MASTER'S DEGREE THESIS

Master of Science in Advanced Mathematics and Mathematical Engineering

Title: Generating Lie Markov models with prescribed symmetry

Author: Ane Santos Herranz

Advisor: Jesús Fernández-Sánchez

Department: Departament de Matemàtica Aplicada I

Academic year: 2011-2012

Facultat de Matemàtiques i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master Thesis

Generating Lie Markov models with prescribed symmetry

Ane Santos Herranz

Advisor: Jesús Fernández Sánchez

Departament de Matemàtica Aplicada I

Abstract

Recent works discuss the importance of the multiplicative closure of the Markov models used in phylogenetics. For continuous-time Markov chains, a sufficient condition to ensure the multiplicative closure is the requiring the set of rate-matrices of the model to form a Lie algebra. It is known that some well-known Markov models form a Lie algebra, while there are others that do not, as the GTR model (see [**19**]). The [**18**] and [**5**] papers discuss a possible definition for a new type of multiplicatively closed models, that generalizes the definition of group-based models [**15**] and equivariant models [**4**]. In particular, in [**18**] an effective algorithm to compute Lie Markov models with prescribed symmetry given by a permutation group of the A,C,G,T nucleotides is proposed.

In this work, we want to implement this algorithm to generate all models for a given symmetry. This implementation would be important for getting examples and investigate more properties of these models.

Keywords: Phylogenetics, Markov model, Representation theory, Permutation groups, Lie algebras

MSC2000: 60J27,70G65,68R99

Contents

Introduction

Phylogenetics try to reconstruct the relationships between different species. Determining the evolutionary lineage of organisms is one of the hardest and more challenging problems in the study of evolution in general, and of phylogenetics in particular.

While phylogenetic trees are used to represent the evolution of some set of taxa, evolutionary models are needed to describe the evolution of molecular sequences. Most of the commonly implemented molecular evolutionary models in phylogenetics are based on the continuous-time Markov assumption. For these models, substitution events are ruled by substitution rates. For example, for nucleotide models, twelve rates must be specified and precise characteristics of the process can be imposed by constraints on these rates.

The motivation behind the papers [**5**, **18**] was to consider the possibility that at some point of some evolutionary process substitution rates change. The resulting *non-homogenous* evolutionary process can still be modelled as a continuous-time Markov chain, but we must allow the rates to vary as a function of time. This leads to considering evolutionary model classes that are "multiplicatively closed": for them, it is possible to interpret the whole process as a *homogenous* Markov process. Many often-used evolutionary models, such as the general time reversible model, are not multiplicatively closed and this lack of closure poses a problem for phylogenetic analysis in both flexibility of interpretation and as a potential source of model misspecification (see [**19**]). These ideas lead to the definition of Lie Markov models (see Definition 1.4.4): roughly speaking, these are models where the rate matrices form a Lie subalgebra on the space of rate matrices. The reason for this condition is the Baker-Campbell-Hausdorff formula (see Section 1.4) that connects the Lie bracket of rate matrices with the multiplicative closure of substitution matrices. The request of some symmetries for these models appears then as a convenient and practical condition.

In [**18**], the authors suggest a procedure to produce this class of models with the assistance of group representation theory and Lie theory. The goal of this work has been the practical implementation of this procedure. To this aim, it has been crucial to understand and use the tools and technical facts that take part in the theory. Representation and Group action theory play a major role, and so does the

connection between Lie algebras and Lie groups via the exponential map. Geometrical and combinatorial facts are also needed to give a convenient presentation of the models.

The code for this implementation has been written in *Python* language. The main reason for this choice was that Python is the programming language used in the SAGE environment. While we could have used other programming languages, such as **C**, the SAGE environment has been fundamental by permitting the use of many existing open-source packages (in our case, GAP and maxima) combined with Python packages and our own Python functions, and everything in the same environment. Thus, it has been necessary to learn the Python language and some GAP to achieve the goal.

In this memoir we put in writing the work carried on. The memoir has been structured in two parts: Background and Implementation.

The *Background* contains the basic results needed to understand the theory of Lie Markov models. It has been divided into 3 chapters that pave the way towards the definition of Lie Markov model (Chapter 3).

In Chapter 1 we recall the definition of evolutionary model, give some examples and try to motivate the need of multiplicative closure as a good mathematical property evolutionary models should have.

In Chapter 2, we state the main facts on group theory and representation theory needed. The Maschke decomposition into irreducible representations, as well as the use of characters or the Young projections of a permutation group will play a key role in the implementation of Chapter 4.

Finally, in Chapter 3, we introduce the Lie Markov models and the Lie Markov models with prescribed symmetry. We explore the definition and see how these models give rise to convex polyhedral cones with nice properties. At the same time, a procedure taken from [**18**] to generate Lie Markov models is described.

The *implementation* represents the main part of the work and it consists on the description of our code to produce all Lie Markov models with some prescribed symmetry G. This code has been written following to some extent the procedure described in Chapter 3. The chapter has been divided into three sections, each one corresponding to a different line of computations: basic computations, computations depending on the group G and final determination of the Lie Markov models.

As an appendix, the memoir contains some examples of application of the code to determine explicitly the Lie Markov models with

- $n = 2$ states and symmetry $\mathfrak{S}_2$;
- $n = 4$ states and symmetry $\mathfrak{S}_4$;
- $n = 4$ states and symmetry $G = \langle (12), (1324) \rangle < \mathfrak{S}_4$.

Part 1

Background

Chapter 1
Preliminaries and motivation

In this chapter, we introduce the main concepts and facts that will be needed thoughout this memoir. At the same time, we introduce notation and try to explain the original problem that motivates the definition of Lie Markov model in Chapter 3.

1.1. Evolutionary models

An evolutionary model describes the theoretical evolutionary process in which a molecular sequence of some species or taxon evolves into another one. Consider a nucleotide sequence where each site evolves under the same hetereogeneous Markov model. For time $t_0 = 0$ to t, the sequence evolves under a time-homogeneus Markov process so that substitutions are governed by some evolutionary rates, which are arranged in a rate matrix.

DEFINITION 1.1.1. *A* rate matrix *is a square matrix $Q = (q_{ij})$ with rows and columns indexed by the ordered set $\Sigma = \{A, C, G, T\}$, and satisfying*

(1) $q_{x,y} \geq 0, \forall x \neq y$,
(2) $\sum_x q_{x,y} = 0, \forall x$ *(columns sum to 0),*
(3) $q_{x,x} < 0$.

The probabilities of change between nucleotides are the entries of the substitution matrix S:

DEFINITION 1.1.2. *A* substitution matrix *is a matrix formed by conditional probabilities*

$$S = \begin{pmatrix} p_{A|A} & p_{C|A} & p_{G|A} & p_{T|A} \\ p_{A|C} & p_{C|C} & p_{G|C} & p_{T|C} \\ p_{A|G} & p_{C|G} & p_{G|G} & p_{T|G} \\ p_{A|T} & p_{C|T} & p_{G|T} & p_{T|T} \end{pmatrix}$$

where $p_{i|j}$ is the probability that the nucleotide i evolves to the nucleotide j. Note that the sum of each row is 1.

This substitution matrix S is given by exponentiation of Q:

$$S = e^{Qt} = \sum_{n\geq 0} \frac{Q^n}{n!} t^n.$$

According to the structure and symmetry of the rate matrices (or substitution matrices), we have different evolutionary Markov models. In a *continuous-time evolutionary model*, the parameters are taken as the evolutionary rates, together with some base distribution vector $(\pi_A, \pi_C, \pi_G, \pi_T)$, with $\pi_A, \pi_C, \pi_G, \pi_T \geq 0$ and $\pi_A + \pi_C + \pi_G + \pi_T = 1$.

EXAMPLE 1.1.3. Jukes-Cantor model (JC69)*: Assumes equal base frecuencies* $(\pi_A = \pi_C = \pi_G = \pi_T)$ *and equal mutation rates:*

$$Q = \begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix}$$

EXAMPLE 1.1.4. Kimura 2-parameter model (K80)*: The model assumes equal base frequencies and considers different rates for the mutations from purine to purine, and pyrimidine to pyrimidine (transitions) and the mutations from purine to pyrimidine or vice versa (transversions):*[1]

$$Q = \begin{pmatrix} \cdot & \beta & \alpha & \beta \\ \beta & \cdot & \beta & \alpha \\ \alpha & \beta & \cdot & \beta \\ \beta & \alpha & \beta & \cdot \end{pmatrix}$$

where $\cdot = 1 - (\alpha + 2\beta)$.

EXAMPLE 1.1.5. Kimura 3-parameter model (K81)*: The model assumes equal base frequencies and takes a parameter for transitions and a parameter for each transversion. The rate matrix has the form:*

$$Q = \begin{pmatrix} \cdot & \beta & \gamma & \delta \\ \beta & \cdot & \delta & \gamma \\ \gamma & \delta & \cdot & \beta \\ \delta & \gamma & \beta & \cdot \end{pmatrix}$$

where $\cdot = 1 - (\beta + \gamma + \delta)$.

EXAMPLE 1.1.6. Felsenstein 81 model (F81)*: Base frequencies are allowed to vary from* $(\pi_A \neq \pi_C \neq \pi_G \neq \pi_T)$*. The rate matrix has the form:*

$$Q = \begin{pmatrix} \cdot & \pi_C & \pi_G & \pi_T \\ \pi_A & \cdot & \pi_G & \pi_T \\ \pi_A & \pi_C & \cdot & \pi_T \\ \pi_A & \pi_C & \pi_G & \cdot \end{pmatrix}$$

[1]Purines are $\{A, C\}$ and pyrimidines are $\{G, T\}$

EXAMPLE 1.1.7. (HKY85) *[9]: It allows for different transitions and transversions rates and different base frequencies. The rate matrix has the form:*

$$Q = \begin{pmatrix} \cdot & \beta\pi_C & \alpha\pi_G & \beta\pi_T \\ \beta\pi_A & \cdot & \beta\pi_G & \alpha\pi_T \\ \alpha\pi_A & \beta\pi_C & \cdot & \beta\pi_T \\ \beta\pi_A & \alpha\pi_C & \beta\pi_G & \cdot \end{pmatrix}$$

EXAMPLE 1.1.8. General time-reversible model (GTR) *[10] [20]: The parameters for nucleotides consist of an equilibrium base frequency vector* $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)$*, giving the frecuency at which each base occurs at each site.*

$$Q = \begin{pmatrix} \cdot & \alpha\pi_C & \beta\pi_G & \gamma\pi_T \\ \alpha\pi_A & \cdot & \delta\pi_G & \epsilon\pi_T \\ \beta\pi_A & \delta\pi_C & \cdot & \phi\pi_T \\ \gamma\pi_A & \epsilon\pi_C & \phi\pi_G & \cdot \end{pmatrix}$$

In an *algebraic evolutionary model*, the parameters are taken as the probabilities of mutation together with the base frequencies.

EXAMPLE 1.1.9. *Algebraic versions of the previous models JC69, K80 and K81 are obtained when we take the parameters of the models as the probabilities of mutation. Notice that these three models, the structure of the substitution matrices is the same as the structure of the rate matrices.*

EXAMPLE 1.1.10. Strand symmetric model (SSM) *[1, 23]: It takes into account the double strand symmetry of DNA,* $A - T, C - G$*. The substitution matrix has the form:*

$$S = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ h & g & f & e \\ d & c & b & a \end{pmatrix}$$

EXAMPLE 1.1.11. General Markov model (GMM)*: The most general algebraic model. The substitution matrix has the form:*

$$S = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

In this work, we will deal with continuous-time evolutionary models of the form $e^{\mathfrak{L}}$ for some well-defined set $\mathfrak{L}$ of rate matrices. We will call these models *rate-matrix models* $e^{\mathfrak{L}}$. Sometimes, we will abuse our terminology and refer to $\mathfrak{L}$ as the "model".

1.2. Evolutionary models on a tree

DEFINITION 1.2.1. *A* phylogenetic tree *T is a connected graph with no cycles where the nodes represents species or taxons. The interior vertices represent the ancestral species and the terminal vertices or* leaves *represent the current species. The* edges *of the graph represent the evolutinary history.*

When there is a vertex representing the common ancestor of all the species, the tree is a rooted phylogenetic tree *and the vertex is called the* root *of the tree. The root induces an orientation of the tree to the leaves. See Figure 1.2.1 for some examples of rooted phylogenetic trees with 4 leaves.*

FIG. 1. Three different rooted trees.

The topology *of a tree is its topology as subspace of the plane when the leaves are labeled. For example, the three trees of Figure 1 have different topology as rooted trees, while the three trees of Figure 1.2 have the same topology.*

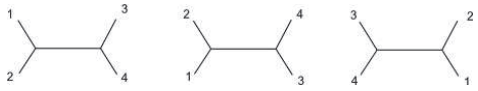


FIG. 2. Three unrooted trees with the same topology.

A phylogenetic tree stores two kinds of information: The topology of the labeled graph represents the speciation events occurred along the evolutionary history, while the length of the edges represent the time elapsed between the two speciation events. The length of an edge of the phylogenetic tree represents the number of mutations that occur between the species at the ends of the edge.

DEFINITION 1.2.2. *A* continuous-time evolutionary model on a tree T *is specified by a phylogenetic tree T with a rate matrix Q_e for each edge e and an initial distribution π for the root of T. The branch lengths $t_e \geq 0$ of the edges and the entries of the rate matrices are the parameters of the model. The substitution matrix of the edge e is given by*

$$S^e = e^{Q_e t_e}.$$

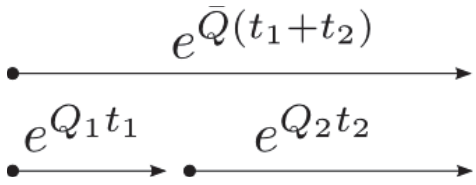
Continuous-time evolutionary models on a tree can be *homogeneous* or *non-homogeneous* models. A *homogeneous* model is an evolutionary model where the rate matrices of all the edges of the phylogenetic tree are equal. Although, this kind of models are useful because they are computationally convenient, they are not realistic since evolution does not occur in a homogeneous way [**11**] and they are taken as an approximation to the real evolutionary process. On the other hand, in the non-homogeneous models, rates may change at branching events. Another approach is to abandon the continuous hypothesis and work with discrete Markov chains, that is, with algebraic evolutionary models (see [**13**]).

DEFINITION 1.2.3. *An* algebraic evolutionary model on a tree T *is specified by a phylogenetic tree T with a substitution matrix S_e for each edge e and an initial distribution π for the root of T. The parameters of the model are taken as the entries of the transition matrices, that is, the probabilities of nucleotide substitutions.*

1.3. Multiplicative closure

Consider that we have a molecular sequence that evolves over time. Suppose that from time $t_0 = 0$ to t_1 the sequence evolves under a homogeneous Markov model with a rate matrix Q_1, so that the corresponding substitution matrix is $M_1 = e^{Q_1 t_1}$. Then, the sequence starts to evolve under a Markov process again, but this time, with a rate matrix Q_2, so that the corresponding substitution matrix is $M_2 = e^{Q_2 t_2}$.

The substitution matrix that describes the evolution of the sequence from $t_0 = 0$ to $t = t_1 + t_2$ is the multiplication of the matrices M_1 and M_2, $\overline{M} = M_1 M_2$. At this point , one question arises: *is there a (unique) real rate matrix Q such that $\overline{M} = e^{\overline{Q}(t1+t2)}$?*(see Figure 1.3). It can be shown that the answer to this question is yes under some restrictions (see [**2**]). But then, if $Q_1 Q_2$ are taken in some evolutionary model, *can we be sure that $\overline{Q}$ will still be in the model?* Notice that an affirmative answer to this question is needed if we want to consider constant rates along each branch of a tree. We would like $\overline{Q}$ to be a rate matrix of the same model of Q_1 and Q_2; that is, we would like our model to be *closed under multiplication*. However, this property is not always true in all the models, and in the case in which the model is not multiplicatively closed, it is possible that some error in estimation of substitution rates or speciation times occur (see [**19**]).



Definition 1.3.1. *An evolutionary Markov model is* multiplicatively closed *if for any couple of substitution matrices S_1, S_2, we also have $S_1 S_2$ is still in the model.*

An example of a not multiplivatively closed model is the GTR model of Example 1.1.8 (see [**18**])).

1.4. Lie algebras and the Baker-Campbell-Hausdorff formula.

A *Lie Algebra* over $\mathbb{C}$ is a vector space $\mathfrak{g}$ over $\mathbb{C}$ with a binary operation

$$[.,.] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g},$$

called the *Lie bracket*, which satisfies:

(1) Bilinearity:

$$[ax + by, z] = a\,[x, z] + b\,[y, z]$$
$$[z, ax + by] = a\,[z, x] + b\,[z, y]$$

$\forall a, b \in \mathbb{C}, \forall x, y, z \in \mathfrak{g}$,

(2) Alternating on $\mathfrak{g}$:
$$[x, x] = 0, \forall x \in \mathfrak{g},$$
(3) The Jacobi identity:
$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0, \forall x, y, z \in \mathfrak{g}.$$

A subspace $\mathfrak{h} \subseteq \mathfrak{g}$ that is closed under the Lie braket is called a *Lie subalgebra*.

EXAMPLE 1.4.1. *Given two matrices $A, B \in M_n(\mathbb{C})$, their* conmutator *or* Lie bracket *is defined by*
$$[A, B] = AB - BA.$$
Then, $M_n(\mathbb{C})$ is a Lie algebra with this operator. The subspace of all the matrices in $M_n(\mathbb{C})$ with trace cero forms a Lie subalgebra of $M_n(\mathbb{C})$.

EXAMPLE 1.4.2. *The elementary rate matrices are defined as $\{L_{ij} : 1 \leq i \neq j \leq n\}$, where each L_{ij} is a $n \times n$ matrix with all its entries zero, except for the entry ij, which is 1, and the entry jj, which is -1. For example, if $n = 3$:*
$$L_{12} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad L_{31} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

LEMMA 1.4.3. *([**18**, Result 2])The elementary rate-matrices provided with the commutator $[A, B] = AB - BA$ generate a Lie subalgebra of $M_n(\mathbb{C})$.*

PROOF. The claim is a consequence of the following equality:
$$[L_{ij}, L_{kl}] = (\delta_{jk} - \delta_{jl})(L_{il} - L_{jl}) - (\delta_{il} - \delta_{jl})(L_{kj} - L_{lj}).$$
□

DEFINITION 1.4.4. *We will denote this subalgebra by $\mathfrak{L}_{GMM} = \langle\{L_{ij}\}\rangle_{\mathbb{C}}$ and call it the* general rate-matrix algebra.

By taking the image of a Lie subalgebra $\mathfrak{L} \subset M_n(\mathbb{C})$ under the exponential map
$$exp(A) = \sum_{n \geq 0} \frac{A^n}{n!},$$
we construct a rate-matrix model. Because of the Baker-Campbell-Hausdorff formula (see below), it can be shown that the exponential map and the Lie subalgebra determine the *local* group structure of the model as a (complex) *Lie matrix group*, that is, a subgroup of $GL(n, \mathbb{C})$. Recall that a *Lie group* $\mathcal{G}$ is a group which is also a finite-dimensional smooth manifold and in which the group operations of multiplication ($\mathcal{G} \times \mathcal{G} \to \mathcal{G}$) and inversion ($g \to g^{-1}$) are smooth (they have derivatives of all orders). Conversely, a Lie algebra can be associated to every Lie group $\mathcal{G}$ by taking the tangent space of $\mathcal{G}$ (as a manifold) at the identity:
$$\mathcal{L} = T_{\mathbf{1}}(\mathcal{G}) = \{\gamma'(0) : \gamma : (-\epsilon, \epsilon) \to \mathcal{G} \text{ smooth with } \gamma(0) = \mathbf{1}\}.$$

EXAMPLE 1.4.5. *The elementary rate matrices form a $\mathbb{C}$-basis of the tangent space of $GL_1(n, \mathbb{C})$.*

If $\mathcal{G}$ is a complex matrix group, then its complex dimension is $dim_{\mathbb{C}}G = dim_{\mathbb{C}}T_{\mathbf{1}}(G)$.

Baker-Campbell-Hausdorff formula

The *Baker-Campbell-Hausdorff formula* [**8**] is the solution to

$$Z = log(e^X e^Y)$$

for noncommutative X and Y. The first terms are:

$$\begin{aligned} Z &= log(e^X e^Y) = X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}[X,[X,Y]] - \\ &\quad -\frac{1}{12}[Y,[X,Y]] - \frac{1}{24}[Y,[X,Y]] - \dots \end{aligned}$$

This formula links Lie groups with Lie algebras. It generalizes the well-known rule for the product of two exponentials, $e^X e^Y = e^{X+Y}$, to the case of non-commutative variables. In particular, it implies that if X and Y are in some Lie subalgebra $\mathfrak{L}$ of $\mathfrak{L}_{GMM}$, then $log(e^X e^Y)$ can be written as *a formal infinite sum of elements of* $\mathfrak{L}$.

The Baker-Campbell-Hausdorff formula is the reason to the requirement that rate matrices should form a Lie subalgebra $\mathfrak{L}$ of $\mathfrak{L}_{GMM}$ in the definition of Lie Markov model (see Definition 3.1.2). In this case, the product of two substitution matrices is still in the rate-matrix model $e^{\mathfrak{L}}$ and multiplicative closure is guaranteed (see [**18**] for details).

Chapter 2
Groups and representation theory

We recall basic facts concerning group actions and group representation theory that will be needed in Chapter 3 to describe the procedure to construct Lie Markov models with prescribed symmetries.

2.1. Background on groups and groups actions

Throughout this chapter, G will represent a finite group.

DEFINITION 2.1.1. *Given a subgroup H of G and an element $g \in G$, the set*

$$gH = \{gh : h \in H\}$$

is called a left coset *of H in G. Similarly, the set*

$$Hg = \{hg : h \in H\}$$

is called a right coset *of H in G.*

Any two right (or left) cosets of H in G are either the same or disjoint. Actually, the list of different cosets of H gives a partition of G:

$$G = \cup_{i=1}^{t} \sigma_i H,$$

for convenient $\sigma_1, \sigma_2, \ldots, \sigma_t \in G$. Such a set is called a *transversal* for H in G.

EXAMPLE 2.1.2. *Let G be the additive group $\mathbb{Z}/(4) = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}\}$ and $H = (\overline{2})$. The left cosets of H in G are*

$$\begin{aligned} \overline{0} + H &= \overline{2} + H = \{\overline{0}, \overline{2}\} \\ \overline{1} + H &= \overline{3} + H = \{\overline{1}, \overline{3}\} \end{aligned}$$

EXAMPLE 2.1.3. *Let G be the group $\mathfrak{S}_3$ and $H = \langle (12) \rangle = \{e, (1,2)\}$. There are three different left cosets of H in G:*

$$\begin{aligned} eH &= H = \{e, (12)\} \\ (13)\, H &= \{(13), (123)\} \\ (23)\, H &= \{(23), (132)\} \end{aligned}$$

When there is no risk of confusion, we will write $[\sigma_i]$ for $\sigma_i H$, and

$$G/H = \{[\sigma_1], [\sigma_2], \ldots, [\sigma_t]\}$$

for the set of all left cosets of H in G.

THEOREM 2.1.4 (Lagrange). *Given a finite group G and $H \leq G$, then $|H|$ divides $|G|$ and $|G/H| = |G|/|H|$, where $|\cdot|$ means cardinality.*

If H is normal, the right and the left cosets of H coincide. In this case, the set $G/H = \{gH : g \in G\}$ inherits a group structure from that of G, and G/H is called the *quotient group*.

EXAMPLE 2.1.5. *Let G be $\mathfrak{S}_4$ and $H = \{e, (12)(34), (13)(24), (14)(23)\}$, which is a normal subgroup of G. The quotinet group is $\mathfrak{S}_4/H$ and the cosests are:*

$$\begin{aligned}
eH &= \{e, (12)(34), (13)(24), (14)(23)\} = (14)(23)H = (12)(34)H = (13)(24)H \\
(12)H &= \{(12), (34), (1324), (1423)\} = (34)H = (1324)H = (1423)H \\
(13)H &= \{(13), (1234), (24), (1432)\} = (234)H = (24)H = (1432)H \\
(23)H &= \{(23), (1342), (1243), (14)\} = (1342)H = (1243)H = (14)H \\
(123)H &= \{(123), (134), (243), (142)\} = (134)H = (243)H = (143)H \\
(132)H &= \{(132), (234), (124), (143)\} = (234)H = (124)H = (143)H
\end{aligned}$$

Two elements $g_1, g_2 \in G$, are called *conjugate* if there exists an element $g \in G$ such that $gg_1g^{-1} = g_2$. This is denoted by $g_1 \sim g_2$. *Conjugacy* is an equivalence relation and the equivalence class of an element $g \in G$,

$$[g] = \left\{hgh^{-1} : h \in G\right\}$$

is called the *conjugacy class* of g. The identity element is the only element in its own class: $[e] = \{e\}$.

EXAMPLE 2.1.6. *The conjugacy classes of $\mathfrak{S}_4$ are:*

$$\begin{aligned}
&\{e\} \\
&\{(12), (13), (14), (23), (24), (34)\} \\
&\{(12)(34), (14)(23), (13)(24)\} \\
&\{(123), (132), (234), (243), (341), (314), (412), (421)\} \\
&\{(1234), (1243), (1324), (1342), (1423), (1432)\}
\end{aligned}$$

DEFINITION 2.1.7. *Given a set S,* a group action of G on S *is a binary operator*

$$\circ : G \times S \longrightarrow S$$

that satisfies:

(1) $e \circ b = b$, $\forall b \in S$, and
(2) $(gh) \circ b = g \circ (h \circ b)$, $\forall g, h \in G$ and $b \in S$.

When this happens, G is said to act on *S.*

EXAMPLE 2.1.8. *(*Defining action*). The symetric group $\mathfrak{S}_n$ acts on $S = \{1, \ldots, n\}$ in the natural way. Let us check that this action satisfies the conditions of Definition 2.1.7:*

(1) for any $r \in \{1, \ldots, n\}, e \circ r = r$,
(2) for any $\sigma, \tau \in \mathfrak{S}_n, (\sigma\tau) \circ r = \sigma(\tau(r)) = \sigma \circ (\tau \circ r)$.

REMARK 2.1.9. *Notice that whenever a group G acts on a finite set $S = \{s_1, \ldots, s_n\}$, this action induces a group homomorphism $\rho : G \longrightarrow \mathfrak{S}_n$.*

EXAMPLE 2.1.10. *Let G be the group $\mathbb{Z}/(2) = \{-1, +1\}$ and $S = \mathbb{R}$. An action of G on $\mathbb{R}$ is induced by multiplication of real numbers.*

EXAMPLE 2.1.11. *(*Regular action*). Let G be a group and take $S = G$. Then G acts on S by left multiplication. That is, $g \circ s$ is defined as the ordinary product of g and s: $g \circ s = gs$.*

EXAMPLE 2.1.12. *(*Conjugacy on elements*). Let G be a group and take $S = G$. Define an action of G on S by conjugation. That is, if $g \in G$ and $s \in S$, then the action is defined by*

$$(g, s) \longmapsto g \circ s = gsg^{-1}$$

Let us check that this definition satisfies the conditions to be an action:

(1) for any $s \in S, e \circ s = ese^{-1} = s$,
(2) for any $g, h \in G, s \in S$, $(gh) \circ s = (gh)s(gh)^{-1} = g(hsh^{-1})g^{-1} = g \circ (h \circ s)$.

EXAMPLE 2.1.13. *(*Conjugation on subgroups*). If H is a subgroup of G and $g \in G$, $g \circ H = gHg^{-1}$ is called the* conjugate subgroup of H by g*. Thus, we have an action of G on the set of subsets of G.*

EXAMPLE 2.1.14. *Let H be a subgroup of G. Then, there is an action of G in the (left) cosets of H in G by taking $g \circ [\sigma_i] = [g\sigma_i]$, where $[\tau]$ represents τH.*

DEFINITION 2.1.15. *Given a group G that acts on a set S, an element $s \in S$ is mapped by the elements of G to other elements in S. The set of all the images of s is called the* orbit of s under G *and is denoted by*

$$Gs := \{g \circ s : g \in G\}.$$

The set of all elements in G that fix a specific point $s \in S$ is called the stabilizer of s in G *and is denoted by*

$$Stab_G(s) := \{g \in G : g \circ s = s\}.$$

REMARK 2.1.16. *If $s_1, s_2 \in S$, then the two orbits Gs_1 and Gs_2 are either equal or disjoint, so the set of all orbits is a partition of S.*

EXAMPLE 2.1.17. *When we consider the actions by conjugation of G on itself (see Example 2.1.12) we obtain that*

(1) $Gs = \left\{gsg^{-1} : g \in G\right\}$ is the conjugacy class of s,
(2) $Stab_G(s) = \left\{g : gsg^{-1} = s\right\} = \{g : gs = sg\}$ is the centralizer of s.

Similarly, when we consider the action by conjugation on subgroups (see Example 2.1.8):

(1) $GH = \left\{gHg^{-1} : g \in G\right\}$ *is the collection of conjugate subgroups of* H*,*
(2) $Stab_G(H) = \left\{g : gHg^{-1} = H\right\}$*, which is called the* normalizer *of* H*,* $N_g(H)$*.*

EXAMPLE 2.1.18. *Let* G *be the group* $\{(1), (12), (34), (12)(34)\}$ *and* $S = \{1, 2, 3, 4\}$*. The orbit of* 1 *and* 2 *is* $\{1, 2\}$ *and the orbit of* 3 *and* 4 *is* $\{3, 4\}$*.*

The *orbit stabilizer theorem* states that up to bijective correspondence, every orbit in S under G has the form of a quotient G/H for some subgroup H of G, and then, H is the stabilizer of some element $s \in S$.

THEOREM 2.1.19 (Orbit-stabilizer theorem). *[3] Given* $s \in S$*, there is a natural bijection*

$$\begin{array}{ccc} G/Stab_G(s) & \longrightarrow & Gs \\ [h] & \longmapsto & hs \end{array}.$$

Conversely, if $H \leq G$ *is a subgroup, there is an action of* G *on the cosets* $G/H = \{[e], [\sigma_1], [\sigma_2], \ldots, [\sigma_q]\}$ *by taking* $\sigma \circ [\sigma_i] = [\sigma \circ \sigma_i]$*.*

REMARK 2.1.20. *The orbit-stabilizer theorem together with the Lagrange theorem gives*

$$|Gs| = |G/Stab_G(s)| = |G|/|Stab_G(s)|.$$

Since there are only finitely many subgroups of G, it is possible to give a complete list of all the orbits under G by listing all possible G/H for H a subgroup of G. This will be specially important in the procedure described in Section 3 of Chapter 3.

2.2. Representation theory

A *representation* of a group G is a group homomorphism from G to the general linear group $GL(V)$ of some finite-dimensional vector space V over $\mathbb{C}$, that is, a map

$$\rho : G \longrightarrow GL(V)$$

such that

$$\rho(g_1 g_2) = \rho(g_1)\rho(g_2),$$

for all $g_1, g_2 \in G$. We refer to V as the *module* associated to the representation ρ, or as the representation itself when the homomorphism is clear. If V has dimension n, we choose a basis for V and identify $GL(V)$ with $GL(n, \mathbb{C})$, the group of $n \times n$ invertible matrices with entries in the field $\mathbb{C}$. The *degree* of a representation is the dimension of the corresponding module.

EXAMPLE 2.2.1. *If* G *acts on a finite set* S*, we can turn this action into a representation of* G *by considering the vector space generated by* S*,* $V = \langle S \rangle_{\mathbb{C}}$ *and extending the action linearly.*

- *(*Trivial representation*). Let* G *be a finite group. The trivial representation of* G *is given by*

$$\begin{array}{rcl} \rho_{id} & : & G \longrightarrow GL(\mathbb{C}) \\ & & g \longmapsto [a \to a]. \end{array}$$

This is induced by the trivial action of G in the set $S = 1$.

- *(*Regular representation*). Let G be a finite group and $V = \langle G \rangle_{\mathbb{C}} = \left\{ \sum_{g \in G} a_g g : a_g \in \mathbb{C} \right\}$. The regular representation of G is given by*

$$\begin{aligned} \rho \quad : \quad & G \longrightarrow GL(V) \\ & \sigma \longmapsto [\sum_{g \in G} a_g g \to \sum_{g \in G} a_g (\sigma g)]. \end{aligned}$$

This is induced by the regular action of G (see Example 2.1.11).

- *(*Defining representation*). Let G be the symmetric group $\mathfrak{S}_n$. Take $V = \mathbb{C}^n$ and choose a basis $\{e_1, e_2, \ldots, e_n\}$ for V. The* defining representation *of $\mathfrak{S}_n$ is given by*

$$\begin{aligned} \rho \quad : \quad & \mathfrak{S}_n \longrightarrow GL(V) \\ & \sigma \longmapsto [\sum_{i=1}^{n} a_i e_i \to \sum_{i=1}^{n} a_i e_{\sigma(i)}]. \end{aligned}$$

This representation is induced by the defining action (see Example 2.1.8). For example, if $n = 2$, the defining representation maps the identity element of $\mathfrak{S}_2$ to the identity of V and the permutation (12) to the linear map $f : V \longrightarrow V$ defined by $f(e_1) = e_2$, $f(e_2) = e_1$.

- *If H is a subgroup of G, we turn the set G/H into a representation of G taking*

$$V = \langle G/H \rangle_{\mathbb{C}} = \langle [\sigma_1], [\sigma_2] \ldots, [\sigma_t] \rangle_{\mathbb{C}}.$$

This representation is induced by the action described in Example 2.1.14.

Example 2.2.2. *(*Sign representation*). Take $G = \mathfrak{S}_n$ and $V = \mathbb{C}$. The* sign representation *of G is given by*

$$\begin{aligned} \rho_{sgn} \quad : \quad & \mathfrak{S}_n \longrightarrow GL(\mathbb{C}) \\ & g \longmapsto [a \to sgn(g)a], \end{aligned}$$

where $sgn(g)$ means the sign of the permutation g:

$$\begin{aligned} sgn(g) &= +1, \text{ if } g \text{ is even}, \\ sgn(g) &= -1, \text{ if } g \text{ is odd}. \end{aligned}$$

Definition 2.2.3. *A representation $\rho : G \to GL(V)$ is said to be a* permutation representation *if it acts by permuting the elements of a basis B of V. For example, the trivial or the defining representation are examples of permutation representation, while the sign representation is not. Notice that every permutation representation induces an action of the group G in B, so we can decompose B into G-orbits. Conversely, every permutation representation arises in this way: we take a collection of G-orbits: $G/H_1, G/H_2, \ldots, G/H_t$ and define the abstract vector space generated by all the elements in these orbits.*

Definition 2.2.4. *Let $\rho : G \longrightarrow GL(V)$ be a representation of G. A subspace W of V that is invariant under G is called a* subrepresentation *of G. If the only subrepresentations of V are the zero-dimensional subspace and V itself, the representation ρ is called* irreducible.

EXAMPLE 2.2.5. *The trivial representation and the sign representation of $\mathfrak{S}_n$ are irreducible representations. The defining representation of $\mathfrak{S}_n$ (see Example 2.2.2) is never irreducible because the subspace $[e_1 + e_2 + \ldots + e_n]$ is invariant.*

DEFINITION 2.2.6. *Two representations $\rho_1 : G \longrightarrow GL(V_1)$ and $\rho_2 : G \longrightarrow GL(V_2)$ are said to be* isomorphic *if there is an isomorphism $h : V_1 \longrightarrow V_2$ such that*

$$\rho_2(g) \circ h = h \circ \rho_1(g).$$

PROPOSITION 2.2.7. *[**6**, Proposition 2.30] The number of irreducible representations of G (up to isomorphism) is equal to the number of conjugacy classes of G.*

EXAMPLE 2.2.8. *There are five irreducible representations of $\mathfrak{S}_4$ (cf. Example 2.1.6). The partitions of 4 (different ways of writting 4 as a sum of possitive integers) label these five representations:* [1]

$$\begin{array}{ccc} 4 & \to & \{4\} \\ 3+1 & \to & \{31\} \\ 2+1+1 & \to & \{21^2\} \\ 2+2 & \to & \{2^2\} \\ 1+1+1+1 & \to & \{1^4\} \end{array}$$

The character table in Example 2.2.17 describes the behaviour of these irreducible representations.

EXAMPLE 2.2.9. *The permutation group $\langle (12), (1324) \rangle$ of $\mathfrak{S}_4$ has five irreducible representations, that will be denoted id, sgn, d_1, d_2 and ξ. The character table in Example 2.2.18 describes the behaviour of these irreducible representations.*

From now on, we will denote by $V_1, \ldots, V_k$ the irreducible representations of G, considered up to isomorphism.

THEOREM 2.2.10 (Maschke's theorem). *[**14**, Theorem 1.5.3] Every representation of G is completely reducible. That is, given a representation $\rho : G \longrightarrow GL(V)$, there is a unique decomposition*

$$V = M_1 \oplus \ldots \oplus M_k$$

where each M_i is isomorphic to $d_i V_i$ for some non-negative integer d_i.

EXAMPLE 2.2.11. *The defining representation of $\mathfrak{S}_2$ (see Example 2.2.1) decomposes $V = \mathbb{C}^2$ as $V = M_1 \oplus M_2$ where*

$$M_1 = [(1,1)] \cong id \text{ and } M_2 = [(1,-1)] \cong sgn.$$

EXAMPLE 2.2.12. *(see [**18**])It can be seen that the decomposition of $\mathfrak{L}_{GMM}$ into the irreducible representations of $\mathfrak{S}_4$ is*

$$\mathfrak{L}_{GMM} \cong \{4\} \oplus 2\{31\} \oplus \{2^2\} \oplus \{21^2\}.$$

EXAMPLE 2.2.13. *(see [**5**]) The decomposition of $\mathfrak{L}_{GMM}$ (see Definition 1.4.4) into irreducible representations of $\langle (12), (1324) \rangle$ is*

$$\mathfrak{L}_{GMM} \cong 2id \oplus sgn \oplus d_1 \oplus d_2 \oplus 3\xi.$$

[1] This is a general fact: the irreducible representations of the symmetric group $\mathfrak{S}_n$ are labeled by the partitions of n. In particular, the trivial representation id and the sign representation sgn are referred to as $\{n\}$ and $\{1^n\}$, respectively.

Characters and Young operators

DEFINITION 2.2.14. *Given a finite-dimensional vector space V over $\mathbb{C}$ and a representation $\rho : G \longrightarrow GL(V)$, the* character *of ρ is the function of G defined by*

$$\begin{array}{rccc} \chi_\rho : & G & \longrightarrow & \mathbb{C} \\ & g & \longmapsto & tr(\rho(g)), \end{array}$$

where tr is the trace of $\rho(g) \in GL(V)$. The character χ_ρ is called irreducible *if ρ is an irreducible representation.*

PROPOSITION 2.2.15. *(1) Characters are class functions in the sense that they take a constant value on each conjugacy class of the group.*
(2) Isomorphic representations have the same characters.
(3) If a representation is the direct sum of subrepresentations, then the corresponding character is the sum of the characters of those subrepresentations.

The *character table* of G is a way of expressing the basic information about the irreducible representations of G. The rows are labelled by the irreducible representations of G (or characters), and the columns are labelled by representatives of the conjugacy classes. Usually the first row is labelled by the trivial representation and the first column by the conjugacy class of the identity of G. The entries of the character table are the characters of each conjugacy class on each irreducible representation. The entries of the first column give the degrees of the irreducible representations. For further information, the reader is referred to Chapter 2 of [**16**].

EXAMPLE 2.2.16. *[**16**] Character table of $\mathfrak{S}_2$:*

$\mathfrak{S}_2$	1	(12)
$id = \{2\}$	1	1
$sgn = \{1^2\}$	1	-1

EXAMPLE 2.2.17. *[**16**] Character table of $\mathfrak{S}_4$:*

$\mathfrak{S}_4$	1	(12)	(123)	(12)(34)	(1234)
$\{4\}$	1	1	1	1	1
$\{31\}$	3	1	0	-1	-1
$\{2^2\}$	2	0	-1	2	0
$\{21^2\}$	3	-1	0	-1	1
$\{1^4\}$	1	-1	1	1	-1

EXAMPLE 2.2.18. *[**16**] Character table of $\langle (12), (1324) \rangle$:*

$\langle (12), (1324) \rangle$	1	(12)	(12)(34)	(13)(24)	(1334)
id	1	1	1	1	1
sgn	1	-1	1	1	-1
d_1	1	-1	1	-1	1
d_2	1	1	1	-1	-1
ξ	2	0	-2	0	0

DEFINITION 2.2.19. *Every irreducible module V_i of G has a* Young projection operator *associated to it, defined by*

$$\Theta_i = \frac{1}{|G|} \sum_{\sigma \in G} \chi^i(\sigma) \rho(\sigma)$$

where χ^i is the character of the irreducible representation ρ_i.

Keeping the notation of Theorem 2.2.10, Young operators project the space V into the subspaces M_i:

$$\Theta_i : V \longrightarrow M_i$$

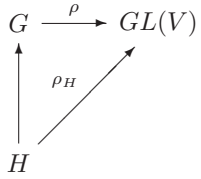
and they are useful to find the explicit decomposition

$$V = M_1 \oplus \ldots \oplus M_k$$

in an effective way.

Branching rule

Finally, another ingredient that we will need is known as the *branching rule* for the restriction of representations [**22**]. Given a representation $\rho : G \longrightarrow GL(V)$ and a subgroup $H \leq G$, the map ρ can be restricted to H according to the diagram



making a H-module of V. By Maschke's theorem, the space V can also be decomposed into the irreducible H-modules. The *branching rule* describes the decomposition into irreducible representations of H of the irreducible representations of G when they are restricted to H. For example, the branching rule of $\mathfrak{S}_4$ to the subgroup $H = \langle (12), (1324) \rangle$

$$\mathfrak{S}_4 \downarrow H : \begin{array}{l} \{4\} \downarrow id \\ \{1^4\} \downarrow sgn \\ \{31\} \downarrow sgn \\ \{2^2\} \downarrow \xi + d_2 \\ \{21^1\} \downarrow \xi + d_1 \end{array}$$

By aplying this rule, we can derive the decomposition of Example 2.2.13 from that in Example 2.2.12.

Chapter 3
Lie Markov Models with prescribed symmetry

In this chapter, we introduce the Lie Markov models and Lie Markov models with prescribed symmetry, and we discuss some examples that generalize other classes of models (equivariant models, group-based models).

We describe a procedure to generate all possible models with some given dimension. In the last section, we describe the space of *real* rate-matrices on a Lie Markov model in terms of convex polyhedral cones.

3.1. Lie Markov models

In this section we finally introduce Lie Markov models. Although we are primarily concerned with the case of nucleotides (4 states), the whole theory remains unaffected if we consider an arbitrary number n of states.

We will refer to a $n \times n$ matrix with complex entries as a *Markov matrix* if the sum of each column is 1. If all the entries of a Markov matrix M are real and lie in [0,1] we say that M is a *stochastic matrix*. We define the *general Markov model* as the set of all Markov matrices, that is

$$\mathfrak{M}_{GMM} := \left\{ M \in M_n(\mathbb{C}) : \Sigma^t M = \Sigma^t \right\}$$

where $\Sigma^t = (1, \ldots, 1)$ (cf. Example 1.1.11). We consider the subset of all Markov matrices whose determinant is non-zero:

$$GL_1(n, \mathbb{C}) := \left\{ M \in M_n(\mathbb{C}) : \Sigma^t M = \Sigma^t, det(M) \neq 0 \right\}.$$

Although there are Markov matrices with determinant zero, this condition is not too restrictive as the set of such matrices has measure zero in $\mathfrak{M}_{GMM}$, and Markov matrices that arise from a continuous-time formulation have non-zero determinant. Notice that $GL_1(n, \mathbb{C})$ is a subgroup of the general linear group, $GL(n, \mathbb{C})$, and contains the set of matrices that can be obtained by the exponentiation of a rate matrix in $\mathfrak{L}_{GMM} = \{M \in M_n(\mathbb{C}) : \Sigma^t M = 0^t\}$ (see Definition 1.4.4).

$$e^{\mathfrak{L}_{GMM}} := \left\{ e^Q : Q \in \mathfrak{L}_{GMM} \right\}.$$

We are interested in studying Markov models being multiplicatively closed (see Section 3 of Chapter 1). To this aim, we need to introduce the *stochastic generating set* for a model. Note that any rate matrix can be expressed as a linear sum of the elementary rate matrices introduced in the Section 1.4. Namely,

$$Q = \sum_{i \neq j} \alpha_{ij} L_{ij}.$$

The unique condition for Q for being *stochastic* is that all α_{ij} are real and non-negative.

Definition 3.1.1. *A vector subspace $\mathfrak{L} \subset \mathfrak{L}_{GMM}$ has a* stochastic generating set *if there exists a generating set of $\mathfrak{L}$, $B_{\mathfrak{L}} = \{L_1, \ldots, L_d\}$, such that each L_k is a convex linear combination of the elementary rate matrices, that is,*

$$L_k = \sum_{i \neq j} \alpha_{ij} L_{ij}$$

where $\alpha_{ij} \geq 0$. A stochastic generating set is a stochastic basis *if the matrices L_k are linearly independent as vectors of $\mathfrak{L}_{GMM}$.*

Definition 3.1.2. *A* Lie Markov model *is a rate-matrix model $e^{\mathfrak{L}}$, where $\mathfrak{L}$ is a Lie subalgebra of $\mathfrak{L}_{GMM}$ that has a stochastic basis. The* dimension *of a Lie Markov model is the dimension of $\mathfrak{L}$ as a complex vector space.*

Because of the Baker-Campbell-Hausdorff formula (see Section 4 of Chapter 1), a Lie Markov model will be multiplicatively closed. The reason to require $\mathfrak{L}$ has a stochastic basis is to ensure that the dimension of the stochastic cone is equal to the dimension of the model (see Consequence 1).

A natural question at this point would be which models are Lie Markov models and which are not. However, the explicit computation of all the Lie Markov models is infeasible for a big number of states [**18**]. To simplify this problem, we request the model to have some symmetry and, thus, limit the number of models we will obtain. In addition to this, the request of symmetry may have biological interest since it allows us to group certain states according to convenient properties (in the case of nucleotides, for example, we may wish to group $\{A, G\}$ and $\{C, T\}$ to reflect the purine/pyrimidine dichotomy).

Given a permutation $\sigma \in \mathfrak{S}_n$, its *permutation matrix* K_σ is the $n \times n$ matrix whose entries are all 0, except for the entry in row i, and column $\sigma(i)$, which is equal to 1. For example, recall that the symmetric group $\mathfrak{S}_n$ acts on $\mathfrak{L}_{GMM}$ 2.1.8 by

$$\sigma Q = K_\sigma Q K_\sigma^{-1} = \sum_{i \neq j} \alpha_{ij} L_{\sigma(i)\sigma(j)}$$

where $Q = \sum_{i \neq j} \alpha_{ij} L_{ij}$.

Definition 3.1.3. *Given a permutation group $G \leq \mathfrak{S}_n$ we say that, a* Lie Markov model $\mathfrak{L}$ has the symmetry of G *if there exists a basis $B_{\mathfrak{L}} = \{L_1, \ldots, L_d\}$ of $\mathfrak{L}$ that is invariant as a set under the action of G, that is*

$$\sigma B_{\mathfrak{L}} := \left\{K_\sigma L_1 K_\sigma^{-1}, \ldots, K_\sigma L_d K_\sigma^{-1}\right\} = B_{\mathfrak{L}}, \forall \sigma \in G.$$

In this case, we say that $B_{\mathfrak{L}}$ is a permutation basis of $\mathfrak{L}$.

Notice that, in particular, the representation of G in $\mathfrak{L}$ is a permutation representation (see Definition 2.2.3).

3.2. Examples

The aim of this section is to show that well-known classes of evolutionary models (namely, equivariant models and group-based models) are particular examples of Lie Markov models.

3.2.1. Equivariant models. Let G be a subgroup of $\mathfrak{S}_n$. The *G-equivariant model* $\mathbb{M}^G$ is defined as the *algebraic* evolutionary model obtained by taking as transition matrices those matrices that are invariant under the action of G, that is (see [**4**]),

$$\mathbb{M}^G = \{A \in M_n(\mathbb{C}) : K_\sigma A K_\sigma^{-1} = A\}.$$

It can be shown that if we define $\mathfrak{M}^G = \mathbb{M}^G \cap GL_1(n, \mathbb{C})$ the the set $\mathfrak{L}^G = \left\{Q \in \mathfrak{L}_{GMM} : K_\sigma Q K_\sigma^{-1} = Q\right\}$ of G-equivariant rate matrices is the tangent vector space of $\mathfrak{M}^G$ at the identity: $T_{\mathbf{1}}(\mathfrak{M}^G) = \mathfrak{L}^G$ (see Lemma 4.3 of [**18**]). Indeed, we have that

PROPOSITION 3.2.1. *[**18**, Proposition 4] Considered as continuous-time rate-matrix models, the equivariant models are Lie Markov models.*

PROOF. We show that $\mathfrak{L}^G$ is a Lie algebra. If X and Y are G-equivariant ($X, Y \in \mathfrak{L}^G$), we need to show that $[X, Y] \in \mathfrak{L}^G$. To this aim, let σ be an element of G. Then,

$$\begin{aligned} K_\sigma [X,Y] K_\sigma^{-1} &= K_\sigma \left(XY - YX\right) K_\sigma^{-1} \\ &= (K_\sigma X K_\sigma^{-1})(K_\sigma Y K_\sigma^{-1}) - (K_\sigma Y K_\sigma^{-1})(K_\sigma X K_\sigma^{-1}) \\ &= XY - YX = [X,Y] \end{aligned}$$

This shows that $\mathfrak{L}^G$ is closed for the Lie bracket. □

It can be seen that the G-equivariant model will correspond to the Lie Markov model $\mathfrak{L}$ with symmetry G such that the decomposition into irreducible representations of G is equal to $d_{id} id$ where $d_{id} > 0$ is the number of copies of id in the decomposition of $\mathfrak{L}_{GMM}$.

3.2.2. Group-based models. Let G be a finite group of n elements. We identify the elements of the group G with the n states of the Markov model. On a *group based* model, the entries of the substitution matrix only depends on the differences between states (according to the operation of G) of the corresponding row and column, that is,$[Q]_{ab} = [Q]_{b-a}$, for all $a, b \in G$.

For example, the Kimura model with 3 parameters is the group-based model obtained by taking the abelian group $G = \langle (12)(34), (13)(24) \rangle \cong \mathbb{Z}_2 \times \mathbb{Z}_2$.

Group-based models can be understood as the rate-matrix models $e^{\mathfrak{L}_G}$, where $\mathfrak{L}_G = \langle \{L_\sigma\}_{\sigma \neq e, \sigma \in G} \rangle$ and $L_\sigma = K_\sigma - 1$ (K_σ is the permutation matrix associated to σ). Then, we have

PROPOSITION 3.2.2. *[**18**, Proposition 4.12] The rate matrix group-based model $e^{\mathfrak{L}_G}$ is a Lie Markov model.*

3.3. Procedure

In this section we will describe the general procedure to obtain Lie Markov models with n states with some prescribed symmetry given by a permutation group $G \leq \mathfrak{S}_n$. This is the procedure that has been implemented in Chapter 4.

(1) First of all, we have to descompose the general Lie Markov model, $\mathfrak{L}_{GMM}$ into irreducible representations of G,
$$\mathfrak{L}_{GMM} = M_1 \oplus \ldots \oplus M_t$$
where $M_k \cong d_k V_k$ and the V_k are the irreducible representations of G, and d_k are the number of copies of the V_k in the descomposition of $\mathfrak{L}_{GMM}$. This can be achieved by applying the Young projection operators (see Definition 2.2.19).
(2) On the other hand, take all the subgroups, $H_i \leq G$ and apply the orbit stabilizer theorem to construct *all possible* G/H_i. Any such set is isomorphic (as a set) to a G-orbit (see Theorem 2.1.19). For each subgroup $H_i \leq G$, extend linearly over $\mathbb{C}$ the G-orbit obtained in (2) and descompose $\langle G/H_i \rangle_{\mathbb{C}}$ into irreducible representations of G,
$$\langle G/H_i \rangle_{\mathbb{C}} \cong \oplus_k b_k^i V_k$$
where the b_k^i are non-negative integers specifying the number of copies of the irreducible representation V_k in the descomposition of $\langle G/H_i \rangle_{\mathbb{C}}$.
(3) Consider all the possible unions of G-orbits
$$S := (G/H_{i_1}) \cup \ldots \cup (G/H_{i_q})$$
such that the decomposition into irreducible representations of G,
$$\langle S \rangle_{\mathbb{C}} \cong \oplus_k a_k V_k \text{ where } a_k = b_k^{H_{i_1}} + \ldots + b_k^{H_{i_q}}$$
satisfies that $a_k \leq d_k$ for all k.
(4) Finally, for these cases, consider a vector subspace $\mathfrak{L} \subset \mathfrak{L}_{GMM}$ with $\mathfrak{L} \cong \oplus_k a_k V_k$ and check it forms a Lie algebra.

3.4. Stochastic cone of a Lie Markov model

Roughly speaking, a Lie Markov model has been defined as a *complex* Lie algebra with some properties. However, for practical purposes and biological applications, we do not want to deal with complex entries, but with real ones, and we need to

restrict the Lie algebra to the set of real rate matrices with non-negative entries outside the diagonal. In other words, we need to consider the *stochastic cone of the model*. To introduce this stochastic cone, we recall the concept of convex polyhedral cone.

DEFINITION 3.4.1. *A* convex polyhedral cone *in* $\mathbb{R}^n$ *is a set*

$$C = \{\lambda_1 v_1 + \ldots + \lambda_r v_r : \lambda_i \geq 0\}$$

generated by a finite set of vectors $\{v_1, v_2, \ldots, v_r\}$ *of* $\mathbb{R}^n$*. These vectors are called the* generators *of the cone* C*.*

DEFINITION 3.4.2. *The* dimension *of a cone* C *is defined as the dimension of the lineal space* $\mathbb{R}C = C + (-C)$ *spanned by* C*. We will denote it by* $\dim(\mathrm{C}) = \dim_{\mathbb{R}}(\mathbb{R}\mathrm{C})$.

EXAMPLE 3.4.3. *(1) With this definition, any real vector subspace of* $\mathbb{R}^n$ *is a complex polyhedral cone.*

(2) The space $\mathfrak{L}^+_{GMM} = \left\{ Q = \sum_{i \neq j} a_{ij} L_{ij} \mid a_{ij} \geq 0 \right\}$ *is a convex polyhedral cone.*

DEFINITION 3.4.4. *A* strongly convex *polyhedral cone is a convex polyhedral cone without nonzero linear subspaces.*

DEFINITION 3.4.5. *The* rays *of a cone are the positive span of each vector in a minimal generator system and correspond to the 1-dimensional faces of the cone. In particular, these minimal generators are unique up to multiplications by positive scalars.*

Note that the set of generators of a cone C is a generator system of the linear space $\mathbb{R}C$, so the number of rays of a cone is greater or equal than the dimension of the cone.

DEFINITION 3.4.6. *If* $\mathfrak{L} \subset \mathfrak{L}_{GMM}$ *is a vector subspace, the* stochastic cone of $\mathfrak{L}$ *is defined as*

$$\mathfrak{L}^+ = \mathfrak{L} \cap \mathfrak{L}^+_{GMM}.$$

RESULT 3.4.7. *[5] If* $\mathfrak{L} \subset \mathfrak{L}_{GMM}$ *is a* $\mathbb{C}$*-vector subspace, then* $\mathfrak{L}^+$ *is a strongly convex polyhedral cone and* $dim(\mathfrak{L}^+) \leq dim(\mathfrak{L})$*. Equality holds if and only if* $\mathfrak{L}$ *has a stochastic generating set.*

CONSEQUENCE 1. *For the Lie Markov models,* $\dim(\mathfrak{L}^+) = \dim_{\mathbb{R}}(\mathfrak{L})$.

This consequence is the reason why we request Lie Markov models to have a stochastic basis.

REMARK 3.4.8. *Notice that if* $\mathfrak{L}$ *is a Lie Markov model with symmetry* G*, then the stochastic cone* $\mathfrak{L}^+$ *is invariant (as a set) under the action of* G*. In particular, the rays of the cone will be arranged in* G *orbits.*

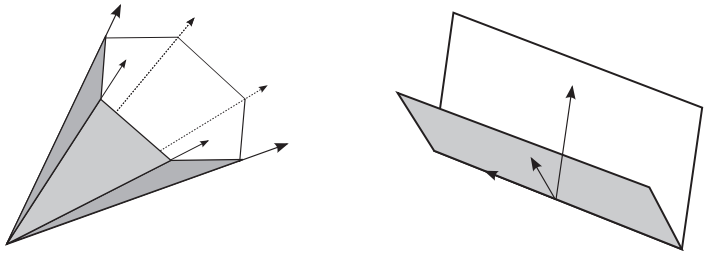


FIG. 1. A strongly convex polyhedral cone of dimension 3 with 6 rays and a convex polyhedral cone which is not strongly convex.

Part 2

Implementation

Chapter 4
Implementation

In this part we show and describe the implementation to compute the Lie Markov models with some prescribed symmetry given by a permutation group $G \leq \mathfrak{S}_n$. To this aim, we follow the procedure describes in Section 3 of Chapter 3. The code is written in *Python* and it has been programmed in the *SAGE* environment combining *Python* with *GAP*. We use *Maxima* software to solve equations.

The reason of using *SAGE* for the implementation, besides being open source software, is that we needed a sotware capable of combining itself with group computations softwares as *GAP*.

SAGE [**17**] (System for Algebra and Geometry Experimentation) is a mathematical software with features covering many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory and calculus. It was created as an open source alternative to *Magma*, *Maple*, *Mathematica* and *Matlab*. *SAGE* uses the *Python* programming language, supporting procedural, functional and object-oriented constructs.

Python [**21**] is a high-level programming language. It permits several styles as object-oriented programming, structured programming, functional programming and aspect-oriented programming.

GAP [**7**] (Groups and Permutations) is a computer algebra system for computational discrete algebra with particular emphasis on computational group theory. *GAP* is used in research and teaching for studying groups and their representations, rings, vector spaces, algebras, combinatorial structures and more.

Maxima is a free computer algebra system for the manipulation of symbolic and numerical expressions. It yields high precision numeric results by using exact fractions, arbitrary precision integers, and variable precision floating point numbers.

4.1. Basic definitions and functions

In this section, we list and describe the basic variables and functions used in the code.

4.1.1. Variables.

(1) *mat0*: A 4×4 matrix of zeros.
(2) *e*: The canonical base of $\mathbb{C}^4$.
(3) *listL*: A list with all the elementary rate matrices (see Example 1.4.2).
(4) *GMM*: The complex vector space $\mathfrak{L}_{GMM}$, which has dimension 12 (see Definition 1.4.4).
(5) *Letters*: These are parameters needed to write the models.

```
import string

mat0=[[0]*4]*4
e=[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]

L=dict()
listL=[]
for i in range(1,5):
    for j in range (1,5):
        M=matrix(mat0)
        M[i-1,j-1]=1
        M[j-1,j-1]=-1
        L[i,j]=M
        if not(i==j):
            listL.append(M)

GMM=VectorSpace(CC,12)

Letters_cap=string.uppercase
Letters_low=string.lowercase
Letters=[var(Letters_cap[j]+Letters_low[j])
         for j in range(len(Letters_cap))]
```

4.1.2. Functions.

(1) *mat_perm*: It returns the permutation matrix of the permutation introduced as an input.
(2) *coord*: It gives the coordinates of a rate-matrix in the basis $\{L_{ij}\}$.
(3) *vectorize*: It constructs a column vector from the non-diagonal entries of the rate-matrix introduced.
(4) *mat_from_coord*: It constructs a matrix from the vector introduced as input.
(5) *repetidos*: It erases the repeated elements of the list introduced.
(6) *reduce*: It returns a basis of the vector space generated by the matrices introduced.

Here is the definition of these functions:

```
def mat_perm(perm):
    return matrix([e[int(gap(j)^perm)-1] for j in range(1,5)])
```

```
def coord(M):
    v=[M[i,j] for i in range(4) for j in range(4) if not(i==j)]
    return vector(v)

def vectorize(M):
    return Matrix([coord(M)]).transpose()

def mat_from_coord(w):
    return sum([t*j for t,j in zip(w,listL)])

def repetidos(R):
    seen=[]
    for e in R:
        if e in seen:
            continue
        seen.append(e)
    return(seen)

def reduce(list):
    aux=[coord(j) for j in list]
    reduced_matrix=matrix(aux).echelon_form()
    basis=reduced_matrix[0:rank(reduced_matrix)]
    return [j for j in basis]
```

4.2. Variables and functions related to the permutation group G

We will use the diagram of Figure 4.2 to explain the functions implemented and variables that depend on the permutation group G. To this aim, we have organised the corresponding code into three different parts.

Part 1. In this part, we define variables with the main aspects of the group, which will be used later.

(1) *gen*: A system of generators of G.
(2) G: The permutation group.
(3) *CT_l*: The character table of G written as a list of lists. Each list corresponds to an irreducible representation of G and its elements are the values of the associated character at the conjugacy classes of G. That is,

$$CT\_l[i][j] \quad = \quad \text{character of the } i\text{-th irreducible character of the } j\text{-th conjugacy class.}$$

(4) *numb_classes*: The number of conjugacy clases of G (recall that this number is equal to the number of irreducible representations of G, see Proposition 2.2.7).

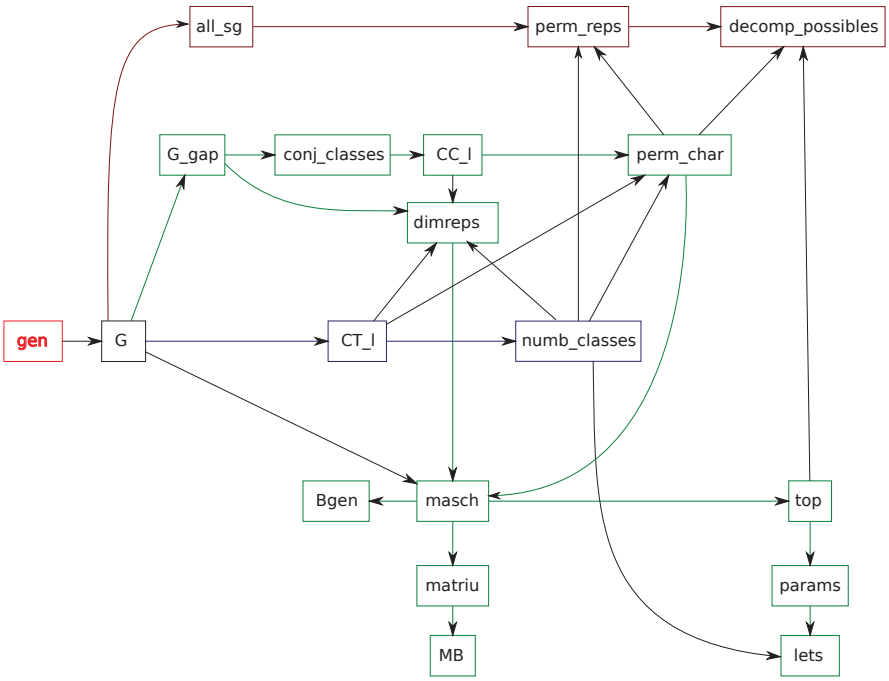


FIG. 1. Diagram for the varibles related to the computational group.

```
gen=[(1,2),(1,3,2,4)]

G=PermutationGroup(gen)

#We obtain the character table and we turn it into a list:
CT_l=list(G.character_table())
numb_classes=len(CT_l)
```

Part 2. The main goal is to obtain the Maschke decomposition (Theorem 2.2.10) of the space $\mathfrak{L}_{GMM}$ (Definition 1.4.4)

$$\mathfrak{L}_{GMM} = M_1 \oplus \ldots \oplus M_t. \tag{4.2.1}$$

This decomposition is given as a list, where each element is a list with a basis of some M_i consistent with the isomorphism $M_i \cong d_i V_i$ (see Theorem 2.2.10 for details). This correspond to the step (1) of the procedure described in Section 3 of Chapter 3. To this aim, we need the GAP software , which allows to obtain the

dimension of the irreducible representations (*dimreps*) and define Young operators (*proj*).

(1) *G_gap*: The permutation group in GAP format.
(2) *conj_classes*: It stores the conjugacy classes of G by means of a representative of each class.
(3) *CC_l*: A list with all the elements of the group G, arranged by conjugacy classes. Namely,

$$CC\_l[i][j] \quad = \quad \text{the } j\text{-th element of the } i\text{-th conjugacy class.}$$

```
G_gap=gap(G)

# We obtain the conjugacy classes of G:

conj_classes=G_gap.ConjugacyClasses()

# We create a list with the conjugacy classes and all
# the elements:

CC_l=[list(i.Elements()) for i in conj_classes]
```

(4) *dimreps*: A list with the degree (=dimension) of the irreducible representations of G.

REMARK 4.2.1. *For computing the dimensions of the irreducible representations we first have to know which is the position of the trivial conjugacy class inside* CC_l *because the value of the associated character in the trivial conjugacy class gives the degree of the irreducible representation(usually, the trivial conjugacy class is in the first position of the list, but that is not always the case when using GAP). This position holds for the charater table* CT_l*, so we now see which are the numbers that appear in the character table for the trivial conjugacy class in each irreducible representation.*

```
e_trivial=list(G_gap.Elements())[0]
[pos]=[j for j in range(len(CC_l)) if CC_l[j] == [e_trivial]]
dimreps=[CT_l[rep][pos] for rep in range(numb_classes)]
```

(5) *perm_char*: A list with the elements of the group together with the corresponding value by the irreducible characters of the group. Namely

perm_char$[i][j]$ = a 2-element list where the second element is the j-th element of the group G and the first element is its value by the i-th irreducible character.

```
#We make a list with all the elements of the group and the
#corresponding number in the character table for each of the
#irreducible representations of G:
```

```
perm_char=[]
for i in range(numb_classes):
#i is the irreducible representation
#j is the conjugacy class
#k is the element of the conjugacy class
    aux=[[CT_l[i][j],CC_l[j][k]] for j in range(numb_classes)
         for k in range(len(CC_l[j]))]
    perm_char.append(aux)
```

(6) *masch*: It provides the Maschke decomposition of $\mathfrak{L}_{GMM}$ as in (4.2.1). It is given by a list of lists, where the i-th sublist is a basis for M_i. This basis is obtained by putting together the basis for the different copies of V_i in $\mathfrak{L}_{GMM}$. Namely,

$$masch[i][j][k] \quad = \quad \text{the } k\text{-th matrix of a basis of the } j\text{-th copy of the } i\text{-th irreducible representation } V_i.$$

(7) *Bgen*: A list with all the matrices of the Maschke decomposition of $\mathfrak{L}_{GMM}$ in a list. In other words, it is a basis of $\mathfrak{L}_{GMM}$ consistent with the decomposition in *masch*.
(8) *MB*: it is the change-of-basis matrix from $\{L_{ij}\}_{i\neq j}$ to the basis *Bgen*.
(9) *top*: A list with the number of copies of each irreducible representation in the Maschke's decomposition of $\mathfrak{L}_{GMM}$. Keeping the notation of Theorem 2.2.10, this is, $[d_1, d_2, \ldots, d_t]$.
(10) *params*: A list that stores the parameters constructed for each irreducible representation and the degrees of the representations. These parameters are used to construct the Lie Markov models.
(11) *lets*: It is obtained from *params* by removing the degrees of the representations.

```
# We compute the Maschke's decomposition of GMM and
# the dimension of each component:
masch=maschke()

S=[]
Bgen=[]
for rep in masch:
    for b in rep:
        for j in b:
            Bgen.append(j)
            S.append(coord(j))

matriu=Matrix(S)
MBgen=matriu.transpose()
MB=MBgen.inverse()

top=[len(j) for j in masch]

params=[]
cont=0
```

```
for rep in range(numb_classes):
    for j in range(1,top[rep]):
        letters_local=[]
        for k in range(1,top[rep]+1):
            aux=var(Letters_low[cont]+str(k))
            letters_local.append(aux)
        params.append([dimreps[rep],letters_local])
        cont+=1
lets=[A[1] for A in params]
```

The functions we need to compute these variables are:

(1) *proj*: It defined and applies the Young projections (see Definition 2.2.19) and returns the matrices obtained by applying these projections to the matrices L_{ij}.

```
def proj():
    lproj=[]
    # k is the position of the irreducible representation
    for k in perm_char:
        # aux_k will be the list of the 12 proyections respect
        # to a proyection operator.
        aux_k=[]
        #i and j are the coordinates for the elementary matrices
        for i in range(1,5):
            for j in range(1,5):
                if not(i==j):
                    sumij=matrix(mat0)
                    # h is a permutation of G (h[1]) with
                    # the character for the character
                    # k (h[0]).
                    for h in k:
                        A=mat_perm(h[1])
                        sumij+=h[0]*A*L[i,j]*(A.transpose())
                    aux_k.append(sumij)
        lproj.append(aux_k)
        # each k is the list of the 12 proyections of
        # the elementary matrices expressed in the base
        # of the elementary matrices
    return [reduce(k) for k in lproj]
```

(2) *find_copies*: From the list of projections of $\{L_{ij}\}$ by the Young projections θ_i (which are a system of generators of M_i), this function obtains a decomposition

$$M_i = \oplus_{j=1}^{d_i} W_j^{(i)},$$

where each $W_j^{(i)}$ is isomorphic to the irreducible representations V_i. The function returns a basis of M_i obtained by putting together the basis for the $W_j^{(i)}$.

```
def find_copies(lista_proj,rep):
    # Returns de position of the conjugation where
    # the trivial permutation is

    # Finds the dimensions of the irreducible
    # representations

    lista_vec=lista_proj[rep]
    perms=G.list()
    dim_rep=dimreps[rep]
    card_copies=len(lista_vec)/dim_rep
    copies=[]
    basis_copies=[]
    for generator in lista_vec:
        gen_mat=mat_from_coord(generator)
        d_generator=dim_orbit(gen_mat)
        j=0
        gen_old=gen_mat
        while not(d_generator==dim_rep): # it will always be >=
            j+=1
            H=G.subgroup([perms[j]])
            gen_mat=sum(orbit(gen_old,H))
            d_generator=dim_orbit(gen_mat)
        orb=orbit(gen_mat,G)
        list_basis=reduce(orb)
        F=GMM.subspace(list_basis)
        if not(F in copies):
            copies.append(F)
            mat_basis=[mat_from_coord(u) for u in list_basis]
            basis_copies.append(mat_basis)
        if len(copies)==card_copies:
            break
    return(basis_copies)
```

(3) *orbit*: It computes the orbit of a matrix under the action of a group. The matrix and the group are the input of the function.
(4) *dim_orbit*: It computes the dimension of the space generated by the G-orbit of the introduced matrix.

```
def orbit(A,H):
    aux=[]
    for j in gap(H).Elements():
        P=mat_perm(j)
        aux.append(P*A*P.transpose())
    return(aux)

def dim_orbit(A):
    orbit_gen=orbit(A,G)
    list_basis=reduce(orbit_gen)
    F=GMM.subspace(list_basis)
```

```
    return(F.dimension())
```

(5) *maschke*: It returns the Maschke's decomposition of $\mathfrak{L}_{GMM}$.

```
# Returns a list whose elements are the isotipic components
# of the GMM algebra's descomposition.
# Each component is given by a sublist with as many elements
# as copies of the irreducible representations.
# Each element of the sublist is a base of the copy expressed
# in coordinates.

def maschke():
    u=proj()
    return [find_copies(u,j) for j in range(numb_classes)]
```

Part 3. In Part 3, we compute all the subgroups $H \leq G$ and for each of them, we obtain the Maschke decomposition of the space $\langle G/H \rangle_{\mathbb{C}}$. This corresponds to the step (2) of the procedure described in Section 3.3. Since we only consider permutation representations, we take all possible sums of these decompositions as long as they are allowed by the decomposition of $\mathfrak{L}_{GMM}$. In the end, we obtain a list of non-negative integer vectors, each of them representing a decomposition of some permutation representation in $\mathfrak{L}_{GMM}$. This is the step (3) of the procedure described in Section 3.3.

(1) *all_sg*: A list with all the subgroups of G and, within each list, the elements of each subgroup. Namely,

$$all\_sg[i][j] \quad = \quad \text{the } j\text{-th element of the } i\text{-th subgroup of } G.$$

(2) *perm_reps*: It stores the number of copies for each irreducible representation in the decomposition of $\langle G/H \rangle_{\mathbb{C}}$. Namely,

$$perm\_reps[i][j] \quad = \quad \begin{array}{l}\text{the number of copies of the } j\text{-th irreducible} \\ \text{representation of } G \text{ of the } i\text{-th subgroup of } G.\end{array}$$

(3) *decomp_possibles*: It provides all the possible decompositions of an invariant subspace of $\mathfrak{L}_{GMM}$.

(4) *cosH*: It stores the cosets of a subgroup H and the elements of each coset. It is a list of lists:

$$cosH[i][j] \quad = \quad j\text{-th element of the } i\text{-th coset of } H \text{ in } G.$$

```
conj_sg = G.conjugacy_classes_subgroups()
all_sg=[]
for representative_sg in conj_sg:
    for g in G:
        new_sg=[g^-1*h*g for h in representative_sg]
        new_sg.sort()
        if not(new_sg in all_sg):
            all_sg.append(new_sg)
```

```
perm_reps=copy(repetidos([decomposition(g) for g in all_sg]))

list_permreps=perm_reps
sumas_posibles=list_permreps
aux=add_lists(sumas_posibles,list_permreps)
while not(aux==[]):
    sumas_posibles=sumas_posibles+aux
    aux=add_lists(aux,list_permreps)
decomp_posibles=repetidos(sumas_posibles)
```

The functions needed to compute these variables are:

(1) *add_lists*: From a couple of lists l_1 and l_2 of vectors of non-negative integers, the function computes all sums of one vector from l_1 with another vector from l_2 the second list, as long as this sum is not bigger than the vector *top*. This allows us to determine from the set of all possible decompositions of G-orbits into irreducible representations, the list of all possible decompositions into irreducible representations of a permutation representation.
(2) *Young_cosets*: It returns a table with the corresponding character for each coset.
(3) *decomposition*: It returns the decomposition into irreducible representations of the G-module $\langle G/H\rangle_{\mathbb{C}}$ for the subgroup $H < G$ introduced as input.

```
def add_lists(l1,l2):
    suma=[]
    for m in l1:
        for n in l2:
            aux_sum=[]
            for k in range(numb_classes):
                aux=m[k]+n[k]
                if (int(aux) > top[k]):
                    break
                aux_sum.append(aux)
            else:
                suma.append(aux_sum)
    return(suma)

def Young_cosets(N,coset):
    # N is the projection
    #  coset is the coset
    coordenada=[0]*len(cosH)
    # coordenada is a list containing the projecction
    for perm in perm_char[N]:
        aux=perm[1]*coset[0]
        for j in cosH:
         # we go through the list of cosets and we identify
         # which one is perm*coset
            if aux in j:
                new_coset=cosH.index(j)
```

```
            coordenada[new_coset]+=perm[0]
        return(coordenada)

    def decomposition(gen_H):
        # "gen" are the generators of a subgroup H of G
        # global cosH
        H=G.subgroup(gen_H) #introduce the subgroup
        cosH=G.cosets(H) # define the cosets of G/H
        V=VectorSpace(CC, len(cosH))
        dim_decomp=[]
        for rep in range(numb_classes):
            # numb_classes is the number of irreducible
            # representations of G
            k=[V(Young_cosets(rep,coset)) for coset in cosH]
            E=V.subspace(k)
            aux=int(E.dimension()/dimreps[rep])
            dim_decomp.append(aux)
        return(dim_decomp)
```

Some examples

Here, we show some examples of the execution of the previous functions in the case $G = \langle (12), (1324) \rangle$.

The following Figure shows the output of *all_sg*, that is, the list of all the subgroups of G given by generators:

```
all_sg
evaluate
    [[()], [(), (1,2)], [(), (3,4)], [(), (1,2)(3,4)], [(), (1,4)(2,3)],
    [(), (1,3)(2,4)], [(), (3,4), (1,2), (1,2)(3,4)], [(), (1,2)(3,4),
    (1,3,2,4), (1,4,2,3)], [(), (1,2)(3,4), (1,3)(2,4), (1,4)(2,3)], [(),
    (3,4), (1,2), (1,2)(3,4), (1,3)(2,4), (1,3,2,4), (1,4,2,3), (1,4)(2,3)]]
```

The Figure shows the output of *decomp_posibles*, that is, the list of all possible permutation representations for a subspace in $\mathfrak{L}_{GMM}$:

```
decomp_posibles
    [[1, 1, 1, 1, 2], [1, 0, 0, 1, 1], [1, 1, 1, 1, 0], [1, 0, 1, 0, 1], [1,
    0, 0, 1, 0], [1, 1, 0, 0, 0], [1, 0, 1, 0, 0], [1, 0, 0, 0, 0], [2, 1,
    1, 2, 3], [2, 1, 1, 2, 2], [2, 1, 1, 1, 2], [2, 0, 0, 2, 2], [2, 1, 1,
    2, 1], [2, 0, 1, 1, 2], [2, 0, 0, 2, 1], [2, 1, 0, 1, 1], [2, 0, 1, 1,
    1], [2, 0, 0, 1, 1], [2, 1, 1, 2, 0], [2, 1, 1, 1, 0], [2, 1, 1, 0, 1],
    [2, 0, 1, 0, 1], [2, 0, 0, 2, 0], [2, 1, 0, 1, 0], [2, 0, 1, 1, 0], [2,
    0, 0, 1, 0], [2, 1, 1, 0, 0], [2, 1, 0, 0, 0], [2, 0, 1, 0, 0], [2, 0,
    0, 0, 0]]
```

4.3. Construction of the Lie Markov models with prescribed symmetry

Once we have computed the list *decomp_possible* with all possible decompositions for a permutation representation, we can start to look for the Lie Markov models. This corresponds to the step (4) of the procedure described in Section 3.3.

First of all, we consider a vector of *decomp_possible*, that is, a vector of non-negative integers

$$v = [a_1, a_2, \ldots, a_t].$$

Such a vector is the input of the function *define_algebra*, that produces all invariant subspaces $\mathfrak{L}$ of $\mathfrak{L}_{GMM}$ whose decomposition into irreducible representation of G is isomorphic to

$$a_1V_1 + a_2V_2 + \ldots + a_tV_t$$

(where the V_i's denote the irreducible representations of G). These subspaces are given in terms of the basis *Bgen*.

REMARK 4.3.1. *If $a_i = d_i$ (that is, the number of copies of V_i in $\mathfrak{L}_{GMM}$), all the matrices of the Maschke decomposition of $\mathfrak{L}_{GMM}$ are taken as part a basis for $\mathfrak{L}$. Otherwise, the function takes linear combinations of these matrices with some parameters that will be determined later on.*

```
def define_algebra(decomp):
    algebra=[]
    cont=0
    for rep in range(numb_classes):
    #rep is the irreducible representation
        aux=matrix(mat0)
        if 1<=decomp[rep]<top[rep]:
            for w in range(decomp[rep]):
            # we execute this as many times as
            # "decomp[rep]" says
                for j in range(dimreps[rep]):
                    list_mat=[copy[j] for copy in masch[rep]]
                    new_matrix=sum([coef*mat for coef,mat
                                in zip(lets[cont],list_mat)])
                    algebra.append(new_matrix)
                cont+=1
                continue
        elif top[rep]>1 :
            cont+=1
        if decomp[rep]==top[rep]:
            for copy in range(top[rep]):
                algebra=algebra+masch[rep][copy]
    return algebra
```

The output of *define_algebra* is used as input of the function *represent_models*.

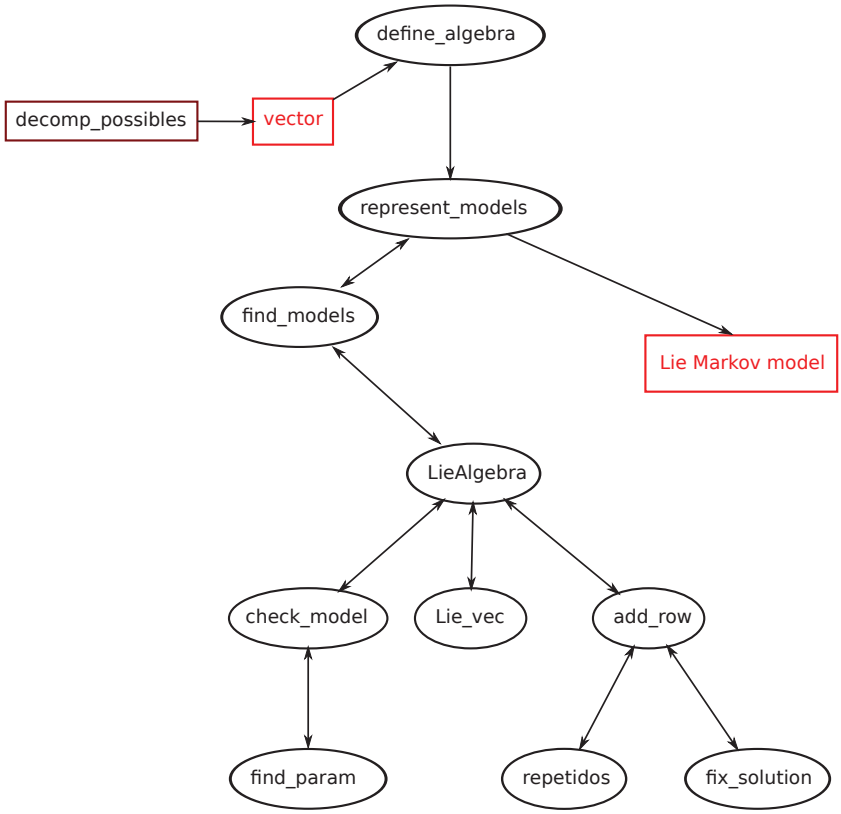


FIG. 2. Diagram for the functions related to the construction of the Lie Markov models with prescribed symmetry given by G. The input is a vector with non-negative integer entries, which is one of the possible decompositions obtained in *decomp_posibles*. The output is the list of all Lie Markov models with that decomposition.

```
def represent_models(algebra):
    list_model=find_models(algebra)
    cont=1
    for model in list_model:
        [mat,ineq]=model
        p=Polyhedron(ieqs=ineq)
        if (p.dim()==len(algebra)):
            print 'Model ', len(algebra), '.', cont
            show(mat)
```

```
            print p
            R=p.rays()
            lettes=[var(Letters[u])
                    for u in range(len(ineq[0])-1) ]
            # this shows the rays
            aux=[change2(mat,lettes,r) for r in R]
            show(aux)
            cont+=1
    return()

def change2(M,alpha,sol):
    N=M
    for i in range(len(alpha)):
        N=N.subs(alpha[i]==sol[i])
    return(N)
```

The output of *represent_models* is the final product of our code: it is a list of all possible Lie Markov models with decomposition given by the vector $[a_1, a_2, \ldots, a_t]$. Each Lie Markov model is described by its general rate-matrix and the list of the ray generators of the stochastic cone.

We describe the implementation of *represent_models*. First of all, it calls *LieAlgebra*.

```
def LieAlgebra(algebra): # we introduce algebra: a possible
                         # decomposition into irreps.
    N=[]
    rk=len(algebra) # this number should be equal to the
                    # dimension of the model
    # to convert each matrix into coordinates in the basis
    # Bgen and to construct a list "N" with all these
    # coordinates
    for x in range(rk):
        coord_algebrax=MB*vectorize(algebra[x])
        aux=Matrix(transpose(coord_algebrax)[0])
        N=N+[aux[0]]
    # this generates the matrix N
    q=[Lie_vec(N[x],N[y]) for x in range(rk)
                          for y in range(x+1,rk)]
    # to construct a list "q" with the coordinates of the
    # Lie brackets of each couple of generators of algebra
    models=[[N,q]]
    for y in range(len(q)):
        models=add_row(models,y)

    models_finals=[]
    for NQ in models:
        [control,mat]=check_model(NQ[0])
        if (control==1):
            models_finals.append(mat)
    return(models_finals)
```

First, this function forms a matrix N where the rows are given by the coordinates of the matrices in *algebra* in the basis *Bgen*. Then, it forms another matrix q whose rows are the Lie brackets of all the possible pairs of rows of N. These Lie brackets are also given by their coordinates in the basis *Bgen*. Secondly, the function calls *add_row* for each row of q.

```
def add_row(models,y):
    new_models=[]
    for NQ in models:
        [N,q]=NQ
        new_row=q[y]
        rk=len(N)
        Nq=matrix(N+[new_row])
        if (rank(Nq)>rk):
            pre_min=Nq.minors(rk+1) #we can say that we
                                    # only check the minors
                                    # that contains the last row
            aux = [w for w in pre_min if w != 0]
            list_min=repetidos(aux) # this is not empty because
                                    # rank(Nq)>rk

            param_model=find_parameters(Nq)
            sol_NQ=sageobj(maxima.solve(list_min,param_model))
            # this solves the equations in "equations"
            # according to the variables in "param" and
            # returns the numerical solutions
            # we obtain a list with the numerical solutions
            # for the equations
            # this is the list of "lists of substitutions"
            # to do in model NN
            sol_NQ=fix_solutions(sol_NQ)
            for b in sol_NQ: # each b is a list of possible
                             # substitutions to do in model NN
                Nb=N
                Qb=q
                for z in b: # each z is a substitution
                    AuxNz=[row.subs(z) for row in Nb]
                    # for each row in NN[0]; impose the
                    # substitution z;  keep the rows once
                    # modified in a new matrix AuxNb; and we
                    # redefine Nb as this matrix
                    Nb=AuxNz
                    AuxQz=[row.subs(z) for row in Qb]
                    # for each row in NN[1]; impose the
                    #substitution z; keep the rows modified
                    # in a new matrix AuxNq
                    Qb=AuxQz
                if (rank(matrix(Nb))==rk) :
                    new_models.append([Nb,Qb])
        else :
            new_models.append([N,q])
    return repetidos(new_models)
```

```
import re
def fix_solutions(sol_NQ):
    copy_solNQ=[]
    length=len(sol_NQ)
    for b in range(length):
        for t in range(len(sol_NQ[b])):
            w=sol_NQ[b]
            copy_w=w
            t_rhs=w[t].rhs()
            if re.match('r\d+',str(t_rhs)):
                t_lhs=w[t].lhs()
                aux_copyw=[tt.subs(t_rhs==t_lhs)
                           for tt in copy_w]
                sol_NQ[b]=aux_copyw
    return(sol_NQ)
```

add_row constructs a new matrix adding to N one of the rows of q, and computes the minors of order $rk(N)+1$. The list of all these minors is a system of equations for the parameters, and to solve this system, we need the software *maxima* (see [**12**]). Notice that in this way, we force the matrices in algebra to generate a Lie subalgebra of $\mathfrak{L}_{GMM}$. A new function *fix_solutions* will fix the presentation of the solutions of the parameters given by maxima in a convenient way for our purposes. Finally, *add_row* returns the values of these parameters, which are replaced in N and q.

At this point, it may happen that some of the Lie Markov models obtained by this procedure still depend on some parameters. In this case, we have an infinite family of Lie Markov models, all of them with the same dimension and the same decomposition. To avoid redundancies, we discard these cases as they will appear again when we compute Lie Markov models of bigger dimension. To this aim, *LieAlgebra* calls *check_model*. This last function looks for free parameters and if any, the model is rejected.

```
def check_model(model):
    still_in_model=find_parameters(model)
    control=1 # First, we acept "model" as a model
    params_to_change=[]
    for parameters in params:
        int=[x for x in parameters[1] if x in still_in_model]
        # they are the correspondent parameters to the copy
        # "parameters" that still remain in the model
        if len(int) in [parameters[0],0]:
            params_to_change=params_to_change+int
        else:
            control=0 #we throw away "model" as a model
            break
    if (control==1):
        aux_model=[]
        for entry in model:
            aux_entry=entry
```

```
            for p in params_to_change:
                aux_entry=[j.subs(p==1) for j in aux_entry]
            aux_model.append(aux_entry)
        model=aux_model
    return([control,model])

def find_parameters(model):
    appear=[]
    for parameter in params:
        for letter in parameter[1]:
            for entry in model:
                if str(letter) in str(entry):
                    appear.append(letter)
    return repetidos(appear)
```

The Lie Markov models obtained are taken by the function *find_models*, which writes the general rate-matrix and obtains the collection of inequalities that will give rise to the stochastic cone.

```
def find_models(algebra):
    list_of_matrix_models=[]
    model_list=LieAlgebra(algebra)
    for N in model_list:
        fixed_N=[]
        z=0
        for vec in N:
            changed_vec=[var(Letters[z])*vec[u]
                         for u in range(12)]
            fixed_N.append(changed_vec)
            z+=1
            # fixed_N is the model "N" where each row has
            # been multiplied by some capital letter
        list_model=[]
        for x in range(12):
            suma_x=sum([vec[x] for vec in fixed_N])
            list_model.append(suma_x)
        matrix_model=scalar_prod(list_model)
        Kt=transpose(matrix(N)*matriu)
        ineq_model=[]
        for x in range(12):
            u=list(Kt[x])
            u.insert(0,0)
            ineq_model.append(u)
        ineq_model=repetidos(ineq_model)
        list_of_matrix_models.append([matrix_model,ineq_model])
    return(list_of_matrix_models)
def scalar_prod(list_model):
    aux=[list_model[x]*Bgen[x] for x in range(12)]
    return sum(aux)
```

Some examples

We show some examples of the execution of the previous functions in the case $G = \langle (12), (1324) \rangle$. In each case, the vector *algebra* contains the possible decomposition into irreducible representations of G and the output of *represent_models* is the general rate matrix of the model together with a description of the stochastic cone and generators of its rays.

```
algebra=define_algebra([2,0,0,0,0])
represent_models(algebra)
```

```
Model  2 . 1
```

$$\begin{pmatrix} -Aa-2Bb & Aa & Bb & Bb \\ Aa & -Aa-2Bb & Bb & Bb \\ Bb & Bb & -Aa-2Bb & Aa \\ Bb & Bb & Aa & -Aa-2Bb \end{pmatrix}$$

```
A 2-dimensional polyhedron in QQ^2 defined as the convex hull of 1
vertex and 2 rays.
```

$$\left[\begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \begin{pmatrix} -2 & 0 & 1 & 1 \\ 0 & -2 & 1 & 1 \\ 1 & 1 & -2 & 0 \\ 1 & 1 & 0 & -2 \end{pmatrix}\right]$$

```
()
```

```
algebra=define_algebra([2,1,0,0,0])
represent_models(algebra)
```

```
Model  3 . 1
```

$$\begin{pmatrix} -Aa-2Bb & Aa & Bb+Cc & Bb-Cc \\ Aa & -Aa-2Bb & Bb-Cc & Bb+Cc \\ Bb-Cc & Bb+Cc & -Aa-2Bb & Aa \\ Bb+Cc & Bb-Cc & Aa & -Aa-2Bb \end{pmatrix}$$

```
A 3-dimensional polyhedron in QQ^3 defined as the convex hull of 1
vertex and 3 rays.
```

$$\left[\begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \begin{pmatrix} -2 & 0 & 0 & 2 \\ 0 & -2 & 2 & 0 \\ 2 & 0 & -2 & 0 \\ 0 & 2 & 0 & -2 \end{pmatrix}, \begin{pmatrix} -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 2 \\ 0 & 2 & -2 & 0 \\ 2 & 0 & 0 & -2 \end{pmatrix}\right]$$

```
()
```

```
algebra=define_algebra([2,0,1,0,0])
represent_models(algebra)
```

```
Model  3 . 1
```

$$\begin{pmatrix} -Aa-2Bb & Aa & Bb+Cc & Bb-Cc \\ Aa & -Aa-2Bb & Bb-Cc & Bb+Cc \\ Bb+Cc & Bb-Cc & -Aa-2Bb & Aa \\ Bb-Cc & Bb+Cc & Aa & -Aa-2Bb \end{pmatrix}$$

```
A 3-dimensional polyhedron in QQ^3 defined as the convex hull of 1
vertex and 3 rays.
```

$$\left[\begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \begin{pmatrix} -2 & 0 & 0 & 2 \\ 0 & -2 & 2 & 0 \\ 0 & 2 & -2 & 0 \\ 2 & 0 & 0 & -2 \end{pmatrix}, \begin{pmatrix} -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 2 \\ 2 & 0 & -2 & 0 \\ 0 & 2 & 0 & -2 \end{pmatrix}\right]$$

```
()
```

```
algebra=define_algebra([2,0,0,2,0])
represent_models(algebra)
```

```
evaluate
```

```
Model  4 . 1
```

$$\begin{pmatrix} -Aa-2Bb-Cc+2Dd & Aa+Cc & Bb+Dd & Bb+Dd \\ Aa+Cc & -Aa-2Bb-Cc+2Dd & Bb+Dd & Bb+Dd \\ Bb-Dd & Bb-Dd & -Aa-2Bb+Cc-2Dd & Aa-Cc \\ Bb-Dd & Bb-Dd & Aa-Cc & -Aa-2Bb+Cc-2Dd \end{pmatrix}$$

```
A 4-dimensional polyhedron in QQ^4 defined as the convex hull of 1
vertex and 4 rays.
```

$$\left[\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 \\ 0 & 0 & 2 & -2 \end{pmatrix}, \begin{pmatrix} -4 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \end{pmatrix}, \begin{pmatrix} -2 & 2 & 0 & 0 \\ 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix}\right]$$

```
()
```

Some examples of Lie Markov models

The previous code has been successfully applied to determine all Lie Markov models with $n = 2$ states and symmetry $G = \mathfrak{S}_2$, and $n = 4$ states with symmetry $G = \mathfrak{S}_4$ and $G = \langle (12), (1324) \rangle \leq \mathfrak{S}_4$.

Lie Markov models with 2 states and symmetry $G = \mathfrak{S}_2$. This case is quite easy and can be studied by direct computation. First of all, the space $\mathfrak{L}_{GMM}$ has dimension 2. The group $G = \mathfrak{S}_2$ has only two irreducible representations (see Example 2.2.16) and the Maschke decomposition is given in Example 2.2.11:

$$\mathfrak{L}_{GMM} = M_1 \oplus M_2,$$

where

$$M_1 = \langle L_{12} + L_{21} \rangle \quad \text{and} \quad M_2 = \langle L_{12} - L_{21} \rangle.$$

There are only two possible decompositions for a G-orbit. Namely,

$H \leq \mathfrak{S}_2$	Cardinality= $\frac{\lvert\mathfrak{S}_2\rvert}{\lvert H\rvert}$	Decomposition of $\langle \mathfrak{S}_2/H \rangle_{\mathbb{C}}$
$\{e\}$	2	$\{2\}\oplus\{1^2\}$
$\mathfrak{S}_2$	1	$\{2\}$

In this case, these are the only possibilities for a permutation representation in $\mathfrak{L}_{GMM}$, so the list of Lie Markov models is reduced to only two models.

- If we consider the decomposition id, we obtain the 1-dimensional model $\mathfrak{L} = \langle L_{12} + L_{21} \rangle$.
- If we take the decomposition $id + sgn$, we obtain the 2-dimensinal model $\mathfrak{L} = \langle L_{12} + L_{21}, L_{12} - L_{21} \rangle$, which of course is equal to $\mathfrak{L}_{GMM}$.

Lie Markov models with 4 states and symmetry $\mathfrak{S}_4$. In the case of $n = 4$ states, the dimension of $\mathfrak{L}_{GMM}$ is 12. The irreducible representations of $\mathfrak{S}_4$ are described in Example 2.2.17 and the Maschke decomposition of $\mathfrak{L}_{GMM}$ is given in Example 2.2.12:

$$\mathfrak{L}_{GMM} \cong \{4\} \oplus 2\{31\} \oplus \{2^2\} \oplus \{21^2\}.$$

In this case, the possible decompositions for the $\mathfrak{S}_4$-orbits are given in the following table (this is the content of the variable *perm_reps* in the previous code):

$H \leq \mathfrak{S}_4$	$\frac{\lvert\mathfrak{S}_4\rvert}{\lvert H\rvert}$	Decomposition of $\langle \mathfrak{S}_4/H\rangle_{\mathbb{C}}$
$\{e\}$	24	$\{4\}\oplus 3\{31\}\oplus 2\{2^2\}\oplus 3\{21^2\}\oplus\{1^4\}$
$\mathbb{Z}_2 \cong \langle(12)\rangle \cong \ldots \cong \langle(34)\rangle$	12	$\{4\}\oplus 2\{31\}\oplus\{2^2\}\oplus\{21^2\}$
$\mathbb{Z}_2 \cong \langle(12)(34)\rangle \cong \ldots \cong \langle(14)(23)\rangle$	12	$\{4\}\oplus\{31\}\oplus 2\{2^2\}\oplus\{21^2\}\oplus\{1\}$
$\mathbb{Z}_3$	8	$\{4\}\oplus\{31\}\oplus\{21^2\}\oplus\{1^4\}$
$\mathbb{Z}_4 \cong \langle(1234)\rangle \cong \ldots \cong \langle(1423)\rangle$	6	$\{4\}\oplus\{2^2\}\oplus\{21^2\}$
$\mathbb{Z}_2\times\mathbb{Z}_2 \cong \langle(12)(34)\rangle \cong \ldots \cong \langle(14)(23)\rangle$	6	$\{4\}\oplus 2\{2^2\}\oplus\{1^4\}$
$\mathbb{Z}_2\times\mathbb{Z}_2 \cong \langle(12)(34),(13)(24)\rangle$	6	$\{4\}\oplus\{31\}\oplus\{2^2\}$
$\mathfrak{S}_3$	4	$\{4\}\oplus\{31\}$
$\mathbb{Z}_2 \wr \mathbb{Z}_2$	3	$\{4\}\oplus\{2^2\}$
A_4	2	$\{4\}\oplus\{1^4\}$
$\mathfrak{S}_4$	1	$\{4\}$

TABLE 1. Decomposition of the orbits of $\mathfrak{S}_4$ into irreducible modules. Notice that the first decomposition is not possible for an invariant subspace $\mathfrak{L}$ in $\mathfrak{L}_{GMM}$ since it contains 3 copies of $\{3,1\}$ while there are only 2 in the Maschke decomposition of $\mathfrak{L}_{GMM}$.

Now, consider all unions of $\mathfrak{S}_4$-orbits $S := (\mathfrak{S}_4/H_1)\cup(\mathfrak{S}_4/H_2)\cup\ldots\cup(\mathfrak{S}_4/H_q)$ such that $|S| = \sum_{1\leq i\leq q}|G/H_i| = d$ and if $\langle S\rangle_{\mathbb{C}} \cong \oplus_j a_j V^j$, then

$$[a_{\{4\}}, a_{\{31\}}, a_{\{2^2\}}, a_{\{21^2\}}, a_{\{1^4\}}] \leq [1,2,1,1,0].$$

Finally, by checking which are the former give rise to Lie subalgebras, we derive the whole list of Lie Markov models with $\mathfrak{S}_4$. This list is shown in Table 2. Notice that we recover the Jukes-Cantor model (JC69), the Kimura model with 3 parameters (K81), the Felsenstein 81 model (F81) and the general Markov model (see Section 1.1 for details). In addition, we get a *new* 6-dimensinoal model, which we denote by $K81 + F81$ since it results from the addition of the Lie algebras of the Kimura model with 3 parameters and the Felsenstein 81 model.

Model	Dimension
GMM	12
K81+F81	6
Felsenstein 81 (F81)	4
Kimura 3ST (K81)	3
Jukes Cantor (JC69)	1

TABLE 2. The complete list of four state Lie Markov models with $\mathfrak{S}_4$ symmetry.

Lie Markov models with 4 states and symmetry $G = \langle(12),(1324)\rangle$. As above, the dimension of $\mathfrak{L}_{GMM}$ is 12. The irreducible representations of G are described in Example 2.2.18 and the Maschke decomposition of $\mathfrak{L}_{GMM}$ is given in Example 2.2.13:

$$\mathfrak{L}_{GMM} \cong 2id \oplus sgn \oplus d_1 \oplus d_2 \oplus 3\xi.$$

In this case, the possible decompositions for the G-orbits are given in the following table:

$H \leq G$	Cardinality=$\lvert G\rvert/\lvert H\rvert$	Decomposition of $\langle G/H\rangle_{\mathbb{C}}$
$\{e\}$	8	$id \oplus sgn \oplus d_1 \oplus d_2 \oplus 2\xi$
$\mathfrak{S}_2 \cong \langle(12)\rangle \cong \langle(34)\rangle$	4	$id \oplus d_2 \oplus \xi$
$\mathfrak{S}_2 \cong \langle(14)(23)\rangle \cong \langle(13)(24)\rangle$	4	$id \oplus sgn \oplus \xi$
$\mathfrak{S}_2 \cong \langle(12)(34)\rangle$	4	$id \oplus sgn \oplus d_1 \oplus d_2$
$\mathbb{Z}_4 \cong \langle(1324)\rangle$	2	$id \oplus d_1$
$\mathfrak{S}_2 \times \mathfrak{S}_2 \cong \langle(12),(34)\rangle$	2	$id \oplus d_2$
G	1	id

TABLE 3. Decomposition of the orbits of G into irreducible modules.

We proceed as above by considering all unions of G-orbits

$$S := (G/H_1) \cup (G/H_2) \cup \ldots \cup (G/H_q)$$

such that $|S| = \sum_{1\leq i\leq q} |G/H_i| = d$, and if $\langle S\rangle_{\mathbb{C}} \cong \oplus_j a_j V^j$, then

$$[a_{id}, a_{sgn}, a_{d_1}, a_{d_2}, a_\xi] \leq [2,1,1,1,3].$$

Finally, by checking which of the former decompositions give rise to Lie subalgebras, we obtain a list with more than 30 Lie Markov models. Among them, we obtain the Lie Markov models of the previous section (those with symmetry $\mathfrak{S}_4$) and some new models. These models are referred to according to the following convention: a model with name $d.r$ means that has dimension d and r is the number of rays in the stochastic cone. In case there is more than one model with that dimension and that number of rays, we differentiate them by using letters: for example, $5.7a$, $5.7b$ and so on.

Here, we present a selection of these models.

- *Model 1.1*: a model with decomposition id. Take $\mathfrak{L} = \langle \sum_{i\neq j} L_{ij}\rangle$ This is the Jukes-Cantor model again.
- *Model 2.2a*: a model with decompostion $2id$. Take $\mathfrak{L} = \langle L_{12}+L_{21}, L_{34}+L_{42}\rangle$. The general rate matrix in the stochatic cone is

$$\begin{pmatrix} * & \alpha & 0 & 0 \\ \alpha & * & 0 & 0 \\ 0 & 0 & * & \beta \\ 0 & 0 & \beta & * \end{pmatrix}, \quad \alpha, \beta \geq 0.$$

  This model gives a reducible Markov chain, that is, it is not possible to get to some states from some other states. If we identify $1,2,3,4$ with A, G, C, T, we see that the purine states A and G communicate with each other, and the same for the pyrimidine states C and T (transitions) while no replacement between purines are pyrimidines (transversions) is allowed.

- *Model 2.2b* $\mathfrak{L} = \langle L_{12} + L_{21}, L_{13} + L_{14} + L_{23} + L_{24} + L_{31} + L_{32} + L_{41} + L_{42} \rangle$. This Lie algebra is abelian and the rate matrices of the stochasatic cone for this model are given by

$$Q = \begin{pmatrix} * & \alpha & \beta & \beta \\ \alpha & * & \beta & \beta \\ \beta & \beta & * & \alpha \\ \beta & \beta & \alpha & * \end{pmatrix}, \quad \alpha, \beta \geq 0.$$

  If we identify our ordered alphabet $\{1, 2, 3, 4\}$ with $\{A, G, C, T\}$, this model corresponds to the Kimura model with 2 parameters (see Section I.1). Notice that this is the equivariant model for the permutation group.
- *Model 3.3a* Take $\mathfrak{L} = \langle L_{12} + L_{21} + L_{34} + L_{43}, L_{13} + L_{24} + L_{31} + L_{42}, L_{14} + L_{23} + L_{32} + L_{41} \rangle$, which is an abelian Lie algebra. The general stochastic rate matrix is

$$\begin{pmatrix} * & \alpha & \beta & \gamma \\ \alpha & * & \gamma & \beta \\ \beta & \gamma & * & \alpha \\ \gamma & \beta & \alpha & * \end{pmatrix}, \quad \alpha, \beta, \gamma \geq 0.$$

  Of course, this is the Kimura model with 3 parameters. At the same time, this is the group-based model corresponding to the copy $\mathbb{Z}_2 \times \mathbb{Z}_2 \cong \langle (13)(24), (14)(23) \rangle$.
- *Model 3.3b* Take $\mathfrak{L} = \langle L_{12} + L_{21} + L_{34} + L_{43}, L_{13} + L_{24} + L_{32} + L_{41}, L_{14} + L_{23} + L_{31} + L_{42} \rangle$, which is a 3-dimensional abelian Lie Markov model. The general rate matrix of the stochastic cone is

$$\begin{pmatrix} * & \alpha & \beta & \gamma \\ \alpha & * & \gamma & \beta \\ \gamma & \beta & * & \alpha \\ \beta & \gamma & \alpha & * \end{pmatrix}, \quad \alpha, \beta, \gamma \geq 0.$$

  This new model may be regarded as a "twisted" version of the Kimura model with three parameters.
  Note that this is the group-based model corresponding to the copy $\langle (1324) \rangle \cong \mathbb{Z}_4$.
- *Model 3.3c* Take $\mathfrak{L} = \langle L_{12} + L_{21}, L_{34} + L_{43}, L_{13} + L_{14} + L_{23} + L_{24} + L_{31} + L_{32} + L_{41} + L_{42} \rangle$, which is a 3-dimensional abelian Lie algebra. The general rate matrix of the stochastic cone is

$$\begin{pmatrix} * & \alpha & \beta & \beta \\ \alpha & * & \beta & \beta \\ \beta & \beta & * & \gamma \\ \beta & \beta & \gamma & * \end{pmatrix}, \quad \alpha, \beta, \gamma \geq 0.$$

- *Model 4.4a* Take $\mathfrak{L} = \langle L_{12} + L_{13} + L_{14}, L_{21} + L_{23} + L_{24}, L_{31} + L_{32} + L_{34}, , L_{41} + L_{42} + L_{43} \rangle$, which is a 4-dimensional Lie algebra. The general rate matrix of the stochastic cone is

$$\begin{pmatrix} * & \alpha & \alpha & \alpha \\ \beta & * & \beta & \beta \\ \gamma & \gamma & * & \gamma \\ \delta & \delta & \delta & * \end{pmatrix}, \quad \alpha, \beta, \gamma, \delta \geq 0$$

- *Model 4.4b* $\mathfrak{L} = \langle L_{12} + L_{21}, L_{34} + L_{43}, L_{13} + L_{14} + L_{23} + L_{24}, L_{31} + L_{32} + L_{41} + L_{42} \rangle$, which is a 4-dimensional Lie algebra. The general rate matrix of the stochastic cone is
$$\begin{pmatrix} * & \alpha & \beta & \beta \\ \alpha & * & \beta & \beta \\ \gamma & \gamma & * & \delta \\ \gamma & \gamma & \delta & * \end{pmatrix}, \quad \alpha, \beta, \gamma, \delta \geq 0.$$
- *Model 12.12* is the general Markov model..

REMARK 4.3.2. *Because the stochastic cone is invariant under the action of G, the set of rays is invariant, too. In fact, rays for the same stochastic cone appear in G-orbits of cardinality 1,2,4 or 8 (as required by Theorem 2.1.19). These families of rays may occur in different models, and this suggests an easy procedure to show nesting between Lie Markov models just by checking that the families of rays of one model are included in the families of rays of the other model. This gives rise to the following list.*

- *1.1 is contained in no one*
- *2.2a is contained in 3.3c, 4.4b, 5.6a, 5.11a, 5.11b, 5.11c, 6.6, 6.8a, 6.8b, 8.16, 8.10a, 8.10b, 9.20b, 10.34;*
- *2.2b is contained in 3.4, 5.6b, 5.16;*
- *3.3a is contained in 4.5a, 5.7a, 5.7b, 5.7c, 6.7a, 6.17a;*
- *3.3b is contained in 4.5b, 6.17b;*
- *3.3c is contained in 5.11a, 5.11b, 5.11c;*
- *3.4 is contained in 5.16;*
- *4.4a is contained in 5.6b, 6.7a, 6.7b, 6.8a, 8.16, 8.10a, 8.17, 8.18, 10.34;*
- *4.4b is contained in 6.8a, 6.8b, 8.16;*
- *4.5a is contained in 6.17a;*
- *4.5b is contained in 6.17b;*
- *5.7c is contained in 6.17a;*
- *6.6 is contained in 8.10a, 8.10b, 10.34;*
- *6.8a is contained in 8.16;*
- *6.8b is contained in 8.16;*
- *8.8 is contained in 10.12;*
- *8.10a is contained in 10.34;*
- *8.10b is contained in 10.34;*
- *the models 5.7a, 5.7b, 5.6a, 5.6b, 5.16, 5.11a, 5.11b, 5.11c, 6.7a, 6.7b, 6.17a, 6.17b, 8.16, 8.17, 8.18, 9.20a, 9.20b, 10.12, 10.34 and 12.12 are not contained in any other model.*

References

[1] M Casanellas and S Sullivant. The strand symmetric model. In L. Pachter and B. Sturmfels, editors, *Algebraic Statistics for computational biology*, chapter 16. Cambridge University Press, 2005.

[2] W. J. Culver. On the existence and uniqueness of the real logarithm of a matrix. *Proc. Amer. Math. Soc.*, 17:1146–1151, 1966.

[3] J. D. Dixon and B. Mortimer. *Permutation groups*, volume 163 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1996.

[4] J. Draisma and J. Kuttler. On the ideals of equivariant tree models. *Mathematische Annalen*, 344:619–644, 2008.

[5] J. Fernández-Sánchez, P. Jarvis, M.Woodhams, and J. Sumner. Lie markov models with symmetry $\mathfrak{S}_2 \wr \mathfrak{S}_2$. *Submitted for publication*, 2012.

[6] W. Fulton. *Young tableaux*, volume 35 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1997. With applications to representation theory and geometry.

[7] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.5.6*, 2012.

[8] B. C. Hall. *Lie groups, Lie algebras, and representations*, volume 222 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2003. An elementary introduction.

[9] M Hasegawa, H Kishino, and T Yano. Dating of the humanape splitting by a molecular clock of mitochondrial dna. *Journal of Molecular Evolution*, 22:160–174, 1985.

[10] CG Lanave, G Preparata, C Saccone, and G Serio. A new method for calculating evolutionary substitution rates. *Journal of Molecular Evolution*, 20:86–93, 1984.

[11] P. J. Lockhart, M. A. Steel, A. C. Barbrook, D. H. Huson, and C. J. Howe. A covariotide model describes the evolution of oxygenic photosynthesis. *Mol. Biol. and Evol.*, 15:1183–1188, 1998.

[12] *Maxima, a Computer Algebra System (Version 5.18.1)*, 2009. `http://maxima.sourceforge.net/`.

[13] L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*. Cambridge University Press, New York, NY, USA, 2005.

[14] B. E. Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions. Second Edition*. Graduate Texts in Mathematics. Springer, 2001.

[15] C. Semple and M. Steel. *Phylogenetics*. Oxford Press, 2003.

[16] J. P. Serre. *Représentations linéaires des groupes finis*. Hermann, Paris, revised edition, 1978.

[17] W. A. Stein et al. *Sage Mathematics Software (Version 4.7.1)*. The Sage Development Team, 2011. `http://www.sagemath.org`.

[18] J. Sumner, J. Fernández-Sánchez, and P. Jarvis. Lie markov models. *J. Theor. Biol.*, 298:16–31, 2012.

[19] J. Sumner, P.Jarvis, J. Fernández-Sánchez, B. Kaine, M. Woodhams, and B. Holland. Is the general time-reversible model bad for molecular phylogenetics? *Systematic Biology, doi: 10.1093/sysbio/sys042; arXiv: 1111.0723*, 2012.

[20] S. Tavaré. Some probabilistic and statistical problems in the analysis of dna sequences. *Lectures on Mathematics in the Life Sciences (American Mathematical Society)*, 17:57–86, 1986.

[21] G. van Rossum and F.L. Drake (eds). *Python Reference Manual*. PythonLabs, Virginia, USA, 2001. Available at `http://www.python.org`.

[22] H. Weyl. *The Theory of Groups and Quantum Mechanics*. Dover Publications, 1950.

[23] VB Yap and L Pachter. Identification of evolutionary hotspotsin the rodent genomes. *Genome Research*, 14(4):574–9, 2004.