \\ \mu_{AT} & \mu_{GT} & \mu_{CT} & -\mu_T \end{pmatrix}$$

where $\mu_{x_i x_j}$ is the transition rate from state x_i to state x_j, and $\mu_{x_i} = \sum_{i \neq j} \mu_{x_i x_j}$. Usually transition rates are normalized such that $\mu_{s(s-1)} = 1$, where s is the number of different states. Since the four stationary frequencies must sum to 1, and the six substitution rates are relative to each other, the GTR model has a total of eight free parameters (five rates + three frequencies). Other nucleotide substitution models can be derived from GTR by simply imposing restrictions on the base frequencies and/or the substitution rates. These derived models are simpler (nested within GTR) and have a lower number of free parameters. For instance, the Jukes-Cantor model [42] is given by assuming equal frequencies and substitution rates ($\pi_A = \pi_C = \pi_G = \pi_T = 0.25$ and $\mu_{AC} = \mu_{AG} = \mu_{AT} = \mu_{CG} = \mu_{CT} = \mu_{GT} = 1.0$).

Models of Amino Acid Replacement

As stated before, for amino acid data it might not be enough data for estimating the 190 substitution rate parameters out of the 20 different states, but also if there is it would be very computationally expensive. Therefore, empirical models are used instead.

Stationary frequencies are usually not optimized by maximizing the likelihood (see Section 1.1.3), but defined by the model or computing the empirical frequencies that can be observed in the data. In the latter, they belong to the set of free

parameters of the model.

It is rather normal, however, that empirical models of amino acid replacement do not specify values for any of the rate heterogeneity parameters. They are thus free parameters.

1.1.2. Phylogenetic Trees

The relations among taxa (e.g., organisms, species, populations, gene copies, proteins, cells) are usually modelled using phylogenetic trees, even though from a biological point of view some events cannot be modelled as a strict branching process. For example, a phylogenetic tree cannot represent some evolutionary events such Horizontal Gene Transfer (HGT) or hybridization.

A phylogenetic tree is a branching diagram or "tree". The taxa joined together in the tree are implied to have descended from a recent common ancestor. The tips of the tree represent the descendent taxa, and the inner nodes represent the common ancestors. The length of the branches represent the expected number of substitutions per site. Figure 1.1 shows an example of a rooted tree. Taxa or species A and B, since they split from the same node, are called "sister taxa" – they are each other's closest relatives among the species included in the tree. The goal of many bioinformatic methods is to infer good phylogenies, or as close as possible to the real tree of life. The shape of the tree is known as *topology* (i.e., the tree ignoring the branch lengths).

A phylogenetic tree can be either bifurcating or multifurcating. A bifurcating topology has exactly two descendants arising from each interior node. A multifurcating tree can have nodes with more than two descendants or sister taxa, and each of those nodes are called a "polytomy". A polytomy can represent the literal hypothesis that a common ancestral population split into multiple lineages through cladogenesis (i.e., speciation). Such a node is referred to as a "hard polytomy". Nevertheless, the vast majority of times politomies do not imply that the same ancestor produces simultaneously all daughter taxa, but rather they represent the uncertainty around which pattern is the best hypothesis. This node is referred to as a "soft polytomy". However, dealing with polytomies is out of the scope of this

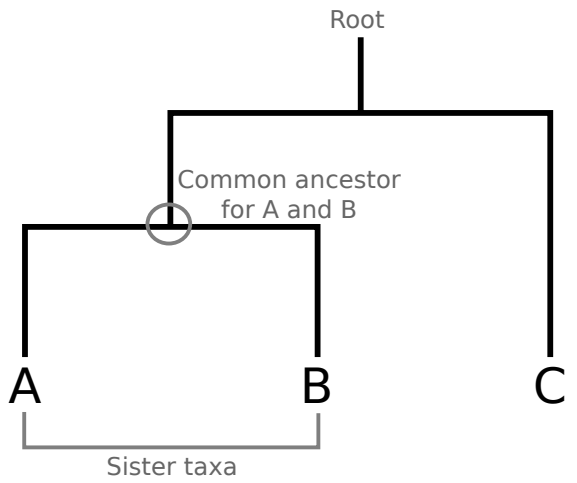

Figure 1.1: Graphic representation of a rooted tree for 3 species.

thesis and we will focus in bifurcating trees.

Other characteristic of trees is that they might or not contain a root node providing an evolutionary direction. According to this, we can differentiate rooted and unrooted topologies. An unrooted tree representation for N taxa is a graph with N tip or leaf nodes with degree 1, and N-2 inner nodes with degree 3. These structures provide information about the relationships between the species, but no direction is given. Given two inner nodes we cannot decide which one is older. Since most popular substitution models consider the evolution as a time-reversible process, it makes sense to work with such structures. Figure 1.2 shows an example of a 4-taxa unrooted tree. Starting from such a topology, we can root the tree at any of the branches. Usually an outgroup (a taxon clearly distant from every other) is included in the analysis in order to determine where to place the root for the group of interest, or ingroup.

Apart of the relationships between species (the *topology*), branch lengths play an important role for understanding the speciation process. In broad words, branch lengths express the amount of evolution. One might expect that the branch lengths are directly related with time, but they do not necessarily present such a direct relationship, since different lineages or even different sites in the DNA might evolve at different rates. A long branch can represent a long time or a high mutation rate

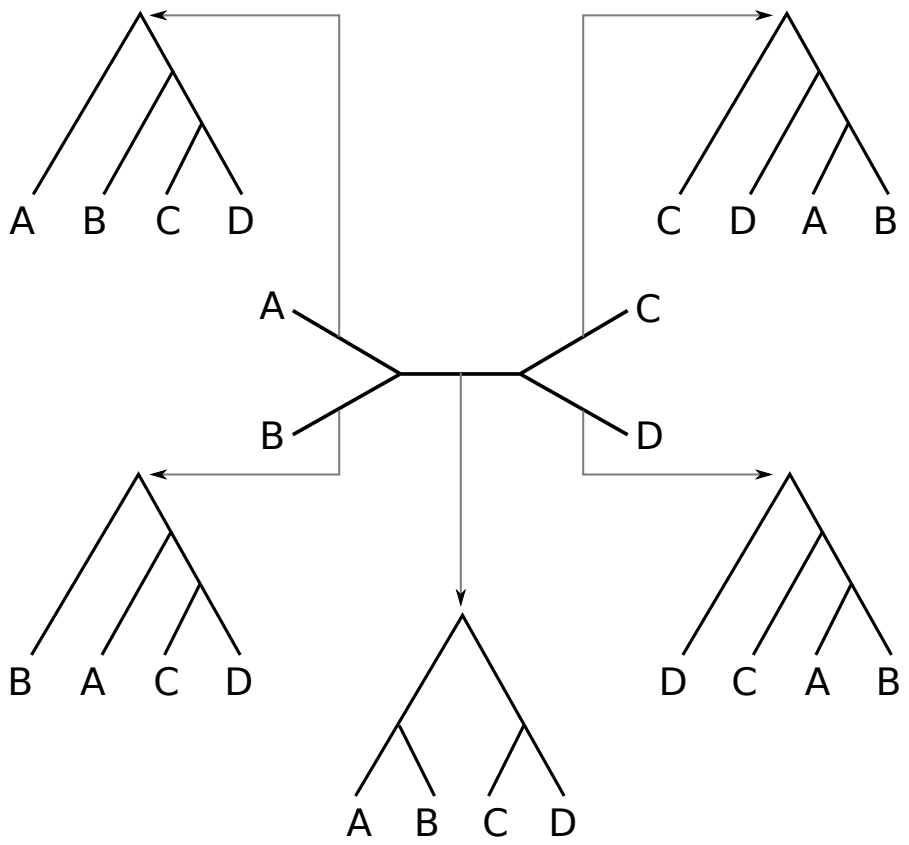

Figure 1.2: Graphic representation of an unrooted tree for 4 species (in the middle). A root can be placed on any of the 5 branches, producing 5 different rooted topologies out of the same unrooted topology.

1.1.3. Computing the Likelihood of a Tree

In 1981, Felsenstein described a Maximum Likelihood (ML) framework for modelling the process of nucleotide substitution combined with phylogenetic tree estimation [24]. As a consequence of the ML search process, applying the PLF (Phylogenetic Likelihood Function) to evaluate alternative trees dominates both the running time and memory requirements of most phylogenetic inference programs.

The computation of the PLF relies on the following assumptions:

1. Sites in the MSA evolve independently from each other.

2. Evolution in different parts of the tree is independent.

3. A comprehensive tree T, including a set of branch lengths b_{ij}.

4. A substitution model is available, and defines the transition probabilities $P_{i \to j}(b)$, that is the probability that j will be the final state at the end of a branch of length b, given that the initial state is i.

5. The substitution model is time-reversible, that is, $\pi_j P_{ij}(b) = \pi_i P_{j \to i}(b)$

These assumptions allow us to compute the likelihood of the tree as the product of the site-likelihoods of each column i of the MSA with n columns, as shown in Equation 1.4. Let T be a (rooted or unrooted) binary tree with n tips. Let θ be a set of (optimized or given) substitution model parameters (see Section 1.1.1). Let $\phi = \{b_{xy}\}$ be a set of (optimized or given) branch length values for tree T , where b_{xy} is the branch length value connecting nodes x and y in tree T ($b_{xy} = b_{yx}, x \neq y$ and $|\phi| = 2n - 3$).

$$L = Pr(D|T, \theta, \phi) = \prod_{i=1}^{n} Pr(D_i|T, \theta, \phi) \tag{1.4}$$

Likelihood scores usually reach very low values, that are either impossible or very computationally expensive to represent with current computer arithmetic. In order to resolve these problems, it is common practice to compute and report log likelihood values:

$$log(L) = log(Pr(D|T, \theta, \phi)) = \sum_{i=1}^{n} log(Pr(D_i|T, \theta, \phi)) \tag{1.5}$$

The PLF is computed in a postorder traversal (i.e., from tip nodes to the root). However, as we stated before, the root of the tree is, in general, unknown. For computing the likelihood on unrooted topologies, we place a virtual root into an arbitrary branch. Because the substitution model is time-reversible, the placement of the virtual root does not affect the likelihood of the tree. Figure 1.2 shows all possible virtual root placements on a 4 taxa tree.

For clarity, let us assume that we are working with a rooted tree and DNA data, where we have an alphabet of size four. Thus, only four states are possible and correspond to the nucleotides A, C, G, and T. For each site i, four entries must be computed to store the conditional, or marginal likelihood for nucleotide states 'a', 'c', 'g', and 't' at node p. The conditional likelihood entry $L_x^{(p)}(i)$ is the probability of everything that is observed from node p on the tree towards the tips, at site i, conditional on node p having state x [26]. We can define the conditional likelihood vector (CLV) at node p and site i as:

$$\vec{L}^{(p)}(i) = \left(L_a^{(p)}(i), L_c^{(p)}(i), L_g^{(p)}(i), L_t^{(p)}(i)\right) \tag{1.6}$$

$$L_x^{(p)}(i) = \left(\sum_{y=a}^{t} P_{x\to y}(b_{qp}) L_x^{(q)}(i)\right)\left(\sum_{x=a}^{t} P_{x\to y}(b_{rp}) L_x^{(r)}(i)\right) \tag{1.7}$$

This equation computes the conditional likelihood vector (CLV) entry $\vec{L}_a^{(p)}$ for observing the nucleotide 'a' at site i of a parent node p, with two child nodes q and r given the respective branch lengths b_{qp} and b_{rp} , the corresponding transition probability matrices $P(b_{qp})$, $P(b_{rp})$, and the conditional likelihood vectors of the children $\vec{L}^{(q)}$, $\vec{L}^{(r)}$ for site i. The children/parent relationship is given by the position of the root (see Figure 1.3).

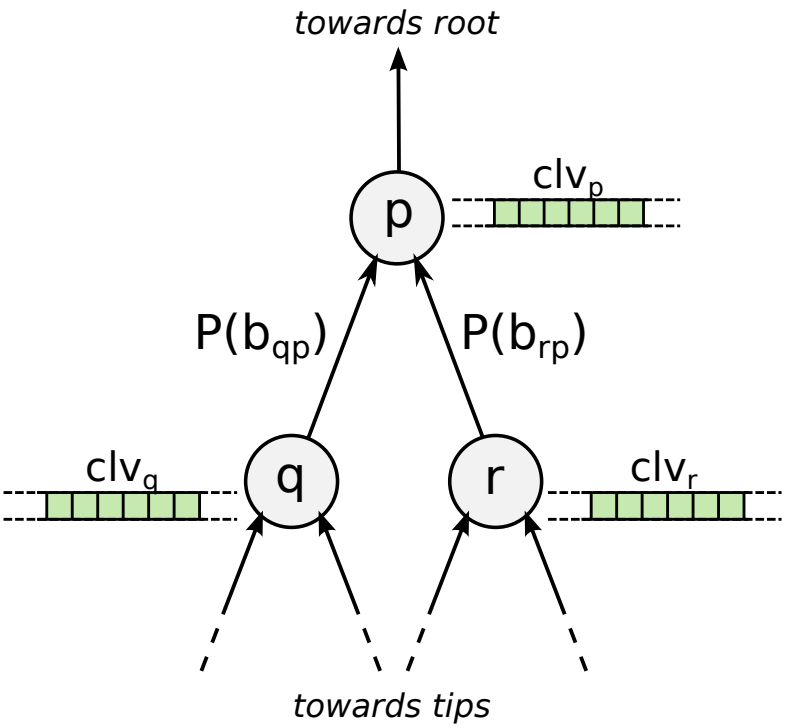

Figure 1.3: Conditional likelihood vectore entries are computed based on the CLVs of the children nodes, the model and the transition probability matrices $P(b_{qp})$ and $P(b_{rp})$.

The tips, which do not have any child nodes, must be initialized. Probability Vectors at the tips are also called tip vectors. In general, the sequence at the tips already have a known value for each site and therefore can be directly assigned a probability. For instance, if site i is an 'a', the tip vector can be directly initialized as: $\left(L_a^{(p)}(i), L_c^{(p)}(i), L_g^{(p)}(i), L_t^{(p)}(i)\right) := (1.0, 0.0, 0.0, 0.0)$.

To efficiently calculate the likelihood of a given, fixed tree, we execute a post-order tree traversal that starts at the root. Via the post-order tree traversal, the probabilities at the inner nodes (conditional likelihood vectors, or ancestral probability vectors) are computed bottom-up from the tips toward the root. This procedure to calculate the likelihood is called the Felsenstein pruning algorithm [24].

At the root node, the likelihood of a specific site in the tree can be computed using Equation 1.8.

$$L(i) = Pr(D_i \mid T, \theta, \phi) = \sum_{x=a}^{t} \pi_x L_x^{root}(i) \tag{1.8}$$

For the unrooted case, the likelihood is computed at a branch instead of a node, and it is computed as shown in Equation 1.9, for site i and the virtual root located between nodes p and q.

$$L(i) = \sum_{x=a}^{t} \pi_x L_x^{(p)}(i) \sum_{y=a}^{t} P_{x \to y}(b_{pq}) L_y^{(q)}(i) \tag{1.9}$$

Addressing Rate Heterogeneity Among Sites

The model initially proposed by Felsenstein assumes a constant rate of substitution among sites. However, this assumption has long been recognized as unrealistic, especially for genes that code for proteins or sequences that are otherwise functional (e.g., [89]). Since the early days of molecular data analysis, it is know that different sites or regions in DNA may evolve at different rates, or remain invariant [28]. The most popular method nowadays to account for rate variation among sites in nucleotide-substitution models consists in using a Γ distribution [41, 86].

In the Γ model, for each site the likelihood is integrated over a continuous Γ distribution of rates. The gamma density is given by Equation 1.10

$$g(r; \alpha, \beta) = \frac{\beta^{\alpha} r^{\alpha} e^{-\beta r}}{\Gamma(\alpha)} \tag{1.10}$$

The mean of a Γ distribution is α/β. A constrain $\alpha = \beta$ is specified, so that the mean rate is 1. Thus, the distribution has only one parameter α, which is usually adjusted by numerical optimization. Figure 1.4 shows the effect of α on the distribution of rates. If $\alpha < 1$, the distribution implies that there is a large amount of rate variation, that is, many sites evolve slowly but some sites may have high rates. The larger α, the smaller rate variation, since most rates fall close to the mean. We can adapt Equation 1.8 to compute the likelihood of the tree integrating over the rate distribution.

$$L(i) = Pr(D_i \mid T, \theta, \phi, \alpha) = \int_0^\infty g(r;\alpha) \sum_{x=a}^{t} \pi_x L_x^{root}(i,r) dr \tag{1.11}$$

However, for computational reasons, a model with equally probable c discrete rate categories is used instead [92]. The likelihood given each rate is computed using the corresponding P matrix, $P^r(b)$, where the branch length has been properly scaled by the discrete rate (Equation 1.13).

$$L_x^{(p)}(i) = \left(\sum_{y=a}^{t} P^r_{x\to y}(b_{qp}) L_x^{(q)}(i) \right) \left(\sum_{x=a}^{t} P^r_{x\to y}(b_{rp}) L_x^{(r)}(i) \right) \tag{1.12}$$

$$P^r(b) = e^{Qrb} \tag{1.13}$$

Equation 1.11 is transformed into Equation 1.14, where the likelihood is the mean among all C discrete rate categories.

$$L(i) = Pr(D_i \mid T, \theta, \phi, \alpha, C) = \frac{1}{C} \sum_{c=1}^{C} \sum_{x=a}^{t} \pi_x L_x^{root}(i, r_c) \tag{1.14}$$

The different models can also assume that a proportion of sites are invariant while others evolve at the same single rate. The per-site likelihood is the sum of the conditional per-site likelihoods assuming the site to be variant, L_V, or invariant, L_I For site i and the virtual root located between nodes p and q, the likelihood considering a proportion of invariant sites is computed as follows:

$$L(i) = L_V(i)(1-p) + L_I(i)(p) \tag{1.15}$$

$$L_V(i) = \sum_{x=a}^{t} \pi_x L_x^{(p)}(i) \sum_{y=a}^{t} P_{x\to y}((1-p)b_{pq}) L_y^{(q)}(i) \tag{1.16}$$

$$L_I(i) = \begin{cases} \pi_{x_i} & \text{if site } i \text{ is invariant} \\ 0 & \text{otherwise} \end{cases} \tag{1.17}$$

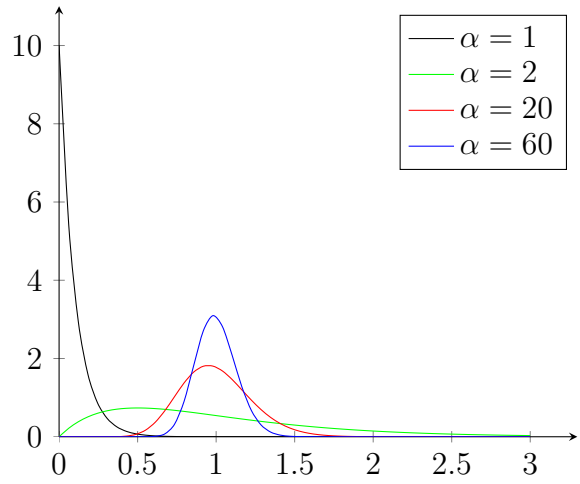

Figure 1.4: Probability density function of the Γ distribution

1.1.4. Selecting the Best-Fit Model of Evolution For Single-Gene Alignments

As it is well established, the use of one model of evolution or another may change the results of the phylogenetic analysis [18, 48, 82]. Especially, estimates of branch length or nodal support can be severely affected [13, 93]. In general, phylogenetic methods may be less accurate (recover an incorrect tree more often) or may be inconsistent (converge to an incorrect tree with increased amounts of data) when the wrong model of evolution is assumed [12, 23, 38]. Because the performance of a method is maximized when it s assumptions are satisfied, some indication of the fit of the data to the phylogenetic model is necessary [37]. Indeed, model selection is not important just because of its consequences in phylogenetic analysis, but because the characterization of the evolutionary process at the sequence level is itself a legitimate pursuit. Moreover, models of evolution are especially critical for estimating substitution parameters or for hypothesis testing [4, 85, 94, 97].

For comparing the fitness of the models, we can use statistical frameworks, such as the likelihood-ratio test (LRT) [31]. However, this method is only appropriate for nested models. Otherwise, there exist other approaches, such as the Akaike information criterion (AIC) [6], the corrected Akaike information criterion

(AICc) [39, 80], or the Bayesian information criterion (BIC) [73]. All these methods can also be used to test for absolute goodness of fit of the model to the data set. Other methods for model selection directed towards phylogenetic analysis are the parametric bootstrapping [31], and the decision theoretic framework [60].

Selecting the best-fit model of evolution requires the scoring of every candidate model. Typically, this requires the optimization and evaluation of each of the competing models such that they can be compared to each other. We cannot ensure that the optimal likelihood score is achieved for each of the models, and also the optimization methods used for optimized each of the parameters (e.g., Newton-Raphson [27], Brent [76], L-BFGS-B [57]) might get stuck in local optima. However, in general the more thorough the models are optimized, the more accurate the selection is, marginally to the selection criteria.

Among the different statistical selection procedures, the use of sequential or hierarchical LRTs often performed poorly compared to other methods [60, 65].

Likelihood Ratio Tests

In traditional statistical theory, a widely accepted statistic for testing the goodness of fit of models is the likelihood ratio test (LRT). LRT is based on the likelihood ratio, denoted by Λ (Equation 1.18).

$$\Lambda(x) = \frac{L(\theta_0|x)}{L(\theta_1|x)} = \frac{f(\cup_i x_i|\theta_0)}{f(\cup_i x_i|\theta_1)} \tag{1.18}$$

where $L(\theta_1|x)$ is the maximum likelihood under the more parameter-rich, complex model (alternative hypothesis) and $L(\theta_0|x)$ is the maximum likelihood under the less parameter-rich simple model (null hypothesis). The LRT provides the decision rule as follows:

- If $\Lambda(x) > c$, do not reject H_0
- If $\Lambda(x) < c$, reject H_0
- If $\Lambda(x) = c$, reject H_0 with probability q

The values $c, \quad q$ are usually chosen to obtain a specified significance level α, through the relation $q \cdot P(\Lambda = c \mid H_0) + P(\Lambda < c \mid H_0) = \alpha$. To preserve the nesting of the models, the likelihood scores need to be estimated upon the same tree.

Likelihood ratio tests can be carried out sequentially by adding parameters (forward selection) to a simple model (JC), or by removing parameters (backward selection) from a complex model (GTR+I+Γ) in a specific order or hierarchy (hLRT). Figure 1.5 illustrates an arbitrary hierarchy of LRTs for six different models. Within each LRT, the null model is depicted above the alternative model. When the LRT is not significant, the null model (above) is accepted (A), and it becomes the null model of the next LRT. When the LRT is significant, the null model is rejected (R) and the alternative model (below) becomes the null model of the next LRT. There are six possible paths depending on the outcome of the individual LRTs, and each path results in the selection of a different model. JC: Jukes-Cantor model [42]; K80: Kimura 1980 model [49], also known as K2P; F81: Felsenstein 81 model [25]; HKY85: Hasegawa-Kishino-Yano model [35]; SYM, symmetrical model [99]; GTR: General Time Reversible model [87], also known as REV. The performance of hierarchical LRTs for phylogenetic model selection has been discussed by Posada and Buckley (2004) [65].

Alternatively, the order in which parameters are added or removed can be selected automatically. One option to accomplish this is to add the parameter that maximizes a significant gain in likelihood during forward selection, or to add the parameter that minimizes a non-significant loss in likelihood during backward selection [67] (dLRT; see Figure 1.6). In this case, the order of the tests is not specified a priori, but it will depend on the particular data. Figure 1.6 shows an example of a dynamical Likelihood Ratio Test with 6 candidate models. Starting with the simplest (JC) or the most complex model (GTR+I+Γ), LRTs are performed among the current model and the alternative model that maximizes the difference in likelihood. Hypotheses tested are: f = base frequencies; s = substitution type; i = proportion of invariable sites; g = rate heterogeneity among sites.

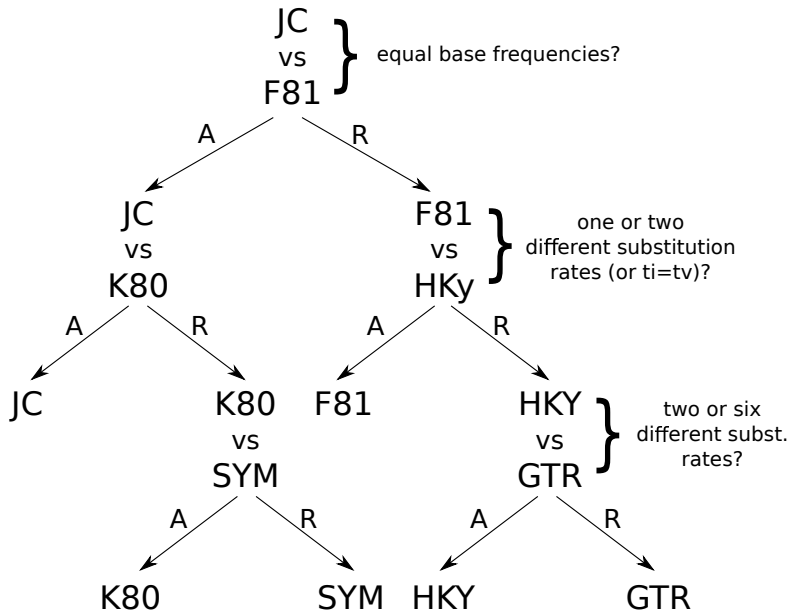

Figure 1.5: Example of a forward hierarchy of Likelihood Ratio Tests (hLRT) for 6 candidate models

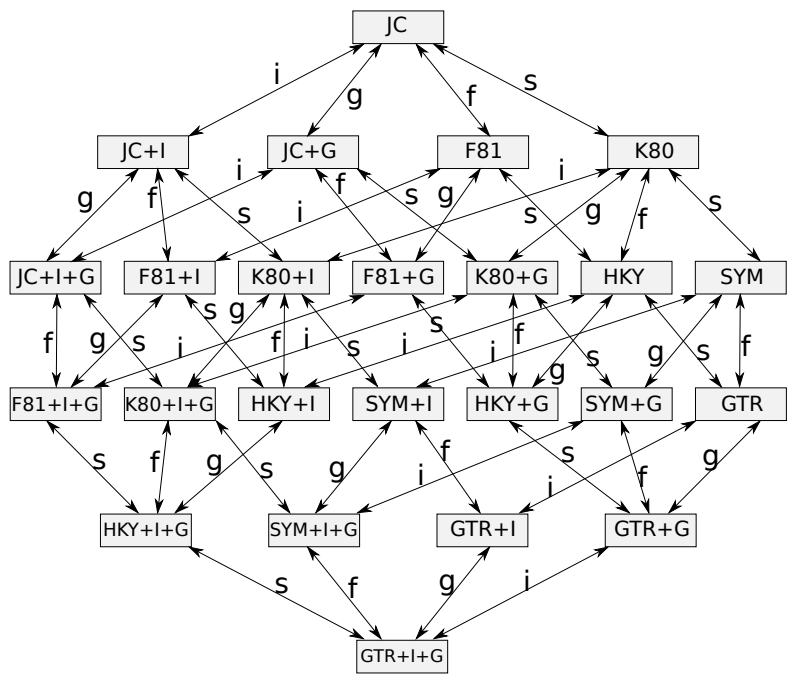

Figure 1.6: Example of dynamical Likelihood Ratio Tests (dLRT)

Information Criteria

The Akaike information criterion (AIC, [6] is an asymptotically unbiased estimator of the Kullback-Leibler information quantity [70]. We can think of the AIC as the amount of information lost when we use a specific model to approximate the real process of molecular evolution. Therefore, the model with the smallest AIC is preferred. The AIC is computed as:

$$AIC = -2l + 2k \tag{1.19}$$

where l is the maximum log-likelihood value of the data under this model and k is the number of free parameters in the model, including branch lengths estimates. When sample size (n) is large compared to the number of parameters (say, $n/k > 40$) the use of a second order AIC, AICc, is recommended:

$$AIC_c = AIC + \frac{(2k(k+1))}{(n-k-1)} \tag{1.20}$$

The AIC compares several candidate models simultaneously, it can be used to compare both nested and non-nested models, and model-selection uncertainty can be easily quantified using the AIC differences and Akaike weights (see Section 1.1.4). Burnham (2004) [16] provide an excellent introduction to the AIC and to model selection in general.

An alternative to the use of the AIC is the Bayesian Information Criterion (BIC) [73]:

$$BIC = -2l + klog(n) \tag{1.21}$$

Given equal priors for all competing models, choosing the model with the smallest BIC is equivalent to selecting the model with the maximum posterior probability. Alternatively, Bayes factors for models of molecular evolution can be calculated using reversible jump MCMC [36]. We can easily use the BIC instead of the AIC to calculate BIC differences or BIC weights (see below).

Minin (2003)[60] developed a novel approach that selects models on the basis of their phylogenetic performance, measured as the expected error on branch lengths estimates weighted by their BIC. Under this decision theoretic framework (DT) the best model is the one which minimizes the risk function:

$$C_i \approx \sum_{j=1}^{n} ||\hat{B}_i - \hat{B}_j|| \frac{e^{\frac{-BIC_j}{2}}}{\sum_{j=1}^{R}(e^{\frac{-BIC_i}{2}})} \tag{1.22}$$

$$||\hat{B}_i - \hat{B}_j||^2 = \sum_{l=1}^{2t-3} (\hat{B}_{il} - \hat{B}_{jl})^2 \tag{1.23}$$

Indeed, simulations suggested that models selected with this criterion result in slightly more accurate branch length estimates than those obtained under models selected with hierarchical LRTs [60, 3].

Model Uncertainty

The AIC, BIC, and DT methods assign a score to each model in the candidate set, therefore providing an objective function to rank them. Using the differences in scores, we can calculate a measure of model support called AIC or BIC weights [15]. For the DT scores, this calculation is not as straightforward, and right now a very gross approach is used instead, where the DT weights are the rescaled reciprocal DT scores.

For example, for the ith model, the AIC (BIC) difference is:

$$\Delta_i = AIC_i - min(AIC) \tag{1.24}$$

where $min(AIC)$ is the smallest AIC value among all candidate models. The AIC differences are easy to interpret and allow a quick comparison and ranking of candidate models. Very conveniently, we can use these differences to obtain the relative AIC (BIC) weight (w_i) of each model:

$$\omega_i = \frac{e^{\frac{-\Delta_i}{2}}}{\sum_{r=1}^{R}(e^{\frac{-\Delta_r}{2}})} \tag{1.25}$$

which can be interpreted, from a Bayesian perspective, as the probability that a model is the best approximation to the truth given the data. The weights for every model add to 1, so we can establish an approximate 95% confidence set of models for the best models by summing the weights from largest to smallest from largest to smallest until the sum is 0.95 [14, 15]. This interval can also be set up stochastically (see above "Model selection and averaging").

Alfaro and Huelsenbeck studied model uncertainty in Bayesian phylogenetic analysis [7]. The BIC posterior probability appears to be more tightly distributed around the generating model, with a smaller credible interval, higher support for the best model, and higher correspondence between the best supported model and the generating model. In their paper, models in the BIC posterior distribution also more closely match the generating model in number of parameters. In contrast, the distribution of AIC-weighted models is more diffuse with lower support for any particular model, more models in the 95% credible interval, and a higher average partition distance. Notably, in a different study the distribution of AIC-weighted models seems to be biased slightly towards models of greater complexity, whereas the BIC posterior distribution of models is unbiased, or slightly biased towards less complex models [21].

1.1.5. Selecting the Best-Fit Model of Evolution For Multigene Alignments: Partitioning

For single-gene model selection we evaluate the score of a set of models applied to a particular data set. For partitioned data, a model of evolution consists in the manner the data is divided into partitions (*partitioning scheme*), and model assigned to each of the partitions. Some parameters might be linked across partitions (e.g., branch lengths). A model of evolution for multigene alignments has as many degrees of freedom as the sum of the free parameters of the per-partition models plus those parameters shared by the different per-partition models. The

goal of selecting the best-fit model for multigene or partitioned data sets is, as in the single-gene case, to find the best trade-off between model parameterization and likelihood score, according to the same criteria described in Section 1.1.4. Therefore, the model selection involves two tasks: (i) how the data should be divided (how many partitions and how they are distributed), and (ii) which substitution model is applied to each partition.

We define "data block" as a user-defined set of sites in the alignment. A data block can be, for example a particular gene, or the set of 3rd codon positions in an alignment. A "partition" (or "subset") will be a set of one or more particular data blocks. A partition can be made of, for example, a single gene, multiple genes, or consist of the set of all first and second codon positions in an alignment. Finally, a set of non-overlapping partitions that cover the whole alignment will be called a "partitioning scheme". Not only is important to decide which models are assigned to each of the partitions, but also how many partitions the data is divided in. The partitioning problems consists of, given a predefined set of data blocks, finding the optimal partitioning scheme for a given alignment.

For phylogenomic studies, such as the 10K vertebrate genome project [40] (http:// www.genome10k.org/) and the 1,000 insect transcriptome evolution project [61] (http://www.1kite.org/), partitioning is part of the routine analyses. However, the number of different possible choices between considering one unique partition for the entire data and one partition per data block present an exponential growth according to the number of initial data blocks, and finding the best-fit partitioning scheme among them is NP-Hard.

Let X be the data, $\mathcal{D} = d_1, .., d_N$ a collection of N non-overlapped data blocks (or subsets of X), and $\mathcal{S}$ the powerset of $\mathcal{D}$. A *partitioning scheme* is a subcollection $\mathcal{S}^*$ of $\mathcal{S}$ such that each element in X is contained in exactly one subset in $\mathcal{S}^*$ (i.e., each element in X is covered by exactly one subset in $\mathcal{S}^*$). Mathematically, a partitioning scheme is an exact cover of X [44]. $\mathcal{S}_k$ is a family of $P(\mathcal{S})$ containing all k-subsets $P(\mathcal{S})^k$, such that $\bigcup_{s \in \mathcal{S}}(s) = D \wedge (s_1 \cap s_2 = \{\phi\}, \forall s_1, s_2 \in \mathcal{S})$ (the disjoint union of its elements is D). The cardinality of $\mathcal{S}_k$ is $|\mathcal{S}_k| = \left\{ {N \atop k} \right\}$.

The number of different partitioning schemes for N data blocks is given by the Nth Bell number, which is the sum from 1 to N of the Stirling Numbers of the Second Kind:

$$B(N) = \sum_{k=1}^{N} \left\{ {N \atop k} \right\} \tag{1.26a}$$

$$\left\{ {N \atop k} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^N \tag{1.26b}$$

The number of partitioning schemes grows very quickly. For example, for 20 data blocks there are 5.8×10^{12} different partitioning schemes. For 100 data blocks there are 4.75×10^{115} different possible ways to arrange the data blocks into partitions that cover the entire data without overlapping. This number is around 10^{30} times greater than the estimated number of atoms in the observable universe.

Finding the best-fit partitioning scheme and assigned models is a very computational and memory intensive task. It is also NP-Hard and therefore finding the absolute optima is not feasible even for a not so large number of single partitions or data blocks. The number of possible combinations grows asymptotically to $((0.792n)/ln(n+1))^n$ [9].

For k groups in each scheme, it is necessary to perform the selection for those groups with $N - k + 1$ data blocks. The number of combinations of k elements without repetition in a set of N elements is given by $\binom{N}{k}$, and therefore the total number of required partition evaluations will be $\sum_{i=0}^{N} \binom{i}{k}$, that equals $(2^N) - 1$ (Newton binomial theorem), giving a computational complexity of $\mathcal{O}(2^N)$. Figure 1.7 shows an example of all possible partitioning schemes for $N = 4$.

$$B(n) = \mathcal{O}\left((\frac{0.792n}{ln(n+1)})^n \right) \tag{1.27}$$

$$|\mathcal{S}(n)| = \sum_{i=0}^{n} \binom{i}{k} \tag{1.28}$$

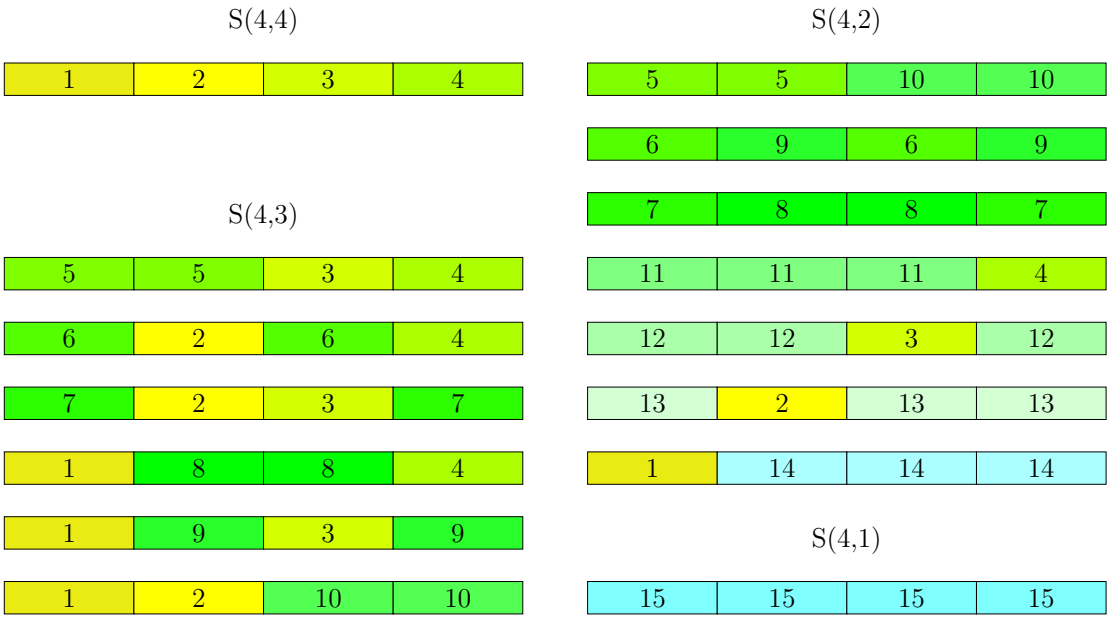

Figure 1.7: Partitioning schemes for $N = 4$. Numbers and colors represent the different partitions. In this example, there are $2^N - 1 = 15$ partitions and $B(N) = 15$ partitioning schemes.

Given these $2^N - 1$ partitions, we can think the best-fit partition search problem as the set cover problem [88] where each partition is given a weight (i.e., the BIC score). Our target is to find the k non-overlapped subsets covering the whole alignment that minimizes the score. The set cover problem is NP-Hard [51], so it is the best-fit partition selection as well.

The set of disjoint partitioning schemes that contains every different possible partition is given by those combinations obtained from the subset $\mathcal{S}_1 \cup \mathcal{S}_2$. Partitions included in every other partitioning scheme will appear in one single partitioning scheme from the subset described before. In the example in Figure 1.7, the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ contain all possible partitions, and for each $s \in \mathcal{S}_3 \cup \mathcal{S}_4$, there is exactly 1 element $x \in \mathcal{S}_1 \cup \mathcal{S}_2$ such that $x \in s$. This subset has cardinality $2^N - 1$, and as long as every element but $\mathcal{S}_1$ includes exactly 2 partitions, the total number of partitions is, as stated before, $2 \times 2^{N-1} - 1 = (2^N) - 1$. This extremely high number of combinations makes impossible to analyse every possible partitioning scheme in a reasonable time.

Going a step further, we can also link parameters across partitions, slightly reducing the total number of parameters, but also increasing the complexity and thus making the best-fit scheme selection much harder.

At the genomic scale the heterogeneity of the substitution process becomes even more apparent that at the single-gene scale, as different genes or genomic regions can have very different functions and evolve under very different constraints [8]. Multilocus substitution models that consider distinct models for different partitions of the data assumed to evolve in an homogeneous fashion have been proposed under the likelihood [69, 95] and Bayesian [63, 79] frameworks. In this case, different loci (or loci by codon position) are typically considered as distinct partitions by default, without further justification, like for example the set of 1st, 2nd or 3rd codon positions in protein-coding sequence alignments [75] or distinct protein domains [100]. However, a number of empirical studies have demonstrated that different partitioning schemes can affect multilocus phylogenetic inference, including tree topology, branch lengths and nodal support [10, 43, 55, 68, 90], with maximal differences occurring when whole datasets are treated as single partitions (i.e., unpartitioned). Using computer simulations, Brown and Lemmon (2007) showed that both over and particularly under-partitioning can lead to inaccurate phylogenetic estimates [11].

The most popular tool for automatic statistical selection of partitioning schemes for multilocus data sets is PartitionFinder [53], released in 2012. PartitionFinder is written in Python, and makes an intensive use of external software such as PhyML [33] or RAxML [77] for the phylogenetic inferences and model optimization. It uses combinatorial optimization heuristics, like hierarchical clustering and greedy algorithms building up on previous ideas raised by Li et al. [56].

1.2. Software for Model Selection

1.2.1. ProtTest 2.x

As briefly introduced in Section 1.1.1, in most cases the 20×20 replacement matrices for amino acid replacement models are not estimated *de novo* for each

data set. Instead, replacement rates previously estimated from large empirical databases are adopted. Among these, some of the more popular are the Dayhoff [22], JTT [32], mtREV [5], WAG [91], mtArt [1] or LG [54] matrices. Importantly, many phylogenetic calculations like the estimation of tree topologies, branch lengths, nodal support, divergence times or replacement rates benefit from the use of explicit models of evolution. Because the use of different models can change the outcome of the analysis [81], different model selection tools for protein alignments have been implemented in the past, like ProtTest [2], Model-Generator [46] or TOPALi [59].

ProtTest is one of the most popular tools for selecting models of protein evolution, with more than 2,000 citations. ProtTest is written in Java and uses the program PhyML [34, 33] for the ML estimates of phylogenetic trees and model parameters. The candidate set of models contain 14 different rate matrices that result in 112 different models when we consider rate variation among sites (+I: invariable sites; +G: Γ-distributed rates) and the observed amino acid frequencies (+F). ProtTest uses the selection criteria described in Section 1.1.4 to find which of the candidate models best fits the data at hand. In addition, it can perform multi-model inference and estimate parameter importances [65]. The time required to complete the likelihood calculations, that take most of the runtime of the program, can be variable depending on the size and complexity of the alignments and models. For large alignments, this task cannot be completed in a reasonable time using a single core. While ModelGenerator/MultiPhyl [47] and TOPALi implement grid computing to speed-up the analyses, they consider fewer models and do not implement model averaging. Also grid implementations usually include a very large overhead compared to HPC-oriented architectures.

1.2.2. ModelTest and jModelTest 1.x

The most popular bioinformatic tool to select appropriate models of DNA substitution for a given DNA sequence alignment is *ModelTest* [66], with more than 17,000 citations. It was superseded in 2008 by *jModelTest* [64]. jModelTest calculates the likelihood score for each model and uses the already described model selection techniques to choose the best-fit model (dLRT, hLRT, AIC, AICc and

BIC). It is written in Java, and makes use of PhyML for phylogenetic calculations.

jModelTest supports 88 submodels of the general time-reversible model. On top of 11 different substitution schemes and stationary frequencies (Table 1.2), each of these models can assume rate variation among sites (+I: invariable sites; +G: Γ-distributed rates).

Table 1.2: Substitution models available in jModelTest.

Model	**Free Parameters**	**Base Frequencies**	**Substitution Rates**	**Substitution Code**
JC	k	Equal	AC = AG = AT = CG = CT = GT	000000
F81	k + 3	Unequal	AC = AG = AT = CG = CT = GT	000000
K80	k + 1	Equal	AC = AT = CG = GT, AG = CT	010010
HKY	k + 4	Unequal	AC = AT = CG = GT, AG = CT	010010
TrNef	k + 2	Equal	AC = AT = CG = GT, AG, CT	010020
TrN	k + 5	Unequal	AC = AT = CG = GT, AG, CT	010020
TPM1	k + 2	Equal	AC = GT, AT = CG, AG = CT	012210
TPM1uf	k + 5	Unequal	AC = GT, AT = CG, AG = CT	012210
TPM2	k + 2	Equal	AC = AT, CG = GT, AG = CT	010212
TPM2uf	k + 5	Unequal	AC = AT, CG = GT, AG = CT	010212
TPM3	k + 2	Equal	AC = CG, AT = GT, AG = CT	012012
TPM3uf	k + 5	Unequal	AC = CG, AT = GT, AG = CT	012012
TIM1ef	k + 3	Equal	AC = GT, AT = CG, AG, CT	012230
TIM1	k + 6	Unequal	AC = GT, AT = CG, AG, CT	012230
TIM2ef	k + 3	Equal	AC = AT, CG = GT, AG, CT	010232
TIM2	k + 6	Unequal	AC = AT, CG = GT, AG, CT	010232
TIM3ef	k + 3	Equal	AC = CG, AT = GT, AG, CT	012032
TIM3	k + 6	Unequal	AC = CG, AT = GT, AG, CT	012032
TVMef	k + 4	Equal	AC, AT, CG, GT, AG = CT	012314
TVM	k + 7	Unequal	AC, AT, CG, GT, AG = CT	012314
SYM	k + 5	Equal	AC, AG, AT, CG, CT, GT	012345
GTR	k + 8	Unequal	AC, AG, AT, CG, CT, GT	012345

1.3. Scope and Motivation

As mentioned before, the use of different models of substitution can change the results of the phylogenetic analysis. Tools for automated selecting the best-fit model of evolution under a Maximum-Likelihood framework exist from the late 90s and they are continuously improving [2, 45, 66].

However, the irruption of Next-Generation Sequencing technologies (NGS) has caused a data avalanche in recent years. The growing rate at which molecular data is being available for researchers encourages the need of developing faster analysis methods. Moreover, it is common in most research groups in the field to have access to local or remote HPC architectures. Thus, HPC and fault tolerance techniques are of great importance.

On top of that, also due to the data explosion, the study of genomic data has become more popular in the last years. Different regions of the genome can evolve in a very different way, and therefore the old approach of analysing the whole data with a single model became obsolete. Selecting the best-fit partitioning scheme (i.e., the best scoring one) is a very computational expensive and memory intensive task, and finding the absolute optima is not feasible for a not so large number of single partitions or genes.

1.4. Main Objectives of the Thesis

The main goal of this thesis is the improvement and evaluation of the existing techniques for selecting the best-fit model of nucleotide substitution and amino acid replacement for single and multigene sequence alignments.

For single-gene sequence alignments, this thesis has three targets: (i) design of new search algorithms for increasing the search space, (ii) optimize the algorithms for multicore desktop computers and HPC architectures, and (iii) the evaluation of the different selection criteria in terms of accuracy finding the true model parameters. This includes the design, development and evaluation of HPC algorithms for model selection and results analysis, incorporated into tools that are nowadays

reference for model selection: *jModelTest* and *ProtTest*. Evaluating the performance of different selection criteria involves testing the accuracy recovering the model parameters out of a set of simulated data, where the true generating model is known.

For multigene sequence alignments the goal is divided in two targets: (i) implementation and evaluation of HPC search algorithms for the best-fit partitioning scheme, and (ii) study the importance of model selection in order to infer better quality phylogenies rather than using trivial approaches. Testing the importance of model selection is based in simulated alignments covering a wide range of input parameter values (e.g, number of genes, alignment size, nucleotide diversity).

Chapter 2

ProtTest 3: Fast Selection of Best-fit Models of Protein Evolution

The content of this chapter corresponds to the following journal paper:

- Title: ProtTest 3: Fast selection of best-fit models of protein evolution
- Authors: **Diego Darriba**, Guillermo L. Taboada, Ramón Doallo and David Posada
- Journal: Bioinformatics
- Editorial: Oxford University Press, ISSN: 1367-4803, EISSN: 1460-2059
- Year: 2011
- Volume(Number):Pages: 27(8):1164–1165
- DOI: 10.1093/bioinformatics/btr088
- Impact factor: 4.981
- Q1; 10th position on Computer Science Applications, 35th position on Molecular Biology

- Number of citations: 547 as of October 2015

The final publication is available at

http://bioinformatics.oxfordjournals.org/content/27/8/1164.full.

A copy of the accepted paper is next included.

Bioinformatics Advance Access published February 17, 2011

ProtTest 3: fast selection of best-fit models of protein evolution

Diego Darriba [1,2], Guillermo L. Taboada [2], Ramón Doallo [2], David Posada [1*]

[1]Department of Biochemistry, Genetics and Immunology, University of Vigo, 36310 Vigo, Spain
[2]Computer Architecture Group, University of A Coruña, 15071 A Coruña, Spain

Associate Editor: Prof. Martin Bishop

ABSTRACT

Summary: We have implemented a High Performance Computing (HPC) version of ProtTest (Abascal *et al.*, 2007) that can be executed in parallel in multi-core desktops and clusters. This version, called ProtTest 3, includes new features and extended capabilities.

Availability: ProtTest 3 source code and binaries are freely available under GNU license for download from `http://darwin.uvigo.es/software/prottest3`, linked to a Mercurial repository at Bitbucket (https://bitbucket.org/).

Contact: dposada@uvigo.es

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

Recent advances in modern sequencing technologies have resulted in an increasing capability for gathering large data sets. Long sequence alignments with hundred or thousands of sequences are not rare these days, but their analysis imply access to large computing infrastructures and/or the use of simpler and faster methods. In this regard, High Performance Computing (HPC) becomes essential for the feasibility of more sophisticated –and often more accurate– analyses. Indeed, during the last years HPC facilities have become part of the general services provided by many universities and research centers. Besides, multicore desktops are now standard.

The program ProtTest (Abascal *et al.*, 2007) is one of the most popular tools for selecting models of amino acid replacement, a routinary step in phylogenetic analysis. ProtTest is written in Java and uses the program PhyML (Guindon and Gascuel, 2003) for the maximum likelihood (ML) estimation of model parameters and phylogenetic trees and the PAL library (Drummond and Strimmer, 2001) to handle alignments and trees. Statistical model selection can be a very intensive task when the alignments are large and include divergent sequences, highlighting the need for new bionformatic tools capable of exploiting the available computational resources.

Here we describe a new version of ProtTest, ProtTest3, that has been completely redesigned to take advantage of HPC environments and desktop multicore processors, significantly reducing the execution time for model selection in large protein alignments.

2 PROTTEST 3

The general structure and the Java code of ProtTest has been completely redesigned from a computer engineering point of view. We implemented several parallel strategies as distinct execution modes in order to make an efficient use of the different computer architectures that a user might encounter:

(1) a Java thread-based concurrence for shared memory architectures (e.g., a multi-core desktop computer or a multi-core cluster node). This version also includes a new and richer Graphical User Interface (GUI) to facilitate its use.

(2) an MPJ (Shafi *et al.*, 2009) parallelism for distributed memory architectures (e.g., HPC clusters).

(3) a hybrid implementation MPJ - OpenMP (Dagum and Menon, 1998) to obtain maximum scalability in architectures with both shared and distributed memory (e.g., multicore HPC clusters).

Moreover, ProtTest 3 includes a number of new and more comprehensive features that significantly extend the capabilities of the previous version: (1) more flexible support for different input alignment formats through the use of the ALTER library (Glez-Peña *et al.*, 2010): ALN, FASTA, GDE, MSF, NEXUS, PHYLIP and PIR; (2) up to 120 candidate models of protein evolution; (3) four strategies for the calculation of likelihood scores: fixed BIONJ, BIONJ, ML or user-defined; (4) four information criteria: AIC, BIC, AICc and DT (see Sullivan and Joyce 2005); (5) reconstruction of model-averaged phylogenetic trees (Posada and Buckley, 2004); (6) fault tolerance with checkpointing; and (7) automatic logging of the user activity.

3 PERFORMANCE EVALUATION

In order to benchmark the performance of ProtTest 3, we computed the running times for the estimation of the likelihood scores of all 120 candidate models from several real and simulated protein alignments (Table 1). When these data were executed in a system with shared memory, e.g., a multicore desktop, the scalability was almost linear as far as there was enough memory to satisfy the requirements. For example, in a shared memory execution in a 24-core node the speedup was almost linear with up to 8 cores, also scaling well with data sets with medium complexity, like HIVML or COXML (Fig. 1). In a system with distributed memory like an cluster, the application scaled well up to 56 processors (Fig. 2). With more processors a theoretical scalability limit exists due to the hetereogeneous nature of the optimization times, from a few seconds for the simplest models to up to several hours for the models that include rate variation among sites (+G). This problem was solved with the hybrid memory approach. In this case, the scalability went beyond the previous limit, reaching up to 150 in the most complex cases with 8-core nodes (Figure 3).

*to whom correspondence should be addressed

© The Author (2011). Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oup.com

Downloaded from http://bioinformatics.oxfordjournals.org/ at 62324608-Swets. Subs. Service on May 15, 2014

Data set Abbreviation	Protein	Size NxL	Base tree	Seq. exec. time
RIB	Ribosomal protein	21x113	Fixed BIONJ	5.5m
RIBML	”	”	ML tree	28m
COX	Cytochrome C oxidase II	28x113	Fixed BIONJ	9.5m
COXML	”	”	ML tree	55m
HIV	HIV polimerase	36x1,034	Fixed BIONJ	44m
HIVML	”	”	ML tree	160m
10K	Simulated aln	50x10K	Fixed BIONJ	9.2h
20K	”	50x20K	”	24.5h
100K	”	50x100K	”	80h

Table 1. Real and simulated alignments used to benchmark ProtTest 3 performance. In column *Size*, *N* indicates the number of sequences and *L* the length of the alignment. *Base tree* is the tree used for model likelihood optimization and *Seq. exec. time* is the time required to calculate the likelihood scores using the sequential version (i.e., a single thread).

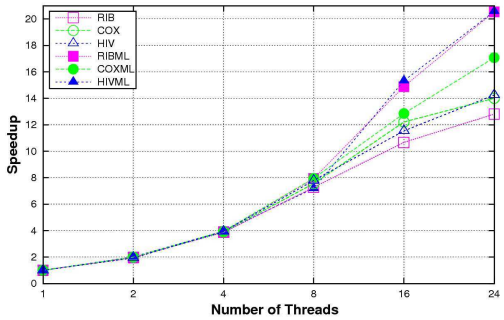

Figure 1. Speed-ups obtained with the shared memory version of ProtTest 3 according to the numbers of threads used in a 24-core shared memory node (4 hexa-core Intel Xeon E7450 processors) with 12GB memory.

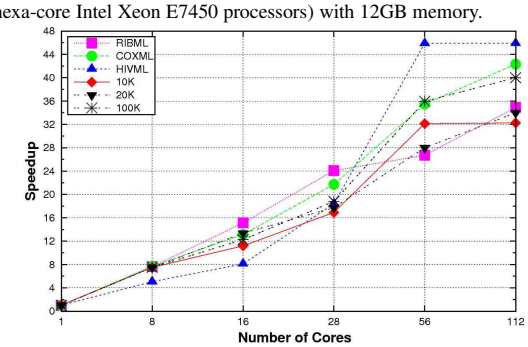

Figure 2. Speed-ups obtained with the distributed memory version of ProtTest 3 according to the numbers of cores used in a 32-node cluster with 2 quad-core Intel Harpertown processors and 8GB memory per node. Up to 4 processes where executed per node because of the memory requirements of the largest datasets (10K, 20K, 100K).

4 CONCLUSIONS

ProtTest 3 can be executed in parallel in HPC environments as: (1) a GUI-based desktop version that uses multi-core processors; (2) a cluster-based version that distributes the computational load among nodes; and (3) as a hybrid multi-core cluster version that achieves speed through the distribution of tasks among nodes while taking advantage of multi-core processors within nodes. The new version has been completely redesigned and includes new capabilities like checkpointing, additional amino acid replacement matrices, new model selection criteria and the possibility of computing model-averaged phylogenetic trees. The use of ProtTest 3 results

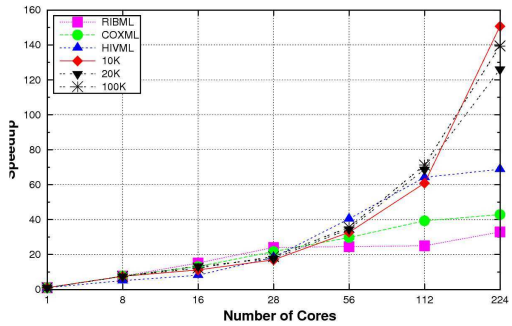

Figure 3. Speed-ups obtained with the hybrid memory version of ProtTest 3 according to the numbers of cores used in a 32-node cluster with 2 quad-core Intel Harpertown processors and 8GB memory per node. Up to 4 MPJ Express processes per node and at least 2 OpenMP threads for each ML optimization were executed.

in significant performance gains, with observed speedups of up to 150 on a high performance cluster. For very large alignments this can be equivalent to a reduction of the running time from more than three days to around half an hour. In this way, statistical model selection for large protein alignments becomes feasible, not only for cluster users, but also for the owners of standard multi-core desktop computers. Moreover, the flexible design of ProtTest-HPC will allow developers to extend future functionalities, whereas third-party projects will be able to easily adapt its capabilities to their requirements.

ACKNOWLEDGEMENTS

Special thanks to Stephane Guindon for his continuous help with PhyML and to Federico Abascal for his help with the previous version of ProtTest. This work was financially supported by the European Research Council [ERC-2007-Stg 203161-PHYGENOM to D.P.], the Spanish Ministry of Science and Education [BFU2009-08611 to D.P.] and by the Xunta de Galicia [Galician Thematic Networks RGB 2010/90 to D.P. and GHPC2 2010/53 to R.D.].

REFERENCES

Abascal, F., Zardoya, R., and Posada, D. (2007). ProtTest: Selection of best-fit models of protein evolution. *Bioinformatics*, **24**(1), 1104–1105.

Dagum, L. and Menon, R. (1998). OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science and Engineering*, **5**(1), 46–55.

Drummond, A. and Strimmer, K. (2001). Pal: an object-oriented programming library for molecular evolution and phylogenetics. *Bioinformatics*, **17**(7), 662–663.

Glez-Peña, D., Gómez-Blanco, D., Reboiro-Jato, M., Fdez-Riverola, F., and Posada, D. (2010). ALTER: program-oriented conversion of DNA and protein alignments. *Nucleic Acids Research*, (38), W14–W18.

Guindon, S. and Gascuel, O. (2003). A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol*, **52**(5), 696–704.

Posada, D. and Buckley, T. R. (2004). Model selection and model averaging in phylogenetics: advantages of akaike information criterion and bayesian approaches over likelihood ratio tests. *Syst Biol*, **53**(5), 793–808.

Shafi, A., Carpenter, B., and Baker, M. (2009). Nested parallelism for multi-core HPC systems using Java. *J Parallel Distr Com*, **69**(6), 532–545.

Sullivan, J. and Joyce, P. (2005). Model selection in phylogenetics. *Annu Rev Ecol Evol S*, **36**, 445–466.

Downloaded from http://bioinformatics.oxfordjournals.org/ at 62324608-Swets. Subs. Service on May 15, 2014

Supplementary material for "ProtTest 3: fast selection of best-fit models of protein evolution"

Diego Darriba, Guillermo L. Taboada, Ramón Doallo, David Posada

February 9, 2011

Abstract

Summary:

This appendix deals with some technical issues that were not specified in the application note for ProtTest 3: how the hybrid version of ProtTest 3 works and how the workload balancing is performed.

Availability:

ProtTest 3 source code and binaries are freely available under GNU license for download from `http://darwin.uvigo.es/software/prottest3`, linked to a Mercurial repository at Bitbucket (https://bitbucket.org/).

1 Hybrid Computation

The scalability of ProtTest 3 using either shared or distributed memory is limited by the replacement models with the highest computational load, usually the "+I+G" models, which could take up to 90% of the overall runtime. In these cases, the runtime was determined by the longest optimization, resulting in poor speedups. Moreover, the higher the number of cores, the higher the workload imbalance due to runtime differences. In fact, ProtTest 3 usually could take advantage of up to 50 cores, approximately. This important limitation prompted us to develop a hybrid (shared/distributed memory) approach, in which we reduced the overhead of the model optimizations using a thread-based executor within the distributed memory implementation.

We parallelized the basic task –ML optimization– to get rid of the limitation of using a single core per model. Thus, we modified PhyML (Guindon and Gascuel, 2003) to produce a thread-based version, using OpenMP (Dagum and Menon, 1998). In this way, the models with the highest computational load could run in parallel, significantly reducing the total runtime. However, this strategy is only possible when memory is shared, like in a cluster node, being limited by the number of available cores per system. Our solution (see Fig. 1)

was to implement a message-passing based distribution of the tasks across a distributed memory machine, using MPJ Express (Shafi *et al.*, 2009). This way, it is possible to take advantage of multi-core clusters through the execution of a thread-based model optimization process together with the message-passing implementation of ProtTest-HPC. This two-level parallelism resulted in a much more efficient exploitation of the available computational resources.

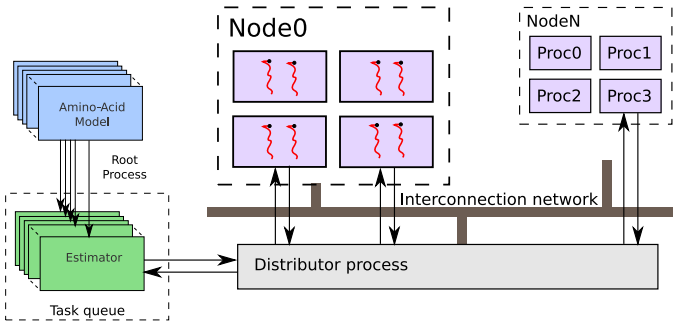

Figure 1: ProtTest 3 hybrid strategy, where two threads are run per external ML optimizer (PhyML) process.

2 Scheduling and Load Balancing

2.1 High Level Scheduling

The load balancing of the model selection task is not trivial, due to the uncertain and highly variable execution times of the model optimizations. Even the use of a dynamic task queue could not be the optimal solution in many cases.

ProtTest 3 attempts to balance the workload in two steps. At first, the workload of each single model optimization is estimated. Since the distribution is performed at a task level (i.e, the 120 models to optimize are distributed among processors), the accuracy of this estimate is essential in order to achieve the optimal performance. ProtTest 3 includes an extensible hierarchy of heuristics to perform this task. For example, the default heuristic uses the model parameters to estimate its relative weight compared to every other model.

The next step is to distribute the set of models among the computational resources, once they are sorted by their computational workload (the relative workload estimation is a good metric for this task). The best strategy is to use a dynamic scheduling of the set of tasks. ProtTest 3 implements this using a Java thread pool in shared memory architectures, and a similar approach in distributed memory. In this case, a distributor thread works as the model distributor and dynamic scheduler. The process starts sending a single model for optimization to each process, and every time a process returns the optimized model to the scheduler, it sends the next model from the sorted queue.

2.2 Hybrid Scheduling

In the hybrid-memory version of ProtTest, the number of shared memory threads is also taken into account to make the distribution among nodes. At execution time, the number of available processors cores per ProtTest 3 process is given. This implies a dependency between the logical distribution of tasks and the physical mapping of the processes (i.e., thread affinity). However, it results into the best parallel efficiency of ProtTest 3.

The scheduler calculates the number of threads that each model optimization should use to get the best overall performance, taking the number and complexity of candidate models and the number of threads per process as parameters.

Figure 2 shows the parallel performance of our OpenMP parallel version of PhyML. PhyML gets an almost linear speedup using up to 4 threads, slightly depending on the input data for a higher number of threads. With this information, ProtTest 3 can aims for the best combination between process-level and thread-level parallelism to attain the best possible performance.

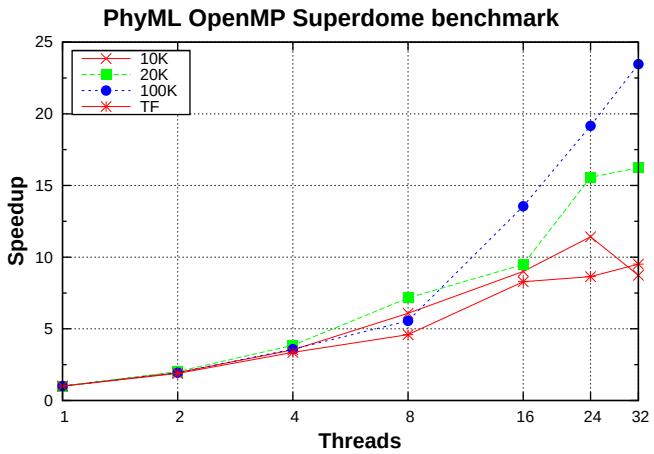

Figure 2: PhyML parallel performance on an HP Superdome system

3 Fault Tolerance

ProtTest 3 can require long execution times and involve a significant number of computing resources. For this reason we also implemented fault tolerance support. Every intermediate running status of the application is held in a serializable Java object, subject to its storage (checkpoint) in a snapshot file by the centralized checkpoint manager (*CPManager*) once it has been notified (through a custom *Observer* pattern) that a task has been completed and the ProtTest 3 status has been validated (Fig. 3). *CPManager* is also in charge of restoring

ProtTest 3 up to the last consistent saved status after a failed execution. This checkpointing system works at a high level, so the application status is verified every time a model optimization is complete.

The overhead of the fault tolerance support is almost negligible compared to the global run times ($< 0.1\%$), so it is enabled by default. Furthermore, it is fully transparent to the user. Every time the application starts, *CPManager* automatically looks up for consistent snapshot files in the snapshot directory and relaunches any previously failed execution.

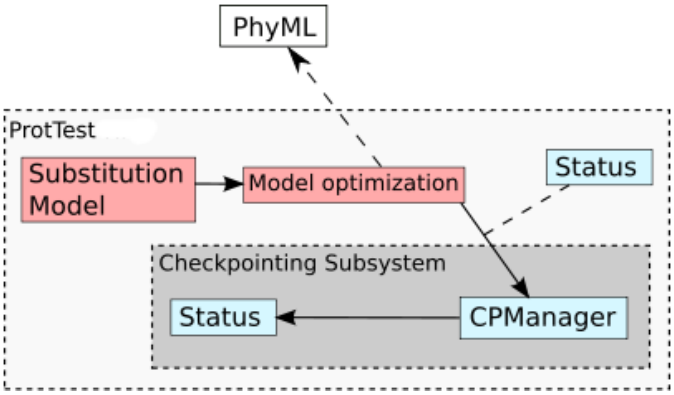

Figure 3: ProtTest 3 fault tolerance subsystem.

References

Dagum, L. and Menon, R. (1998). OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science and Engineering*, **5**(1), 46–55.

Guindon, S. and Gascuel, O. (2003). A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol*, **52**(5), 696–704.

Shafi, A., Carpenter, B., and Baker, M. (2009). Nested parallelism for multi-core HPC systems using Java. *J Parallel Distr Com*, **69**(6), 532–545.

Chapter 3

High-Performance Computing Selection of Models of DNA Substitution for Multicore Clusters

The content of this chapter corresponds to the following journal paper:

- Title: High-performance computing selection of models of DNA substitution for multicore clusters
- Authors: **Diego Darriba**, Guillermo L. Taboada, Ramón Doallo and David Posada
- Journal: International Journal of High Performance Computing Applications
- Editorial: Sage Publications, ISSN: 1094-3420, EISSN: 1741-2846
- Year: 2013
- Volume(Number):Pages: 28(1):112–125
- DOI: 10.1177/1094342013495095

- Impact factor: 1.477
- Q2
- Number of citations: 4 as of October 2015

The final publication is available at

http://hpc.sagepub.com/content/28/1/112

A copy of the accepted paper is next included.

Article

International Journal of HIGH PERFORMANCE COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
2014, Vol. 28(1) 112–125
© The Author(s) 2013
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342013495095
hpc.sagepub.com
SAGE

High-performance computing selection of models of DNA substitution for multicore clusters

Diego Darriba[1,2], Guillermo L Taboada[1], Ramón Doallo[1] and David Posada[2]

Abstract

This paper presents the high-performance computing (HPC) support of jModelTest2, the most popular bioinformatic tool for the statistical selection of models of DNA substitution. As this can demand vast computational resources, especially in terms of processing power, jModelTest2 implements three parallel algorithms for model selection: (1) a multithreaded implementation for shared memory architectures; (2) a message-passing implementation for distributed memory architectures, such as clusters; and (3) a hybrid shared/distributed memory implementation for clusters of multicore nodes, combining the workload distribution across cluster nodes with a multithreaded model optimization within each node. The main limitation of the shared and distributed versions is the workload imbalance that generally appears when using more than 32 cores, a direct consequence of the heterogeneity in the computational cost of the evaluated models. The hybrid shared/distributed memory version overcomes this issue reducing the workload imbalance through a thread-based decomposition of the most costly model optimization tasks. The performance evaluation of this HPC application on a 40-core shared memory system and on a 528-core cluster has shown high scalability, with speedups of the multithreaded version of up to 32, and up to 257 for the hybrid shared/distributed memory implementation. This can represent a reduction in the execution time of some analyses from 4 days down to barely 20 minutes. The implementation of the three parallel execution strategies of jModelTest2 presented in this paper are available under a GPL license at http://code.google.com/jmodeltest2.

Keywords

High-performance computing (HPC), multicore cluster, Message-Passing in Java (MPJ), phylogeny, nucleotide substitution, performance evaluation

1. Introduction

In recent years, DNA sequence data has been accumulated in databases (e.g. GenBank) at an exponential rate. These DNA sequences can be used for example to study the history of the different species that inhabit our planet, for example estimating phylogenetic trees from multiple sequence alignments. All phylogenetic methods make assumptions, whether explicit or implicit, about the process of DNA substitution (Felsenstein, 1988). It is well known that the use of one or another probabilistic model of nucleotide substitution can change the outcome of the analysis (Buckley, 2002; Buckley and Cunningham, 2002; Lemmon and Moriarty, 2004), and model selection has become a routine step for the estimation of molecular phylogenies.

The most popular bioinformatic tool to select appropriate models of DNA substitution for a given DNA sequence alignment is jModelTest (Posada, 2008). This program calculates the likelihood score for each model and uses different model selection techniques to choose the "best" one according to the likelihood and number of parameters. The model selection strategies implemented in jModelTest are the Akaike information criterion (AIC) (Akaike, 1974), Bayesian information criterion (BIC) (Schwarz, 1978) and dynamic likelihood ratio tests (dLRTs) (Posada and Crandall, 2001).

[1] Computer Architecture Group, University of A Coruña, A Coruña, Spain
[2] Bioinformatics and Molecular Evolution Group, University of Vigo, Vigo, Spain

Corresponding author:
Diego Darriba, Computer Architecture Group, University of A Coruña, 15071 A Coruña, Spain.
Email: ddarriba@udc.es

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

Table 1. Substitution models available in jModelTest. Any of these can include a proportion of invariable sites (+*I*), rate variation among sites (+*G*), or both (+*I* + *G*).

Model	Free parameters	Base frequencies	Substitution rates	Substitution code
JC	k	Equal	AC = AG = AT = CG = CT = GT	000000
F81	k + 3	Unequal	AC = AG = AT = CG = CT = GT	000000
K80	k + 1	Equal	AC = AT = CG = GT, AG = CT	010010
HKY	k + 4	Unequal	AC = AT = CG = GT, AG = CT	010010
TrNef	k + 2	Equal	AC = AT = CG = GT, AG, CT	010020
TrN	k + 5	Unequal	AC = AT = CG = GT, AG, CT	010020
TPM1	k + 2	Equal	AC = GT, AT = CG, AG = CT	012210
TPM1uf	k + 5	Unequal	AC = GT, AT = CG, AG = CT	012210
TPM2	k + 2	Equal	AC = AT, CG = GT, AG = CT	010212
TPM2uf	k + 5	Unequal	AC = AT, CG = GT, AG = CT	010212
TPM3	k + 2	Equal	AC = CG, AT = GT, AG = CT	012012
TPM3uf	k + 5	Unequal	AC = CG, AT = GT, AG = CT	012012
TIM1ef	k + 3	Equal	AC = GT, AT = CG, AG, CT	012230
TIM1	k + 6	Unequal	AC = GT, AT = CG, AG, CT	012230
TIM2ef	k + 3	Equal	AC = AT, CG = GT, AG, CT	010232
TIM2	k + 6	Unequal	AC = AT, CG = GT, AG, CT	010232
TIM3ef	k + 3	Equal	AC = CG, AT = GT, AG, CT	012032
TIM3	k + 6	Unequal	AC = CG, AT = GT, AG, CT	012032
TVMef	k + 4	Equal	AC, AT, CG, GT, AG = CT	012314
TVM	k + 7	Unequal	AC, AT, CG, GT, AG = CT	012314
SYM	k + 5	Equal	AC, AG, AT, CG, CT, GT	012345
GTR	k + 8	Unequal	AC, AG, AT, CG, CT, GT	012345

jModelTest supports 88 submodels of the general time-reversible model (Table 1). In top of different substitution schemes and ACGT frequencies, each of these models can assume that some sites do not change between sequences (i.e. are invariant; "+I" parameter), or they do it at different rates (approximated with a discrete gamma distribution "+G"). The estimation of the α shape parameter of the gamma distribution can be complicated, and models that include this parameter ("+G" models) carry an extra computational burden.

We define the basic execution task as the optimization of a single model. jModelTest makes an extensive use of third-party bioinformatics libraries and software, aggregating multiple tasks in a pipeline and providing a high-level view of the analysis. Figure 1 shows the workflow of jModelTest, where the most time-consuming part of the process is the calculation of the likelihood scores (carried out by the PhyML program (Guindon and Gascuel, 2003)). Because this calculation represents more than 99% of the execution time in most cases, our parallel adaptation is focused in this part of the model selection process. The parallel strategies here exposed are implemented in a new version of jModelTest, available at http://code.google.com/jmodeltest2. A preliminary version of the parallelization of jModelTest, including only the shared and distributed memory versions, has been presented by Darriba et al. (2011a). This paper extends the previous work by implementing a hybrid shared/distributed memory version which overcomes the limitations of the previous work, namely the poor scalability and the workload imbalance, achieving 8 times higher performance (from speedups around 30 to speedups around 230).

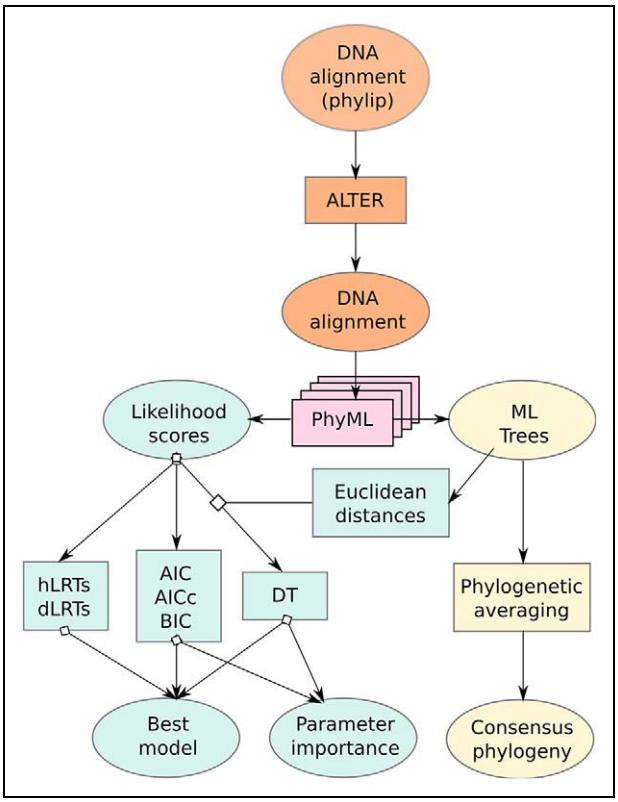

Figure 1. jModelTest algorithm workflow.

2. Parallel algorithm for model selection

Most of the execution time of the model selection analysis is spent optimizing each substitution model from the candidate model set, maximizing the likelihood function (the

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

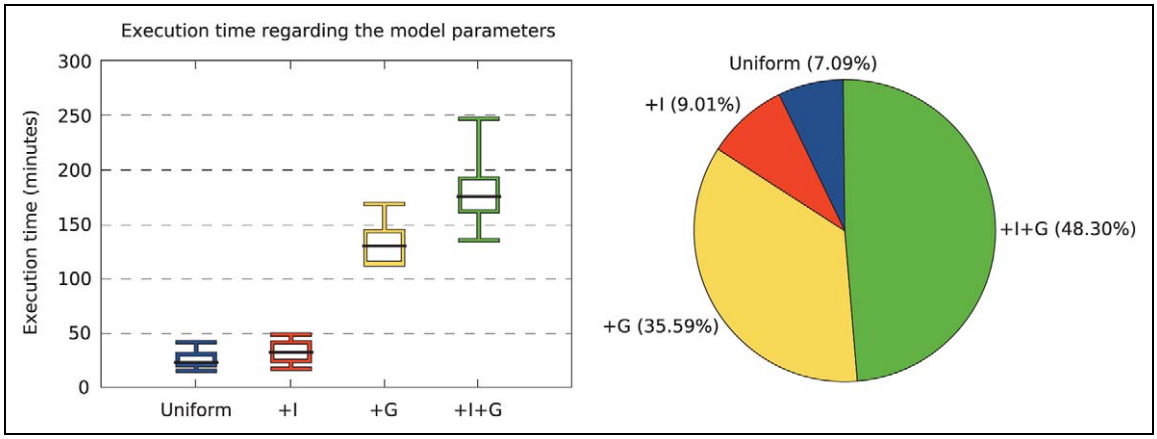

Figure 2. Computational load and execution times of the 88 model optimizations. There are 22 models of each rate variation parameter. The pie graph represents the proportion of the execution times of the models including invariant sites (+I), rate variation among sites (+G) or both (+I+G).

likelihood is the probability of the data, a multiple sequence alignment, given the model), which depends on the size and complexity of the data. For large-scale alignments this cannot be completed in a reasonable time with just a single core. Thus, we implemented a high-performance computing (HPC) version of jModelTest that supports its parallel execution on shared memory systems such as current multicore desktop processors and HPC clusters, distributing the workload among nodes and also taking advantage of multicore processors within nodes.

Maximum likelihood model optimization was proved NP-complete (Chor and Tuller, 2006). Thus, it is really difficult to estimate the runtime of each single task. However, we can estimate the relative workload depending on the model parameters. Figure 2 shows the high variance between task runtimes regarding invariant sites (+I) and discrete rate categories (+G) parameters. This variance slightly depends on the input data characteristics (e.g.number of taxa, sequences length or divergence between sequences). A representative real dataset (91 taxa and 33,148 base pairs). In order to homogeneously distribute the workload, it is better to run the most complex (i.e. +I+G and +G models) tasks at first (reverse complexity estimate). The lightest tasks would take up the remaining computational resources as long as the candidate models are optimized.

This paper presents three parallel algorithms for model selection using asynchronous communication and dynamic load-balancing: (1) a threaded shared-memory approach using Java built-in thread pool; (2) a distributed memory approach using a Message-Passing in Java (MPJ) (Shafi et al., 2009); and (3) a hybrid shared-distributed memory approach using message-passing for inter-node synchronization, a custom thread pool for intra-node synchronization, and OpenMP (Dagum and Menon, 1998) for parallelizing the basic task. The first two approaches are based on the parallel execution of model optimization tasks, presenting a coarser-grained parallelism than the last one, where the model optimization is executed in parallel as well (multilevel parallelism, with message-passing combined with shared memory thread pools and OpenMP base executions).

2.1. Design overview

The original implementation was partially redesigned to grant model extensibility, traceability and encapsulation, taking advantage of the code included in the ProtTest3 API (Darriba et al., 2011b), a similar program for protein sequences already adapted for HPC environments.

Figure 3 shows the high-level design of the HPC version of jModelTest. There is not coupling between the front-end and the back-end layers, delegating communications through a façade design pattern. Some features were organized into a class hierarchy, decoupling the related classes from the controller (i.e. ModelTestService), therefore making the model easier to extend through several interfaces:

1. The execution modes use the *RunPhyml* hierarchy. A common interface hides the model optimization behavior, and internally is able to run several PhyML instances in a shared memory architecture using a thread pool (*RunPhymlThreaded*), synchronize several processes in a distributed memory architecture (*RunPhymlMPJ*) or synchronize multiple thread pools in different nodes (*RunPhymlHybrid*).
2. The model selection task can be performed applying different information criteria that in general terms behaves in the same way. For this reason a common specification (*InformationCriterion*) hides each single criterion. As before, this decouples these classes from the controller, and also brings extensibility to the architecture.
3. The view classes do not directly depend on the inner model. The use cases are implemented in the main application service. Using an observer design pattern the execution information is displayed in real time. This works this way not only for the graphical user interface (GUI), but also for the command

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

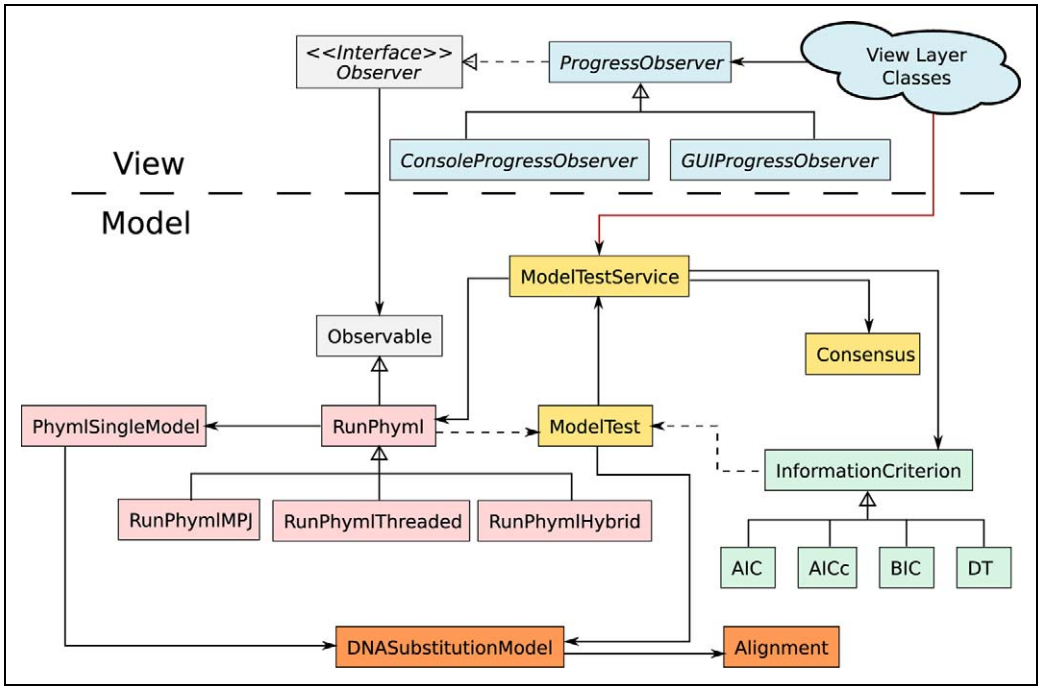

Figure 3. High level design of jModelTest2.

console executions, both threaded and distributed. In the distributed approach, only the root model is in charge of the I/O operations, unless they mean read/write data from/into scratch.

2.2. Shared memory implementation

The shared memory implementation of jModelTest relies on a thread pool to handle the execution of tasks on shared memory architectures. This approach is fully portable as it relies on thread pools from the Java Concurrence API, present in Java 1.5 and higher. Figure 4 and Algorithm 1 present the shared memory parallel operation. The task queue contains the whole set of tasks which will be processed by the thread pool in a particular order (reverse complexity estimate) (Figure 5(a)).

This multithreaded shared memory approach is especially suitable for the parallel execution of jModelTest2 on multicore desktop computers, benefiting also from the availability of a GUI. However, it is limited by the number of available cores in the system, and especially from the memory consumption, directly proportional to the number of cores being used.

2.3. Distributed memory implementation

In order to handle the computation of tasks on distributed memory architectures (e.g. clusters), this implementation manages processes, which rely on message-passing for communication, and uses a dedicated distributor thread to allocate the workload according to a dynamic distribution strategy (Figure 5(b)).

The process synchronization is explicitly achieved through message-passing blocking communication. The models are sequentially distributed and gathered among the processes through non-blocking communications. Only the root process is in charge of I/O, centralizing the displaying of runtime information and results. This central management has a negligible impact on the whole performance as long as there is no output during the model optimization task, which is by far the most time-consuming part of the model selection process. Every process works with its own copy of the input alignment from scratch, avoiding this way read or write conflicts. Figure 6 and Algorithm 2 show the operation in a distributed memory environment.

This approach saves the previous bottleneck with memory requirements. However, because each single task is executed sequentially, the workload imbalance in the model optimization tasks (see Figure 2) represents a new bottleneck. In fact, in most cases half of the candidate models requires more than 80% of the total execution time. The more computational resources are used, the more probably is that the total execution time depends on the optimization of a single model, the one which takes longer to optimize. This way, looking at Figure 2 it can be seen that the maximum runtime of a single model optimization is 248 minutes. The total runtime of the dataset is 135 hours. Thus, dividing the total runtime by the maximum single task runtime we can estimate that the highest speedup achievable is 32.83, no matter how many processes are used. Although this is a particular example, empirical tests show that the workload imbalance usually leads to a maximum speedup below 40 for these task-level parallel strategies.

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

```
Data: Execution parameters, input data, resource information
Result: Best-fit model
begin Application initialization                                   // tasks initialization
    build(model set);
    Estimate workload per model;
    Sort models in reverse complexity estimate;
    begin Initialize parallel environment
        Initialize thread pool;
        Synchronize thread pool;
    end
end
begin Tasks Computation                                            // model optimization
    build(task queue);
    foreach model optimization task do
        Wait for an idle thread;
        Assign the next task to the thread;
        Optimize model;
    end
end
```

Algorithm 1. jModelTest algorithm for shared memory parallel model optimization.

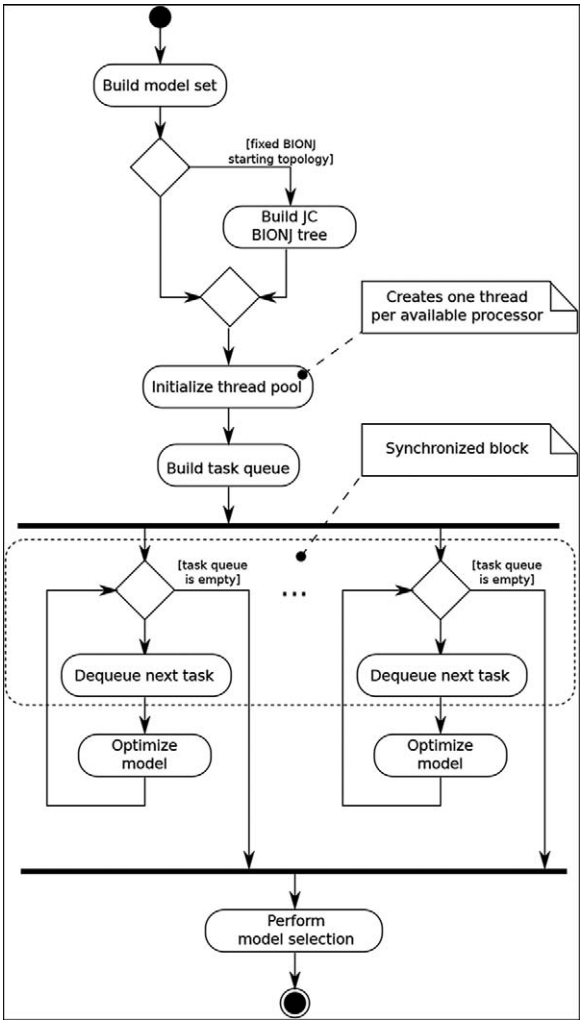

Figure 4. Activity diagram for the shared memory parallel operation algorithm.

$$Speedup_{max} = \frac{Runtime}{Runtime_{max}}$$

2.4. Parallel model optimization implementation

A detailed knowledge of the performance of the PhyML parallel implementation is key for an optimal assignment of resources (processor cores) for each model optimization task. As the +G and +I+G models used to take four times longer to optimize than the rest of the models, a proportion of four-to-one in relative speedups would balance the workload for each task. However, since the parallel efficiency of the PhyML parallel implementation is not optimal a trade-off between the expected speedup and the available computational resources has to be considered.

PhyML uses the Maximum Likelihood algorithm (ML) (Felsenstein, 1981) for finding the model parameters that maximize the likelihood function. The likelihood evaluation algorithm is the most time-consuming part of the ML process, because it is executed for each new model state proposal (i.e. after changing the parameters configuration, the tree topology or the branch lengths). This likelihood evaluation algorithm is highly parallelizable, since it is a site-independent operation performed all along the column patterns in the alignment. We have slightly changed the source code fixing unnecessary carried dependencies in order to parallelize this loop using OpenMP. Further than the source code analysis, the results of the parallel PhyML have been thoroughly validated. The sources of this parallel patch are available from the authors.

Figure 8 shows the performance of this OpenMP-based parallel PhyML version, where the scalability is notably higher for models with rate variation among sites (+G). In these models, the likelihood evaluation for each site is

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

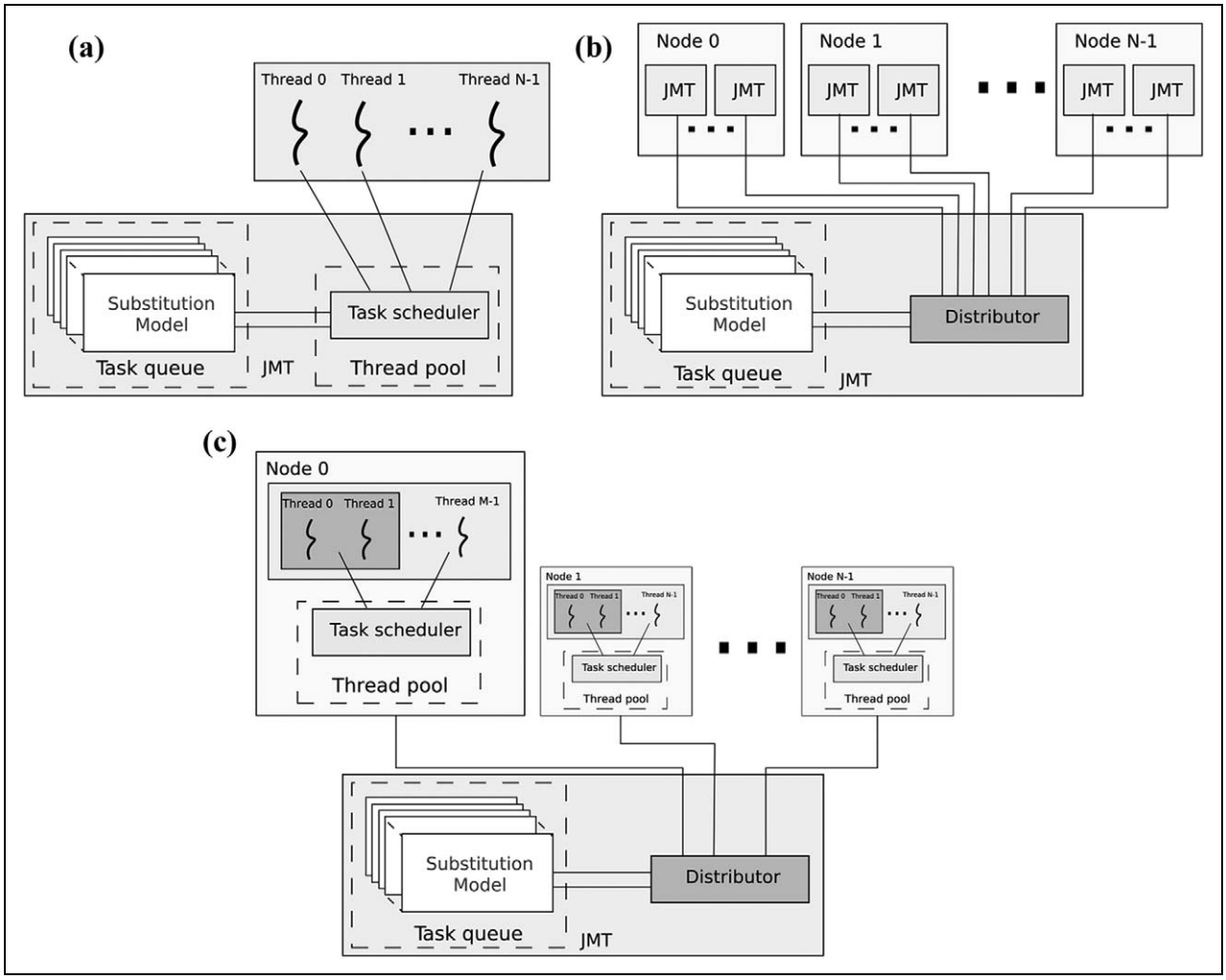

Figure 5. Parallel execution strategies in (a) shared memory, (b) distributed memory and (c) hybrid shared/distributed memory.

repeated for each different rate category (typically four categories are used).

The parallel section here represents around 75% of runtime for single category models and 98% of the total execution time for models with rate variation among sites. Amdahl's law states an expected speedup of 2.91 and 7.02, respectively, using 8 threads, and 3.36 and 12.31, respectively, using 16 threads. However, there is a high parallel overhead as long as the consumed execution time is caused by a large number of sequential repetitions of the call to this function and not so by the computational load of this loop itself.

Looking at these results, it is important to select the best ratio of resources allocated to +G and +I+G models regarding non-gamma models. For example, a four-to-one ratio is expected to balance the workload of the tasks for four- and eight-core nodes. In addition, using a number of threads that is a divisor of the number of available cores per node will maximize the number of +G and +I+G models that can be optimized in parallel. For example, an efficient allocation rule for a cluster of 12-core nodes would be the use of 4 or 6 threads for each gamma model (+G and +I+G) and a single thread for the rest (uniform rates and +I).

2.5. Hybrid shared/distributed memory implementation

The performance limitations of the previous implementations can be solved using the previous parallel implementation of the basic task for avoiding workload imbalance, and relying on a distributed memory approach to cope with memory limitations, thus implementing a three-level hybrid shared/distributed memory implementation (Figure 5(c)). Figure 9 and Algorithm 3 show its operation in a hybrid shared/distributed memory environment, such as a cluster of multicore nodes. A single jModelTest process is executed for each node, containing a custom thread pool implementation that manages the tasks that are executed within the node, and the number of cores allocated for each task (i.e.the number of OpenMP threads used for the model optimization task). This distribution, known as "*thread scheduling*", allows different amounts of computational resources to be assigned depending on the estimated

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

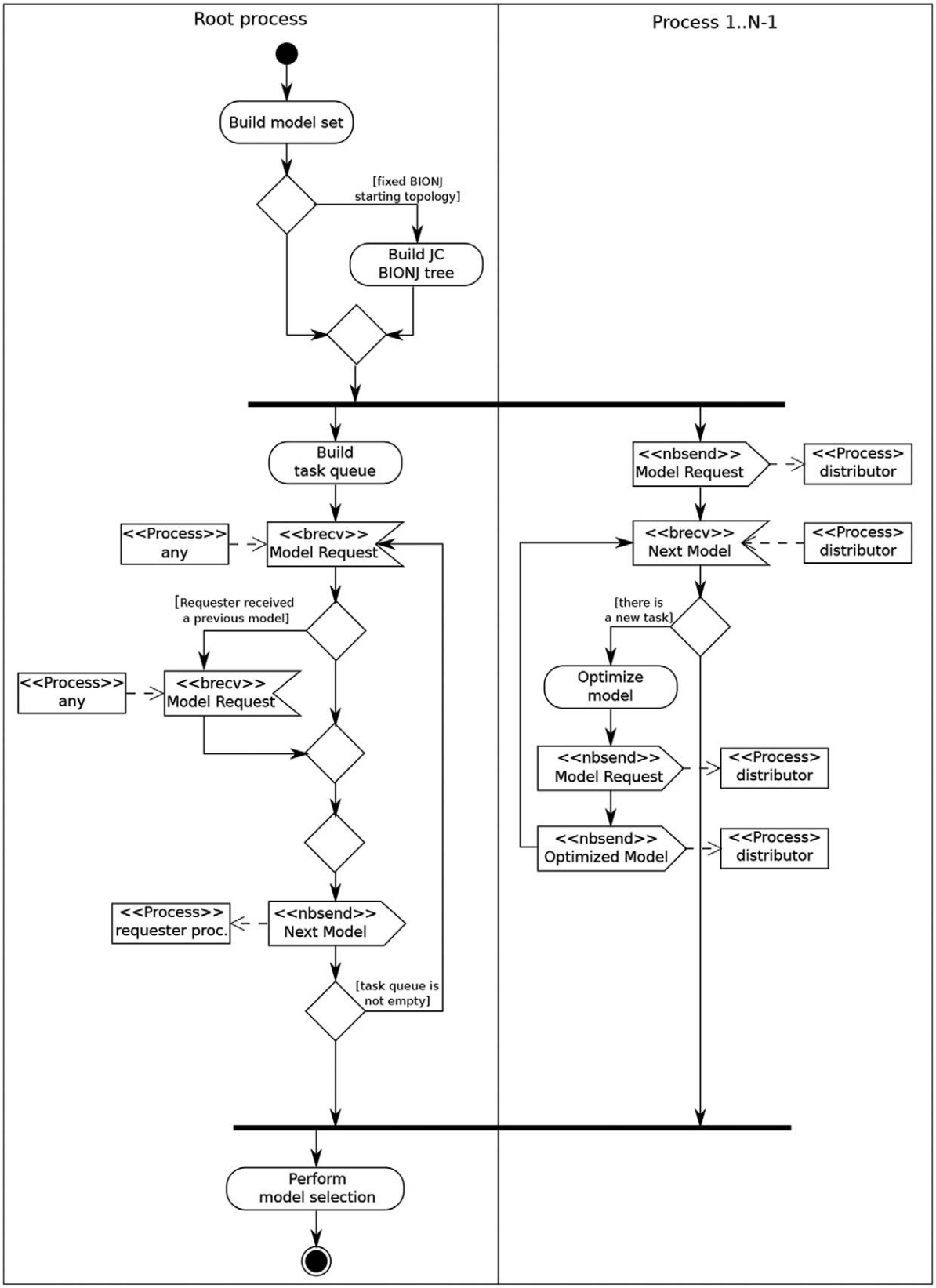

Figure 6. Activity diagram for the distributed memory parallel operation algorithm.

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

```
Data: Execution parameters, input data, resource information
Result: Best-fit model
begin Application initialization                                   // tasks  initialization
    build(model set);
    if Master process then
        Estimate workload per model;
        Sort models in reverse complexity estimate;
        build(task queue);
    else
        Wait for task;
    end
end
begin Tasks Computation                                            // model  optimization
    if Master process then
        while There are active workers do
            Wait for task request;
            if Worker had a previous model then
                Receive results from previous model;
                Update execution progress;
            end
            if Task queue is not empty then
                Send the next task to the requester worker;
            else
                Send termination message to the worker;
            end
        end
    else
        Send task request;
        while There are tasks to execute do
            Receive task;
            Execute task;
            Send task request;
            Send previous task results;
        end
        Finalize;
    end
end
```

Algorithm 2. jModelTest algorithm for distributed memory parallel model optimization.

$$Speedup_{max} = \frac{Runtime}{Runtime_{max}}$$

Figure 7. Maximum reachable speedup using task-level parallelization.

workload of each task. Therefore, the parallel efficiency is maximized in the simplest models by using a reduced number of cores (one or two), while the heaviest tasks are split, thus balancing the global workload.

Parallel synchronization between nodes is performed using MPJ, as in the previous distributed memory version. Each MPJ process uses a custom implementation of the thread pool, where both tasks and cores per task are managed. Thus, the model optimization tasks can be heterogeneously distributed among the total number of cores within the node. The workload is decomposed relying on our custom parallel PhyML version implemented with OpenMP.

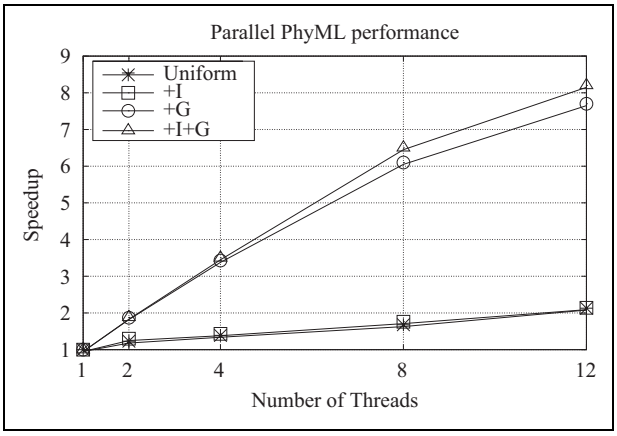

Figure 8. Parallel PhyML performance.

3. Performance evaluation

The performance of the three parallel algorithms for model selection of jModelTest2 has been evaluated on two

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

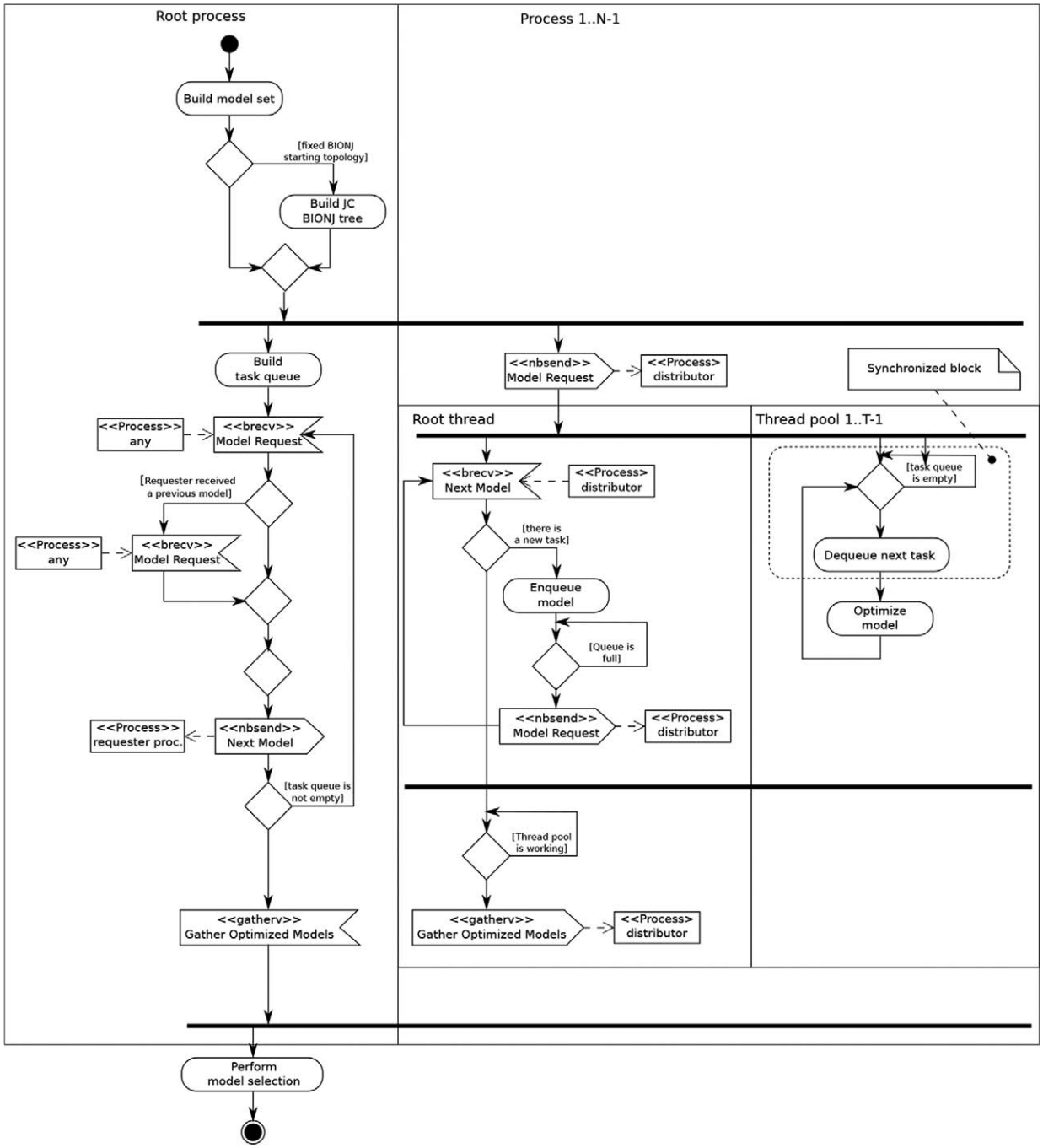

Figure 9. Activity diagram for the hybrid shared/distributed memory parallel operation algorithm.

representative HPC systems, a 40-core shared memory system and a cluster of multicore nodes (44 nodes with 12 cores per node, hence 528 cores). The distributed memory and the hybrid shared/distributed memory algorithm have been evaluated in this latter system.

3.1. Data set configuration

The data sets used in the performance evaluation consist of 4 simulated multiple sequence alignments, covering a wide range of number of sequences, from 12 to 91, and a wide range of sites for each sequence, from 3000 to 33,148 (see Table 2). The largest alignment, ALIGN1 (91 sequences of 33,148 sites), has the largest sequential runtime (5.65 days) whereas the sequential execution time of the smallest one, ALIGN4 (12 sequences of 5831 sites), is around 5 hours.

The calculation of the model likelihood scores requires an initial phylogenetic tree ("base" tree), generated using likelihood estimation (ML) (Felsenstein, 1981). This algorithm provides much more accurate results in the model selection process in exchange of significantly higher runtimes than other algorithms such as BIONJ (Gascuel,

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

```
Data: Execution parameters, input data, resource information
Result: Best-fit model
begin Application initialization                                   // tasks  initialization
    build(model set);
    if Master process then
        Estimate workload per model;
        Set the required threads per task;
        Sort models in reverse complexity estimate;
        build(task queue);
    else
        begin Initialize parallel environment
            Initialize custom thread pool;
        end
        Wait for tasks;
    end
end
begin Tasks Computation                                            // model  optimization
    if Master process then
        while There are active workers do
            Receive task request and number of idle threads on the requester pool;
            if Task queue is not empty then
                Select the most complex task that requires at most the available threads;
                Send the next task to the requester worker;
            else
                Send termination message to the worker;
            end
        end
    else
        while There are tasks to execute do
            while There are idle threads do
                Send task request and number of idle threads;
                Receive task;
                Execute task in parallel asynchronously;
            end
            Wait for idle threads;
        end
        Finalize;
    end
    Gather selection results at the Master process;
end
```

Algorithm 3. jModelTest algorithm for hybrid shared/distributed memory model optimization.

Table 2. Data sets used in the performance evaluation.

Name	Number of sequences	Length	Base Tree	Sequential runtime (hh:mm:ss)
ALIGN1	91	33,148	ML	135:42:01
ALIGN2	40	4,203	ML	15:23:56
ALIGN3	40	3,200	ML	14:33:48
ALIGN4	12	5,831	ML	4:51:13

1997). ML estimation is NP-complete, while BIONJ has a computational complexity of $O(n^3)$, where n is the number of sequences.

The distribution of the tasks is limited by the maximum number of substitution models to be computed, 88 in this case, so it is not possible to take advantage from the use of more than 88 processes. Moreover, the variability of the runtimes across models has a significant impact on performance as workload imbalance reduces the speedup and the parallel efficiency obtained. Thus, Figure 2 presents the overhead of the model likelihood calculation of ALIGN1 for each model type. In this case, when a small number of models per processor is distributed the differences in execution times can delay significantly the completion of the parallel execution.

Furthermore, even the execution times of the optimization of the models with the same parameters (e.g. "+I" and "+I+G" models) present significant variance. This characteristic, together with the fact that their execution time cannot be estimated a priori, contribute to the presence of a performance bottleneck as the number of cores increases (i.e. the fewer models per processor, the less probability the work is balanced). In order to reduce this overhead a

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

heuristic, which consists of starting the optimization with the most complex models, has been proposed. This approach has reported more balanced executions as the main source of imbalance, starting the computation of a complex model at the end of the optimization process, is avoided.

However, the scalability using the shared and distributed memory implementations is limited by the replacement models with the largest execution time, which can account for more than 80% of the overall runtime. In these cases, the runtime is determined by the longest optimization, even if the execution is prioritized efficiently using the proposed heuristic for selecting the models to be optimized first.

3.2. Testbed configuration

The shared memory testbed is a system with 4 Westmere-EX (Westmere-based EXpandable/multiprocessor) Intel Xeon E7-4850@2.0 GHz 10-core processors (hence, 40 cores) and 512 GB memory. The OS is Linux Ubuntu 11.10 64 bits, the C compiler is gcc 4.6 and the JVM is OpenJDK 1.6.0_23 64-bit Server VM.

The second testbed is a cluster of multicore nodes, used for the evaluation of the distributed and hybrid shared/distributed memory implementation. This cluster is also a Westmere-based system, Westmere-EP (Efficient Performance), which consists of 44 nodes, each of them with 2 Intel Xeon X5675@3.06 GHz hexa-core processors (hence 12 cores per node, 528 cores in the cluster) and 24 GB of RAM (1104 GB of RAM in the cluster). The interconnection networks are InfiniBand (DDR 4X: 16 Gbps of maximum theoretical bandwidth), with OFED driver 1.5.3, and Gigabit Ethernet (1 Gbps). The OS is Linux CentOS 5.3, the C compiler is gcc 4.6, the JVM is Oracle 1.6.0_23, and the Java message-passing library is FastMPJ 1.0b.

The performance metrics considered in this performance evaluation are the execution time and its associated speedup, defined as $Speedup(n) = T_{seq}/T_n$, where T_{seq} is the runtime of the sequential execution of jModelTest2, and T_n the time measured when using n cores. Another metric considered is the parallel efficiency, defined as $Efficiency(n) = Speedup(n)/n$, which is 100% in the case of a linear speedup (speedup of n when using n cores) and is close to 0% in case of highly inefficient parallel executions.

3.3. Evaluation of the shared memory algorithm

Figure 10 and Table 3 present, respectively, the speedups and execution times obtained using the shared memory implementation in the 40-core Westmere-EX shared memory system. In this scenario the speedup is close to the ideal case (i.e. obtaining speedups around n with n cores), especially using up to 24 threads. The use of a higher number of threads (32 and 40) results in workload imbalance due to the reduced number of models optimized per thread, which makes it more difficult to balance the workload even with this dynamic distribution.

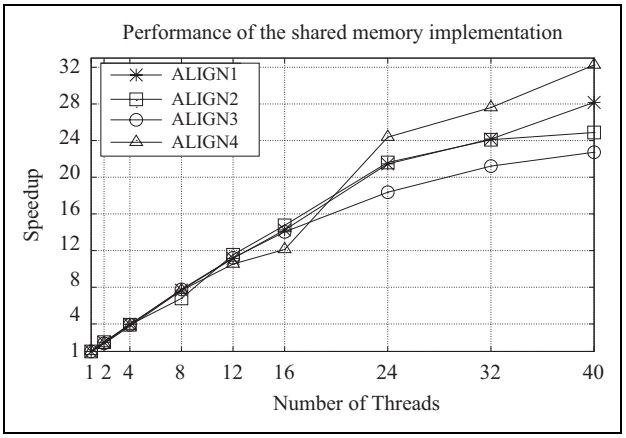

Figure 10. Scalability of the shared memory version (40-core Westmere-EX testbed).

Table 3. Execution times (hh:mm:ss) in a shared memory system (40-core Westmere-EX testbed).

Threads	ALIGN1	ALIGN2	ALIGN3	ALIGN4
1	135:42:01	15:23:56	14:33:48	04:51:13
2	66:11:01	07:39:52	07:45:53	02:37:45
4	34:14:08	03:56:18	03:43:59	01:17:28
8	17:39:16	02:16:03	01:52:30	00:38:01
12	12:10:04	01:20:03	01:17:56	00:27:40
16	09:31:29	01:02:39	01:02:13	00:24:02
24	06:19:31	00:42:45	00:47:34	00:11:57
32	05:36:53	00:38:21	00:41:11	00:10:33
40	04:49:08	00:37:08	00:38:27	00:09:02

The processors of this testbed have simultaneous multithreading (SMT) or hyperthreading, whose activation allows to run 80 threads simultaneously on 40 physical cores (two threads per physical core). However, the use of more than 40 threads in this evaluation has not reported any benefit as the workload imbalance limits the scalability. In fact, the speedups obtained from running 64 and 80 threads (not shown for clarity purposes) are slightly lower than running 40 threads due to the higher overhead of executing twice the number of model optimization tasks, which burdens memory access performance and OS thread scheduling. This result would seem questionable as Intel has reported that hyperthreading can provide up to 30% higher performance and indeed we reported such a performance benefit in our previous work on an 8-core system (Darriba et al., 2011a). Nevertheless, there is no contradiction as hyperthreading increases jModelTest2 shared memory scalability when an increment in the number of threads improves the workload balance, such as running on an 8-core system, whereas it does not yield any benefit when using 40 or more cores.

3.4. Evaluation of the distributed memory algorithm

The distributed memory implementation of jModelTest2 has been evaluated on the Westmere-EP cluster, showing

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

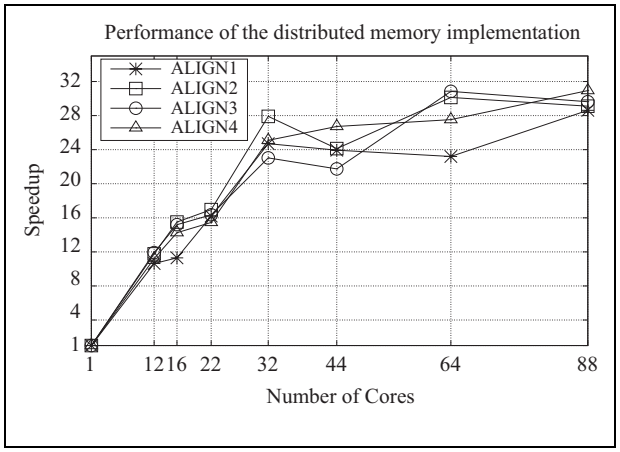

Figure 11. Scalability of the distributed memory version (Westmere-EP testbed).

Table 4. Execution times (hh:mm:ss) of the distributed memory implementation (Westmere-EP testbed).

Threads	ALIGN1	ALIGN2	ALIGN3	ALIGN4
1	100:11:08	12:53:56	12:36:04	04:01:50
12	09:25:32	01:05:50	01:03:29	00:21:34
16	08:51:50	00:49:56	00:49:47	00:16:58
22	06:11:03	00:45:38	00:46:15	00:15:39
32	04:03:22	00:27:44	00:32:50	00:09:38
44	04:11:06	00:32:05	00:34:47	00:09:03
64	04:19:04	00:25:41	00:24:31	00:08:47
88	03:30:02	00:26:36	00:25:31	00:07:49

the measured execution times and the associated speedups in Figure 11 and Table 4, respectively. In this testbed the sequential execution time is around 15–25% faster than on the Westmere-EX system, an expected result according to their respective computational power, measured in terms of the SPEC CPU floating point CFP2006 benchmark (the optimization of models is intensive in floating point operations). Thus, a single core from the Westmere-EX system obtains a result of 49.6 in the CFP2006 whereas a single core from the Westmere-EP system achieves a result of 60.3, a 22% higher.

This implementation distributes the workload among the available message-passing processes, using up to 88 processes, the maximum number of models to be optimized and running each process in a single core. In this testbed the particular allocation of processes among the cluster nodes has a negligible impact on performance ($< 0.1\%$ runtime overhead) due to the computationally intensive nature of the application with respect to the communications required (ML optimization accounts for nearly all of the execution time). In this performance evaluation the processes have been distributed among the cluster nodes using a fill-up allocation rule, minimizing the number of nodes by using the 12 cores available per node (e.g. the execution using 88 processes has distributed 12 processes per node across 7 nodes, and 4 processes in the eighth node).

The workload imbalance presented in the evaluated data sets (22 out of 88 tasks optimizing the most complex models accounts for approximately 50% of the total runtime and 44 out of 88 tasks require more than 80% of the total runtime, as shown in Figure 2) imposes an upper bound in the scalability, limiting the measured speedups to around 30, as for the shared memory implementation. Thus, distributing a small number of models per process severely limits the load balancing benefits, as it is not possible to take advantage of the spare computational power available once a process finishes its task processing and the task queue is empty. In fact, little performance benefits are obtained when using more than 32 processes, the speedups using 32 processes are around 25 (78% parallel efficiency), whereas the speedups using 88 processes are around 30 (34%), the use of 56 additional processes (a 175% resources increase, from 32 up to 88 cores) hardly improves speedups (20% higher, from 25 to 30). When using more than 32 cores most of the processes only compute a single model and finish working earlier than the longest running model, which is the one that determines the overall runtime (and, hence, the speedup) in these scenarios.

3.5. Evaluation of the hybrid shared/distributed algorithm

The limitations of the shared and distributed memory implementations of jModelTest2 have motivated the development of a more scalable implementation based on the decomposition of the model optimization among multiple threads (see Section 2.4). This decomposition depends on the allocated resources, the number of available cores per node and the computational cost of each model optimization. Thus, the longest running model computations, such as the gamma models (+G and +I+G), are split among multiple threads whereas the lightest ones are executed by one or two threads. The final objective is to achieve the workload balance among all of the involved processes. Thus, this new implementation is able to take advantage of hybrid shared/distributed memory architectures, such as clusters of multicore nodes, without compromising significantly its efficiency.

This new implementation of jModelTest2 has been evaluated on the Westmere-EP cluster, where each node has 12 physical cores and can run up to 24 simultaneous threads thanks to hyperthreading. The measured execution times and the associated speedups are shown in Figure 12 and Table 5, respectively. The experimental results have been obtained using 12 threads per node for executions from 192 up to 480 threads, thus 16, 24, 33 and 40 nodes have been used for executions on 192, 288, 396 and 480 threads, respectively. The performance results using 40 threads (4 nodes) and 88 threads (8 nodes) are also shown for comparative purposes against the results of the shared and distributed memory versions, evaluated using up to 40 and 88 cores, respectively. Here the resources allocated for each type of model (i.e. uniform, +I, +G or +I+G) are selected

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

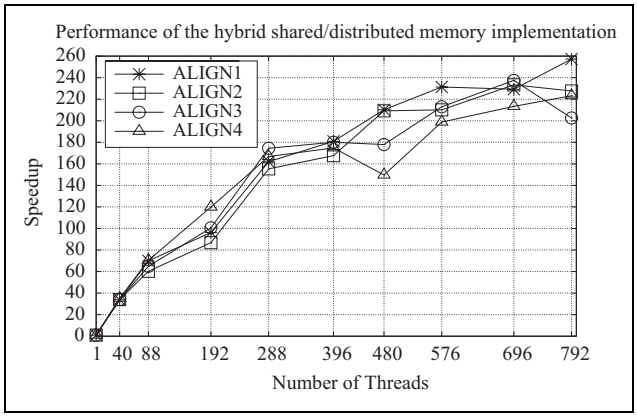

Figure 12. Scalability of the hybrid shared/distributed version (Westmere-EP testbed).

Table 5. Execution times (hh:mm:ss) of the hybrid shared/distributed memory implementation (Westmere-EP testbed).

Threads	ALIGN1	ALIGN2	ALIGN3	ALIGN4
1	100:11:08	12:53:56	12:36:04	04:01:50
40	02:58:04	03:13:29	03:09:01	00:06:52
88	01:25:55	00:12:50	00:11:39	00:03:26
192	01:02:24	00:08:55	00:07:31	00:02:01
288	00:37:07	00:04:59	00:04:20	00:01:27
396	00:33:14	00:04:37	00:04:12	00:01:23
480	00:28:36	00:03:42	00:04:15	00:01:37
576	00:25:59	00:03:41	00:03:33	00:01:13
696	00:26:13	00:03:19	00:03:11	00:01:08
792	00:23:22	00:03:24	00:03:44	00:01:05

depending on the number of total resources. For example, an execution with 4 nodes and 12 threads per node would use 4 threads for each gamma model (+G and +I+G) and a single thread for the rest.

This benchmarking has also taken into account the evaluation of the impact of hyperthreading in the performance of this new implementation. Thus, the executions with 576, 696 and 792 threads have used 32 nodes $\times$ 18 threads per node, 29 nodes $\times$ 24 threads per node and 44 nodes $\times$ 18 threads per node, respectively. The main conclusion derived from the analysis of these performance results is that jModelTest2 takes advantage of hyperthreading, both running the maximum number of simultaneous threads per node (24) or sharing half of the physical cores (18 threads running on 12 physical cores).

The analysis of the results shows that this implementation achieves significantly higher scalability, speedups around 230, in the range 203–257, which is the result of multiplying the scalability of the distributed memory processing (speedups around 30) by the scalability obtained by the parallel execution of the optimization of the models (speedups up to 8). In fact, this multilevel parallel implementation increases 8 times jModelTest2 performance, that is to say, its performance benefits are equivalent to the scalability obtained from the parallelization of the model optimization.

4. Conclusions

A popular tool for the statistical selection of models of DNA substitution is jModelTest, a sequential Java application that requires vast computational resources, especially CPU hours, which has motivated the development of its parallel implementation (jModelTest2, distributed under a GPL license at http://code.google.com/jmodeltest2). This paper presents its three parallel execution strategies: (1) a shared memory multithread GUI-based desktop version; (2) a distributed memory cluster-based version with workload distribution through message-passing; and (3) a multilevel hybrid shared/distributed memory version that distributes the computation of the likelihood estimation task across cluster nodes while taking advantage, through a thread-based parallelization, of the multiple cores available within each node.

The performance evaluation of these three strategies has shown that the hybrid shared/distributed memory implementation of jModelTest2 presents significantly higher performance and scalability than the shared and distributed memory versions, overcoming their limitations that force their execution using only up to 32–40 cores. Thus, the new implementation can take advantage efficiently of the use of up to several hundreds of cores. The observed parallel efficiencies are around 38–49%, with speedups in the range 203–257 on 528 physical cores. This performance has been obtained thanks to the workload balance provided by the thread-based decomposition of the most costly model optimization tasks.

The shared memory implementation of jModelTest2 provides scalable performance although generally up to 40 threads. Nowadays this limitation is becoming more and more important as the number of available cores per system continues increasing, especially with the advent of many-core processors which definitely demand further workload decomposition in jModelTest2.

The distributed memory implementation shows similar performance results as the shared memory version despite supporting distributed memory architectures with hundreds of cores. In fact, using more than 32 cores results in highly inefficient scalability gains due to the significant workload imbalance present in jModelTest2. However, this solution avoid memory limitations with large input data.

The hybrid shared/distributed memory implementation provides a three-level parallelism avoiding memory limitation as the previous strategy while using a heterogeneous computational resource distribution in order to achieve a better load balancing. Although there is a high overhead in the parallel execution of each task, this strategy homogenizes the task execution times, thus balancing the workload.

Acknowledgement

Special thanks are due to Stephane Guindon for his help.

Funding

This work was financially supported by the European Research Council (grant number ERC-2007-Stg 203161-

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

PHYGENOM to D.P.) and the Spanish Ministry of Science and Education (grant numbers BFU2009-08611 to D.P. and TIN2010-16735 to R.D.).

References

Akaike H (1974) A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19(6): 716–723.

Buckley TR (2002) Model misspecification and probabilistic tests of topology: evidence from empirical data sets. *Systematic Biology* 51: 509–523.

Buckley TR and Cunningham CW (2002) The effects of nucleotide substitution model assumptions on estimates of nonparametric bootstrap support. *Molecular Biology and Evolution* 19: 394–405.

Chor B and Tuller T (2006) Finding a maximum likelihood tree is hard. *Journal of the ACM* 53(5): 722–744.

Dagum L and Menon R (1998) OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science and Engineering* 5(1): 46–55.

Darriba D, Taboada GL, Doallo R and Posada D (2011a) HPC selection of models of DNA substitution. In: *Proceedings of the Workshop on High Performance Computational Systems Biology (HiBi'11), 9th International Conference on Computational Methods in Systems Biology (CMSB'11)*, Paris, France, pp. 65–72.

Darriba D, Taboada GL, Doallo R and Posada D (2011b) Prottest 3: fast selection of best-fit models of protein evolution. *Bioinformatics* 27: 1164–1165.

Felsenstein J (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution* 17: 368–376.

Felsenstein J (1988) Phylogenies from molecular sequences: inference and reliability. *Annual Review of Genetics* 22: 521–565.

Gascuel O (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution* 14(7): 685–95.

Guindon O and Gascuel S (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology* 52: 696–704.

Lemmon AR and Moriarty EC (2004) The importance of proper model assumption in Bayesian phylogenetic. *Systematic Biology* 53: 265–277.

Posada D (2008) jModelTest: phylogenetic model averaging. *Molecular Biology and Evolution* 25(7): 1253–1256. DOI: 10.1093/molbev/msn083.

Posada D and Crandall KA (2001) Selecting the best-fit model of nucleotide substitution. *Systematic Biology* 50(4): 580–601.

Schwarz G (1978) Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

Shafi A, Carpenter B and Baker M (2009) Nested parallelism for multi-core HPC systems using Java. *Journal of Parallel and Distributed Computing* 69(6): 532–545.

Author biographies

Diego Darriba received the B.S. (2010) and M.S. (2011) degrees in Computer Science from the University of A Coruña, Spain. Currently he is a Ph.D. student at the Universities of A Coruña and Vigo. His research interests are in the area of bioinformatics, focused on applying HPC to phylogenetics and phylogenomics.

Guillermo L Taboada received the B.S. (2002), M.S. (2004) and Ph.D. (2009) degrees in Computer Science from the University of A Coruña, Spain, where he is currently an Associate Professor in the Department of Electronics and Systems. His main research interest is in the area of HPC, focused on high-speed networks, programming languages for HPC, cluster/Grid/cloud computing and, in general, middleware for HPC.

Ramón Doallo received his B.S. (1987), M.S. (1987) and Ph.D. (1992) degrees in Physics from the University of Santiago de Compostela, Spain. In 1990 he joined the Department of Electronics and Systems at the University of A Coruña, where he became a Full Professor in 1999. He has extensively published in the areas of computer architecture, and parallel and distributed computing. He is coauthor of more than 140 technical papers on these topics.

David Posada received his B.S. (1994) and M.S. (1997) degrees in Biology from the University of Santiago de Compostela, Spain, and his Ph.D. (2001) in Zoology from the Brigham Young University. In 2003 he joined the University of Vigo, where he became a Full Professor in 2010. He has extensively published in the areas of phylogenetics and phylogenomics. He is coauthor of more than 120 technical papers on these topics.

Downloaded from hpc.sagepub.com at Kahramanmara Sutcu Imam on May 5, 2014

Chapter 4

Jmodeltest.org: Selection of Nucleotide Substitution Models on the Cloud

The content of this chapter corresponds to the following journal paper:

- Title: Jmodeltest.org: Selection of nucleotide substitution models on the cloud
- Authors: **Diego Darriba**, Guillermo L. Taboada, Ramón Doallo and David Posada
- Journal: Bioinformatics
- Editorial: Oxford University Press, ISSN: 1367-4803, EISSN: 1460-2059
- Year: 2011
- Volume(Number):Pages: 27(8):1164–1165
- DOI: 10.1093/bioinformatics/btr088
- Impact factor: 4.981

- Q1; 10th position on Computer Science Applications, 35th position on Molecular Biology

The final publication is available at

http://bioinformatics.oxfordjournals.org/content/early/2014/02/07/bioinformatics.btu032.fu

A copy of the accepted paper is next included.

Bioinformatics Advance Access published February 7, 2014

BIOINFORMATICS APPLICATIONS NOTE 2014, pages 1–2
doi:10.1093/bioinformatics/btu032

Phylogenetics

Advance Access publication January 21, 2014

jmodeltest.org: selection of nucleotide substitution models on the cloud

Jose Manuel Santorum[1], Diego Darriba[1,2], Guillermo L. Taboada[1] and David Posada[2,*]

[1]Department of Electronics and Systems, University of A Coruña, 15071 A Coruña and [2]Department of Biochemistry, Genetics and Immunology, University of Vigo, 36201 Vigo, Spain

Associate Editor: Alfonso Valencia

ABSTRACT

Summary: The selection of models of nucleotide substitution is one of the major steps of modern phylogenetic analysis. Different tools exist to accomplish this task, among which jModelTest 2 (jMT2) is one of the most popular. Still, to deal with large DNA alignments with hundreds or thousands of loci, users of jMT2 need to have access to High Performance Computing clusters, including installation and configuration capabilities, conditions not always met. Here we present *jmodeltest*.org, a novel web server for the transparent execution of jMT2 across different platforms and for a wide range of users. Its main benefit is straightforward execution, avoiding any configuration/execution issues, and reducing significantly in most cases the time required to complete the analysis.

Availability and implementation: *jmodeltest*.org is accessible using modern browsers, such as Firefox, Chrome, Opera, Safari and IE from http://jmodeltest.org. User registration is not mandatory, but users wanting to have additional functionalities, like access to previous analyses, have the possibility of opening a user account.

Contact: info@jmodeltest.org

Received on November 18, 2013; revised on January 8, 2014; accepted on January 14, 2014

1 INTRODUCTION

The statistical selection of best-fit models of nucleotide substitution is relevant for the phylogenetic analysis of DNA sequence alignments (Sullivan and Joyce, 2005). With the advent of next-generation sequencing (NGS) technologies, many researches are moving from phylogenetics to phylogenomics, in which large sequence alignments typically include hundreds or thousands of loci. Phylogenetic resources, therefore, need to be adapted to a high-performance computing paradigm so as to allow demanding analyses. To keep up with the increasing availability of genome-wide data, jModelTest 2 (jMT2) (Darriba *et al*., 2012) was recently developed to profit from technical optimizations and parallel computing. jMT2 uses PhyML (Guindon and Gascuel, 2003) to obtain maximum likelihood estimates of model parameters, and implements different statistical criteria for model selection including hierarchical and dynamical likelihood ratio tests, Akaike's and Bayesian information criteria (AIC and BIC) and a performance-based decision theory method (Posada and Buckley, 2004). jMT2 can take advantage of high-performance computing (HPC) environments, such as supercomputers and clusters. However, execution in HPC environments is not trivial: (i) installing, configuring and optimizing parallel software are generally cumbersome for non-HPC experts, (ii) access to HPC resources generally implies long waiting times or at least significant variability in the response time and (iii) it is difficult to estimate in advance the computational resources needed.

To overcome these limitations, we introduce *jmodeltest*.org, a web service for executing jMT2 transparently on HPC infrastructures. *jmodeltest*.org can distribute jMT2 jobs across multiple public or private clouds, such as Amazon Web Services (AWS) EC2, adopting optimal HPC configurations. *jmodeltest*.org considers the available resources at each site to minimize execution times and scales the resources up and down depending on the workload. Such an 'easy' access to HPC resources will allow users to focus more on their research rather than on secondary tasks like resource provision, installation, configuration, execution and optimization of parallel environments.

2 IMPLEMENTATION

jmodeltest.org has been implemented as a web interface for jMT2, plus a task manager. The web interface captures input data and parameters, whereas the task manager divides jMT2 jobs in different subtasks, one per substitution model. *jmodeltest*.org looks for infrastructures, which are ready to execute these subtasks immediately. Currently, *jmodeltest*.org jobs will run in private clouds at the University of Vigo and University of A Coruña, and occasionally at the Galicia Supercomputing Center (CESGA) and Amazon WS EC2 public clouds. When the server workload exceeds the available capacity of the private clouds, resources are requested from the public clouds. The technologies behind *jmodeltest*.org are Tomcat for the web interface, MySQL for handling subtasks, DRMAA for executing tasks on remote servers and *StarCluster* (http://star.mit.edu/cluster/) for managing Amazon WS EC2 resources.

Because the tasks are split, *jmodeltest*.org is able to start large analyses without having yet assigned computational resources for the whole job. Subtasks are sent to the different computational resources through Distributed Resource Management Application API (DRMAA), a high-level Open Grid Forum API specification for the submission and control of jobs to a Distributed Resource Management (DRM) system, such as a Cluster or a Grid computing infrastructure. As the job manager is not aware of the resources required to run a particular task, it

*To whom correspondence should be addressed.

© The Author 2014. Published by Oxford University Press.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on February 10, 2014

will start submitting 1h jobs with 1 GB of memory. This way, cloud schedulers will allocate resources faster. In case these initial requests are not enough, subsequent submissions will double either the time and/or the amount of memory. To save resources, *jmodeltest*.org implements a check-pointing mechanism using Distributed Multi-Threaded Check-Pointing (DMTCP).

Furthermore, users will be able use their own computational resources when running *jmodeltest*.org. The only requirement is that these machines have a resource manager (i.e. SGE, Torque, SLURM) with proper user permissions. After this, the user just needs to register this resource in *jmodeltest*.org. Only the user who registered the resource will be able to execute *jmodeltest*.org jobs on it. Communications with the added resource are secured through a public RSA key 1024 bits. Finally, we are working on a new feature that will allow users to request exclusive access to prepaid AWS EC2 resources for accelerating the jobs.

3 FUNCIONALITY

jmodeltest.org was designed to be completely transparent to the user, who does not need to install, configure or update anything, nor specifying the resources needed in a shared resources infrastructure, like the number of cores or user permissions. *jmodeltest*.org is accessible through any web browser. Users can login anonymously or register. If the login is anonymous, analyses are executed within a web session, until the browser is closed or there is a long inactivity period, losing any resulting jobs. When access occurs through a user account, job settings and results are kept in the server, and registered users can recover these at any time. This can be particularly interesting when analysing large datasets, avoiding accidental interruptions. The user account can be accessed multiple times and from multiple devices. Moreover, *jmodeltest*.org helps users to monitor their jobs, displaying information about their current state ('initializing', 'running', 'done') and resources consumed (CPU time). Once the job is completed, the user can output, view, download or delete the results. By default, *jmodeltest*.org limits the CPU time granted per user to (currently) 500 CPU hours. The web service includes documentation, example files, support tickets and a FAQ section.

4 PERFORMANCE

For benchmarking, we submitted five representative datasets to *jmodeltest*.org. We recorded the time to complete the likelihood calculations, by far the most intensive task, for 88 models using default settings, for (i) the serial version of jMT2 running in a single core, (ii) the parallel version of jMT2 running on 2, 4, 8 and 16 cores on a shared resource and (iii) *jmodeltest*.org running on backend private clouds and public cloud providers (CESGA and AWS), without waiting for resources and virtually running all tasks in parallel. Figure 1 presents the resulting execution times, taking into account both queuing time and runtime. The queuing time increased with the number of cores requested, reducing significantly the benefits of using the parallel version on a shared resource. Here *jmodeltest*.org performed best, as it has multiple resources and can virtually run all the tasks in parallel.

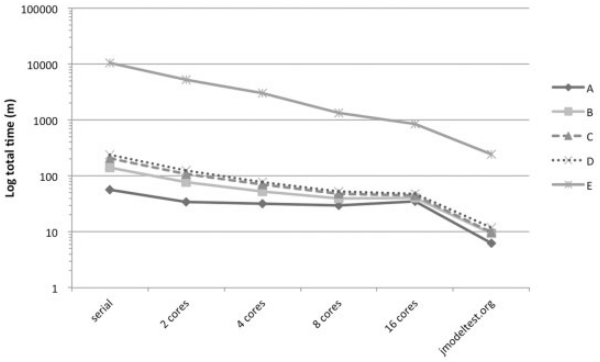

Fig. 1. Execution times for five datasets (alignments A–E, with 35, 43, 9, 44 and 246 sequences and 392, 492, 14 403, 561 and 4465 sites, respectively), using the serial and parallel versions of jModelTest2, and *jmodeltest*.org

ACKNOWLEDGEMENTS

The authors thank multiple anonymous users for testing the web server, and the Galician Bioinformatic Network and Galician Supercomputing Center (CESGA) for allowing the use of shared resources.

Funding: This work was financially supported by the European Research Council (ERC- 2007-Stg 203161- PHYGENOM), by the Ministry of Science and Innovation of Spain under Projects TIN2010-16735, and by an Amazon Web Services (AWS) research grant "EC2 in phylogenomics".

Conflict of Interest: none declared.

REFERENCES

Darriba,D. *et al*. (2012) jModelTest 2: more models, new heuristics and parallel computing. *Nat. Methods*, **9**, 772.

Guindon,S. and Gascuel,O. (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst. Biol.*, **52**, 696–704.

Posada,D. and Buckley,T.R. (2004) Model selection and model averaging in phylogenetics: advantages of akaike information criterion and bayesian approaches over likelihood ratio tests. *Syst. Biol.*, **53**, 793–808.

Sullivan,J. and Joyce,P. (2005) Model selection in phylogenetics. *Annu. Rev. Ecol., Evol. Syst.*, **36**, 445–466.

Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on February 10, 2014

From *jModelTest.org* main page, you can log in with a user account (created under the Register label) or use the anonymous log in. In such a case, the privacy of the data is maintained until log out, removing all data and results at that time (Figure 4.1).

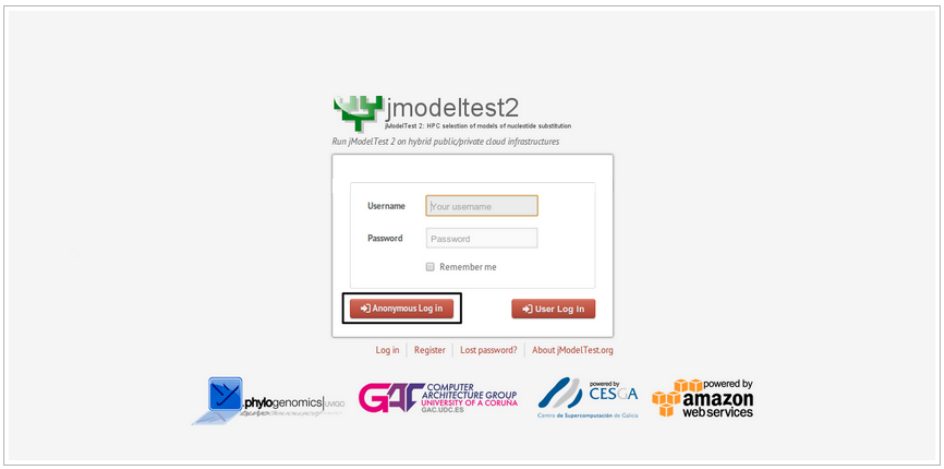

Figure 4.1: jModelTest.org login screen

For submitting a new job, the first task is to select the Multiple Sequence Alignment (Figure 4.2). Name the job 1 and load an input alignment file 2 (with ALN, FASTA, GDE, MSF, NEXUS, PHYLIP and PIR formats). Three sample alignment files are provided for testing purposes 3.

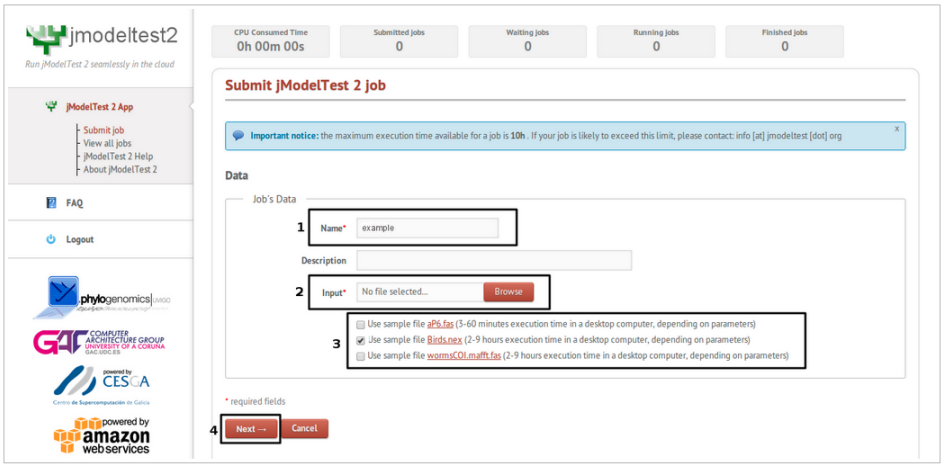

Figure 4.2: jModelTest.org MSA input

Next, fill in the execution options (Figure 4.3). The information about each parameter is self contained. After job submission, the user will be notified about it, as well as important status changes (task finishes or unexpectedly stops).

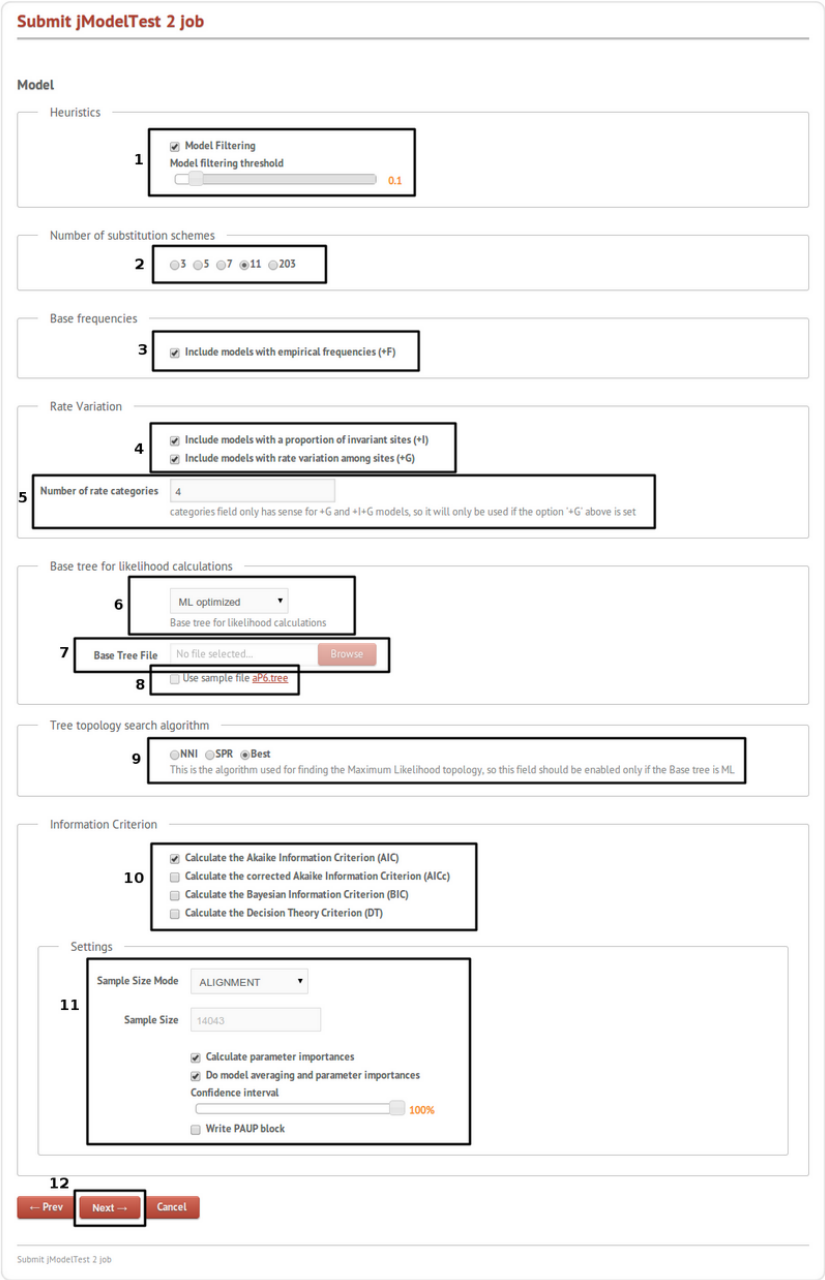

Figure 4.3: jModelTest.org execution options

Finally, the user can check out the status of the submitted jobs 2 at the "view all jobs" submenu 1 (Figure 4.4). The job status is shown with an icon and also a color code:

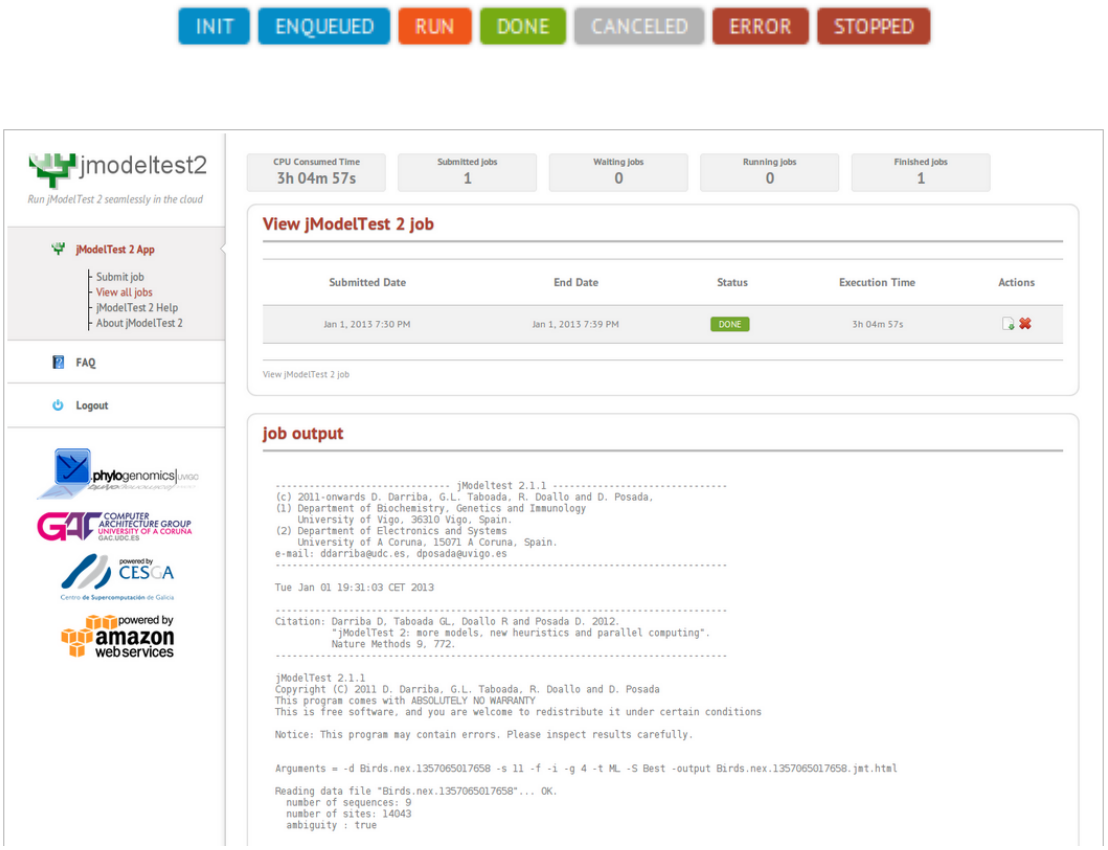

Figure 4.4: jModelTest.org running status

Chapter 5

jModelTest 2: More Models, New Heuristics and Parallel Computing

The content of this chapter corresponds to the following journal paper:

- Title: jModelTest 2: More models, new heuristics and parallel computing
- Authors: Jose Manuel Santorum, **Diego Darriba**, Guillermo L. Taboada and David Posada
- Journal: Nature Methods
- Editorial: Nature Publishing Group, ISSN: 1548-7091, EISSN: 1548-7105
- Year: 2012
- Volume(Number):Pages: 9(8):772–772
- DOI: 10.1038/nmeth.2109
- Impact factor: 32.072
- Q1; 2nd position on Biochemistry and Biotechnology, 4th position on Molecular Biology and Cell Biology

- Number of citations: 1,849 as of October 2015

The final publication is available at

http://www.nature.com/nmeth/journal/v9/n8/full/nmeth.2109.html

A copy of the accepted paper is next included.

Darriba D, Taboada GL, Doallo R, Posada D. 2012. jModelTest 2: more models, new heuristics and parallel computing. Nature Methods 9, 772 (2012)

doi:10.1038/nmeth.2109

http://www.nature.com/nmeth/journal/v9/n8/full/nmeth.2109.html

jModelTest 2: more models, new heuristics and parallel computing

To the Editor: The statistical selection of best-fit models of nucleotide substitution is routine in the phylogenetic analysis of DNA sequence alignments[1]. With the advent of next-generation sequencing technologies, most researchers are moving from phylogenetics to phylogenomics, in which large sequence alignments typically include hundreds or thousands of loci. Phylogenetic resources therefore need to be adapted to a high-performance computing paradigm so as to allow demanding analyses at the genomic level. Here we introduce jModelTest 2, a program for nucleotide-substitution model selection that incorporates more models, new heuristics, efficient technical optimizations and parallel computing.

jModelTest 2 includes important features not present in the previous versions[2,3] (**Supplementary Table 1**). We expanded the set of candidate models from 88 to 1,624, and we implemented two heuristics for model selection: a greedy, hill-climbing hierarchical clustering approach (**Supplementary Note 1**) and a filtering algorithm based on similarity among parameter estimates (**Supplementary Note 2**).

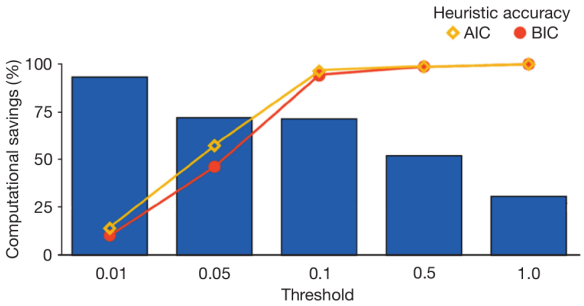

Figure 1 | Benchmarking of the filtering heuristic in jModelTest 2. The threshold of the filtering heuristic (**Supplementary Note 2**) is directly correlated with the probability of finding the true best-fit model (heuristic accuracy) and inversely related to the number of models for which we avoided the likelihood calculation (computational savings). AIC, Akaike information criterion[5]; BIC, Bayesian information criterion.

jModelTest 2 is written in Java, and it can run on Windows, Macintosh and Linux platforms. Source code and binaries are freely available from https://code.google.com/p/jmodeltest2/. The package includes detailed documentation and examples, and a discussion group is available at https://groups.google.com/forum/#!forum/jmodeltest/.

We evaluated the accuracy of jModelTest 2 using 10,000 data sets simulated under a large variety of conditions (**Supplementary Note 3**). Using the Bayesian information criterion[4] for model selection, jModelTest 2 identified the generating model 89% of the time (**Supplementary Table 2**); in the remaining cases, jModelTest 2 selected a model similar to the generating one.

Accordingly, model-averaged estimates of model parameters were highly accurate (**Supplementary Table 3**). In these simulations, the two selection heuristics that we developed were accurate and efficient. Using the hierarchical clustering heuristic, we found the same best-fit model as the full search 95% of the time. With the similarity filtering approach, we reduced the number of models evaluated by 60% on average and found the global best-fit model 99% of the time (**Fig. 1** and **Supplementary Note 2**).

jModelTest 2 can be executed in high-performance computing environments as (i) a desktop version with a user-friendly interface for multicore processors, (ii) a cluster version that distributes the computational load among nodes, and (iii) as a hybrid version that can take advantage of a cluster of multicore nodes. An experimental study with real and simulated data sets showed remarkable computational speedups compared to previous versions (**Supplementary Note 4**). For example, the hybrid approach executed on the Amazon EC2 cloud with 256 processes was 182–211 times faster. For relatively large alignments (138 sequences and 10,693 sites), this could be equivalent to a reduction of the running time from nearly 8 days to around 1 hour.

Note: Supplementary information is available at http://www.nature.com/doifinder/10.1038/nmeth.2109.

ACKNOWLEDGMENTS
This work was financially supported by the European Research Council (ERC-2007-Stg 203161-PHYGENOM to D.P.), Spanish Ministry of Science and Education (BFU2009-08611 to D.P.) and Xunta de Galicia (Galician Thematic Networks RGB 2010/90 to D.P. and GHPC2 2010/53 to R.D.).

COMPETING FINANCIAL INTERESTS
The authors declare no competing financial interests.

Diego Darriba[1,2], Guillermo L Taboada[2], Ramón Doallo[2] & David Posada[1]

[1]Department of Biochemistry, Genetics and Immunology, University of Vigo, Vigo, Spain. [2]Computer Architecture Group, University of A Coruña, A Coruña, Spain.
e-mail: dposada@uvigo.es

1. Sullivan, J. & Joyce, P. *Annu. Rev. Ecol. Evol. Syst.* **36**, 445–466 (2005).
2. Posada, D. & Crandall, K.A. *Bioinformatics* **14**, 817–818 (1998).
3. Posada, D. *Mol. Biol. Evol.* **25**, 1253–1256 (2008).
4. Schwarz, G. *Ann. Stat.* **6**, 461–464 (1978).
5. Akaike, H. *IEEE Trans. Automat. Contr.* **19**, 716–723 (1974).

Supplementary Information

Nature Methods

jModelTest 2: more models, new heuristics and parallel computing

Diego Darriba, Guillermo L. Taboada, Ramón Doallo and David Posada

Supplementary Table 1. New features in jModelTest 2

Supplementary Table 2. Model selection accuracy

Supplementary Table 3. Mean square errors for model averaged estimates

Supplementary Note 1. Hill-climbing hierarchical clustering algorithm

Supplementary Note 2. Heuristic filtering

Supplementary Note 3. Simulations from prior distributions

Supplementary Note 4. Speed-up benchmark on real and simulated datasets

Nature Methods: doi:10.1038/nmeth.2109

Supplementary Table 1 | New features in jModelTest 2. jModelTest 2 implements a number of new features that facilitate model selection among more models and for large data sets.

New feature	Description
1. Exhaustive GTR submodels	All the 203 different partitions of the GTR rate matrix[1] can be included in the candidate set of models. When combined with rate variation (+I,+G, +I+G) and equal/unequal base frequencies the total number of possible models is $203 \times 8 = 1624$.
2. Hill-climbing hierarchical clustering	Calculating the likelihood score for a large number of models can be extremely time-consuming. This hill-climbing algorithm implements a hierarchical clustering to search for the best-fit models within the full set of 1624 models, but optimizing at most 288 models while maintaining model selection accuracy.
3. Heuristic filtering	Heuristic reduction of the candidate models set based on a similarity filtering threshold among the GTR rates and the estimates of among-site rate variation.
4. Absolute model fit	Information criterion distances can be calculated for the best-fit model against the unconstrained multinomial model (based on site pattern frequencies)[2]. This is computed by default when the alignment does not contain missing data/ambiguities, but can also be approximated otherwise.
5. High Performance Computing	Model selection can be executed in parallel in multicore desktop machines and in HPC clusters achieving large speedups.
6. Topological summaries	Tree topologies supported by the different candidate models are summarized in the html log, including confidence intervals constructed from cumulative models weights, plus Robinson-Foulds[3] and Euclidean distances to the best-fit model tree.
7. Alignment sample size	The alignment sample size used for the AICc and BIC frameworks can be calculated according to alignment length (L) as before, but also as the number of variable sites, $L \times$ the number of sequences (N), Shannon entropy and Normalized Shannon entropy multiplied by $N \times L$.
8. User-friendly HTML log	The results of the model selection can be displayed in html format including maximum likelihood trees derived from each model and linked to http://www.phylowidget.org[4] for graphical depiction.

Nature Methods: doi:10.1038/nmeth.2109

Supplementary Table 2 | Model selection accuracy. Model selection accuracy was defined as the number of times the best-fit model selected by jModelTest 2 was the generating model. In case these models differed, we kept track of which components of the generating model were identified correctly (base frequencies, partition, rate variation among sites). In this table we show the model selection accuracy (%) across 10,000 data sets. *Num Params* refers to the mean number of parameters of the best-fit models. *Full Model* refers to the number of times the exact generating model was selected as the best-fit model. *Partition* refers to the number of times the structure of the R-matrix was correctly identified. *Rate Variation* refers to the number of times the rate variation parameter combinations (+I, +G, +I+G) were correctly identified.

Criterion	Num Params	Full Model	Partition	Rate Variation
AIC	5.62	62.36	70.64	93.11
BIC	4.99	89.34	89.87	99.29
DT	4.99	89.30	89.94	99.27

Nature Methods: doi:10.1038/nmeth.2109

Supplementary Table 3 | Mean square errors for model averaged estimates. To obtain the MSEs, and because the generating model and the best-fit model can differ, we did not consider every case for every parameter. For the base frequencies, transition/transversion ratio and R-matrix we considered all cases (see **Supplementary Note 3**). For the proportion of invariable sites in +I models (*p-invI*) we considered only cases where the generating model was M (*p-inv*=0) or M+I (*p-inv* = simulated). For the proportion of invariable sites in +I+G models (*p-invIG*) we considered only cases where the generating model was M+I+G (*p-inv* = simulated). For the alpha shape of the gamma rate variation among sites in M+G models (*alphaG*) we considered only cases where the generating model was M+G (*alpha* = simulated). For the alpha shape of the gamma rate variation among sites in M+I+G models (*alphaIG*) we considered only cases where the generating model was M+I+G (*p-inv* = simulated).

Parameter	MSE (AIC)	MSE (BIC)
fA	0.01	0.01
fC	0.01	0.01
fG	0.01	0.01
fT	0.01	0.01
titv	0.88	0.75
Ra	3.93	2.46
Rb	13.46	12.03
Rc	6.12	10.95
Rd	4.92	3.26
Re	6.16	5.38
p-invI	0.83	0.83
p-invIG	0.02	0.02
alphaG	0.09	0.09
alphaIG	0.14	0.14

Nature Methods: doi:10.1038/nmeth.2109

Supplementary Note 1 | Hill-climbing hierarchical clustering algorithm. Models of DNA substitution are defined by a rate matrix R, which describes the rate at which nucleotides of one type change into another type (e.g., r_{ij} for $i \neq j$), is the rate at which base i goes to base j. Because the models are most of the time assumed time reversible for tractability, this rate matrix is in practice always symmetrical. Therefore, we can define the rate matrix just in terms of the upper triangular matrix as a vector of 6 rates (r_{AC}, r_{AG}, r_{AT}, r_{CG}, r_{CT}, r_{GT}). Note that we assume all rates are relative to r_{GT}, which is set to 1.0. We can reduce the number of free parameters further forcing several of these rates to be the same. For example, we could assume the same rate for the two types of transitions (r_{AG} = r_{CT}). An easy way to label these partitions (i.e., the set of constraints) is indicating with 6 digits which rates are forced to be identical. For example, the JC[5] model has the partition 000000, where all the 6 rates among the 4 nucleotides are identical, while the GTR or SYM[6] models have the partition 012345, where all 6 rates are different. How many partitions are in between? The number of ways in which we can subdivide n elements into (non-empty) groups is given by the Bell numbers, $B(n)$, which in turn is the sum from $k = 1$ to $k = n$ of the number of ways to partition a set of n elements into k (non-empty) groups, which is given by the Stirling numbers of the second kind, $S(n,k)$:

$$B(n)=\sum_{k=0}^{n} S\left(n,k\right)=\sum_{k=0}^{n}\frac{1}{k!}\sum_{j=0}^{k}\left(-1\right)^{k-j}\binom{k}{j}j^{n}$$

$$S\left(n,k\right)=\sum_{j=0}^{k}\left(-1\right)^{k-j}\frac{j^{n-1}}{\left(j-1\right)!\left(k-j\right)!}=\frac{1}{k!}\sum_{j=0}^{k}\left(-1\right)^{k-j}\binom{k}{j}j^{n}$$

Accordingly, the R-matrix can be partitioned in $B(3)$ = 203 different ways. For each R-matrix we can build 8 different models depending on the rate variation parameters (i.e., +I, +G and +I+G models) and whether we considered equal/unequal frequencies. Therefore, the total number of possible time reversible models is 203 × 8 = 1624. Indeed, the exhaustive computation of so many models is only feasible for small alignments or when computer power is not a problem. To alleviate this situation, we have implemented a simple, greedy hill climbing heuristic to search for the best-fit model in large candidate sets of models (up to 1624) without evaluating all of them (i.e., avoiding an *exhaustive* search). The algorithm is as follows:

1. Start with n = 6 and k = 6. There is only a single partition (012345) that fits this condition.
2. Select the best-fit model, $M_{current_best}$, according to the chosen information-theoretic criterion (AIC, AICc or BIC).
3. Set $k = k - 1$.
4. Define a new set of models by exploring all possible merges of two groups into a single group.
5. Select the best-fit model, M_{best_merge}, from this set.

Nature Methods: doi:10.1038/nmeth.2109

6. If $M_{current_best}$ has a better AIC/AICc/BIC score than M_{best_merge}, stop, otherwise continue
7. Set $M_{current_best} = M_{best_merge}$
8. Update the number of elements ($n = n - 1$).
9. Repeat from step 3 until the algorithm finds a local maxima or $k = 1$.

Note that in fact we are travelling in diagonal through the Stirling numbers of the second kind pyramid (**Fig. S1**), evaluating models only at those stages where $k = (n\text{-}1)$ for $n = 1 \ldots 6$.

n \ k	1	2	3	4	5	6
1	1					
2	$\boxed{1}$	1				
3	1	$\boxed{3}$	1			
4	1	7	$\boxed{6}$	1		
5	1	15	25	$\boxed{10}$	1	
6	1	31	90	65	$\boxed{15}$	$\boxed{1}$

Figure S1 | Stirling numbers of the second kind. The rows sum to the nth Bell number (e.g., 203 for $n = 6$). The squared numbers represent the stages of our hierarchical clustering algorithm. As long as our algorithm moves forward, the number of elements and groups are reduced by one until there is a single group.

In this heuristic the number of model partitions evaluated goes from a minimum of 1 to a maximum of 36 (i.e., up to 36 x 8 = 288 different models). Because, as any heuristic, this algorithm can get stuck in local optima, we have evaluated its performance analyzing 2,000 simulated alignments (as in **Supplementary Note 3** but considering all possible 203 *R*-matrices). In this case, our heuristic finds the same model as the exhaustive search 95% of the time. Note that a very similar heuristic to find model partitions across multigene data sets has just been developed[7].

Nature Methods: doi:10.1038/nmeth.2109

Supplementary Note 2 | Heuristic filtering. For very large alignments, even the computation of the likelihood of the 88 standard models implemented in the previous version of jModelTest can take a long time. We have developed a second heuristic to find the best-fit model without evaluating all candidate models. The basic idea is that model selection can be somewhat predicted from the most complex model. Our strategy attempts to significantly reduce the number of candidate models evaluated paying attention to the GTR+I+G rate matrix and the base frequencies estimates. For example, if the maximum likelihood estimates of the transition and transversion rates are very different and the likelihood score is low enough (so we expect noticeable likelihood differences among models), one could obviate the evaluation of a simple model like JC. We perform the filtering process in three main steps: (1) look at the rate matrix, where different enough rates will imply that models with equal rates will be excluded; (2) look at the base frequencies, where different enough frequencies will imply that models with equal base frequencies will be excluded; and (3) look at the among-site rate variation, where small *p-inv* or *alpha* estimates will imply that only site-homogeneous models will be considered.

To decide what can be considered different enough we defined a *filtering threshold* (δ). A higher δ means a larger model set and a smaller probability if getting trapped in local optima (i.e., the model selected is not the best one according to the selection criterion). On the other hand, a lower δmeans more possibilities of selecting the optimal model but less computational load. Although accurate among-site rate variation filtering should be presumably implemented from the GTR+I or GTR+G models, we prefer to use a single model (GTR+I+G) for every dataset. Because these two rate variation parameters (*alpha* and *pinv*) try to model the same thing, a proportion of invariable sites could be theoretically 'converted' into gamma rate variation in the +I+G model. Therefore, we use the two thresholds just described for excluding models at this step. Once the filtering is completed, we will obtain a set of excluded models that it is not necessary to optimize. The whole heuristic is as follows:

1. Optimize the GTR+I+G model, obtaining maximum likelihood estimates of the *R*-matrix (to facilitate notation r_{AC}, r_{AG}, r_{AT}, r_{CG}, r_{CT} and r_{GT} will be referred here as r_1, r_2, r_3, r_4, r_5 and r_6, respectively), base frequencies (π_1, π_2, π_3 and π_4), the proportion of invariant sites (*pinv* = ρ) and the alpha shape of the gamma distribution (*alpha* = α) estimates.
2. Set up a filtering threshold $\delta \in \Re > 0$
3. If the standard deviation of the rates r_i is high enough ($\sigma > 1.0$) standardize them to a *z*-score: $z_i = (r_i - \bar{r}_i) / \sigma_{r_i}$
4. Calculate the pairwise differences between every pair of rates $D_{ij} = \left| z_i - z_j \right|$
5. If $D_{ij} > \delta$, all models with equal *i* and *j* rates will be ignored during the selection process. One special case is for the transition rates (r_2 and r_5) , where the rates are known to be higher than the transversion rates. Therefore in this case we use a more stringent threshold, $D_{25} > 2\delta$.

Nature Methods: doi:10.1038/nmeth.2109

6. Check whether $\frac{\min(\pi_1,\pi_2,\pi_3,\pi_4)}{\max(\pi_1,\pi_2,\pi_3,\pi_4)} < (1-\delta)$
7. In this case, all models with equal frequencies are ignored.
8. Define a gamma shape threshold $\alpha_{min} \in \Re$ (α_m should be big enough, e.g., $\alpha_m = 50$).
9. If $\alpha < \alpha_{min}$ filter out all +G and +I+G models from the candidate set
10. Let $\rho_{min} \in \Re$ and $\alpha_{max} \in \Re$, where ρ_{min} is the minimum ρ and α_{max} the maximum α. Then +I models will be excluded if $\rho < \rho_{min}$ and $\alpha > \alpha_{max}$. Note that α_{max} is not expected to be as big as α_{min}, but the higher it is, the less probably is to exclude the best-fit model.

In addition, our empirical analysis also showed that the effectiveness of this heuristic depends on the 'complexity' of the input data, which is reflected in the likelihood score. For simpler data sets the likelihood is smaller, and the number of parameters become more decisive, favoring the selection of simpler models. However, in this case the reduction of the candidate models set is also less important since the execution times will also be smaller. **Fig. S2** shows the heuristic accuracy as a function of the likelihood score across 4,000 alignments sampled from the 10,000 simulated.

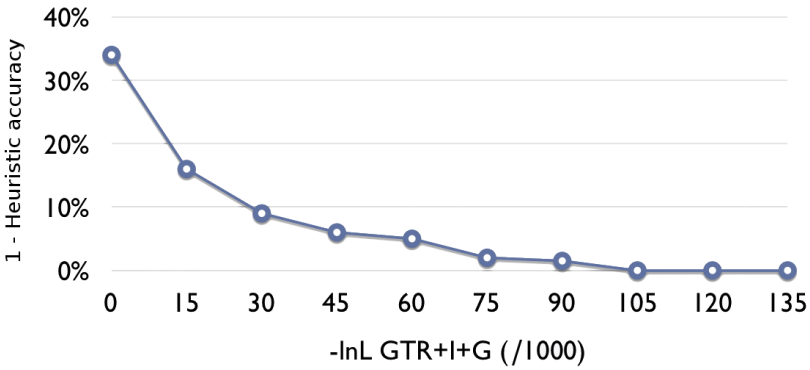

Figure S2 | Likelihood score and heuristic performance. The figure shows the percentage of best-fit models identified during the exhaustive search wrongly filtered out (1 - heuristic accuracy) from the candidate model set in function of the likelihood of the GTR+I+G model for a fixed filtering threshold.

Since it is difficult for the user to estimate *a priori* the likelihood of the models, we aimed to find some kind of general "threshold tuning" that depends on the likelihood score of the GTR+I+G model and guarantees a similar trade-off between the accuracy of the heuristic and the computational savings to that depicted in **Fig. 1** independently of the particular data set. Using logarithmic interpolation we arrived to the following function:

$$f(x) = t\frac{-5\ln(x-1)+6\ln(151)-1\ln(1)}{\ln(151)-\ln(1)}$$

where t is the user-defined filtering threshold and x is the -lnL of the GTR+I+G model for the specific data set divided by 1000.

Nature Methods: doi:10.1038/nmeth.2109

Supplementary Note 3 | Simulations from prior distributions. We simulated 10,000 nucleotide sequence alignments with 40 sequences and 2500 bp each. In order to consider a variety of simulation scenarios we first sampled model parameters and random trees from different statistical distributions using R[8]. Then, we used Seq-Gen[9] to simulate the DNA sequences accordingly and analyzed them with jModelTest 2. We considered 4 model families: without rate variation (M), with a proportion of invariable sites (M+I), with gamma rate variation among sites (+G), and with both a proportion of invariable sites and gamma rate variation among sites (+I+G). The simulation pipeline was repeated 2,500 times for each model family in turn:

1. Select at random one of the 22 possible models implemented in jModelTest 2 for the given family according to a Uniform distribution U(0,21).

2. Assign parameter values according to the model predefined structure:
    - 2.1. The base frequencies (ACGT) are set to 0.25 or sampled from a Dirichlet distribution D(1.0,1.0,1.0,1.0).
    - 2.2. The transition/Transversion rate comes from a Gamma distribution G(2,1) truncated between 2 and 10
    - 2.3. The R-matrix parameters are sampled from a Dirichlet distribution D(6,16,2,8,20,4) scaled with the last rate.
    - 2.4. The gamma shape for rate variation among sites comes from an Exponential distribution E(2) truncated between 0.5 and 5.
    - 2.5. The proportion of invariable sites is sampled from a Beta distribution B(1,3) truncated between 0.2 and 0.8.

3. Generate a random non-ultrametric rooted tree:
    - 3.1. We used the function *rtree* from the ape R package[10] (this works splitting randomly the edges) with branches according to a Exponential distribution E(1,10).
    - 3.2. Total tree length was scaled so the tree length was uniformly distributed in the U[2, 12] interval.

4. Simulate a sequence alignment using the parameter values sampled and the simulated tree using SeqGen[9].

5. Analyze this dataset using jModelTest 2: select the best-fit model under the AIC, BIC and DT criteria and obtain model-averaged parameter estimates.

Nature Methods: doi:10.1038/nmeth.2109

Supplementary Note 4 | Speed-up benchmark on real and simulated datasets. We analyzed several real data sets in order to benchmark the speed-ups obtained with jModelTest 2:

Dataset	Organism	Genes	NumSeq	Length	Reference
A	HIV-1	polimerase	8	3009	http://www.hiv.lanl.gov/
B	HIV-1	whole genome	138	10693	http://www.hiv.lanl.gov/
C	Yeast	106 genes	8	127060	Rokas et al. (2003)
D	simulated	--	40	500	Guindon and Gascuel (2003)
E	simulated	--	100	500	Guindon and Gascuel (2003)

Datasets A and B are trimmed alignments initially downloaded from http://www.hiv.lanl.gov/. Dataset C was provided by Antonis Rokas[11]. Datasets D and E are two alignments already used for benchmarking Phyml_ENREF_11[12], and can be downloaded from http://www.atgc-montpellier.fr/phyml/datasets.php.

The threaded version of jModelTest 2 was executed on CESGA's SVG nodes with 2 AMD Opteron Processors 6174@2.2GHz (2x12 cores, hence 24 cores) and 32GB memory. The MPI-based version of jModelTest 2 was executed on 8 Xeon nodes with 2 Intel Xeon E5420@2.50GHz per node (2x4 cores, hence 8 cores per node) and 16GB memory. These nodes are interconnected via 10 Gigabit Ethernet. The hybrid multithread/MPI-based implementation was executed in a public cloud infrastructure, in 32 Amazon EC2 cluster compute instances (23 GB memory, 33.5 EC2 Compute Units and Linux OS), with two Intel Xeon X5570 quad-core Nehalem processors each instance, hence 8 cores per virtual machine. These systems are interconnected via 10 Gigabit Ethernet, which is the differential characteristic of this resource. In fact, this EC2 instance type has been specifically designed for HPC applications and other demanding latency-bound applications. According to Amazon one EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

In the shared memory architecture with 24 cores, the scalability of the multithreaded implementation was almost linear with up to 8 threads, but also scaled well with 24 threads (**Fig. S3a**). In a cluster –distributed memory– the MPI-based application scaled well up to 32 processes, especially for the largest data sets (**Fig. S3b**). Here, the fact that some models can be optimized much more faster than others –especially when they do not include rate variation among sites–, posed a theoretical limit to the scalability. This problem was circumvented when we implemented a hybrid multithread/MPI-based approach –shared and distributed memory–, executed on Amazon EC2 cloud, which resulted in speedups of 182-211 with 256 processes even for the most complex cases (**Fig. S3c**). For relatively large alignments here (e.g., 138 sequences and 10,693 sites) this could be equivalent to a reduction of the running time from near 8 days to around 1 hour.

Nature Methods: doi:10.1038/nmeth.2109

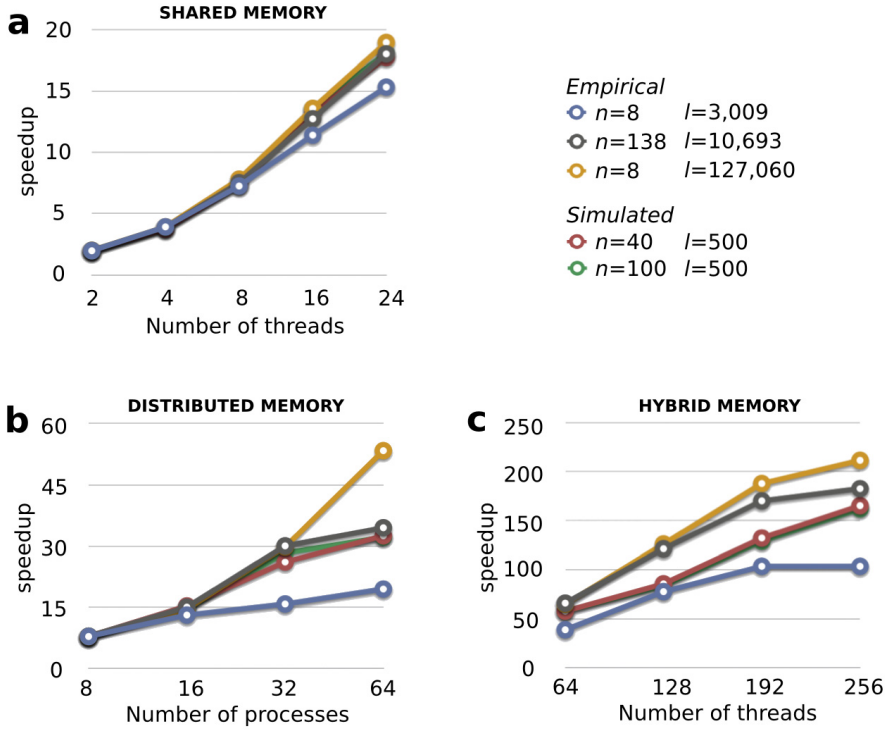

Figure S3. Scalability of jModelTest 2 with real and simulated data. The x-axis represents the number of parallel processes used in executions, and the y-axis represents the speedup regarding the sequential execution. *n* is the number of taxa and *l* is the alignment length.

In addition, we ran additional simulated datasets with up to 100 taxa and 10,000 sites on a different testbed. With shared memory (**Fig. S4a**), jModelTest 2 showed an almost linear speedup up to 8 threads (1 per core). When enabling hyperthreading (12 or 16 threads on 8 physical cores) the scaling-up was less pronounced. With distributed memory (**Fig. S4b**), the scalability was even better, reaching some saturation with 64 processes, mainly due to the serialized execution of each model optimization and the workload imbalance between models. As the 22 +G models represent around 80% of the total execution time, it is expected a theoretical limit 40X speedup in most cases. The hybrid memory version brings a more fine grain parallelism, and therefore overcomes the previous scalability limit (**Fig. S4c**). Those tests with higher computational load reached up to 130X speedup, while the lightest ones showed reduced efficiency due to the low computational load per process, making the parallel overhead (i.e., the cost of communications and synchronization) larger in relative terms. In this experiment, shared memory speedups were obtained in an 8-core node with Hyper-Threading technology (i.e., running up to 16 threads on 8 physical cores). The distributed memory version was executed on 8 Xeon nodes with 2 Intel Xeon E5420@2.50GHz per node (2 7 4 cores, hence 8 cores per node) and 16GB memory. These nodes are interconnected via 10 Gigabit Ethernet. The Hybrid memory version was executed on 32 nodes with the same features.

Nature Methods: doi:10.1038/nmeth.2109

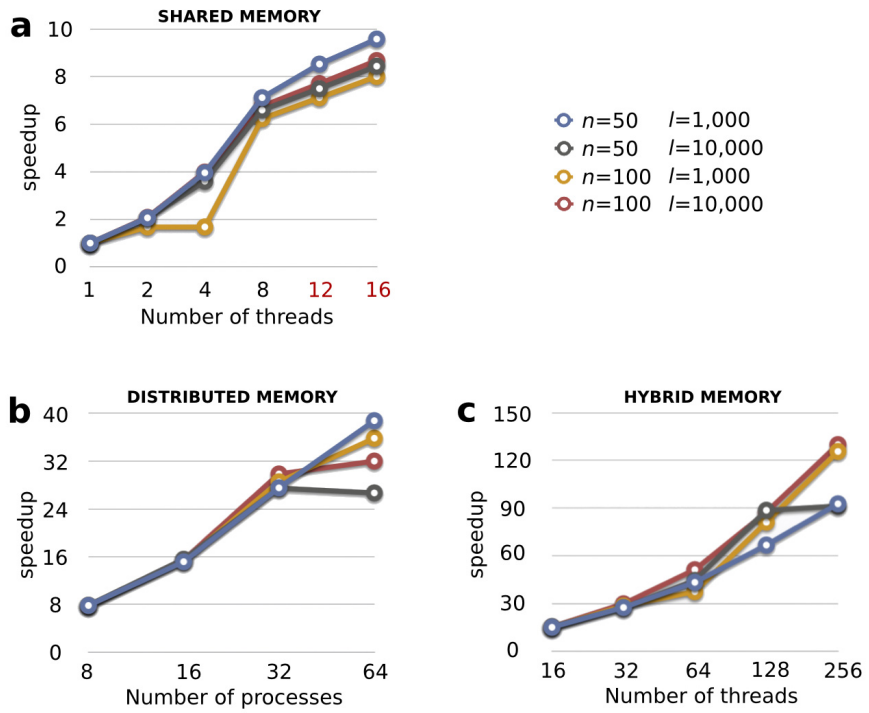

Figure S4. Scalability of jModelTest with simulated data. The speedups reported are for (a) Shared memory (red numbers in the x-axis indicate hyperthreading), (b) Distributed memory and (c) Hybrid memory version. *n* is number of taxa and *l* is the alignment length.

Nature Methods: doi:10.1038/nmeth.2109

References

1. Tavaré, S., in *Some mathematical questions in biology - DNA sequence analysis*, edited by R. M. Miura (American Mathematical Society, Providence, RI, 1986), Vol. 17, pp. 57-86.
2. Goldman, N., *J. Mol. Evol.* **36**, 182-198 (1993).
3. Robinson, D. F. & Foulds, L. R., *Math. Biosci.* **53**, 131-147 (1981).
4. Jordan, G. E. & Piel, W. H., *Bioinformatics* **24**, 1641-1642 (2008).
5. Jukes, T. H. & Cantor, C. R., in *Mammalian Protein Metabolism*, edited by H. M. Munro (Academic Press, New York, NY, 1969), pp. 21-132.
6. Zharkikh, A., *J. Mol. Evol.* **39**, 315-329 (1994).
7. Lanfear, R., Calcott, B., Ho, S. Y. & Guindon, S., *Mol. Biol. Evol.* **29**, 1695-1701 (2012).
8. R Development Core Team, *R Foundation for Statistical Computing* (2008).
9. Rambaut, A. & Grassly, N. C., *Comput. Appl. Biosciences* **13**, 235-238 (1997).
10. Paradis, E., Claude, J. & Strimmer, K., *Bioinformatics* **20**, 289-290 (2004).
11. Rokas, A., Williams, B. L., King, N. & Carroll, S. B., *Nature* **425**, 798-804 (2003).
12. Guindon, S. & Gascuel, O., *Syst. Biol.* **52**, 696-704 (2003).

Nature Methods: doi:10.1038/nmeth.2109

Chapter 6

PartitionTest: The Impact of Partitioning on Phylogenomic Accuracy

The content of this chapter corresponds to the following journal paper:

- Title: The impact of partitioning on phylogenomic accuracy
- Authors: **Diego Darriba** and David Posada
- Journal: bioRxiv
- Year: 2015
- Pages: 26
- DOI: http://dx.doi.org/10.1101/023978

The publication is available at

http://biorxiv.org/content/early/2015/08/19/023978

A copy of the paper is next included.

The impact of partitioning on phylogenomic accuracy

Diego Darriba[1,2] and David Posada[2,*]
[1]Department of Electronics and Systems, University of A Coruña, A Coruña 15071, Spain
[2]Department of Biochemistry, Genetics and Immunology, University of Vigo, Vigo 36310, Spain
***Corresponding author:** E-mail: dposada@uvigo.es
Associate Editor: TBA
04/08/2015

Abstract

Several strategies have been proposed to assign substitution models in phylogenomic datasets, or partitioning. The accuracy of these methods, and most importantly, their impact on phylogenetic estimation has not been thoroughly assessed using computer simulations. We simulated multiple partitioning scenarios to benchmark two *a priori* partitioning schemes (one model for the whole alignment, one model for each data block), and two statistical approaches (hierarchical clustering and greedy) implemented in PartitionFinder and in our new program, PartitionTest. Most methods were able to identify optimal partitioning schemes closely related to the true one. Greedy algorithms identified the true partitioning scheme more frequently than the clustering algorithms, but selected slightly less accurate partitioning schemes and tended to underestimate the number of partitions. PartitionTest was several times faster than PartitionFinder, with equal or better accuracy. Importantly, maximum likelihood phylogenetic inference was very robust to the partitioning scheme. Best-fit partitioning schemes resulted in optimal phylogenetic performance, without appreciable differences compared to the use of the true partitioning scheme. However, accurate trees were also obtained by a "simple" strategy consisting of assigning independent GTR+G models to each data block. On the contrary, leaving the data unpartitioned always diminished the quality of the trees inferred, to a greater or lesser extent depending on the simulated scenario. The analysis of empirical data confirmed these trends, although suggesting a stronger influence of the partitioning scheme. Overall, our results suggests that statistical partitioning, but also the *a priori* assignment of independent GTR+G models, maximize phylogenomic performance.

Key words: partitioning scheme, PartitionTest, PartitionFinder, multilocus, phylogenetics.

Article

Introduction

Statistical inference of phylogenetic trees from sequence alignments requires the use of probabilistic models of molecular evolution (Felsenstein, 2004). It is well established that the choice of a particular model of molecular evolution can change the results of the phylogenetic analysis, and not surprisingly, one of the most active areas of research in phylogenetics in recent years has been the development of more

© The Author 2013. Published by Oxford University Press on behalf of the Society for Molecular Biology and Evolution. All rights reserved.
For permissions, please email: journals.permissions@oup.com

realistic models of nucleotide, codon and amino acid substitution/replacement, together with the implementation of statistical methods for the selection of best-fit models for the data at hand (Joyce and Sullivan, 2005; Posada, 2012).

A key aspect in the development of these models has been the consideration of the heterogeneity of the substitution process among sites. Several mixture models have been proposed that assign each site within a locus a probability of evolving under a given rate (Yang, 1994), substitution pattern (Lartillot and Philippe, 2004; Pagel and Meade, 2004), or both (Wu *et al.*, 2013). In particular, a discrete gamma distribution to consider rate variation among sites (Yang, 1996) is used nowadays in practically any phylogenetic analysis. A different approach to account for the heterogeneity of the substitution process consists of defining *a priori* groups of sites (so called partitions) that evolve under the same substitution model, like for example the set of 1st, 2nd or 3rd codon positions in protein-coding sequence alignments (Shapiro *et al.*, 2006) or distinct protein domains (Zoller *et al.*, 2015).

At the genomic scale the heterogeneity of the substitution process becomes even more apparent that at the single-gene scale, as different genes or genomic regions can have very different functions and evolve under very different constraints (Arbiza *et al.*, 2011). Multilocus substitution models that consider distinct models for different partitions of the data assumed to evolve in an homogeneous fashion have been proposed under the likelihood (Ren *et al.*, 2009; Yang, 1996) and Bayesian (Nylander *et al.*, 2004; Suchard *et al.*, 2003) frameworks. In this case, different loci (or loci by codon position) are typically considered as distinct partitions by default, without further justification. However, a number of empirical studies have demonstrated that different partitioning schemes can affect multilocus phylogenetic inference, including tree topology, branch lengths and nodal support (Brandley *et al.*, 2005; Kainer and Lanfear, 2015; Leavitt *et al.*, 2013; Powell *et al.*, 2013; Ward *et al.*, 2010), with maximal differences occurring when whole datasets are treated as single partitions (i.e., unpartitioned). Using computer simulations, Brown and Lemmon (2007) showed that both over and particularly under-partitioning can lead to inaccurate phylogenetic estimates.

If the partitioning scheme can affect phylogenetic analysis, we should try to identify the best-fit partitioning scheme for the data at hand. In principle, predefined partitioning schemes might not be included within the optimal ones, and some statistical model selection procedure needs to be implemented to justify the choice of a particular partitioning scheme, just as it happens when finding the best-fit model of evolution for a single locus (Posada and Crandall, 2001). Unfortunately, the number of partitioning schemes for a multilocus data set can be huge, ranging from considering that a single model fits

the whole alignment to assigning a different model to each site/region/gene, and until very recently in practice model selection in phylogenomics was restricted to the comparison of a fixed number of alternative partitions in relatively modest data sets, often using Bayes factors (Bao *et al.*, 2007; Brandley *et al.*, 2005; Brown and Lemmon, 2007; Castoe and Parkinson, 2006; Fan *et al.*, 2011; McGuire *et al.*, 2007; Nylander *et al.*, 2004; Pupko *et al.*, 2002);but see Li *et al.* (2008). Opportunely, the release of PartitionFinder (Lanfear *et al.*, 2012, 2014) made a big difference in this regard, facilitating the automatic statistical selection of partitioning schemes for relatively large multilocus data sets. For this task, PartitionFinder uses combinatorial optimization heuristics like clustering and greedy algorithms, building up on previous ideas raised by Li *et al.* (2008). Also, Wu *et al.* (2013) recently described a sophisticated Bayesian approach for the identification of optimal partitioning scheme, but its heavy computational requirements seem to have prevented its general use. While automated statistical model choice procedures have been shown to result in partitioning schemes with a better fit in real data, often resulting in distinct tree topologies when compared to unpartitioned schemes (Kainer and Lanfear, 2015; Wu *et al.*, 2013), the accuracy of these inferences has not been thoroughly assessed. In order to fill this gap we present here a computer simulation study designed to evaluate (i) the precision of the best-fit multilocus partitioning schemes identified by PartitionFinder and by a new tool for multilocus model selection developed by us, called PartitionTest, and (ii) the accuracy of the phylogenetic trees derived from best-fit and *a priori* partitioning schemes. In this article we evaluate the accuracy of PartitionTest and PartitionFinder under different conditions representing biologically realistic scenarios, including rate variation among loci and lineages, non-homogenous data blocks, and large data sets. In addition, we also analyze some of the real datasets previously used in the evaluation of PartitionFinder. Our results suggest that best-fit partitioning schemes can lead to accurate trees, but also that the *a priori* assignment of independent GTR+G models to each locus performs equally well.

Results

Simulation 1: multi-gene phylogenetics

The greedy strategy implemented in PartitionTest (PT-G) recovered most often the true partitioning scheme (PPR=0.305), followed by the greedy strategy implemented in PartitionFinder (PF-G) (PPR=0.255) and the hierarchical clustering implemented in PartitionTest (PT-C) (PPR= 0.200) (table 1). The hierarchical clustering implemented in PartitionFinder (PF-C) did much worse (PPR=0.013). PT-C, PT-G, PF-G recovered accurate partitioning schemes, with RI (Rand, 1971) values above 0.93. PF-C performed

clearly worse ($RI = 0.852$), also evident from its low ARI (Hubert and Arabie, 1985) values. In general, the hierarchical clustering algorithms overestimated the number of partitions while the greedy algorithms underestimated it. The hierarchical clustering algorithms were several times faster than the greedy algorithms. Overall, PartitionTest was on average 2.6 and 1.5 times faster finding the optimal partition than PartitionFinder, for the greedy and hierarchical clustering algorithms, respectively.

Most strategies performed also well recovering the exact true topology (PTR, average perfect topology recovery = $0.820-0.890$), in particular when using FT-C. The largest differences were observed when a single partition was assumed to underlie the data (K=1), which resulted in an PTR of 0.787. The average RF distances to the true topologies were very small ($RF = 0.007-0.013$) except when K=1, which performed worse ($RF = 0.018$) (table 1). The average number of distinct topologies per replicate across methods was 1.31. Regarding the branch lengths, PT-C, PT-G, PF-G performed as well as using the true partitioning scheme (K=T), while PF-G, K=N, and especially K=1, did worse.

Simulation 2: mosaic data blocks

In this case, where sites inside the simulated data blocks evolved under two different models, there is not a true partitioning scheme so only the accuracy of the trees inferred from the selected partitioning scheme was evaluated. The different strategies did well recovering the exact true topology ($PTR \geq 0.827$), although K=1 did slightly worse ($PTR = 0.787$) (table 2). The average RF distances were larger than in the previous simulation but still reasonably small ($RF = 0.012-0.014$), with K=1 doing slightly worse again (RF=0.018). The average number of distinct topologies per replicate across methods was 1.02. Branch lengths estimate were quite accurate ($BS = 0.014-0.020$), with the greedy algorithms performing best. In this simulation PartitionTest was on average 2.1 and 2.0 times faster finding the optimal partition than PartitionFinder, for the greedy and hierarchical clustering algorithms, respectively.

Simulation 3: large-scale phylogenomic study

For large data sets (500,000-1,500,00 bp) the greedy algorithms can take very long, and only the hierarchical clustering algorithms were evaluated. In fact, even in this case PartitionFinder was not able to evaluate 20 out of the 200 replicates due to execution errors, while only 1 replicate failed for PartitionTest. All the comparisons in table 3 refer to the 180 replicates in common. The clustering algorithm implemented in PartitionTest ($PPR = 0.056$; $RI = 0.989$) was more accurate than its analog in PartitionFinder ($PPR = 0.011$; $RI = 0.846$) finding the true partitioning scheme,

Table 1. Partitioning and phylogenetic accuracy for Simulation 1 (multi-gene phylogenetics).

		K=1	K=T	K=N	PT-C	PT-G	PF-C	PF-G
Partitioning accuracy	**PPR**	N/A	N/A	N/A	0.2	0.305	0.013	0.255
	RI	N/A	N/A	N/A	0.97	0.931	0.852	0.951
	ARI	N/A	N/A	N/A	0.775	0.696	0.031	0.771
	Kdiff	N/A	N/A	N/A	2.011	-1.71	13.682	-1.773
	Kmse	N/A	N/A	N/A	29.294	5.855	297.321	5.761
Phylogenetic accuracy	**PTR**	0.787	0.89	0.89	0.892	0.842	0.885	0.82
	RF	0.018	0.007	0.007	0.007	0.012	0.007	0.013
	BS	0.019	0.006	0.01	0.007	0.006	0.011	0.006
Average Run Time		N/A	N/A	N/A	01:20:25	05:25:50	01:59:00	14:31:20

NOTE.— Different partitioning strategies were evaluated: a single partition (K=1), the "true" partitioning scheme (K=T), each data block as a GTR+G partition (K=N), PartitionTest using the Hierarchical Clustering (PT-C) and Greedy (PT-G) algorithms, and PartitionFinder using the Hierarchical Clustering (PF-C) and Greedy (PF-G) algorithms (all of them assuming independent branch lengths). The Greedy algorithms were used only for simulation replicates with up to 20 partitions (¿1,000 replicates). The accuracy of the selected partitions was evaluated by the number of times the exact true partitioning scheme was identified (PPR = Perfect Partitioning Recovery), the Rand index (RI) and the adjusted Rand index (ARI). The accuracy of the RAxML trees inferred from the selected partitions was evaluated with the average Robinson-Foulds distance (RF) (scaled per branch), a measurement of the number of times the exact true topology was estimated (PTR = Perfect Topology Recovery), and the average Branch Score difference (BS) (scaled per branch). The average time required to identify the optimal partitioning scheme (Average Run time) was measured in hours, minutes and seconds.

Table 2. Phylogenetic accuracy for Simulation 2 (mosaic data blocks).

		K=1	K=N	PT-C	PT-G	PF-C	PF-G
Phylogenetic accuracy	**PTR**	0.787	0.827	0.829	0.856	0.828	0.841
	RF	0.018	0.014	0.014	0.012	0.014	0.012
	BS	0.019	0.020	0.019	0.016	0.020	0.014
Average Run Time		N/A	N/A	01:03:15	03:10:47	02:13:16	06:15:50

NOTE.—For further explanations please refer to table 1.

and 6.2 times faster on average (table 3). Both algorithms overestimated the true number of partitions, specially in the case of PF-C. The K=T and K=N *a priori* strategies always recovered the true topology, while K=1 failed in a few occasions. The average number of distinct topologies per replicate across methods was 1.05. Using the true partitioning scheme (K=T) resulted in very accurate branch lengths. The phylogenetic accuracy of PartitionTest and PartitionFinder was also very high, with the former providing slightly better branch length estimates and the latter finding the true topology in one additional replicate. Overall, PartitionTest was ∼7 times faster than PartitionFinder.

Simulation 4: rate variation

In the presence of rate variation among lineages and partitions the true partitioning scheme was never found ($PPR=0$), although the RI scores were still high ($RI=0.932-0.953$) (table 4). PartitionTest was slightly more accurate than PartitionFinder (∼0.95 vs ∼0.93), and the accuracy of the best-fit partitioning schemes did not change when assuming independent vs. proportional branch lengths. The ARI values for PartitionFinder were low or very low. Both PartitionTest and PartitionFinder overestimated the true number of partitions, specially in the latter case. The different strategies showed very

Table 3. Partitioning and phylogenetic accuracy for Simulation 3 (phylogenomics).

		K=1	K=T	K=N	PT-C	PF-C
Partitioning accuracy	**PPR**	N/A	N/A	N/A	0.056	0.011
	RI	N/A	N/A	N/A	0.989	0.846
	ARI	N/A	N/A	N/A	0.864	0.003
	Kdiff	N/A	N/A	N/A	60.112	374.855
	Kmse	N/A	N/A	N/A	9667.229	337010.9
Phylogenetic accuracy	**PTR**	0.979	1.000	1.000	0.989	0.994
	RF	0.004	0.000	0.000	0.003	0.003
	BS	0.097	0.003	0.021	0.044	0.054
Average Run Time		N/A	N/A	N/A	06:55:27	42:49:51

NOTE.—For further explanations please refer to table 1.

similar phylogenetic accuracy, with K=1 doing slightly worse. The true topology was found in most occasions (PTR=0.895−0.907 for most strategies except for K=1, with PTR=0.839), with small RF distances (0.005 for all strategies except PT-C-p and K=1, with RF=0.009) and similar BS (∼0.428 for all strategies except PT-C-p and K=1, with BS=0.483). The average number of distinct topologies per replicate across methods was 1.25. Note that the branch length estimates were much worse than in previous scenarios in which the simulated branch lengths were the same across partitions. As expected, assuming proportional branch lengths resulted in faster run times. Overall, PartitionTest was ∼4 times faster than PartitionFinder.

Simulation 5: the effect of the likelihood optimization threshold

Changing the ML optimization threshold did not have a noticeable impact on the final inferences but dramatically influenced the running times. Higher epsilon thresholds (i.e., less thorough optimization) did not seem to influence much the resulting optimal partitioning schemes (figure 1a-c) or the resulting trees (with identical inferred topologies and very similar branch length estimates). However, the partitioning search algorithm was up to 4 times faster on average in this case. (figure 1d).

Analysis of real data

The optimal partitioning schemes identified in the real datasets were often different depending on the exact implementation (program and method) used, and in most cases without particularly obvious trends. With some exceptions, the assumption of proportional branch lengths across partitions resulted in more partitions in the optimal partitioning scheme (table S1). The number of model parameters in the optimal partitioning schemes was very variable. The greedy algorithms resulted in more or less partitions in the optimal partitioning scheme than the clustering algorithms depending on the data set. To make a legit comparison of the optimal BIC (Schwarz, 1978) scores

Table 4. Partitioning and phylogenetic accuracy for Simulation 4 (rate variation).

		K=1	K=T	K=N	PT-C	PT-G	PT-C-p	PT-G-p	PF-C	PF-G	PF-C-p	PF-G-p
Partitioning accuracy	PPR	N/A	N/A	N/A	0	0	0	0	0	0	0	0
	RI	N/A	N/A	N/A	0.952	0.91	0.953	0.906	0.932	0.908	0.935	0.9
	ARI	N/A	N/A	N/A	0.539	0.529	0.441	0.506	0.007	0.527	0.005	0.455
	Kdiff	N/A	N/A	N/A	7.239	-15.564	9.677	-12.141	17.331	-17.638	18.826	-11.308
	Kmse	N/A	N/A	N/A	73.269	307.855	120.923	201.002	378.886	386.877	428.989	179.385
Phylogenetic accuracy	PTR	0.839	0.907	0.904	0.895	0.866	0.903	0.888	0.904	0.865	0.904	0.888
	RF	0.009	0.005	0.005	0.005	0.007	0.009	0.005	0.005	0.007	0.005	0.005
	BS	0.483	0.429	0.428	0.428	0.43	0.483	0.431	0.428	0.431	0.428	0.43
Average Run Time		N/A	N/A	N/A	00:19:49	07:16:42	00:15:15	01:22:34	01:26:07	10:38:39	01:01:55	06:07:45

NOTE.—PT-C-p and PF-C-p assume that branch lengths are proportional across partitions. For further explanations please refer to table 1.

found by the different algorithms we recomputed all the BIC scores in RAxML (Stamatakis, 2006) assuming proportional branch lengths (BIC*). No significant or consistent differences were observed. Regarding running times, the hierarchical clustering algorithms were clearly faster than the greedy algorithms, while assuming proportional branch lengths further reduced the computation time. On average, PartitionTest was 2.5 times faster than PartitionFinder. The optimal partitioning schemes found by the different algorithms were often quite distinct (table S2), being most similar in general when the same algorithm but a different program was used (e.g., PartitionTest greedy vs. PartitionFinder greedy). The assumption of proportional/independent branch lengths across partitions often resulted in quite different partitioning schemes, with a slightly bigger influence than the program or algorithm used (table S3).

The ML trees estimated under the best-fit partitioning schemes found by the different methods were more or less distinct depending on the specific data set, with RF values ranging from 0 to 0.37. For data sets like *Endicott* or *Li*, the topological differences were highest, in particular regarding the assumption of proportional/independent branch lengths across partitions, which had the most noticeable effect across all data sets. For data sets like *Fong* all trees estimates were very similar, independently of the partitioning selection strategy. The branch scores were very low in practically every case, suggesting in principle that branch length estimates were not affected by the partitioning strategy, although we should note that in some cases the tree length was very small –like for *Endicott*– preventing large BS scores.

In addition, we also compared the ML trees found under the optimal partitioning schemes or under *a priori* partitioning scheme with a single partition (K=1), against the ML trees inferred using the K=N strategy (each partition assumed to evolve under an independent GTR+G model) (table 5). Again, the different strategies often resulted in different trees, except for the *Fong* data

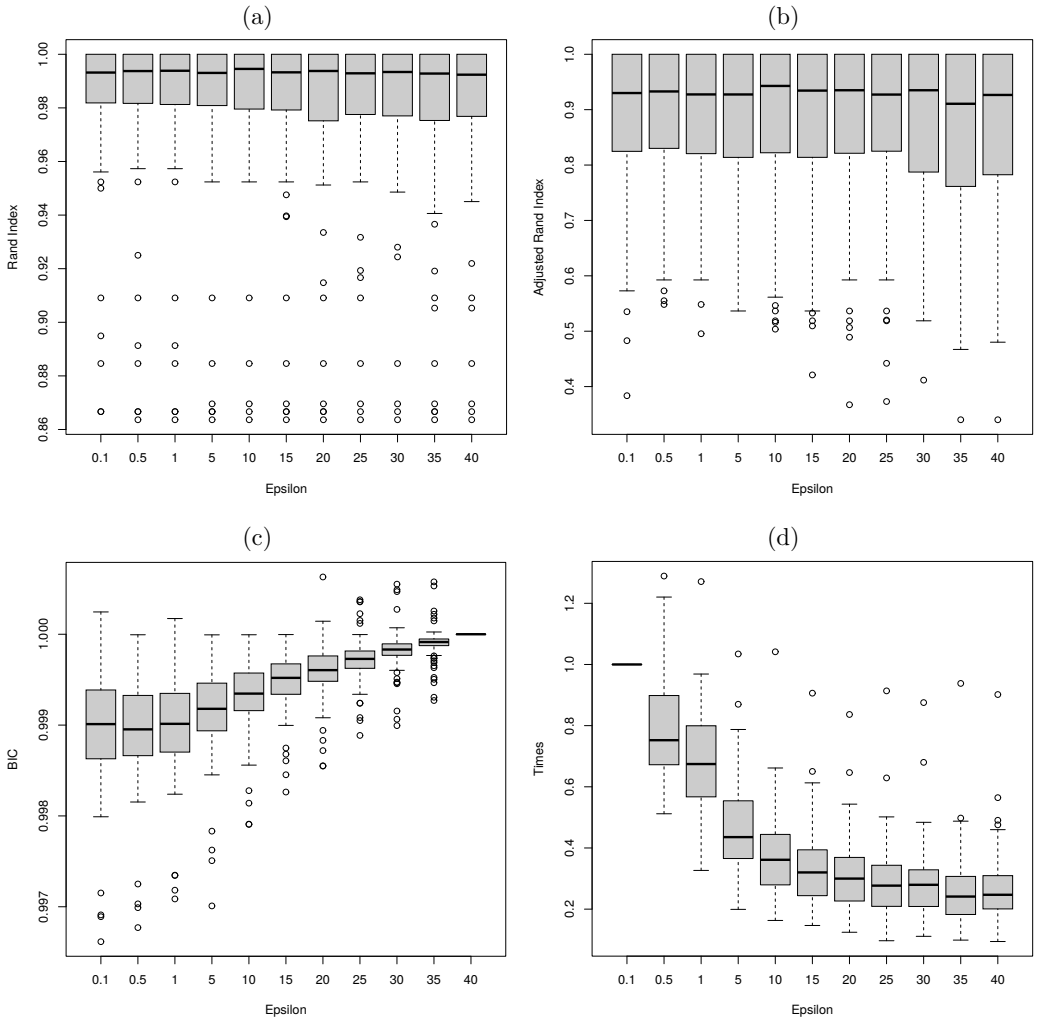

FIG. 1. Partitioning sensitivity and running times as a function of the optimization threshold epsilon. (a) Rand Index. (b) Adjusted Rand Index. (c) Relative BIC scores (normalized to epsilon = 40) (d) Relative execution times (normalized to epsilon = 0.1).

set, where all but one of the strategies resulted in the same topology.

Discussion

Identifying optimal partitioning schemes

Identifying the optimal partitioning scheme is not an easy task, but the different selection strategies studied here seem to perform quite well. In our simulations the exact true partitioning scheme was recovered up to 30% of the time when the number of data blocks was relatively small, and up to 5% of the time when this number was larger, which is still a remarkable result given the vast number of potential solutions. In the presence of rate variation among lineages and partitions the problem becomes much harder and the true

Table 5. Comparison of the RAxML trees inferred under the optimal partitioning schemes found by different strategies in the 6 empirical data sets in the study.

RF scores	Endicott	Fong	Hackett	Kaffenberger	Li	Wainwright
K=1	0.29 [tree A]	0.00 [tree A]	0.08 [tree A]	0.13 [tree A]	0.17 [tree A]	0.09 [tree A]
PT-C	0.29 [tree C]	0.00 [tree A]	0.08 [tree A]	0.19 [tree B]	0.03 [tree C]	0.09 [tree A]
PT-G	0.29 [tree A]	0.07 [tree B]	0.08 [tree C]	0.07 [tree D]	0.03 [tree E]	0.05 [tree D]
PT-C-p	0.36 [tree B]	0.00 [tree A]	0.11 [tree B]	0.19 [tree B]	0.24 [tree B]	0.12 [tree B]
PT-G-p	0.05 [tree D]	0.00 [tree A]	0.11 [tree B]	0.05 [tree C]	0.00 [tree D]	0.01 [tree C]
PF-C	0.29 [tree C]	0.00 [tree A]	0.08 [tree A]	0.19 [tree B]	0.17 [tree A]	0.09 [tree A]
PF-G	0.29 [tree A]	0.00 [tree A]	0.04 [tree F]	0.07 [tree D]	0.07 [tree G]	0.07 [tree G]
PF-C-p	0.12 [tree E]	0.00 [tree A]	0.02 [tree D]	0.19 [tree B]	0.01 [tree F]	0.03 [tree E]
PF-G-p	0.05 [tree F]	0.00 [tree A]	0.02 [tree E]	0.13 [tree A]	0.01 [tree F]	0.00 [tree F]
Num. different topologies	6/9	2/9	6/9	4/9	7/9	7/9

NOTE.—Cells contain the normalized RF scores against the RAxML trees assuming and independent GTR+G model per data bock (K=N). Letters in brackets indicate the different topologies found.

partitioning scheme was never found. Still, most methods were able to identify most of the time and in most conditions optimal partitioning schemes closely related to the true partitioning scheme, as judged by the generally high Rand Index (RI) scores. Now, we should consider that the RI works a little bit different than what we might intuitively think. This index considers pairs of data blocks that belong to the same partition in both partitioning schemes, but also those belong to different partitions in the two schemes being compared. In contrast, our perception might be that a similarity measure between two partitioning schemes should count only those data blocks belong to the same partition in both schemes. Thus, some implementations like the clustering algorithm of PartitionFinder, that significantly overestimates the number of partitions, can still display high RI scores because many pairs of data blocks will belong to different partitions in both partitioning schemes. However, in this case the Adjusted RI scores were very low, highlighting the problem. Also, note that the RI does not consider the particular substitution models assigned to each partition. Hence, as long as two data blocks are grouped together, they are assumed to belong to the same partition, even if the best-fit models assigned to their partitions are different. The performance of the greedy and clustering algorithms was slightly different. Greedy algorithms tended to underestimate the number of partitions while the hierarchical clustering algorithms usually overestimated it. On average, greedy algorithms identified the true partitioning scheme more frequently than the clustering ones, but selected slightly less accurate partitioning schemes. Nevertheless, the computational complexity of the hierarchical clustering algorithms is much lower than that of the greedy algorithms (linear versus quadratic), so they were several times faster. Even though one might think of the schemes evaluated by the clustering algorithms as strict subsets of those evaluated by the greedy algorithm, this is not

necessary the case. In a particular iteration where the initial partitioning scheme is the same for both algorithm it is true that hierarchical clustering inspects a subset of the candidate partitioning schemes in the greedy algorithm. However, the selected partitioning in that iteration scheme might not be the same for both algorithms, and hence from there they can take different paths that might result in different probabilities of getting stuck in local maxima. Therefore, the greedy algorithms do not necessarily reach always a better partitioning scheme than the clustering algorithms.

Effect of partitioning on phylogenetic accuracy

In the simulations the different partitioning methods resulted in the inference of the same ML tree topology, with the only exception of the single partition strategy, which led to a different topology in some cases. Phylogenetic accuracy was quite high overall, even under rate variation among lineages and/or among partitions, or when there was obligate model misspecification (i.e., with mosaic data blocks). Although the greedy and hierarchical clustering strategies did not return the true partitioning scheme in most occasions, they still resulted in practically the same trees as those obtained under the true partitioning scheme. Only when a single partition was assumed *a priori* (i.e., the data was left unpartitioned), phylogenetic accuracy dropped down to some extent, up to 10% when the number of data blocks was not very large. This is in concordance with a previous simulation study that suggested that underpartitioning could negatively affect phylogenetic estimates under a Bayesian framework (Brown and Lemmon, 2007). However, while in the Bayesian case overly complex partitioning schemes also had some effect on posterior probabilities, in our simulations the ML trees did not seem to be much sensitive to overparameterization, an observation already made with real data by Li *et al.* (2008). In general, it is well known that ML phylogenetic estimation can be very robust to model misspecification when the phylogenetic problem is not too complex (e.g., Sullivan and Joyce, 2005) and this might at least partially explain why in the simulations ML phylogenetic estimation seems quite robust also to the partitioning scheme chosen and subsequent model assignment, despite for example having introduced branch length variation in the simulated trees or used mosaic data blocks.

Remarkably, the *a priori* assigning of independent GTR+G model to each data block led to very similar trees than the greedy and hierarchical clustering strategies, which advocates this as a very convenient strategy for the analysis of phylogenomic datasets. An alternative *a priori* K=N option not evaluated here might have been the independent assignment of best-fit models, for example identified with jModelTest (Darriba *et al.*, 2012), to each data block. While this is an

obvious strategy, it was not explored here mainly because it cannot be currently implemented in RAxML, difficulting then a fair comparison among approaches. In any case, this strategy would require much more computation than the assignment of independent GTR+G models to each data block, which already results in optimal performance –practically the same as when using the true partitioning scheme.

Phylogenetic accuracy was almost perfect when the number of data blocks was very large, even for the single partition case. This is the expected behaviour because in our simulations all the data blocks were evolved under the same tree, so there was no phylogenetic incongruence among the different partitions. In real life phenomena like incomplete lineage sorting, gene duplication and loss and horizontal gene transfer can lead to a different scenario in which different partitions evolve under different gene trees, embedded within a single species tree (e.g., Martins *et al.* 2014). Thus, further studies could focus on evaluating the impact of partitioning on the inference of species trees.

Empirical data analysis

The analysis of real data can be very helpful to show the relative fit of the different partitioning schemes and/or the congruence of the different phylogenetic estimates derived from them, beyond the simplicity of simulated data. On the other hand, in this case neither true partitioning scheme nor the true phylogeny is known, and accuracy cannot be directly measured. However, in our analyses of the six empirical data sets we saw more topological variation than in the simulations. In this case the partitioning selection strategy had a stronger effect than in the simulations, and the final tree estimates varied more depending on the method chosen. Nevertheless, this was not true for every dataset, and in all cases there was at least one selection strategy that resulted in the same tree as the unpartitioned scenario. These results are in agreement with previous empirical studies in which different partitioning schemes sometimes resulted in different trees, but also where the main differences were also observed when the data was left unpartitioned (Brandley *et al.*, 2005; Kainer and Lanfear, 2015; Leavitt *et al.*, 2013; Powell *et al.*, 2013; Ward *et al.*, 2010). For example, Kainer and Lanfear analyzed 34 data sets with different partitioning strategies that half or more of the time resulted in different trees, albeit the differences among them were not significant, with average RF distances smaller than 10%, except for the unpartitioned case, which implied significant differences. The same was true for branch lengths and bootstrap values, were only the use of a single partition made a difference in some cases..

Alternative partitioning approaches

The definition of homogeneous data blocks can be a problem under certain circumstances, like in the case of non-coding regions, or when there is significant heterogeneity at a local scale. However, in our (simple) simulation of non-homogeneous data blocks, phylogenetic accuracy was still reasonably high. Wu *et al.* (2013) described an elegant Bayesian framework in which the partitioning scheme is treated as a random variable using a Dirichlet process prior (DPP). This method is able to simultaneously identify the number of partitions, their best-fitting models, and assign sites to each one of them. While this approach is certainly much more flexible than previous strategies, explicitly considers the uncertainty associated with partitioning and improves model fit, it has not been demonstrated yet to lead to more accurate trees. Unfortunately, its heavy computational requirements, and the restriction of only sites being the units if the assignment seem to have limited for now its widespread application to real data. Indeed, it might be very interesting to see a DPP method -in fact a special case of the one just described- that works with user-defined data blocks. Very recently, Frandsen *et al.* (2015) introduced a promising algorithm for phylogenetic partitioning that uses rates of evolution instead of substitution patterns, also avoiding the need for an arbitrary delimitation of data blocks. While this method can increase model fit, again its advantage over data block methods like those studied here has not been demonstrated in terms of phylogenetic accuracy.

PartitionTest vs. PartitionFinder

In the majority of the conditions explored in the simulations PartitionTest was slightly more accurate than PartitionFinder both regarding the identification of optimal partitioning schemes and tree estimation. Although these differences were small, they were consistent. Importantly, PartitionTest is much faster than PartitionFinder, between 1.5 and 7 times faster, in particular with large data sets. PartitionFinder is implemented in Python and delegates the phylogenetic calculations on external third party software, like PhyML (Guindon and Gascuel, 2003) or RAxML. On the other hand, PartitionTest is implemented in C++ and keeps a finer control over the phylogenetic calculations through the use use of the PLL (Flouri *et al.*, 2015).

Conclusions

Several strategies for the selection of best-fit partitioning schemes have been recently introduced for the phylogenetic analysis of multilocus data sets. Here we evaluated different partitioning approaches using comprehensive computer simulations and real data. We conclude that hierarchical clustering algorithms should be preferred over existing greedy algorithms for the selection of best-fit partitioning scheme,

because under the conditions explored, they were much faster with practically the same accuracy. However, our simulations also suggest that ML phylogenetic inference is quite robust to the partitioning scheme, at least as far as single models are assigned to the final partitions using a statistical procedure. In this case, any reasonable partitioning scheme seems to perform reasonably well, including the *a priori* assignment of GTR+G model to each data block. To be on the safe side, leaving the data unpartitioned should be avoided.

Materials and methods

Partitions and partitioning schemes

Let us consider set of aligned nucleotide or amino acid sequences of any length (the "data"). Following Lanfear *et al.* (2012), we define "data block" as a set of alignment sites that are assumed to evolve in a similar way, normally user-defined. Typical examples of data blocks in phylogenetics are amplified gene fragments, gene families, assembled loci from NGS reads, introns/exons, or sets of codon positions. A "partition" ("subset" in Lanfear *et al.* (2012) will be a set of one or more particular data blocks. A partition can be made of, for example, a single gene fragment, family or locus, multiple gene fragments, families or loci, or consist of the set of all 1st and 2nd codon positions in an alignment. Finally, a set of non-overlapping partitions that cover the whole alignment will be called a "partitioning scheme". The partitioning problem consists of, given a predefined set of data blocks, finding the optimal partitioning scheme for a given alignment. In our case, we want to optimize the partitions with regard to the assignment of substitution models. Note that a "model" here will be a particular model of nucleotide substitution or amino acid replacement together with parameter values. That is, K80 ($ti/tv=2$) would be a different model than K80+G or JC, but also than K80 ($ti/tv=8$). For example, if we have a multilocus alignment with, say 100 concatenated genes, our aim is to find out whether we should use 100 different substitution models, just one (all genes evolving under exactly the same model), or something in between, in which case we would need to assign 2-99 models to the 100 genes. Note that there are two related questions here, which are: (i) how many different models should we use (i.e., the number of partitions), and (ii) which partitions evolve under which model (i.e., the partitioning scheme). In general, given n initial data blocks, the number of possible partitioning schemes, $B(n)$, is given by the Bell numbers, which are the sum from 1 to n of the Stirling numbers of the second kind, $S(n,k)$, where k is the number of partitions:

$$B_n = \sum_{k=1}^{n} S(n,k) \tag{1a}$$

$$\begin{aligned} S(n,k) &= \sum_{j=1}^{k} (-1)^{k-j} \frac{j^{n-1}}{(j-1)!(k-j)!} \\ &= \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \end{aligned} \tag{1b}$$

The number of partitioning schemes grows very quickly. For example, for 4 data blocks there

are 15 different partitioning schemes, but for 20 there are already 5.8×10^{12}. Clearly, finding the optimal partitioning scheme and assigned models is a very intensive task, and rapidly becomes computationally unfeasible.

Selecting optimal partitioning schemes

In order to select optimal partitioning schemes at the phylogenomic level, we have implemented *de novo* (in the program PartitionTest, see below) a set of heuristic algorithms that are very similar to those already available in the software PartitionFinder. The main steps in these algorithms are:

1. Estimate an initial tree.
2. Define a set of candidate partitioning schemes.
3. Select the best-fit substitution/replacement model for each partition.
4. Compute the score of each partitioning scheme and select the best one accordingly.
5. Return to step 2 until there is no score improvement or until the current partitioning scheme includes a single partition.

Step 1. Initial tree estimate . The starting tree topology can be user-defined or estimated using a particular phylogenetic method.

Step 2. Define a set of candidate partitioning schemes . The initial partitioning scheme is the set of data blocks defined by the user. For each iteration, new partitioning schemes are proposed as potentially better candidates, given the currently best partitioning scheme, and using a greedy or a hierarchical clustering algorithm.

The greedy algorithm defines $\binom{k}{2}$ candidate partitioning schemes of size $(k-1)$ by merging all possible pairs of partitions, where k is the number of partitions in the current best partitioning scheme. This algorithm is identical to the greedy algorithm implemented in PartitionFinder. Its computational complexity is $\mathcal{O}(n^2)$, so if the number of initial partitions (n) is large the required computational time will be considerable.

The hierarchical clustering algorithm defines r candidate partitioning schemes of size $(k-1)$ by merging the r closest pairs of partitions, given a matrix of pairwise distances between partitions ($D(m_i,m_j)$, see below). The parameter r is defined by the user. A strict hierarchical clustering (i.e., $r=1$) will evaluate a maximum of n candidate partitioning schemes. Although r can be defined in the range $(0,\infty)$, only a maximum of $n(n-1)/2$ new candidate partitioning schemes can be proposed, so if $r \geq n(n-1)/2$, this algorithm will behave exactly as the greedy algorithm. The pairwise distances between partitions i and j are calculated using the maximum likelihood estiamtes (MLEs) of the best-fit substitution

model parameters for each partition:

$$D(m_i,m_j)=\omega_r d(R_iR_j)+\omega_f||F_i-F_j||+\omega_a|\alpha_i-\alpha_j| \quad (2)$$

where $R=\{r_{ac},r_{ag},r_{at},r_{cg},r_{ct},r_{gt}\}$ are the substitution rates, $F=\{f_a,f_c,f_g,f_t\}$ are the base frequencies, and α is the alpha shape for the gamma rate variation among sites (+G). Because the substitution rates are usually estimated relative to each other, we first scale them such that their euclidean distance is minimized:

$$d(R_i,R_j)=\sum_{n=1}^{6}(\lambda R_{i,n}-R_{j,n})^2 \quad (3)$$

Deriving this function we obtain:

$$\frac{\delta d(R_i,R_j)}{\delta\lambda}=\lambda\sum_{n=1}^{6}(R_{i,n}^2)-\sum_{n=1}^{6}(R_{i,n}R_{j,n}) \quad (4)$$

whose minimum is located at:

$$\lambda=\frac{\sum_{n=1}^{6}(R_{i,n}R_{j,n})}{\sum_{n=1}^{6}(R_{i,n}^2)} \quad (5)$$

We include different weights (ω_r, ω_f, ω_a and ω_p) for each part of the distance formula, that the user can specify. By default these values are set to those that maximized accuracy (finding the true partitioning scheme) in pilot simulations. Note that the hierarchical clustering algorithm implemented PartitionFinder specifies a slightly different formulae than PartitionTest for the distance calculation. The computational complexity of the hierarchical clustering algorithm is $\mathcal{O}(rn)$, so the required computational time should be affordable even for very large data sets (e.g., with $>1{,}000$ initial partitions).

Step 3. Select a substitution model for each partition . For each partition, likelihood scores are calculated given a tree and a model of substitution/replacement. Best-fit substitution/replacement models with associated parameter values are then identified using the Akaike Information Criterion (AIC, Akaike, 1973), corrected AIC (AICc, Sugiura, 1978), Bayesian Information Criterion (BIC, Schwarz, 1978) or Decision Theory (DT, Minin *et al.*, 2003). Alternatively, a fixed substitution/replacement model can be assigned for every partition, with unlinked parameter values that are independently optimized. For the likelihood calculations the tree topology can be fixed (i.e., the starting tree topology is used for every calculation) or reoptimized using maximum likelihood for each partition. Branch lengths across partitions can be assumed to be independent for each partition (unlinked) or proportional among partitions (linked). In the independent model the branch lengths are reoptimized for every new partition. In the proportional model a set of global branch lengths is estimated at the beginning for the whole data set, with a scaling parameter being optimized for every new partition.

Step 4. Compute the score of each partitioning scheme . The score of a partitioning scheme will be calculated in two different ways depending on the occurrence of linked/unlinked branch lengths.

If branch lengths are unlinked across partitions, model parameters and branch lengths are optimized independently for each partition.

Therefore, the BIC score of a partitioning scheme is simply the sum of the individual scores of its partitions:

$$BIC = \sum_{i=0}^{N} (p_i ln(s_i) - 2lnLK_i) \quad (6)$$

where p_i is the number of parameters, s_i is the sample size, and LK_i is the likelihood score of partition i.

However, if branch lengths are linked proportionally across partitions, the score of the partitioning scheme with linked parameters is computed as follows:

$$BIC = \left[p^* + \sum_{i=0}^{N} (p_i) \right] ln(s) - 2\sum_{i=0}^{N} (lnLK_i) \quad (7)$$

where p_i is the number of parameters of partition i, p^* is the number of parameters globally optimized for the partitioning scheme, and s is the sample size of the entire partitioning scheme.

PartitionTest software

We have implemented the algorithms described above in the program PartitionTest, available from *https://github.com/ddarriba/partitiontest* The greedy search algorithm in PartitionTest is essentially the same as the one implemented in PartitionFinder, but the hierarchical clustering algorithm uses slightly different distances.

PartitionTest makes an intensive use of the Phylogenetic Likelihood Library (PLL) for carrying out all likelihood computations, including tree estimation. The PLL speeds up the calculations considerably. During ML estimation of model parameters and trees with PLL, a parameter ϵ regulates how thorough are the mathematical optimizations. Basically, epsilon is a numerical threshold under which the improvement in likelihood is considered not worthy and the optimization stops. When epsilon is small the optimization is more thorough and takes more time. PartitionTest tests implements several computational strategies to avoid repeated calculations, including checkpoint and restarting capabilities, allowing its use in systems with per-job time restriction, like many High Performance Computing (HPC) clusters. In order to choose the best substitution/replacement model for each partition, PartitionTest considers 22 models of DNA substitution (the +G models in jModelTest2) and 36 empirical models of amino acid replacement (the same as RAxML excluding LG4M and LG4X). All of them assume rate heterogeneity among sites using a discrete gamma distribution with four categories. If desired, PartitionTest is also able to estimate ML trees from the optimal partitioning scheme.

Benchmarking of partitioning algorithms

We devised a set of experiments with simulated and real DNA sequence data to compare different partitioning strategies. The main questions asked were (i) how accurate (close to truth) are the optimal partitions identified by the different algorithm, and (ii) what is the impact of the

different partitioning strategies on phylogenetic accuracy. The different partitioning strategies evaluated were three *a priori* partitioning schemes plus different algorithms implemented in PartitionTest and PartitionFinder:

1. A single partition, or unpartitioned (K=1)
2. One partition for each data block (K=N)
3. The simulated partitioning scheme (K=T)
4. PartitionTest hierarchical clustering with independent branch lengths across partitions (PT-C)
5. PartitionTest greedy with independent branch lengths across partitions (PT-G)
6. PartitionTest hierarchical clustering with proportional branch lengths across partitions (PT-C-p)
7. PartitionTest greedy with proportional branch lengths across partitions (PT-G-p)
8. PartitionFinder greedy with independent branch lengths across partitions (PF-G)
9. PartitionFinder hierarchical clustering with independent branch lengths across partitions (PF-C)
10. PartitionFinder hierarchical clustering with proportional branch lengths across partitions (PF-C-p)
11. PartitionFinder greedy with proportional branch lengths across partitions (PF-G-p)

Strategies 6-7 and 10-11 were only evaluated in Simulation 4 (see below). All the analyses were carried out in a computer with 2 hexa-core Intel Xeon X5675 @ 3.07GHz processors (12 cores) and 50GB of memory, with Hyper-threading disabled. We used a single core per run to facilitate running time comparisons.

Computer simulations

The first four experiments consisted of a series of computer simulations aiming to recreate different biological scenarios, while the last one was designed to assess the sensitivity of the results to the level of parameter optimization (table 6). In our simulations, parameter values were not fixed along a grid but sampled from predefined statistical distributions, allowing us to explore a large parameter space and to carry out *ad hoc* analyses of the results.

- Simulation 1: with a limited number of data blocks, typical of a multi-gene phylogenetic study.
- Simulation 2: with pairs of data blocks merged at random before the analysis. Our intention was to represent an scenario where sites inside data blocks did not evolve in an homogeneous fashion, as assumed by definition. Instead, in this simulation data blocks are mosaics of two distinct evolutionary processes.
- Simulation 3: with a large number of data blocks, typical of a large-scale phylogenomic study.

- Simulation 4: with rate variation among partitions and lineages. In this case the branch lengths for each partition were scaled using two random multipliers. A global multiplier ∼U(0.25, 4) was applied to all branches, while a local multiplier ∼U(0.8, 1.2) was chosen for each branch. For the analysis of the simulated data, we used both the independendent and proportional branch length models.

Simulation 5: here we tested the impact of the optimization threshold epsilon on the resulting partitioning schemes and topologies, in order to find a good compromise between computational time and accuracy.

For each replicate the simulation proceeded as follows:

1. N data blocks were generated according to U[10,50] with variable lengths chosen from U[500,1500].
2. Data blocks were randomly assigned to K partitions, where K ∼U[1,N].
3. Each partition was assigned a random model of nucleotide substitution.
   (a) A model family (M) is chosen from the 22 nucleotide substitution model families ∼U(0,21).
   (b) A model of rate variation was chosen among 4 possibilities ∼U(0,3): no rate variation (M), including a proportion of invariable sites (M+I), including gamma rate variation among sites (+G), and including both a proportion of invariable sites and gamma rate variation among sites (+I+G).
4. Specific model parameter values were chosen from prior distributions.
   (a) Nucleotide frequencies: equal or ∼Dirichlet(1.0,1.0,1.0,1.0).
   (b) Transition/transversion rate: ∼Gamma(2,1) truncated between 2 and 10
   (c) R-matrix parameters ∼Dirichlet(6,16,2,8,20,4) scaled with the last rate (taking free parameters as necessary).
   (d) Proportion of invariable sites ∼Beta(1,3) truncated between 0.2 and 0.8.
   (e) Gamma shape for rate variation among sites ∼Exponential(2) truncated between 0.5 and 5.
5. A random non-ultrametric rooted tree topology with number of taxa ∼U(6,40) and branch lengths ∼Exponential(1,10) was simulated with the function rtree from the ape package (Paradis *et al.*, 2004) in R. The total tree length is scaled so tree length ∼U[2, 12].
6. Each partition was evolved under this tree according to the chosen substitution model

Table 6. Simulation summary. Parameter values were chosen to reflect a range of plausible biological scenarios.

	Sim1	Sim2	Sim3	Sim4	Sim5
N, number of genes	U(10,50)	U(5,25)	1000	U(10,50)	U(5,50)
K, number of partitions	U(1,N)	NA	U(1,N)	U(1,N)	U(1,N)
Gene length	U(500,1500)	U(1000,3000)	U(500,1500)	U(500,1500)	U(500,1500)
Number of taxa	U(6,40)	U(6,40)	U(6,40)	U(6,40)	U(8,140)
Topology	Fixed	Fixed	Fixed	Branch Length multiplier	Fixed
Number of replicates	4,000	4,000	200	1,000	100
Tree length	U(0.5,15)	U(0.5,12)	U(0.5,12)	U(0.5,12)	U(0.5,12)

parameters using INDELible (Fletcher and Yang, 2009), resulting in a multiple sequence alignment.

7. Optimal partitions were identified from this alignment according to the partitioning strategies listed above, using the default settings in each software. See Lanfear *et al.* (2012) for details.

8. A ML tree was estimated from this alignment according to the optimal partitioning schemes identified under each strategy, using RAxML.

Analysis of real data

We also reanalyzed some of the real datasets previously used in the evaluation of PartitionFinder (table 7). As in the simulations, optimal partitioning schemes were selected under the different partitioning strategies evaluated, and used to infer ML trees with RAxML.

Evaluation of partitioning and phylogenetic accuracy

Partitioning accuracy In order to compare the selected partitioning schemes obtained under the different partitioning strategies with the true partitions (simulation), or among themselves (real data), we computed different statistics. We counted how many times the exact true partitioning scheme was identified (PPR = Perfect Partitioning Recovery). We also calculated the Rand Index (Rand, 1971) (RI), a measure of the similarity between two clusterings that is constructed as follows. Given a set of n data blocks $S=\{o_1,\ldots,o_n\}$ and two partitioning schemes of S named X and Y with r and s partitions, respectively, $X=\{X_1,\ldots,X_r\}$ and $Y=\{Y_1,\ldots,Y_s\}$, define the following:

- a, the number of pairs of data blocks in S that are in the same partition in X and in the same partition in Y.
- b, the number of pairs of data blocks in S that are in different partition in X and in different partition in Y.
- c, the number of pairs of data blocks in S that are in the same partition in X and in different partition in Y.
- d, the number of pairs of data blocks in S that are in different partition in X and in the same partition in Y.

Intuitively, $a+b$ can be considered as the number of agreements between X and Y and $c+d$

Table 7. Description of the empirical datasets evaluated in this study.

Short name	Clade	Number of taxa	Sequence length	Number of data blocks	Average number of sites per data block	Reference
Endicott	Humans (Homo sapiens)	179	13857	42	329.92	Endicott and Ho (2008)
Fong	Vertebrates (Vertebrata)	16	25919	168	154.28	Fong et al. (2012)
Hackett	Birds (Aves)	171	52383	168	277.16	Hackett et al. (2008)
Kaffenberger	Frogs (Gephyromantis)	54	6145	27	277.59	Kaffenberger et al. (2012)
Li	Ray-finned fishes (Actubioterygii)	56	7995	30	266.5	Li et al. (2008)
Wainwright	Ray-finned fishes (Acanthomorpha)	188	8439	30	281.3	Wainwright et al. (2012)

as the number of disagreements between X and Y. With these counts in place, the RI is computed as follows:

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{8}$$

The RI is a value between 0 and 1, with 0 indicating that the two partitioning schemes are completely different and 1 that they are identical. In addition, we calculated the adjusted Rand index (Hubert and Arabie, 1985) (ARI), which measures the probability that a given RI was achieved by chance. The ARI can yield negative values if the observed RI is smaller than the expected RI. In this case the overlap between two partitioning schemes X and Y can be summarized in a contingency table where each entry n_{ij} denotes the number of data blocks in common between partition X_i and Y_j, like this:

X/Y	Y_1	Y_2	...	Y_s	Sums
X_1	n_{11}	n_{12}	...	n_{1s}	a_1
X_2	n_{21}	n_{22}	...	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	...	n_{rs}	a_r
Sums	b_1	b_2	...	b_s	

Then the ARI is calculated as:

$$ARI = \frac{RI - exp_{RI}}{max_{RI} - exp_{RI}} \tag{9a}$$

$$max_{RI} = \frac{1}{2}\left(\sum_{i=1}^{r}\binom{a_i}{2} + \sum_{j=1}^{s}\binom{b_j}{2}\right) \tag{9b}$$

$$exp_{RI} = \frac{\sum_{i=1}^{r}\binom{a_i}{2}\sum_{j=1}^{s}\binom{b_j}{2}}{\binom{n}{2}} \tag{9c}$$

where a_i and b_j are values from the contingency table.

We also computed two statistics that reflect if the number of partitions is under or overestimated (Kdiff = average number of true partitions – number of partitions in the optimal partitioning scheme), and the mean square error of this deviation (Kmse).

Phylogenetic accuracy .

In order to compare the inferred ML trees obtained under the different partitioning strategies with the true, generating trees (in the case of computer simulations), or among themselves (in the case of real data), we calculated: (i) how many times the exact true tree topology was identified (PTR = Perfect Topology Recovery), (ii) the Robinson-Foulds metric (RF) (Robinson and Foulds, 1981), that only considers the topology, and (iii) the branch score difference (BS) (Kuhner and Felsenstein, 1994), which takes also into account the branch lengths. In order to compare measurements from trees with different sizes, we scaled both the RF and BS so they were expressed per branch. We consider as outliers those simulation replicates that resulted in any BS difference (per-branch) higher than three. Even if the tree topologies were completely different, such a large BS distance could only be caused by a extremely long average branch length in one of the trees, suggesting an optimization error. This threshold resulted in only less than 1% of the replicates being treated as outliers.

Supplementary Material

Supplementary tables S1–S3 are available at Molecular Biology and Evolution online (http://www.mbe.oxfordjournals.org/).

Acknowledgments

This work was supported by the European Research Council (ERC-2007Stg 203161-PHYGENOM to D.P.) and the Spanish Government (research grant BFU2012-33038 to D.P.).

References

Akaike, H. 1973. Information theory and an extension of the maximum likelihood principle. In *2nd Intl Symp on Information Theory. Budapest (Hungary)*, pages 267–281.

Arbiza, L., Patricio, M., Dopazo, H., and Posada, D. 2011. Genome-wide heterogeneity of nucleotide substitution model fit. *Genome biology and evolution*, 3: 896.

Bao, L., Gu, H., Dunn, K. A., and Bielawski, J. P. 2007. Methods for selecting fixed-effect models for heterogeneous codon evolution, with comments on their application to gene and genome data. *BMC evolutionary biology*, 7(Suppl 1): S5.

Brandley, M. C., Schmitz, A., and Reeder, T. W. 2005. Partitioned Bayesian analyses, partition choice, and the phylogenetic relationships of scincid lizards. *Systematic biology*, 54(3): 373–390.

Brown, J. M. and Lemmon, A. R. 2007. The importance of data partitioning and the utility of bayes factors in bayesian phylogenetics. *Systematic Biology*, 56(4): 643–655.

Castoe, T. A. and Parkinson, C. L. 2006. Bayesian mixed models and the phylogeny of pitvipers (viperidae: Serpentes). *Molecular phylogenetics and evolution*, 39(1): 91–110.

Darriba, D., Taboada, G. L., Doallo, R., and Posada, D. 2012. jModelTest 2: more models, new heuristics and parallel computing. *Nature Methods*, 9(8): 772.

Endicott, P. and Ho, S. Y. 2008. A bayesian evaluation of human mitochondrial substitution rates. *The American Journal of Human Genetics*, 82(4): 895–902.

Fan, Y., Wu, R., Chen, M.-H., Kuo, L., and Lewis, P. O. 2011. Choosing among partition models in bayesian phylogenetics. *Molecular biology and evolution*, 28(1): 523–532.

Felsenstein, J. 2004. *Inferring phylogenies*. Sinauer associates Sunderland.

Fletcher, W. and Yang, Z. 2009. INDELible: A flexible simulator of biological sequence evolution. *Molecular Biology and Evolution*, 26(8): 1879–1888.

Flouri, T., Izquierdo-Carrasco, F., Darriba, D., Aberer, A., Nguyen, L.-T., Minh, B., Von Haeseler, A., and Stamatakis, A. 2015. The phylogenetic likelihood library. *Systematic biology*, 64(2): 356–362.

Fong, J. J., Brown, J. M., Fujita, M. K., and Boussau, B. 2012. A phylogenomic approach to vertebrate phylogeny supports a turtle-archosaur affinity and a possible paraphyletic lissamphibia. *PLoS One*, 7(11): e48990.

Frandsen, P. B., Calcott, B., Mayer, C., and Lanfear, R. 2015. Automatic selection of partitioning schemes for phylogenetic analyses using iterative k-means clustering of site rates. *BMC evolutionary biology*, 15(1): 13.

Guindon, S. and Gascuel, O. 2003. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic biology*, 52(5): 696–704.

Hackett, S. J., Kimball, R. T., Reddy, S., Bowie, R. C., Braun, E. L., Braun, M. J., Chojnowski, J. L., Cox, W. A., Han, K.-L., Harshman, J., *et al.* 2008. A phylogenomic study of birds reveals their evolutionary history. *science*, 320(5884): 1763–1768.

Hubert, L. and Arabie, P. 1985. Comparing partitions. *Journal of classification*, 2(1): 193–218.

Joyce, P. and Sullivan, J. 2005. Model selection in phylogenetics. *Annual Review of Ecology, Evolution, and Systematics*, 36: 445–466.

Kaffenberger, N., Wollenberg, K. C., Köhler, J., Glaw, F., Vieites, D. R., and Vences, M. 2012. Molecular phylogeny and biogeography of malagasy frogs of the genus gephyromantis. *Molecular phylogenetics and evolution*, 62(1): 555–560.

Kainer, D. and Lanfear, R. 2015. The effects of partitioning on phylogenetic inference. *Molecular biology and evolution*, page msv026.

Kuhner, M. K. and Felsenstein, J. 1994. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11(3): 459–468.

Lanfear, R., Calcott, B., Ho, S. Y. W., and Guindon, S. 2012. PartitionFinder: Combined selection of partitioning schemes and substitution models for phylogenetic analyses. *Molecular Biology and Evolution*, 29(6): 1695–1701.

Lanfear, R., Calcott, B., Kainer, D., Mayer, C., and Stamatakis, A. 2014. Selecting optimal partitioning schemes for phylogenomic datasets. *BMC evolutionary biology*, 14(1): 82.

Lartillot, N. and Philippe, H. 2004. A bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. *Molecular biology and evolution*, 21(6): 1095–1109.

Leavitt, J. R., Hiatt, K. D., Whiting, M. F., and Song, H. 2013. Searching for the optimal data partitioning strategy in mitochondrial phylogenomics: A phylogeny of Acridoidea (Insecta: Orthoptera: Caelifera) as a case study. *Molecular Phylogenetics and Evolution*, 67(2): 494–508.

Li, C., Lu, G., and Ortí, G. 2008. Optimal data partitioning and a test case for ray-finned fishes (Actinopterygii) based on ten nuclear loci. *Systematic biology*, 57(4): 519–539.

Martins, L. D. O., Mallo, D., and Posada, D. 2014. A bayesian supertree model for genome-wide species tree reconstruction. *Systematic biology*, page syu082.

McGuire, J. A., Witt, C. C., Altshuler, D. L., and Remsen, J. 2007. Phylogenetic systematics and biogeography of hummingbirds: Bayesian and maximum likelihood analyses of partitioned data and selection of an appropriate partitioning strategy. *Systematic Biology*, 56(5): 837–856.

Minin, V., Abdo, Z., Joyce, P., and Sullivan, J. 2003. Performance-based selection of likelihood models for

phylogeny estimation. *Systematic Biology*, 52(5): 674–683.

Nylander, J. A., Ronquist, F., Huelsenbeck, J. P., and Nieves-Aldrey, J. 2004. Bayesian phylogenetic analysis of combined data. *Systematic biology*, 53(1): 47–67.

Pagel, M. and Meade, A. 2004. A phylogenetic mixture model for detecting pattern-heterogeneity in gene sequence or character-state data. *Systematic Biology*, 53(4): 571–581.

Paradis, E., Claude, J., and Strimmer, K. 2004. Ape: Analyses of phylogenetics and evolution in r language. *Bioinformatics*, 20(2): 289–290.

Posada, D. 2012. Inferring the history of species using many genes. In *FEBS JOURNAL*, volume 279, pages 22–22. WILEY-BLACKWELL 111 RIVER ST, HOBOKEN 07030-5774, NJ USA.

Powell, A. F. L. A., Barker, F. K., and Lanyon, S. M. 2013. Empirical evaluation of partitioning schemes for phylogenetic analyses of mitogenomic data: An avian case study. *Molecular Phylogenetics and Evolution*, 66(1): 69–79.

Pupko, T., Huchon, D., Cao, Y., Okada, N., and Hasegawa, M. 2002. Combining multiple data sets in a likelihood analysis: which models are the best? *Molecular Biology and Evolution*, 19(12): 2294–2307.

Rand, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66: 846–850.

Ren, F., Tanaka, H., and Yang, Z. 2009. A likelihood look at the supermatrix–supertree controversy. *Gene*, 441(1): 119–125.

Robinson, D. and Foulds, L. R. 1981. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1): 131–147.

Schwarz, G. 1978. Estimating the dimension of a model. *Ann Stat*, 6: 461–464.

Shapiro, B., Rambaut, A., and Drummond, A. J. 2006. Choosing appropriate substitution models for the phylogenetic analysis of protein-coding sequences. *Molecular Biology and Evolution*, 23(1): 7–9.

Stamatakis, A. 2006. Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21): 2688–2690.

Suchard, M. A., Kitchen, C. M., Sinsheimer, J. S., and Weiss, R. E. 2003. Hierarchical phylogenetic models for analyzing multipartite sequence data. *Systematic Biology*, 52(5): 649–664.

Sugiura, N. 1978. Further analysts of the data by akaike's information criterion and the finite corrections: Further analysts of the data by akaike's. *Communications in Statistics-Theory and Methods*, 7(1): 13–26.

Wainwright, P. C., Smith, W. L., Price, S. A., Tang, K. L., Sparks, J. S., Ferry, L. A., Kuhn, K. L., Eytan, R. I., and Near, T. J. 2012. The evolution of pharyngognathy: a phylogenetic and functional appraisal of the pharyngeal jaw key innovation in labroid fishes and beyond. *Systematic Biology*, 61(6): 1001–1027.

Ward, P. S., Brady, S. G., Fisher, B. L., and Schultz, T. R. 2010. Phylogeny and biogeography of dolichoderine ants: Effects of data partitioning and relict taxa on historical inference. *Systematic Biology*, 59(3): 342–362.

Wu, C. H., Suchard, M. A., and Drummond, A. J. 2013. Bayesian selection of nucleotide substitution models and their site assignments. *Molecular Biology and Evolution*, 30(3): 669–688.

Yang, Z. 1994. Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: approximate methods. *Journal of Molecular evolution*, 39(3): 306–314.

Yang, Z. 1996. Among-site rate variation and its impact on phylogenetic analyses.

Zoller, S., Boskova, V., and Anisimova, M. 2015. Maximum-likelihood tree estimation using codon substitution models with multiple partitions. *Molecular biology and evolution*, page msv097.

Table S1. Optimal partitioning schemes for the empirical datasets.

Endicott	**PT-C**	**PT-G**	**PT-C-p**	**PT-G-p**	**PF-C**	**PF-G**	**PF-C-p**	**PF-G-p**
Npar	364	364	427	400	364	364	400	409
K	1	1	8	5	1	1	5	6
BIC*	71409	71409	69439	69328	71409	71409	70193	69316
Run Time	00:02:22	00:26:30	00:01:09	00:16:53	00:04:17	00:55:42	00:03:41	00:46:44
Fong								
Npar	497	128	182	164	128	110	218	137
K	52	11	17	15	11	9	21	12
BIC*	285842	283542	284787	283508	293635	284146	285117	283358
Run Time	00:03:04	02:02:24	00:03:16	01:25:42	00:07:03	06:18:12	00:03:51	03:18:30
Hackett								
Npar	348	366	1311	627	348	366	753	555
K	1	3	108	32	1	3	46	24
BIC*	1862900	1844431	1841869	1836829	1862900	1844255	1843320	1837617
Run Time	04:11:04	72:52:40	00:41:17	13:17:33	04:23:35	81:28:25	01:35:34	52:32:22
Kaffenberger								
Npar	132	141	186	204	114	132	168	186
K	3	4	9	11	1	3	7	9
BIC*	130682	129538	129635	128791	128950	130169	131406	128955
Run Time	00:01:48	00:12:34	00:01:20	00:05:35	00:05:06	00:32:28	00:03:05	00:21:43
Li								
Npar	190	145	280	208	163	127	217	199
K	9	4	19	11	6	2	12	10
BIC*	257289	256878	256095	255849	260116	257749	256437	255787
Run Time	00:02:10	00:20:28	00:02:03	00:07:46	00:05:59	00:50:18	00:03:40	00:31:54
Wainwright								
Npar	382	400	553	481	382	391	571	490
K	1	3	20	12	1	2	22	13
BIC*	486930	480516	477486	477661	486930	480892	477932	477702
Run Time	00:09:14	01:23:36	00:09:05	00:29:15	00:25:19	03:25:53	00:13:49	01:59:03

NOTE.—For each dataset the table includes the total number of parameters in the optimal partitioning schemes (Npar), the number of partitions in the optimal partitioning schemes (K), the BIC scores of the optimal partitioning schemes assuming proportional branch lengths and recomputed in RaxML (BIC*), and the time to select an optimal partitioning schemes (Run Time) in hours, minutes and seconds (hh:mm:ss).

Table S2. Rand-Index between the optimal partitioning schemes found under different strategies in the real datasets studied.

	PT-C	PT-G	PT-C-p	PT-G-p	PF-C	PF-G	PF-Cp
Endicot							
PT-G	1.00(0)						
PT-C-p	0.27(6)	0.27(6)					
PT-G-p	0.21(5)	0.21(5)	0.78(-1)				
PF-C	1.00(0)	1.00(0)	0.27(-6)	0.21(-5)			
PF-G	1.00(0)	1.00(0)	0.27(-6)	0.21(-5)	1.00(0)		
PF-Cp	0.29(4)	0.29(4)	0.88(-2)	0.79(-1)	0.29(4)	0.29(4)	
PF-G-p	0.23(5)	0.23(5)	0.77(-1)	0.97(0)	0.23(5)	0.23(5)	0.77(1)
Fong							
	PT-C	**PT-G**	**PT-C-p**	**PT-G-p**	**PF-C**	**PF-G**	**PF-C-p**
PT-G	0.79(-8)						
PT-C-p	0.8(4)	0.83(4)					
PT-G-p	0.78(-6)	0.93(2)	0.8(-2)				
PF-C	0.67(6)	0.73(14)	0.71(10)	0.73(12)			
PF-G	0.78(-11)	0.90(-3)	0.83(-7)	0.87(-5)	0.73(-17)		
PF-C-p	0.72(1)	0.85(9)	0.78(5)	0.84(7)	0.74(-5)	0.84(12)	
PF-G-p	0.77(-7)	0.91(1)	0.82(-3)	0.90(-1)	0.74(-13)	0.91(4)	0.87(-8)
Hackett							
	PT-C	**PT-G**	**PT-C-p**	**PT-G-p**	**PF-C**	**PF-G**	**PF-C-p**
PT-G	0.36(3)						
PT-C-p	0.06(55)	0.68(52)					
PT-G-p	0.06(27)	0.69(24)	0.91(-28)				
PF-C	1.00(0)	0.36(-3)	0.06(-55)	0.06(-27)			
PF-G	0.37(2)	0.57(-1)	0.62(-53)	0.62(-25)	0.37(2)		
PF-C-p	0.05(44)	0.63(41)	0.90(-11)	0.91(17)	0.05(44)	0.65(42)	
PF-G-p	0.05(23)	0.62(20)	0.90(-32)	0.90(-4)	0.05(23)	0.67(21)	0.92(-21)
Kaffenberger							
	PT-C	**PT-G**	**PT-C-p**	**PT-G-p**	**PF-C**	**PF-G**	**PF-Cp**
PT-G	0.47(1)						
PT-C-p	0.35(9)	0.86(8)					
PT-G-p	0.38(7)	0.90(6)	0.96(-2)				
PF-C	0.79(-1)	0.45(-2)	0.34(-10)	0.37(-8)			
PF-G	0.64(0)	0.80(-1)	0.67(-9)	0.71(-7)	0.57(1)		
PF-C-p	0.44(4)	0.72(3)	0.71(-5)	0.73(-3)	0.37(5)	0.77(4)	
PF-G-p	0.49(5)	0.70(4)	0.71(-4)	0.72(-2)	0.36(6)	0.73(5)	0.73(1)
Li							
	PT-C	**PT-G**	**PT-C-p**	**PT-G-p**	**PF-C**	**PF-G**	**PF-C-p**
PT-G	0.71(-11)						
PT-C-p	0.82(5)	0.75(16)					
PT-G-p	0.74(-4)	0.75(7)	0.90(-9)				
PF-C	0.21(-14)	0.28(-3)	0.04(-19)	0.09(-10)			
PF-G	0.67(-13)	0.74(-2)	0.50(-18)	0.55(-9)	0.54(1)		
PF-C-p	0.76(-3)	0.72(8)	0.86(-8)	0.83(1)	0.11(11)	0.57(10)	
PF-G-p	0.76(-5)	0.75(6)	0.90(-10)	0.86(-1)	0.08(9)	0.54(8)	0.85(-2)
Wainwright							
	PT-C	**PT-G**	**PT-C-p**	**PT-G-p**	**PF-C**	**PF-G**	**PF-C-p**
PT-G	0.50(2)						
PT-C-p	0.04(18)	0.54(16)					
PT-G-p	0.06(11)	0.56(9)	0.94(-7)				
PF-C	1.00(0)	0.50(-2)	0.04(-18)	0.06(-11)			
PF-G	0.54(1)	0.96(-1)	0.50(-17)	0.52(-10)	0.54(1)		
PF-C-p	0.03(21)	0.52(19)	0.94(3)	0.94(10)	0.03(21)	0.49(20)	
PF-G-p	0.06(12)	0.55(10)	0.94(-6)	0.94(1)	0.06(12)	0.52(11)	0.93(-9)

NOTE.—In brackets, the difference in the number of partitions (row - column).

Table S3. Average Rand-Index across data sets between PartitionTest and PartitionFinder, between hierarchical clustering and Greedy algorithms, and proportional versus independent branch lengths.

Dataset	PartitionTest vs. PartitionFinder	Clustering vs. Greedy	Proportional vs. Independent
Endicott	0.4693	0.5707	0.2515
Fong	0.8030	0.7941	0.8090
Hackett	0.4523	0.4935	0.3504
Kaffenberger	0.5858	0.6160	0.5716
Li	0.4514	0.6193	0.5346
Wainwright	0.4260	0.5074	0.2856
Average	0.5313	0.6002	0.4671

Chapter 7

Discussion

This section presents an overview of the whole research carried out in the Thesis. Basically, we describe how the main objectives of the thesis have been accomplished by the different journal articles that make up this work, together with an overall discussion of the main research results that have been achieved.

For single-gene model selection, we devised and implemented in ProtTest and jModelTest three parallelization approaches (Chapters 2 and 3): (i) a shared memory implementation for multicore desktop computers, (ii) distributed memory implementation for HPC architectures, and (iii) a hybrid memory strategy for multicore cluster architectures. ProtTest and jModelTest are tools implemented in Java, which provides several totally portable options for developing parallel algorithms. Java provides several programming options for HPC [83]. As Java has built-in multithreading support, the use of threads is quite extended due to its portability and high performance, although it is a rather low-level option. Nevertheless, Java provides concurrency utilities, such as thread pools, tasks, blocking queues, and low-level high-performance primitives (e.g., *CyclicBarrier*), for a higher level programming. However, this option is limited to shared memory machines, which provide less computational power than distributed memory architectures. On the other hand, message-passing is the preferred programming model for distributed memory architectures (e.g., clusters) due to its portability, scalability and usually good performance, although it generally requires significant development efforts. Among currently available Java Message-Passing Java

(MPJ) libraries, F-MPJ [84] and MPJ Express [74] deserve to be mentioned for their nested parallelism (MPJ+threads) support for exploiting performance on clusters of multi-core processors.

The shared memory approach uses a thread pool to handle the execution of tasks on shared memory architectures. Java thread pools are included in the SDK. The task queue contains the set of candidate models to optimize which will be processed by the thread pool (Figure 7.1). The implementation can take advantage of simultaneous multithreading (SMT) obtaining up to 30% of additional speed-up.

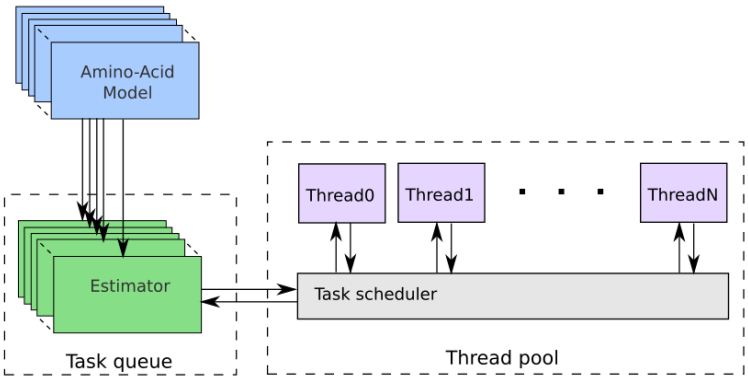

Figure 7.1: ProtTest 3 shared memory strategy.

For distributed memory architectures (e.g., clusters), we used message-passing communication. The implemented parallel approach is compatible with F-MPJ [84] and MPJ Express [74] libraries. A dynamic master-worker approach, analogous to the thread pool strategy, turned out to be the most efficient and scalable. However, due to the limited number of models and the workload imbalance (e.g., Γ models require a longer time to optimize), the number of models with Γ rate heterogeneity is a limiting factor.

A heterogeneous hybrid parallel approach with dynamic scheduling overcomes the limitations in the scalability and the workload imbalance. Models are weighted according to the estimated computational load according to their parameters. We assume equal computational costs of each parameter, except the shape of the Γ distribution (α). The core model parameter and likelihood computations are carried out using a custom parallel PhyML implementation with OpenMP [19]. The hybrid approach overcomes the speed-up limitations of the task-level parallel

approaches, achieving speed-ups of up to 260 in our benchmarks.

We implemented also an online service, *jmodeltest.org*, oriented towards High Performance Computing clusters for the transparent execution of jModelTest 2 (Chapter 4). Its main benefit is straightforward execution across different platforms, avoiding any configuration/execution issues. Unlike CIPRES, the advantage of *jmodeltest.org* is that users can aggregate their own private computational resources. The web interface facilitates the execution for users that are not familiar with command console interfaces.

For the final release of jModelTest2 (Chapter 5), we implemented a set of new features. Among those, we should highlight the capability of searching among the entire set of 203 substitution schemes nested within GTR (i.e., all possible combinations of rate matrix symmetries). When combined with rate variation (proportion of invariant sites and discrete Γ rate categories) and equal/unequal base frequencies the total number of possible models is $203 \times 8 = 1624$. Other interesting additions in this paper are the topological support and the reliability of the selection criteria. The topological support summarizes which topologies are supported by which model, including confidence intervals constructed from cumulative models weights. Measuring the stability of the inferences according to the model parameters gives the user an idea about the reliability of the results. Among the different selection criteria (Section 1.1.4), BIC turned out to be the most accurate for retrieving the true generating model in a simulation study of 10,000 samples.

For multigene model selection (Chapter 6), loop level parallelism via a fine-grain parallelization of the PLF is provided by the Phylogenetic Likelihood Library (PLL) [29]. PLL exploits state-of-the-art vector instructions (SSE3, AVX, AVX-2 intrinsics) to accelerate computations. PartitionTest is one of the first tools incorporating this library, although it has already been successfully deployed to substantially accelerate DPPDIV [20], a Bayesian program for divergence time(s) estimates, and IQ-Tree [62], a novel but very promising tool for phylogenetic inference. We estimate *a priori* the balance of task-level and loop-level distribution of cores/processes such that the parallel efficiency is maximized.

The analysis of multigene model selection approaches were directed towards

evaluating the accuracy of the heuristic algorithms for finding the "true" partitioning schemes (partitioning accuracy), and for the phylogenetic reconstructions (phylogenetic accuracy). PartitionTest is between 1.5 and 2.6 times faster than the current state-of-the-art (PartitionFinder) for alignments with a reduced set of partitions (< 100), and 7 times faster for 1,000 partitions. Moreover, PartitionTest is more accurate finding the true partitioning scheme.

Greedy algorithm performs better in terms of partitioning accuracy than hierarchical clustering algorithm. However, we observed the opposite results in the phylogenetic accuracy. In general, hierarchical clustering tends to select partitioning schemes with a higher number of parameters than the "true" partitioning scheme, while greedy algorithm tends to select underparameterized models. Remarkably, we obtained similar phylogenetic inferences with the *a priori* assigning of independent GTR+Γ model to each data block. Using one GTR+Γ model per data block ($K = N$) lead to similar tree inferences than the search algorithms, and even compared to the true partitioning scheme. On the other hand, using one single partition ($K = 1$) performed always noticeably worse than every other approach. Our results, together with results obtained by Kainer and Lanfear (2015) [43] in their analyses on real data sets, suggest that $K = N$ is a very convenient strategy for the ML analysis of not very complex phylogenomic datasets.

Chapter 8

Conclusions

This PhD thesis, *"Selection of Models of Genomic Evolution in HPC Environments"*, has been conducted with two main purposes: On the one hand, develop and implement HPC algorithms for the selection of the most suitable models of evolution, nucleotide substitution and amino acid replacement, for single and also multigene data. On the other hand, we discussed the impact of different partitioning scheme in the phylogenetic inferences.

1. ProtTest 3 and jModelTest 2 are efficient and accurate HPC tools for single-gene model selection in phylogenetics.

2. Both tools provide a flexible best-fit model selection framework, allowing for a wide range of tuning that is not present in other tools. For example, the topology can be (i) fixed by using a distance method like BIONJ [30], or the ML topology inferred from the null model (JC for nucleotide data), or (ii) ML estimated.

3. A hybrid shared/distributed memory parallelization approach presents significantly higher performance and scalability than the shared and distributed memory versions. The tools can also take advantage of simultaneous multithreading (SMT).

4. They provide fault tolerance capabilities through a checkpointing system, which is essential for running on large scale data sets.

5. jModelTest.org is a web service that allows for executing instances of jModelTest in multiple HPC architectures (public or private) transparent to the user. It takes advantage of the scalability of the public cloud while maximizes the available slots in clusters.

6. jModelTest 2 incorporates a fast selection algorithm among the entire set of GTR submodels (203 substitution schemes), and extended user-readable analysis features for DNA sequence alignments, such as HTML reports and topological support summary.

7. BIC shows better performance for retrieving the true generating model than AIC or AICc.

8. PLL is a suitable library for model parameter optimization and topological searches.

9. PartitionTest is an efficient and accurate tool for multigene model selection in phylogenetics, providing fast approximations to the best-fit partitioning scheme.

10. For not very complex multigene data sets, phylogenetic inference is not very sensitive to overparameterization. Hierarchical clustering approaches can perform better than greedy algorithms, even though achieving a lower score. Nevertheless, using a single GTR+Γ model for each data block provides a phylogenetic accuracy similar to the best-fit partitioning scheme.

Chapter 9

Future Work

In terms of future research work, the current avalanche of genetic data due to NGS technologies is generating a large number of challenges in the field of phylogenetics and phylogenomics, and therefore in bioinformatics. Current analyses, apart from presenting a growth in terms of per-species data, comprise also a larger number of species, or taxa, and it has been convincingly demonstrated that the evolutionary rate of a given position is not always constant throughout time. Within-site rate variations are called heterotachy (for "different speed" in Greek). Yet, heterotachy was found among homologous sequences of distantly related organisms, often with different functions. In such cases, the functional constraints are likely different, which would explain the different distribution of variable sites. Therefore, for analyses with a large number of taxa or distant species, addressing heterotachy (assuming different substitution models for different lineages) in the selection of the evolutionary models might improve the phylogenetic inferences.

Regarding partitioning in phylogenomic analyses, there are still many unknown issues. In real life, phenomena like incomplete lineage sorting, gene duplication and loss and horizontal gene transfer can lead to a different scenario in which different partitions evolve under different gene trees, embedded within a single species tree. Thus, further studies could focus on evaluating the impact of partitioning on the inference of species trees.

We are already working in a low-level replacement for PLL, intended to effi-

ciently support arbitrary data types (e.g., DNA, proteins, codons), and rate heterogeneity models (discrete Γ rates, proportion of invariant sites, mixture models and heterotachy). The new implementation facilitates improving the efficiency of the current phylogenetic analysis software, and also extending their features. We are also working in a new model selection tool, as a replacement for *jModelTest*, *ProtTest* and *PartitionTest* incorporating the new low-level PLL.

For obtaining "good" load balance for partitioned datasets, when optimizing ML or proposing new model parameters (Bayesian Inference) for partitioned datasets, it is important to propose and evaluate these new values (e.g., α-shape) simultaneously for all partitions to improve parallel efficiency [78]. In addition, a well-balanced number of sites must be assigned to each process, and the number of distinct partitions per process [98] needs to be minimized. Kobert et al. (2014) [50] showed that this load-balancing problem is NP-hard and designed a polynomial time approximation algorithm that is guaranteed to yield a near-optimal distribution of sites and partitions to processes. The algorithm balances the number of sites among processes and at the same time minimizes the number of partitions assigned to each process. Note that, in this setting, a single partition can be split among several processes. The implementation of this new method in PLL is underway.

Capítulo 10

A Summary in Spanish

Este resumen se compone de una introducción, que explica la motivación y contexto de la Tesis, seguida de una sección sobre su organización en partes y capítulos. A continuación, sigue una enumeración de los medios que han sido necesarios para llevarla a cabo, para finalizar con las conclusiones, trabajo futuro y las principales contribuciones recogidas en ella.

10.1. Introducción

La irrupción de las tecnologías de secuenciación de nueva generación (NGS, por sus siglas en inglés –Next Generation Sequencing) en el campo de la biología molecular computacional ha cambiado drásticamente el escenario de la filogenética. La generación de datos moleculares continúa creciendo más y más rápido, y esto facilita la exploración de nuevas áreas de investigación, nuevos métodos, y genera la necesidad de mejorar el rendimiento de los ya existentes, tanto en precisión como en eficiencia. El análisis de datos está cambiando rápidamente desde la filogenética (i.e., estudio de un único gen, o un conjunto reducido de genes) a la filogenómica (i.e., estudio de un gran conjunto de genes, o many-gene analysis).

Muchos métodos actuales requieren el uso de modelos de evolución, y está demostrado que el uso de un modelo u otro puede llevar a diferentes resultados en el proceso de inferencia filogenética. Desde los comienzos del análisis de datos mole-

culares, se conoce que diferentes sitios (posiciones concretas en las cadenas de ADN o proteínas), o regiones del ADN pueden evolucionar a diferentes ritmos, o permanecer invariantes. Por tanto, desde modelos muy simples (e.g., Jukes-Cantor 1969) hasta los más complejos (e.g., mixture models), hay un gran número de posibilidades. Tanto la sub- como la sobreparametrización en el modelo escogido presenta desventajas en términos de precisión. Por este motivo, se han creado herramientas que utilizan entornos estadísticos para seleccionar el modelo de evolución que más se ajuste a los datos, buscando un balance correcto entre el likelihood (verosimilitud) y la parametrización.

Un método para abordar el problema de la heterogeneidad en los datos es el particionado, donde se asume que diferentes grupos de sitios evolucionan bajo la influencia de diferentes modelos de evolución. Una práctica común es concatenar alineamientos de diferentes fuentes (e.g., diferentes genes) para el mismo conjunto de taxa (i.e., especies u organismos). Estos datos, generalmente de gran longitud, se pueden organizar posteriormente en particiones. Por ejemplo, cada gen, o cada posición de codón para cada gen, puede constituir un bloque de datos (o partición) separados. En un análisis de datos particionados en un entorno de Maximum Likelihood (ML – máxima verosimilitud), el modelo evolutivo se evalúa bajo un conjunto de parámetros, que son optimizados para cada partición. En este caso, no sólo es importante decidir qué modelos se asignan a cada partición, sino también en cuántas particiones dividimos los datos. Para estudios filogenómicos, como por ejemplo 10K vertebrate genome project (http:// www.genome10k.org/) y el 1,000 insect transcriptome evolution project (http://www.1kite.org/), que analizan la historia evolutiva de 10,000 vertebrados y 1,000 insectos respectivamente, el particionado es una práctica rutinaria en los análisis. Sin embargo, el número de posibilidades que existen entre considerar una única partición para todo el conjunto de datos, a una partición para cada uno de los bloques de datos propuestos, presenta un crecimiento exponencial con respecto al número de bloques inicial, y encontrar el esquema de particionado más adecuado es un problema NP-Difícil. Por ejemplo, para 100 bloques existen $4{,}75 \times 10^{115}$ formas diferentes de distribuirlos en particiones de forma que abarquen todo el conjunto de datos sin solapamiento. Este número es alrededor de 10^{30} veces mayor que el número estimado de átomos en el universo observable.

Esta Tesis presenta un estudio de métodos de computación de altas prestaciones (HPC – High Performance Computing) para la selección de modelos tanto para un único gen, como para alineamientos genómicos, así como su impacto en los posteriores análisis en un marco ML, en términos de precisión o eficiencia filogenética.

Hemos desarrollado versiones HPC de jModelTest y Protest, así como un gestor online multi-plataforma de jModelTest para entornos HPC. Actualmente, ambas herramientas son una referencia en la comunidad bioinformática para selección de modelos de evolución de datos sin particionar, para dominios de ADN y proteínas respectivamente.

Para análisis multigen, hemos desarrollado PartitionTest, que incluye un conjunto de heurísticas que pueden ser calculadas en tiempo lineal o polinomial, para encontrar el modelo que mejor se ajusta a los datos. PartitionTest hace un uso intensivo de la librería PLL (Phylogenetic Likelihood Library), en conjunto con técnicas HPC de más alto nivel (e.g., paralelismo a nivel de tarea) para las operaciones en un entorno ML.

Finalmente, hemos diseñado un estudio basado en simulaciones, donde los datos reales son conocidos, para evaluar el impacto de las diferentes técnicas de selección de modelos en la precisión al recuperar el verdadero modelo utilizado para generar los datos, y, más importante, las filogenias.

10.2. Organización de la Tesis

De acuerdo con la regulación actual de la Universidade da Coruña, la Tesis doctoral se ha estructurado como una Tesis por compendio de publicaciones de investigación. Concretamente se compone de 3 artículos publicados en revistas indexadas en el Journal Citation Reports (JCR), y que se han agrupado en dos partes diferenciadas.

La Tesis comienza con un capítulo de introducción, orientado a proporcionar al lector una visión general de los aspectos importantes dentro del área de estudio, así como de toda la investigación llevada a cabo en los artículos. En primer lugar,

este capítulo introduce el alcance y las motivaciones de la Tesis, con el objetivo de delimitar su contexto y proporciona una clara descripción de los objetivos principales a conseguir.

A continuación, se presentan los artículos de investigación que conforman la tesis, cada uno en un capítulo separado (Capítulos 2- 6):

- **Darriba, D.**, Taboada, G. L., Doallo, R., & Posada, D. (2011). ProtTest 3: fast selection of best-fit models of protein evolution. *Bioinformatics, 27(8), 1164-1165.*

  Factor de impacto: **4.981**, **547** citas

- **Darriba, D.**, Taboada, G. L., Doallo, R., & Posada, D. (2013). High-performance computing selection of models of DNA substitution for multicore clusters. *International Journal of High Performance Computing Applications, 1094342013495095.*

  Factor de impacto: **1.477**, **4** citas

- Santorum, J. M., **Darriba, D.**, Taboada, G. L., & Posada, D. (2014). jmodeltest.org: selection of nucleotide substitution models on the cloud. *Bioinformatics, btu032.*

  Factor de impacto: **4.981**

- **Darriba, D.**, Taboada, G. L., Doallo, R., & Posada, D. (2012). jModelTest 2: more models, new heuristics and parallel computing. *Nature methods, 9(8), 772-772.*

  Factor de impacto: **32.072**, **1,849** citas

- **Darriba, D.**, & Posada, D. (2015). The impact of partitioning on phylogenomic accuracy. *bioRxiv, 023978.*

En el Capítulo 7 se incluye una discusión general de los artículos de investigación anteriores enlazando los contenidos y dotando de coherencia al conjunto. El Capítulo 8 describe las conclusiones de la tesis. Finalmente, en el Capítulo 9 se describe el trabajo futuro.

10.3. Medios

- Material de trabajo y financiación principalmente soportados por el Grupo de Arquitectura de Computadores de la Universidade de A Coruña, y el Grupo de Filogenómica de la Universidade de Vigo.
- Acceso a material bibliográfico a través de las bibliotecas de las Universidades de A Coruña y Vigo.
- Financiación adicional a través de los siguientes proyectos de investigación:
  - Financiamiento regional por la Xunta de Galicia bajo el Programa de Consolidación y Estructuración de Unidades de Investigación Competitivas, Modalidad de Redes de Investigación (Grupo Arquitectura de Computadores, refs. GRC2013/055 y 2010/6), Red Gallega de Computación de Altas Prestaciones (ref. 2010/53), y Red Gallega de Bioinformática (Grupo Filogenómica, ref. 2010/90).
  - European Research Council (Grupo Filogenómica, ref. ERC-2007-Stg 203161-PHYGENOM).
  - Ministerio de Ciencia e Innovación bajo los proyectos BFU2009-08611 (Grupo de Filogenómica) y TIN2010-16735 (Grupo de Arquitectura de Computadores).
  - Amazon Web Services (AWS) research grant "EC2 in phylogenomics".
  - Ministerio de Educación y Ciencia (Grupo de Filogenómica, ref. BFU2009-08611)
- Acceso a clusters, supercomputadores y plataformas de computación en la nube:
  - *Cluster Pluton* (Grupo Arquitectura de Computadores, Universidade de A Coruña) Inicialmente, 16 nodos con 2 procesadores Intel Xeon quad-core Nehalem-EP y 16 GB de memoria, todos los nodos conectados vía InfiniBand DDR y 2 de ellos vía 10 Gigabit Ethernet. Adicionalmente, 2 nodos con un procesador Intel Xeon quad-core Sandy Bridge-EP y 32 GB de memoria, interconectados vía InfiniBand FDR, RoCE e iWARP,

y 4 nodos con un procesador Intel Xeon hexa-core Westmere-EP, 12 GB de memoria y 2 GPUs NVIDIA Tesla "Fermi" 2050 por nodo, interconectados vía InfiniBand QDR. Además, se han añadido 16 nodos, cada uno de ellos con 2 procesadores Intel Xeon octa-core Sandy Bridge-EP, 64 GB de memoria y 2 GPUs NVIDIA Tesla "Kepler" K20m, interconectados por InfiniBand FDR.

- *Cluster Diploid* (Grupo de Filogenómica, Universidade de Vigo). (1) Un *fat node* con 4 procesadores Intel Xeon Westmere-EX de 10 núcleos y 512 GB de memoria (80 núcleos en total), (2) 30 nodos con 2 procesadores Intel Xeon E5-420 quad-core Harpertown (8 núcleos en totoal) y 16 GB de memoria, y (3) 44 nodos con 2 procesadores Intel Xeon X5675 hexa-core Westmere-EP (12 núcleos en total); 50 GB de memoria y *Hyperthreading* deshabilitado.

- *Supercomputador Finis Terrae* (Centro de Supercomputación de Galicia, CESGA): 144 nodos con 8 procesadores Intel Itanium-2 dual-core Montvale y 128 GB de memoria, interconectados vía InfiniBand DDR. Adicionalmente, hemos usado un sistema Superdome con 64 procesadores Intel Itanium-2 dual-core Montvale y 1 TB de memoria.

- Plataforma de *cloud computing* Amazon EC2 IaaS (Amazon Web Services, AWS). Se han empleado diferentes tipos de instancias: (1) CC1, 2 procesadores Intel Xeon quad-core Nehalem-EP (8 núcleos en total), 23 GB de memoria y 2 discos de almacenamiento local por instancia; (2) CC2, 2 procesadores Intel Xeon octa-core Sandy Bridge-EP processors, 60.5 GB de memoria y 4 discos de almacenamiento local por instancia; (3) CG1, 2 procesadores Intel Xeon quad-core Nehalem-EP (8 núcleos en total), 22 GB de memoria, 2 GPUs NVIDIA Tesla "Fermi" 2050 y 2 discos de almacenamiento local por instancia; (4) HI1, 2 procesadores Intel Xeon quad-core Westmere-EP (8 núcleos en total), 60.5 GB de memoria y 2 discos de almacenamiento local SSD por instancia; (5) CR1, 2 procesadores Intel Xeon octa-core Sandy Bridge-EP (16 cores en total), 244 GB de memoria y 2 discos de almacenamiento local SSD por instancia; Todos estos tipos de instancias están conectadas vía 10 Gigabit Ethernet.

- Beca predoctoral por la Universidade da Coruña
- Estanca de investigación de 6 meses de duración en el Grupo de Filogenómica de la Universidade de Vigo.
- Estancia de investigación de 3 meses de duración en el Heidelberg Institute for Theoretical Studies (HITS) en Heidelberg, Alemania, que ha permitido la colaboración en el desarrollo de la librería de análisis filogenético (PLL), añadiendo flexibilidad para trabajar con diferentes particiones y adecuándola para el desarrollo de *PartitionTest*. La estancia fue financiada por la Universidade da Coruña.
- Contrato de investigador asociado en el Heidelberg Institute for Theoretical Studies (HITS) en Heidelberg, Alemania.

10.4. Discusión

Para la selección de modelos de un único gen, hemos diseñado e implementado tres esquemas de paralelización (Capítulos 2 y 3): (i) una implementación en memoria compartida para computadoras de escritorio multinúcleo, (ii) una implementación en memoria distribuida para arquitecturas HPC, y (iii) una implementación híbrida para arquitecturas cluster multinúcleo. ProtTest and jModelTest son herramientas implementadas en Java, que proporciona diversa opciones totalmente portables para el desarrollo de algoritmos paralelos. Java proporciona diferentes opciones para la Computación de Altas Prestaciones [83]. Gracias al soporte multihilo nativo, su portabilidad y su alto rendimiento, el uso de hilos en Java está bastante extendido, a pesar de que es una opción de programación de bajo nivel. Sin embargo, para programación de más alto nivel Java proporciona proporciona herramientas de concurrencia como *thread pools*, tareas, listas de bloqueo y primitivas de alto rendimiento (e.g., *CyclicBarrier*). No obstante, esta opción se limita al uso de memoria compartida, lo que proporciona menor potencia computacional que las arquitecturas de memoria distribuida. Por otro lado, el paso de mensajes es el modelo de programación más adecuado para las arquitecturas de memoria distribuida (e.g., clusters) gracias a su portabilidad, escalabilidad, y

habitualmente buen rendimiento, aunque en general requiere un mayor esfuerzo en el desarrollo. Entre las librearías de paso de mensajes disponibles en Java (MPJ), merecen ser mencionadas F-MPJ [84] y MPJ Express [74] por su soporte de paralelismo anidado (MPJ+threads) para explotar mejor el rendimiento en clusters multinúcleo.

La implementación de memoria compartida tanto para ProtTest como para jModeltest utilizan un thread pool para manejar la ejecución de las diferentes tareas. Los thread pools están integrados en el SDK. La implementación también puede aprovecharse de las tecnologías de multithreading simultáneo (SMT), obteniendo hasta un 30 % de aceleración adicional.

Para las arquitecturas de memoria distribuida (e.g., cluster), hemos utilizado comunicaciones por paso de mensajes. La estrategia paralela implementada es compatible con las librerías F-MPJ y MPJ Express. Una implementación maestro-esclavo, análogo a la estrategia basada en thread pool, ha resultado ser la más eficiente y escalable. Sin embargo, debido al limitado número de modelos y a la carga desbalanceada (e.g., los modelos con una distribución Γ requieren un tiempo de procesamiento mucho mayor), el número de modelos con categorías discretas son un factor limitante.

Una implementación paralela híbrida con planificación dinámica permite solucionar las limitaciones en la escalabilidad y el desbalanceo de la carga de trabajo. Los modelos se ponderan de acuerdo a la carga computacional estimada según los parámetros que contengan. Asumimos un coste homogéneo para cada uno de los parámetros, excepto para aquellos con distribución Γ. Los cálculos necesarios para la optimizacion de los parámetros, y computación de la verosimilitud se efectúan utilizando una implementación paralela propia de PhyML, haciendo uso de directivas OpenMP [19]. La versión híbrida soluciona las limitaciones de speed-up que encontrábamos en las versiones con paralalelización a nivel de tarea, alcalnzando aceleraciones de hasta 260 en nuestros benchmarks.

Hemos implementado un servicio online, *jmodeltest.org*, orientado hacia la computación en clusters HPC para ejecutar jModelTest 2 de forma transparente (Capítulo 4). El mayor beneficio radica en su ejecución sencilla a través de diferentes plataformas, evitando cualquier problema relacionado con la configuración/ejecución.

A diferencia de CIPRES, la ventaja de *jmodeltest.org* es que los usuarios pueden agregar sus propios recursos computacionales. La interfaz web facilita la ejecución para los usuarios que no están familiarizados con las interfaces de consola de comandos.

Para el lanzamiento de la versión final de jModelTest 2 (Capítulo 5), hemos implementado una serie de nuevas características. Entre éstas, debemos destacar la capacidad de búsqueda entre el conjunto completo de 203 esquemas de sustitución anidados con GTR (i.e., todas las posibles combinaciones de simetrías en la matriz de tasas de sustitución). Cuando combinamos estos esquemas de sustitución con los diferentes parámetros (proporción de sitios invariantes, distribución Γ y frecuencias estacionarias iguales/estimadas), el número de modelos candidatos es $203 \times 8 = 1624$. Otras incorporaciones interesantes en este Capítulo son el soporte topológico y la fiabilidad de los diferentes criterios de selección. El soporte topológico resume qué topologías son apoyadas por qué modelos, incluyendo intervalos de confianza construidos a partir de los pesos de los modelos según el criterio de selección elegido. Por otro lado, medir la estabilidad de las inferencias según los parámetros de los modelos nos proporciona una idea acerca de la fiabilidad de los resultados. Entre los diferentes criterios de selección, BIC ha resultado ser el más preciso para recuperar el modelo verdadero en un estudio sobre 10.000 conjuntos de datos simulados.

Para la selección de modelos en alineamientos multi-gen (Capítulo 6), el paralelismo a nivel de bucle gracias a una paralelización de grano fino de la función de likelihood (PLF), viene proporcionado por la librería de análisis filogenético PLL [29].

PLL explota el estado del arte en instrucciones vectoriales (SSE3, AVX, AVX-2) para acelerar las computaciones. PartitionTest es una de las primeras herramientas en incorporar esta librería, aunque ya ha sido integrada con éxito para acelerar sustancialmente DPPDIV [20], un programa para estimar los tiempos de divergencia en árboles filogenéticos, y en IQ-TREE, una nueva pero muy prometedora herramienta de inferencia filogenética. En PartitionTest, estimamos *a priori* la distribución de los procesadores/núcleos de computación entre las estrategias de paralelismo a nivel de bucle y a nivel de tarea, con el objetivo de maximizar la eficiencia paralela.

El análisis de las estrategias de selección de modelos multi-gen se ha dirigido a la evaluación de la precisión de las diferentes heurísticas para encontrar el esquema de particionado verdadero, así como la reconstrucción filogenética. PartitionTest es entre 1,5 y 2,6 veces más rápido que el estado del arte actual (PartitionFinder) para alineamientos con un conjunto reducido de particiones (< 100), y 7 veces para conjuntos más grandes (1.000 particiones). Además, PartitionTest es más preciso encontrando el esquema de particionado verdadero.

El algorithm Greedy obtiene mejores resultados que el clustering jerárquico en la búsqueda del esquema de particionado correcto. El cambio, hemos observado lo contrario cuando lo que buscamos es una correcta inferencia filogenética. En general, el clustering jerárquico tiende a seleccionar esquemas de particionado con un mayor número de parámetros que el esquema de particionado verdadero, mientras que el algoritmo Greedy tiende a la subparametrización. Destacablemente, hemos obtenido inferencias similares con la asignación *a priori* de modelos GTR+Γ para cada uno de los bloques de datos. Empleando un modelo GTR+Γ para cada bloque de datos ($K = N$) se traduce en inferencias filogenéticas similares a las de los algoritmos de búsqueda, e incluso en comparación con el esquema de particionado verdadero. Utilizando una partición única ($K = 1$), los resultados han sido notablemente peores que cualquier otra opción. Nuestros resultados, en conjunto con los obtenidos por Kainer y Lanfear (2015) [43] en sus análisis en sets de datos reales, sugieren que $K = N$ es una buena opción para el análisis ML de conjuntos de datos filogenéticos de complejidad reducida.

10.5. Conclusiones

Esta tesis doctoral, *Selección de Modelos de Evolución Genómica en Entornos HPC*, se ha llevado a cabo con 2 objetivos: En primer lugar, desarrollar e implementar algoritmos para computación de altas prestaciones para la selección de los modelos de evolución más adecuados, tanto de sustitución de nucleótidos como de reemplazo de aminoácidos, para datos de un único o varios genes. En segundo lugar, hemos discutido el impacto de los diferentes esquemas de particionado en las inferencias filogenéticas.

1. ProtTest 3 y jModelTest 2 son herramientas HPC eficientes y precisas para la selección de modelos de un único gen en filogenética.

2. Ambas herramientas proporcionan un marco flexible para seleccionar el modelo que mejor se ajusta a los datos, permitiendo un amplio rango de ajuste que no está presente en otras herramientas. Por ejemplo, la topología puede ser (i) fijada empleando métodos basados en matriz de distancias, como BIONJ [30], o utilizando la topología con la máxima verosimilitud inferida desde el modelo nulo (JC para datos de ADN), o (ii) recalculada para cada modelo.

3. Una implmentación híbrida en memoria compartida/distribuida presenta significativamente mejor rendimiento y escalabilidad que las implementaciones en memoria compartida o distribuida. Las herramientas pueden también aprovecharse de las tecnologías de multihilo simultáneo (simultaneous multithreading, o SMT).

4. Proporcionan tolerancia a fallos a través de un sistema de checkpointing, lo cual es esencial para la ejecución sobre datos de gran escala.

5. jModelTest.org es un servicio web que permite la ejecución de instancias de jModelTest en múltiples arquitecturas HPC (públicas o privadas) de forma transparente al usuario.

6. jModelTest 2 incorpora un algoritmo de selección rápido para el conjunto completo de submodelos de GTR (203 esquemas de particionado),uuu y herramientas de análisis inteligibles por el usuario para alineamientos de secuencias de ADN, como reportes HTML y un resumen del soporte topológico.

7. BIC presenta un mejor rendimiento que AIC o AICc para recuperar el modelo verdadero que ha generado los datos.

8. PLL es una librería adecuada para la optimización de parámetros y la búsqueda de topologías.

9. PartitionTest es una herramienta eficiente y precisa para la selección de modelos multigénicos en filogenética, proporcionando una aproximación rápida al mejor esquema de particionado.

10. Para conjuntos de datos multigénicos de baja complejidad, la inferencia filogenética es poco sensible a la sobreparametrización. Los algoritmos de clustering jerárquico pueden obtener mejores resultados que los algoritmos greedy, a pesar de obtener generalmente esquemas de particionado con un peor valor del criterio de selección. Sin embargo, utilizar un modelo GTR+Γ para cada bloque de datos proporciona unos resultados similares a los de los esquemas de particionado seleccionados por los diferentes algoritmos.

10.6. Trabajo Futuro

En términos de trabajo futuro de investigación, la avalancha actual de datos genéticos gracias a las tecnologías de sequenciación de nueva generación (NGS) está generando un gran número de retos en los campos de la filogenética y la filogenómica, y por lo tanto en la bioinformática. Los análisis actuales, además de presentar un importante crecimiento en términos de datos por especia, comprenden también un mayor número de especies, o *taxa*, y se ha demostrado convincentemente que el ritmo de evolución de las diferentes posiciones en el ADN no son siempre constantes a lo largo del tiempo. La variación en la velocidad de evolución dentro de un mismo sitio se conoce como *heterotaquia* (del griego "diferente velocidad"). La heterotaquia se ha podido ver entre sequencias homólogas de organismos relacionados de una forma distante, a menudo con diferentes funciones. En esos casos, las restricciones funcionales son probablemente diferentes, lo que explicaría las diferentes distribuciones de los sitios variantes. Por lo tanto, para análisis con un número elevado de taxa o especies distantes, teniendo en cuenta la heterotaquia (asumiendo diferentes modelos de sustitución para diferentes linajes) en la selección de modelos de evolución podría mejorar las inferencias filogenéticas.

En lo referente al particionado en el análisis filogenómico, todavía quedan muchas cuestiones por resolver. En la vida real, fenómenos como la ordenación incompleta de linajes, duplicación y pérdida de genes y la transferencia genética lateral pueden derivar en un escenario diferente en el cual diferentes particiones evolucionan bajo diferentes árboles (i.e., el árbol de especies y el árbol de un gen particular pueden diferir). Por tanto, estudios posteriores podrían focalizarse en

evaluar el impacto del particionado en la obtención de árboles de especies cuando se asumen diferentes topologías para cada una de las particiones.

Estamos trabajando en una nueva librería de bajo nivel, que reemplazará a PLL, con la intención de soportar eficientemente tipos de datos (e.g., ADN, proteínas, codones) y modelos de heterogeneidad arbitrarios (tasas de sustitución siguiendo una distribución Γ discretizada, porporción de sitios invariantes, mixture models y heterotaquia). La nueva implementación facilita la construcción de herramientas de análisis filogenético más eficientes, así como la extensión de sus capacidades. Estamos también trabajando en una nueva herramienta de selección de modelos que sustituirá a *jModelTest*, *ProtTest* y *PartitionTest* incorporando la nueva librería PLL de bajo nivel.

Para obtener un buen balanceo de carga en los sets de datos particionados, cuando se optimizan los parámetros de los modelos vía máxima verosimilitud (ML, por sus iniciales en inglés) o se proponen nuevos valores para los parámetros (inferencia Bayesiana), es importante proponer y evaluar estos nuevos valores de forma simultánea para todas las particiones con el objetivo de mejorar la eficiencia [78]. Adicionalmente, se debe asignar las particiones a los diferentes procesadores de modo que se minimice el número de particiones que son compartidas entre varios procesadores, a la vez que se homogeneiza el número de sitios asignados a cada procesador [98]. Kobert et al. (2014) [50] mostraron que este problema de balanceo de carga es NP-Dificil, y han diseñado un algoritmo de aproximación en tiempo polinomial que garantiza una distribución cuasi-óptima de los sitios y las particiones a los procesadores. El algoritmo balancea la cantidad de sitios, y al mismo tiempo minimiza el número de particiones que son asignados a cada procesador. En esta configuración, una partición puede dividirse entre varios procesos. La implementación de este nuevo método en PLL está en proceso.

10.7. Principales Contribuciones

Las principales contribuciones de esta tesis son:

1. Diseño, implementación y evaluación de algoritmos de Computación de Al-

tas Prestaciones para la selección estadística del modelo de reemplazo de aminoácidos más adecuado para alineamientos de proteínas, incorporados en *ProtTest 3.0*.

2. Diseño, implementación y evaluación de algoritmos de Computación de Altas Prestaciones para la selección estadística de modelos de sustitución mecanísticos para alineamientos de ADN, incorporados en *jModelTest 2.0*. Este trabajo facilita el uso de arquitecturas HPC para seleccionar el modelo más adecuado. Proporciona también tolerancia a fallos a través de un sistema de *checkpointing*.

3. Diseño de nuevas técnicas de visualización para selección de modelos.

4. Métodos extendidos para la selección de modelos de ADN, soportando todos los 203 posibles esquemas de sustitución derivados de GTR (modelo reversible generalizado).

5. Métodos de análisis extendidos del impacto de los modelos de sustitución en conjuntos de datos particulares, tales como reportes HTML y un resumen topológico sobre los diferentes modelos candidatos.

6. Extensión de la librería de análisis filogenético PLL para dar añadir flexibilidad en el manejo de datos particionados.

7. Diseño, implementación y evaluación de algoritmos de Computación de Altas Prestaciones para evaluar correctamente la heterogeneidad proporcionada por el particionado de datos, incorporados en *PartitionTest*.

8. Evaluación del impacto de los diferentes esquemas de particionado en la inferencia filogenética.

Bibliography

[1] F. Abascal, D. Posada, and R. Zardoya. Mtart: a new model of amino acid replacement for arthropoda. *Molecular biology and evolution*, 24(1):1–5, 2007. pages 32

[2] F. Abascal, R. Zardoya, and D. Posada. Prottest: selection of best-fit models of protein evolution. *Bioinformatics*, 21(9):2104–2105, 2005. pages 32, 34

[3] Z. Abdo, V. Minin, P. Joyce, and J. Sullivan. Accounting for uncertainty in the tree topology has little effect on the decision-theoretic approach to model selection in phylogeny estimation. *Molecular Biology and Evolution*, 22:691–703, 2005. pages 26

[4] J. Adachi and M. Hasegawa. Improved dating of the human/chimpanzee separation in the mitochondrial dna tree: heterogeneity among amino acid sites. *Journal of Molecular Evolution*, 40(6):622–628, 1995. pages 21

[5] J. Adachi and M. Hasegawa. Model of amino acid substitution in proteins encoded by mitochondrial dna. *Journal of molecular evolution*, 42(4):459–468, 1996. pages 32

[6] H. Akaike. A new look at the statistical model identification. *IEEE T Automat Contr*, 19(6):716–723, 1974. pages 21, 25

[7] M. E. Alfaro and J. P. Huelsenbeck. Comparative performance of bayesian and aic-based measures of phylogenetic model uncertainty. *Systematic Biology*, 55(1):89–96, 2006. pages 27

[8] L. Arbiza, M. Patricio, H. Dopazo, and D. Posada. Genome-wide heterogeneity of nucleotide substitution model fit. *Genome biology and evolution*, 3:896, 2011. pages 31

[9] D. Berend and T. Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010. pages 29

[10] M. C. Brandley, A. Schmitz, and T. W. Reeder. Partitioned Bayesian analyses, partition choice, and the phylogenetic relationships of scincid lizards. *Systematic biology*, 54(3):373–390, 2005. pages 31

[11] J. M. Brown and A. R. Lemmon. The importance of data partitioning and the utility of bayes factors in bayesian phylogenetics. *Systematic Biology*, 56(4):643–655, 2007. pages 31

[12] W. J. Bruno and A. Halpern. Topological bias and inconsistency of maximum likelihood using wrong models. *Molecular Biology and Evolution*, 16:564–566, 1999. pages 21

[13] T. R. Buckley, C. Simon, and G. K. Chambers. Exploring among-site rate variation models in a maximum likelihood framework using empirical data: effects of model assumptions on estimates of topology, branch lengths, and bootstrap support. *Systematic Biology*, 50(1):67–86, 2001. pages 21

[14] K. Burnham and D. Anderson. Model selection and inference: a practical information-theoretic approach. *Springer-Verlag, New York, NY*, 1998. pages 27

[15] K. Burnham and D. Anderson. Model selection and multimodel inference: a practical information-theoretic approach. *Springer-Verlag, New York, NY*, 2003. pages 26, 27

[16] K. Burnham and D. Anderson. Multimodel inference: understanding aic and bic in model selection. *Sociological Methods and Research*, 33(2):261–304, 2004. pages 25

[17] F. H. Crick et al. The origin of the genetic code. *Journal of molecular biology*, 38(3):367–379, 1968. pages 8

[18] C. Cunningham, H. Zhu, and D. Hillis. Best-fit maximum-likelihood models for phylogenetic inference: empirical tests with known phylogenies. *Evolution*, pages 978–987, 1998. pages 21

[19] L. Dagum and R. Enon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998. pages 114, 128

[20] D. Darriba, A. Aberer, T. Flouri, T. Heath, F. Izquierdo-Carrasco, A. Stamatakis, et al. Boosting the performance of bayesian divergence time estimation with the phylogenetic likelihood library. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 539–548. IEEE, 2013. pages 115, 129

[21] D. Darriba, G. L. Taboada, R. Doallo, and D. Posada. jmodeltest 2: more models, new heuristics and parallel computing. *Nature methods*, 9(8):772–772, 2012. pages 27

[22] M. O. Dayhoff and R. M. Schwartz. A model of evolutionary change in proteins. In *In Atlas of protein sequence and structure*. Citeseer, 1978. pages 32

[23] J. Felsenstein. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Biology*, 27(4):401–410, 1978. pages 21

[24] J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376, Nov. 1981. pages 15, 18

[25] J. Felsenstein. A likelihood approach to character weighting and what it tells us about parsimony and compatibility. *Biological Journal of the Linnean Society*, 16(3):183–196, 1981. pages 23

[26] J. Felsenstein. *Inferring phylogenies*. Sinauer associates Sunderland, 2004. pages 17

[27] J. Felsenstein and G. A. Churchill. A hidden markov model approach to variation among sites in rate of evolution. *Molecular Biology and Evolution*, 13(1):93–104, 1996. pages 22

[28] W. M. Fitch and E. Margoliash. A method for estimating the number of invariant amino acid coding positions in a gene using cytochrome c as a model case. *Biochemical genetics*, 1(1):65–71, 1967. pages 19

[29] T. Flouri, F. Izquierdo-Carrasco, D. Darriba, A. Aberer, L.-T. Nguyen, B. Minh, A. Von Haeseler, and A. Stamatakis. The phylogenetic likelihood library. *Systematic biology*, 64(2):356–362, 2015. pages 115, 129

[30] O. Gascuel. Bionj: an improved version of the nj algorithm based on a simple model of sequence data. *Molecular biology and evolution*, 14(7):685–695, 1997. pages 117, 131

[31] N. Goldman. Statistical tests of models of dna substitution. *Journal of Molecular Evolution*, 36(2):182–198, 1993. pages 21, 22

[32] G. H. Gonnet, M. A. Cohen, and S. A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256(5062):1443–1445, 1992. pages 32

[33] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0. *Systematic biology*, 59(3):307–321, 2010. pages 31, 32

[34] S. Guindon and O. Gascuel. Phyml - a simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 52(5):696–704, 2003. pages 32

[35] M. Hasegawa, H. Kishino, and T.-a. Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *Journal of molecular evolution*, 22(2):160–174, 1985. pages 23

[36] J. Huelsenbeck, B. Larget, and M. Alfaro. Bayesian phylogenetic model selection using reversible jump markov chain monte carlo. *Molecular Biology and Evolution*, 21:1123–1133, 2004. pages 25

[37] J. P. Huelsenbeck. Performance of phylogenetic methods in simulation. *Systematic biology*, 44(1):17–48, 1995. pages 21

[38] J. P. Huelsenbeck and D. M. Hillis. Success of phylogenetic methods in the four-taxon case. *Systematic Biology*, 42(3):247–264, 1993. pages 21

[39] C. Hurvich and C. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76:297–307, 1989. pages 22

[40] E. D. Jarvis, S. Mirarab, A. J. Aberer, B. Li, P. Houde, C. Li, S. Y. Ho, B. C. Faircloth, B. Nabholz, J. T. Howard, et al. Whole-genome analyses resolve early branches in the tree of life of modern birds. *Science*, 346(6215):1320–1331, 2014. pages 28

[41] L. Jin and M. Nei. Limitations of the evolutionary parsimony method of phylogenetic analysis. *Molecular biology and evolution*, 7(1):82–102, 1990. pages 19

[42] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. *Mammalian protein metabolism*, 3:21–132, 1969. pages 12, 23

[43] D. Kainer and R. Lanfear. The effects of partitioning on phylogenetic inference. *Molecular biology and evolution*, page msv026, 2015. pages 31, 116, 130

[44] R. M. Karp. *Reducibility among combinatorial problems.* Springer, 1972. pages 28

[45] T. Keane, T. Naughton, and J. McInerney. Modelgenerator: amino acid and nucleotide substitution model selection. *National University of Ireland, Maynooth, Ireland*, 2004. pages 34

[46] T. M. Keane, C. J. Creevey, M. M. Pentony, T. J. Naughton, and J. O. Mclnerney. Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified. *BMC evolutionary biology*, 6(1):29, 2006. pages 32

[47] T. M. Keane, T. J. Naughton, and J. O. McInerney. Multiphyl: a high-throughput phylogenomics webserver using distributed computing. *Nucleic acids research*, 35(suppl 2):W33–W37, 2007. pages 32

[48] C. R. Kelsey, K. A. Crandall, and A. F. Voevodin. Different models, different trees: the geographic origin of ptlv-i. *Molecular phylogenetics and evolution*, 13(2):336–347, 1999. pages 21

[49] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16(2):111–20, 1980. pages 23

[50] K. Kobert, T. Flouri, A. Aberer, and A. Stamatakis. The divisible load balance problem and its application to phylogenetic inference. In *Algorithms in Bioinformatics*, pages 204–216. Springer, 2014. pages 120, 133

[51] B. Korte and J. Vygen. Combinatorial optimization: Theory and algorithms, 2012. pages 30

[52] C. Lanave, G. Preparata, C. Sacone, and G. Serio. A new method for calculating evolutionary substitution rates. *Journal of molecular evolution*, 20(1):86–93, 1984. pages 12

[53] R. Lanfear, B. Calcott, S. Y. Ho, and S. Guindon. Partitionfinder: combined selection of partitioning schemes and substitution models for phylogenetic analyses. *Molecular biology and evolution*, 29(6):1695–1701, 2012. pages 31

[54] S. Q. Le and O. Gascuel. An improved general amino acid replacement matrix. *Molecular biology and evolution*, 25(7):1307–1320, 2008. pages 32

[55] J. R. Leavitt, K. D. Hiatt, M. F. Whiting, and H. Song. Searching for the optimal data partitioning strategy in mitochondrial phylogenomics: A phylogeny of Acridoidea (Insecta: Orthoptera: Caelifera) as a case study. *Molecular Phylogenetics and Evolution*, 67(2):494–508, 2013. pages 31

[56] C. Li, G. Lu, and G. Orti. Optimal data partitioning and a test case for ray-finned fishes (actinopterygii) based on ten nuclear loci. *Systematic Biology*, 57(4):519–539, 2008. pages 31

[57] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. pages 22

[58] E. R. Mardis. The impact of next-generation sequencing technology on genetics. *Trends in genetics*, 24(3):133–141, 2008. pages 8

[59] I. Milne, D. Lindner, M. Bayer, D. Husmeier, G. McGuire, D. F. Marshall, and F. Wright. Topali v2: a rich graphical interface for evolutionary analyses of multiple alignments on hpc clusters and multi-core desktops. *Bioinformatics*, 25(1):126–127, 2009. pages 32

[60] V. Minin, Z. Abdo, P. Joyce, and J. Sullivan. Performance-based selection of likelihood models for phylogeny estimation. *Systematic Biology*, 52(5):674–683, 2003. pages 22, 26

[61] B. Misof, S. Liu, K. Meusemann, R. S. Peters, A. Donath, C. Mayer, P. B. Frandsen, J. Ware, T. Flouri, R. G. Beutel, et al. Phylogenomics resolves the timing and pattern of insect evolution. *Science*, 346(6210):763–767, 2014. pages 28

[62] L.-T. Nguyen, H. A. Schmidt, A. von Haeseler, and B. Q. Minh. Iq-tree: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular biology and evolution*, 32(1):268–274, 2015. pages 115

[63] J. A. Nylander, F. Ronquist, J. P. Huelsenbeck, and J. Nieves-Aldrey. Bayesian phylogenetic analysis of combined data. *Systematic biology*, 53(1):47–67, 2004. pages 31

[64] D. Posada. jmodeltest: phylogenetic model averaging. *Molecular biology and evolution*, 25(7):1253–1256, 2008. pages 32

[65] D. Posada and T. Buckley. Model Selection and Model Averaging in Phylogenetics: Advantages of Akaike Information Criterion and Bayesian Approaches Over Likelihood Ratio Tests. *Systematic Biology*, 53(5):793–808, 2004. pages 22, 23, 32

[66] D. Posada and K. Crandall. Modeltest: testing the model of dna substitution. *Bioinformatics*, 14(9):817–818, 1998. pages 32, 34

[67] D. Posada and K. Crandall. Selecting the best-fit model of nucleotide substitution. *Systematic Biology*, 50(4):580–601, 2001. pages 23

[68] A. F. L. A. Powell, F. K. Barker, and S. M. Lanyon. Empirical evaluation of partitioning schemes for phylogenetic analyses of mitogenomic data: An avian case study. *Molecular Phylogenetics and Evolution*, 66(1):69–79, 2013. pages 31

[69] F. Ren, H. Tanaka, and Z. Yang. A likelihood look at the supermatrix–supertree controversy. *Gene*, 441(1):119–125, 2009. pages 31

[70] R. L. S. Kullback. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951. pages 25

[71] F. Sanger, S. Nicklen, and A. R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977. pages 8

[72] S. C. Schuster. Next-generation sequencing transforms today's biology. *Nature*, 200(8):16–18, 2007. pages 8

[73] G. Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978. pages 22, 25

[74] A. Shafi, B. Carpenter, and M. Baker. Nested parallelism for multi-core HPC systems using Java. *J Parallel Distr Com*, 69(6):532–545, 2009. pages 114, 128

[75] B. Shapiro, A. Rambaut, and A. J. Drummond. Choosing appropriate substitution models for the phylogenetic analysis of protein-coding sequences. *Molecular Biology and Evolution*, 23(1):7–9, 2006. pages 31

[76] A. Stamatakis. Phylogenetic models of rate heterogeneity: a high performance computing perspective. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8–pp. IEEE, 2006. pages 22

[77] A. Stamatakis. Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, 2006. pages 31

[78] A. Stamatakis and M. Ott. Load balance in the phylogenetic likelihood kernel. In *Parallel Processing, 2009. ICPP'09. International Conference on*, pages 348–355. IEEE, 2009. pages 120, 133

[79] M. A. Suchard, C. M. Kitchen, J. S. Sinsheimer, and R. E. Weiss. Hierarchical phylogenetic models for analyzing multipartite sequence data. *Systematic Biology*, 52(5):649–664, 2003. pages 31

[80] N. Sugiura. Further analysts of the data by Akaike' s information criterion and the finite corrections. *Communication in Statistics - Theory and Methods*, 7(1):13–26, 1978. pages 22

[81] J. Sullivan and P. Joyce. Model selection in phylogenetics. *Annual Review of Ecology, Evolution, and Systematics*, pages 445–466, 2005. pages 32

[82] J. Sullivan, J. Markert, and C. Kilpatrick. Phylogeography and molecular systematics of the Peromyscus aztecus species group (Rodentia: Muridae) inferred using parsimony and likelihood. *Syst Biol*, 46:426–440, 1997. pages 21

[83] G. Taboada, J. T. no, and R. Doallo. Java for high performance computing: assessment of current research and practice. In *PPPJ '09: Proceedings of the 7th International Conference on Principles and Practice of Programming in Java*, pages 30–39, New York, NY, USA, 2009. ACM. pages 113, 127

[84] G. Taboada, J. T. no, and R. Doallo. F-mpj: Scalable java message-passing communications on parallel systems. *Journal of Supercomputing*, In press, 2011. pages 114, 128

[85] K. Tamura. Estimation of the number of nucleotide substitutions when there are strong transition-transversion and g+ c-content biases. *Molecular biology and evolution*, 9(4):678–687, 1992. pages 21

[86] K. Tamura and M. Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Molecular biology and evolution*, 10(3):512–526, 1993. pages 19

[87] S. Tavaré. Some probabilistic and statistical problems in the analysis of dna sequences. *Lectures on mathematics in the life sciences*, 17:57–86, 1986. pages 12, 23

[88] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2001. pages 30

[89] J. Wakeley. Substitution rate variation among sites in hypervariable region 1 of human mitochondrial dna. *Journal of Molecular Evolution*, 37(6):613–623, 1993. pages 19

[90] P. S. Ward, S. G. Brady, B. L. Fisher, and T. R. Schultz. Phylogeny and biogeography of dolichoderine ants: Effects of data partitioning and relict taxa on historical inference. *Systematic Biology*, 59(3):342–362, 2010. pages 31

[91] S. Whelan and N. Goldman. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular biology and evolution*, 18(5):691–699, 2001. pages 32

[92] Z. Yang. Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: approximate methods. *Journal of Molecular evolution*, 39(3):306–314, 1994. pages 20

[93] Z. Yang. Statistical properties of the maximum likelihood method of phylogenetic estimation and comparison with distance matrix methods. *Systematic biology*, 43(3):329–342, 1994. pages 21

[94] Z. Yang. A space-time process model for the evolution of dna sequences. *Genetics*, 139(2):993–1005, 1995. pages 21

[95] Z. Yang. Maximum-likelihood models for combined analyses of multiple sequence data. *Journal of Molecular Evolution*, 42(5):587–596, 1996. pages 31

[96] Z. Yang. *Computational molecular evolution*, volume 21. Oxford University Press Oxford, 2006. pages 12

[97] J. Zhang. Performance of likelihood ratio tests of evolutionary hypotheses under inadequate substitution models. *Molecular biology and evolution*, 16(6):868–875, 1999. pages 21

[98] J. Zhang and A. Stamatakis. The multi-processor scheduling problem in phylogenetics. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 691–698. IEEE, 2012. pages 120, 133

[99] A. Zharkikh. Estimation of evolutionary distances between nucleotide sequences. *Journal of Molecular Evolution*, 39(3):315–329, 1994. pages 23

[100] S. Zoller, V. Boskova, and M. Anisimova. Maximum-likelihood tree estimation using codon substitution models with multiple partitions. *Molecular biology and evolution*, page msv097, 2015. pages 31