Assessing the robustness of parsimonious predictions for gene neighborhoods from reconciled phylogenies

Ashok Rajaraman, Cedric Chauve, Yann Ponty

Assessing the robustness of parsimonious predictions for gene neighborhoods from reconciled phylogenies

Ashok Rajaraman[1], Cedric Chauve[1], and Yann Ponty[1,2,3]

[1] Department of Mathematics, Simon Fraser University, Burnaby, Canada
[2] Pacific Institute for Mathematical Sciences, CNRS UMI3069, Vancouver, Canada
[3] CNRS/LIX, Ecole Polytechnique, Palaiseau, France

Abstract. The availability of a large number of assembled genomes opens the way to study the evolution of syntenic character within a phylogenetic context. The DeCo algorithm, recently introduced by Bérard *et al.* allows the computation of parsimonious evolutionary scenarios for gene adjacencies, from pairs of reconciled gene trees. Following the approach pioneered by Sturmfels and Pachter, we describe how to modify the DeCo dynamic programming algorithm to identify classes of cost schemes that generates similar parsimonious evolutionary scenarios for gene adjacencies, as well as the robustness to changes to the cost scheme of evolutionary events of the presence or absence of specific ancestral gene adjacencies. We apply our method to six thousands mammalian gene families, and show that computing the robustness to changes to cost schemes provides new and interesting insights on the evolution of gene adjacencies and the DeCo model.

1 Introduction

Reconstructing evolutionary histories of genomic characters along a given species phylogeny is a long-standing problem in computational biology. This problem has been studied for several types of genomic characters (DNA sequences and gene content for example), for which efficient algorithms exist to compute parsimonious evolutionary scenarios. Recently, [2] extended the corpus of such results to syntenic characters. They defined a model for the evolution of gene adjacencies within a species phylogeny, together with an efficient dynamic programming (DP) algorithm, called `DeCo`, to compute parsimonious adjacency evolutionary histories. Reconstructing evolutionary scenarios for syntenic characters is an important step toward more comprehensive models of genome evolution, going beyond classical sequence/content frameworks as it implicitly integrates genome rearrangements [4]. The `DeCo` algorithm is a DP algorithm that takes as input a pair of reconciled gene trees, which represent the evolution of two gene families and a list of extant gene adjacencies, and compute, in polynomial time, an evolutionary scenario for adjacencies that minimizes the number of adjacency gains and breaks.

The evolutionary events considered by `DeCo`, gene adjacency gain and break, caused by genome rearrangement, are rare evolutionary events, compared to gene-family specific events. It is then important to assess the robustness of inferences made by `DeCo`, whether it is of a parsimony cost, of an event count, or of an individual feature such as the presence of a specific ancestral adjacency. We explored recently an approach that considers the set of all possible evolutionary scenarios under a Boltzmann probability distribution [5]. A second approach consists of considering changes in the cost associated to evolutionary events (the cost scheme) and in assessing how features of evolutionary scenarios are robust to such changes. Such approaches have recently been considered for the gene tree reconciliation problem and have been shown to improve significantly the results obtained from purely parsimonious approaches [1, 9]. A second motivation is more specific to the `DeCo` model: as it models gene adjacencies, each ancestral gene can only be adjacent

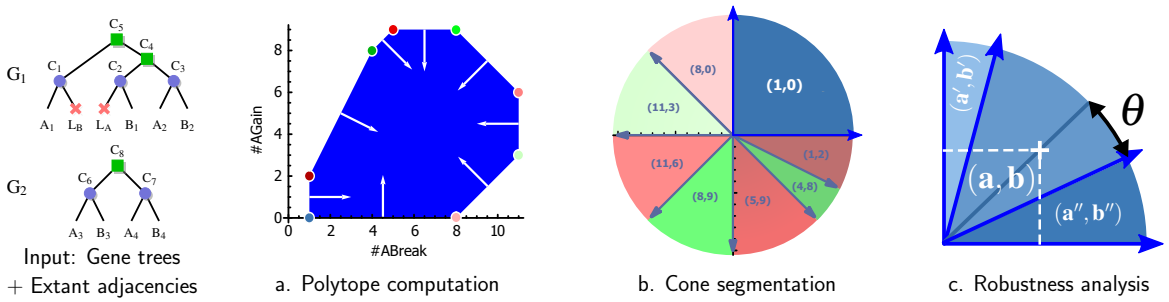

Fig. 1. Outline of our method: Starting from two reconciled gene trees and a set of extant adjacencies, the polytope of parsimonious signatures is computed (a.). Its normal vectors define a segmentation of the space of cost schemes into cones (b.), each associated with a signature. Here, the positive quadrant is fully covered by a single cone, meaning that the parsimonious prediction does not depend on the precise cost scheme. In general, the robustness of a prediction to perturbations of the scoring scheme can be measured as the smallest angle θ such that a cost scheme at angular distance θ no longer predicts its signature (c.).

to at most two other genes, which is not considered in DeCo. However, the initial experiments using DeCo on mammalian gene trees resulted in hundreds of ancestral genes were involved in more than two ancestral gene adjacencies. This raises the question of filtering inferred ancestral adjacencies to reduce the level of syntenic conflict, which can be done on the basis of their robustness. We reason that some of the erroneously-predicted adjacencies may result from an imperfectly calibrated evolutionary model and that features of a gene adjacency parsimonious evolutionary scenario that are not robust to perturbations of the cost scheme should be considered as dubious.

These observations motivate the precise question tackled in this work: how robust is an inferred parsimonious ancestral gene adjacency to a change of the costs associated to adjacency gains and breaks? We address this problem using a the the *polytope approach*, a methodology that has been formalized into a rigorous algebraic framework by Pachter and Sturmfels [13, 12, 11], building on the seminal work of Gusfield [6]. Its main features, summarized in Fig. 1, are (1) associating each evolutionary scenario to a *signature*, a vector of two integers (g, b) where g is the number of adjacency gains and b the number of adjacency breaks; and (2) partitioning the space of cost schemes into convex regions such that, for all the cost schemes within a region, all optimal solutions obtained with such cost schemes have the same signature. This partition computed by an algorithm that is a direct translation of the DP algorithm into a polytope framework. After introducing the DeCo model and algorithm, we describe the corresponding polytope approach, apply it on a mammalian dataset, and conclude by discussing the obtained results.

2 Preliminary: models and problems

A *phylogeny* is a rooted tree which describes the evolutionary relationships of a set of elements (species, genes, ...) represented by its nodes: internal nodes correspond to ancestral elements, leaves to extant elements, and edges represent direct descents between parents and children. For a node v of a phylogeny, we denote by $s(v)$ the species it belongs to. For a tree T and a node x of T, we denote by $T(x)$ the subtree rooted at x. If x is an internal node, we assume it has either one child, denoted by $a(x)$, or two children, denoted by $a(x)$ and $b(x)$.

2.1 Species tree and Reconciled gene trees.

A species tree S is a binary tree that describes the evolution of a set of related species from a common ancestor (the root of the tree), through the mechanism of *speciation*. A reconciled gene tree is a binary tree that describes the evolution of a set of genes, called a *gene family*, through the evolutionary mechanisms of speciation, *gene duplication* and *gene loss*, within a given species tree S. Therefore, each node of a gene tree G represents either a gene loss, an extant gene or an ancestral gene. Ancestral genes are represented by the internal nodes of G, while gene losses and extant genes are represented by the leaves of G.

In a reconciled gene tree, we associate every ancestral gene (an internal node g) to an evolutionary event $e(g)$ that leads to the creation of the two children $a(g)$ and $b(g)$: $e(g)$ is a speciation (denoted by Spec) if the species pair $\{s(a(g)), s(b(g))\}$ is equal to the species pair $\{a(s(g)), b(s(g))\}$, $s(a(g)) \neq s(b(g))$, or a gene duplication (GDup) if $s(a(g)) = s(b(g)) = s(g)$. If g is a leaf, then $e(g)$ indicates either a gene loss (GLoss) or an extant gene (Extant), in which case $e(g)$ is not an evolutionary event *stricto sensu*. A *pre-speciation* ancestral gene is an internal node g such that $e(g) = \mathsf{Spec}$. See Fig. 3 (Appendix) for an illustration.

2.2 Adjacency trees and forests.

We consider now that we are given two reconciled gene trees G_1 and G_2, representing two gene families evolving within a species tree S. A *gene adjacency* is a pair of genes (one from G_1 and one from G_2) that appear consecutively along a chromosome, for a given species, ancestral or extant. Gene adjacencies evolve within a species tree S through the evolutionary events of speciation, gene duplication, gene loss (these three events are modeled in the reconciled gene trees), and *adjacency duplication* (ADup), *adjacency loss* (ALoss) and *adjacency break* (ABreak), that are adjacency-specific events.

Following the model introduced in [2], we represent such an evolutionary history using an *adjacency forest*, composed of *adjacency trees*. An adjacency tree represents the evolution of an ancestral gene adjacency (located at the root of the tree) through the following events: (1) The duplication of an adjacency $\{g_1, g_2\}$, where g_1 and g_2 are respectively genes from G_1 and G_2 such that $s(g_1) = s(g_2)$, follows from the simultaneous duplication of both its genes g_1 and g_2 (so $e(g_1) = e(g_2) = \mathsf{GDup}$), resulting in the creation of two distinct adjacencies each belonging to $\{a(g_1), b(g_1)\} \times \{a(g_2), b(g_2)\}$; (2) The loss of an adjacency, which can occur due to several events, such as the loss of exactly one of its genes (gene loss, GLoss), the loss of both its genes (adjacency loss, ALoss) or a genome rearrangement that breaks the contiguity between the two genes (adjacency break, ABreak); (3) The creation/gain of an adjacency (denoted by AGain), for example due to a genome rearrangement, that results in the creation of a new adjacency tree whose root is the newly created adjacency.

With this model, one can model the evolution of two gene families along a species phylogeny by a triple (G_1, G_2, A): G_1 and G_2 are reconciled gene trees representing the evolution of these families in terms of gene-specific events and A is an adjacency forest consistent with G_1 and G_2. Similar to species and reconciled gene trees, internal nodes of an adjacency tree are associated to ancestral adjacencies, while leaves are associated to extant adjacencies or lost adjacencies (due to a gene loss, adjacency loss or adjacency break), and are labeled by evolutionary events. The label $e(v)$ of an internal node v of an adjacency forest A belongs to $\{\mathsf{Spec}, \mathsf{GDup}, \mathsf{ADup}\}$, while the label $e(v)$ of a leaf belongs to $\{\mathsf{Extant}, \mathsf{GLoss}, \mathsf{ALoss}, \mathsf{ABreak}\}$.

2.3 Signatures, descriptors and parsimonious scenarios.

The *signature* (or event count) of an adjacency forest A is an ordered pair of integers $\sigma(A) = (g_A, b_A)$ where g_A (resp. b_A) is the number of adjacency gains (resp. adjacency breaks) in A. A *cost scheme* is a pair $\mathbf{x} = (x_0, x_1)$ of non-negative real numbers, where x_0 is the cost of an adjacency gain and x_1 the cost of an adjacency break. The *cost* of an adjacency forest A for a given cost scheme $\mathbf{x}$ is the number $S(A) = x_0 \times g_A + x_1 \times b_A$. The adjacency forest A in an evolutionary scenario (G_1, G_2, A) is *parsimonious for* $\mathbf{x}$ if there is no other evolutionary scenario (G_1, G_2, B) such that $S(B) < S(A)$.

A *descriptor* of a scenario is a boolean or integer valued feature of the solution which does not contribute to the cost of the scenario, but rather represent a feature of a scenario. For instance, the presence/absence of an ancestral adjacency $a = (g_1, g_2)$, in a given adjacency forest A can be described as a boolean. Given k descriptors $a_1, \ldots, a_k$, we can define an *extended signature* of a scenario A as a tuple $\sigma_{a_1,\ldots,a_k}(A) = (g, b, s_{a_1}, \ldots, s_{a_k})$, where g, b are the event counts for adjacency gains and breaks in A respectively, and s_{a_i} is the value of the descriptor a_i for A.

2.4 The `DeCo` and `DeClone` algorithms.

Bérard *et al* [2] showed that, given a pair of reconciled gene trees G_1 and G_2, a list L of extant gene adjacencies, and a cost scheme $\mathbf{x}$, one can compute, using a DP algorithm, an evolutionary scenario (G_1, G_2, A) where A is a parsimonious adjacency forest such that L is exactly the set of leaves of A labeled `Extant`. The `DeCo` algorithm computes, for every pair of nodes g_1 (from G_1) and g_2 (from G_2) such that $s(g_1) = s(g_2)$, two quantities denoted by $c_1(g_1, g_2)$ and $c_0(g_1, g_2)$, that correspond respectively to the cost of a parsimonious adjacency forest for the pairs of subtrees $G(g_1)$ and $G(g_2)$, under the hypothesis that g_1 and g_2 do form (for c_1) or do not form (for c_0) an ancestral adjacency. As usual in dynamic programming along a species tree, the cost of a parsimonious adjacency forest for G_1 and G_2 is given by $\min(c_1(r_1, r_2), c_0(r_1, r_2))$ where r_1 is the root of G_1 and r_2 the root of G_2. Fig. 4, in the Appendix presents the `DeCo` dynamic programming equations.

2.5 Robustness problems.

Assuming that a cost scheme $\mathbf{x} = (x_0, x_1) \in \mathbb{R}^2$ provides sufficient information to evaluate the objective function (here, the cost of an adjacency tree), then the predictions under such a model remain unchanged upon multiplying $\mathbf{x}$ by any positive number. We may therefore assume that $\|\mathbf{x}\| = 1$ without loss of generality. In two dimensions, $\mathbf{x} = (x_0, x_1)$ can be summarized as a simple angle θ (expressed in radians), and the difference between two cost schemes is indicated by their associated angular distance.

The first problem we are interested in is the *signature robustness problem*. A signature $\sigma = (g, b)$ is parsimonious for the cost scheme $\mathbf{x}$ if there exists at least one adjacency forest A that is parsimonious for $\mathbf{x}$ and has signature $\sigma(A) = \sigma$. The robustness of the signature σ can be quantified as the difference (i.e angular distance) from $\mathbf{x}$ to the closest cost scheme for which σ is no longer parsimonious.

However, signatures only provide a quantitative summary of the evolutionary events described by a parsimonious adjacency forest. In particular, signatures discard any information about predicted sets of ancestral adjacencies. This leads us to consider the *parsimonious adjacency robustness problem*, where we address the robustness of inferred parsimonious ancestral adjacencies. Let $a = (g_1, g_2)$ be an ancestral adjacency featured in a parsimonious adjacency forest for a cost

scheme **x**. We say that a is parsimonious for a cost scheme **y** if a belongs to every adjacency forest that is parsimonious for **y**. The robustness of a can then be defined as the angular distance from **x** to the closest cost scheme **y** for which a is no longer parsimonious. This notion of robustness can be extended to sets of ancestral adjacencies **a** by considering that a set of ancestral adjacencies is parsimonious for a cost scheme **y** if every adjacency in **a** is parsimonious for **y**.

3 Methods

We now show how to compute the robustness measures introduced in the previous section. If the signature for a given adjacency forest A is given by the vector $\sigma(A) = \big(g, b\big)$, and the cost scheme is given by the vector $\mathbf{x} = (x_0, x_1)$, then the parsimony cost of `DeCo` can be written as the inner product $\langle \mathbf{x}, \sigma(A)\rangle = g \times x_0 + b \times x_1$. `DeCo` computes the following quantity for a pair of gene trees G_1 and G_2.

$$c(G_1, G_2) = \min_{A \in \mathscr{F}(G_1, G_2)} \langle \mathbf{x}, \sigma(A)\rangle, \tag{1}$$

where $\mathscr{F}(G_1, G_2)$ denotes the set all possible adjacency forests that can be constructed from G_1 and G_2, irrespective of the cost scheme.

We will consider a single descriptor a, indicating the presence or absence of an ancestral adjacency $a = \big(g_1, g_2\big) \in G_1 \times G_2$ of interest in the given instance, where $s_a = 1$ if it is present in A, and 0 otherwise. Since, by definition, a descriptor does not contribute to the cost, we will only consider cost schemes of the form $\mathbf{x} = (x_0, x_1, 0)$, and `DeClone` will compute Eq. (1) as usual.

An adjacency forest in the set $\mathscr{F}(G_1, G_2)$ may be associated with the same signature, as signatures provide only an event count. Furthermore, for a given cost scheme **x**, two adjacency forests A_1 and A_2 $\sigma(A_1) = \sigma(A_2)$ also have the same associated cost. We can thus define an equivalence class in $\mathscr{F}(G_1, G_2)$ based on the signatures. However, the adjacency forests in this equivalence class may have different extended signatures, differing only in the last coordinate. Thus, there may be two adjacency forests A_1 and A_2 with extended signatures $\big(g, b, 1\big)$ and $\big(g, b, 0\big)$ respectively, and they will have the same cost for all cost schemes. Evolutionary scenarios with the same extended signature also form an equivalence class in $\mathscr{F}(G_1, G_2)$.

3.1 Convex polytopes from signatures.

Let us denote the set of signatures of all scenarios in $\mathscr{F}(G_1, G_2)$ by $\sigma(\mathscr{F}(G_1, G_2))$, and the set of extended signatures for a given adjacency a by $\sigma_a(\mathscr{F}(G_1, G_2))$. Each of these is a point in $\mathbb{R}^d$, where $d = 2$ for signatures and $d = 3$ for extended signatures. In order to explore the parameter space of parsimonious solutions to `DeCo`, we use these sets of points to construct a *convex polytope* in $\mathbb{R}^d$. A convex polytope is simply the set of all convex combinations of points in a given set, in this case the set of signatures or extended signatures [13]. Thus, for each pair of gene trees G_1, G_2 and a list of extant adjacencies, we can theoretically construct a convex polytope in $\mathbb{R}^2$ by taking the convex combinations of all signatures in $\sigma(\mathscr{F}(G_1, G_2))$. This definition generalizes to a convex polytope in $\mathbb{R}^3$ when extended signatures $\sigma_a(\mathscr{F}(G_1, G_2))$ are considered for some signature a. Viewing the set of evolutionary scenarios as a polytope allows us to deduce some useful properties:

1. Any (resp. extended) signature that is parsimonious for some cost scheme **x** lies on the surface of the polytope;
2. If a (resp. extended) signature is parsimonious for two cost schemes **x** and **x**′, then it is also parsimonious for any cost scheme *in between* (for any convex combination of **x** and **x**′).

Traditionally, a polytope is either represented as a set of inequations, which is inappropriate for our intended application. Therefore, we adopt a slighty modified representation, and denote the polytope of $\mathscr{F}(G_1, G_2)$ as a list of signatures that are represented within $\mathscr{F}(G_1, G_2)$ and lie on its convex hull.

A *vertex* in a polytope is a signature (resp. extended signature) which is the optimal for some cost scheme. The domain of parsimony of a vertex $\mathbf{v}$ is the set of cost schemes for which $\mathbf{v}$ is optimal. A consequence of Property 2 is that the domain of parsimony for a vertex $\mathbf{v}$ is a *cone* in $\mathbb{R}^d$, formally defined as:

$$Cone(\mathbf{v}) = \left\{ \mathbf{x} \in \mathbb{R}^d : \langle \mathbf{x}, \mathbf{v} \rangle \leq \langle \mathbf{x}, \mathbf{w} \rangle \ \forall \ \mathbf{w} \in P \right\}. \tag{2}$$

The set of cones associated with the vertices of a polytope form a partition of the cost schemes space [13]. This allows us to study the parameter space, and assess the effect of perturbing the parameters on the optimal solution of `DeCo`.

3.2 Computing the polytope.

Pachter and Sturmfels [12, 11, 13] introduced the concept of *polytope propagation*, based on the observation that the polytope of a dynamic programming (minimization) scheme can be computed through an algebraic substitution. Accordingly, any point that lies strictly within the polytope is suboptimal for any cost scheme, and can be safely discarded by a procedure that repeatedly computes the *convex hull* $H(P)$ of the (intermediates) polytopes produced by the modified DP scheme. In the context of the `DeCo` DP scheme, the precise modifications are:

1. Any occurrence of the + operator is replaced by $\oplus$, the (convex) *Minkowski sum* operator, defined for P_1, P_2 two polytopes as
$$P_1 \oplus P_2 = H(\{p_1 + p_2 \mid (p_1, p_2) \in P_1 \times P_2\});$$
2. Any occurrence of the min operator is replaced by $\Cup$, the *convex union* operator, defined for P_1, P_2 two polytopes as
$$P_1 \Cup P_2 = H(P_1 \cup P_2);$$
3. Any occurrence of an *adjacency gain* cost is replaced by the vector $(1,0)$ (resp. $(1,0,0)$ for extended signatures);
4. Any occurrence of an *adjacency break* cost is replaced by the vector $(0,1)$ (resp. $(0,1,0)$ for extended signatures);
5. (Extended signatures only) Any production that corresponds to the prediction of a fixed ancestral adjacency a in a scenario is replaced by the vector $(0,0,1)$;

Through this substitution, we can efficiently compute the polytope associated with two input gene trees G_1 and G_2, having sizes n_1 and n_2 respectively, through $O(n_1 \times n_2)$ executions of the convex hull procedure. In place of the integers used by the original minimization approach, intermediate convex polytopes are now processed by individual operations, and stored in the dynamic programming tables, so the overall time and space complexities of the algorithm critically depend on the size of the polytopes, i.e. its number of vertices. Pachter and Sturmfels proved that, in general, the number of vertices on the surface of the polytope is $O\left(n^{d-1}\right)$, where d is the number of dimensions, and n is the size of the dynamic programming table. Thus, in our case, the number of vertices in the 2D polytope associated with simple signatures is in $O(n_1 \times n_2)$. This upper

bound also holds for extended signatures, as the third coordinate is boolean, and the resulting 3D polytope is in fact the union of two 2D polytopes. The total cost of computing the convex hull is therefore bounded by $O\left(n_1^2 \times n_2^2 \times \log(n_1 \times n_2)\right)$, e.g. using Chan's convex hull algorithm [3].

As for the computation of the cones, let us remark that the cone of a vertex v in a given polytope P is fully delimited by a set of vectors, which can be computed from P as the normal vectors, pointing towards the center of mass of P, of each of the facets in which v appears. This computation can be performed as a postprocessing using simple linear algebra, and its consumption will remain largely dominated by that of the DP-fueled polytope consumption.

3.3 Assessing signature and adjacency robustness.

The cones associated with the polytopes cover all the real-valued cost schemes, including those associating negative costs to events. These cost schemes are not valid, and so, we only consider cones which contain at least one positive cost scheme.

Given a fixed cost scheme $\mathbf{y}$, the vertex associated to the cone containing this cost scheme corresponds to the signature of all parsimonious scenarios for this cost scheme. In order to assess the robustness of this signature, we can calculate the smallest angular perturbation needed to move from $\mathbf{y}$ to a cost scheme whose parsimonious scenarios do not have this signature. This is simply the angular distance from $\mathbf{y}$ to the nearest boundary of the cone which contains it. Using this methods, we assign a numerical value to the robustness of the signatures of parsimonious scenarios on a number of instances for a particular cost scheme.

In the case of extended signatures $\sigma_a(\mathscr{F}(G_1, G_2))$ for an adjacency a, the polytope output by the DP equation is 3-dimensional. The cones associated with the vertices, as defined algebraically, now partition $\mathbb{R}^3$, the set of cost schemes (x_0, x_1, x_2), where x_2 indicates the cost of a distinguished adjacency. Since the third coordinate is a descriptor, it should not contribute to the cost scheme, and we must therefore restrict our analysis to the $\mathbb{R}^+ \times \mathbb{R}^+ \times \{0\}$ subset of the cost scheme space. Therefore we take the intersection with the plane $x_2 = 0$ of each cone associated with a vertex $\left(g, b, s_a\right)$, and obtain the region in which the extended signature $\left(g, b, s_a\right)$ is parsimonious. Note that this region is a 2D cone.

However, the cost of an extended signature is now independent of the entry in its last coordinate, and there may exist two different extended signatures $\left(g, b, 0\right)$ and $\left(g, b, 1\right)$, both parsimonious for all the cost schemes found in the 2D cone. It is also possible for adjacent cones to have different signatures, yet feature a given adjacency. The robustness of a given adjacency a is therefore computed from the cones using a greedy algorithm which, starting from the cone of $\mathbf{x}$, explores the adjacent cones in both directions (clockwise/counter-clockwise) until it finds one that no longer predicts a, i.e. is associated with at least one signature $\left(g', b', 0\right)$.

4 Results

The data set we considered is composed of 5,039 reconciled gene trees and 50,389 extant gene adjacencies, forming 6,074 `DeCo` instances, with genes taken from 36 extant mammalian genomes from the Ensembl database in 2012. In [2], these data were analyzed using the `DeCo` algorithm that computed a single parsimonious adjacency forest per instance. All together, these adjacency forests defined 96,482 ancestral adjacencies, covering 112,188 ancestral genes, where an "ancestral adjacency"is an adjacency that involves two genes g_1 and g_2 whose descendants in their re-

spective gene trees do not belong to the same species $s(g_1)$ (equal to $s(g_2)$), *i.e.* g_1 and g_2 are pre-speciation genes, that were not duplicated within their species.

Besides our notions of robustness, an indirect validation criterion used to assess the quality of an adjacency forest is the limited presence of syntenic conflicts. An ancestral gene is said to participate to a syntenic conflict if it belongs to three or more ancestral adjacencies, as a gene can only be adjacent to at most two neighboring genes along a chromosome. An ancestral adjacency participates to a syntenic conflict if it contains a gene that does. Among the ancestral adjacencies inferred by DeCo, 16,039 participate to a syntenic conflict, covering 5,817 ancestral genes. This represents a significant level of syntenic conflict.

We first considered all 6,074 instances, and computed for each signature the robustness of the parsimonious signature obtained with the cost scheme $(1,1)$ used in the DeCo experiment. Interestingly, we observe (Fig. 2(A)) that for more than half of the instances, the parsimonious signature is robust to a change of cost scheme, as the cone associated to this signature is the complete first quadrant of the real plane. On the contrary, for 945 instances the parsimonious signature for the cost scheme $(1,1)$ is not robust to any change in the cost scheme; these cases correspond to interesting instances where the cost scheme $(1,1)$ lies at the border of two cones, meaning that two parsimonious signatures exist for the cost scheme $(1,1)$, and any small change of cost scheme tips the balance towards one of these two signatures. More generally, as revealed by Fig. 2(A), the robustness landscape of the considered instances indicates an extreme robustness of parsimonious signatures. There is a $\sim 80\%$ overlap between the sets of signatures that are parsimonious for any (positive) cost scheme, and for the $(1,1)$ cost scheme.

Next, to evaluate the stability of the total number of evolutionary events inferred by parsimonious adjacency forests, we recorded three counts of evolutionary events for each instance: the total number of gene duplications in the reconciled trees, the number of syntenic events (adjacencies gains and breaks) of the parsimonious signature (called the parsimonious syntenic events count), and the maximum number of syntenic events taken over all signatures that are parsimonious for some cost scheme (called the maximum syntenic events count). We observe that the average number of gene duplications per instance is 3.38, and the average parsimonious (resp. maximum) syntenic events count is 1.25 (resp. 1.66). This shows a strong robustness of the number of syntenic events to changes in the cost scheme, with a significant difference in the number of syntenic events compared to gene-specific events.

We then considered the robustness of individual adjacencies. Using the variant DeClone of DeCo that explores the set of all evolutionary scenarios [5], we extracted adjacencies that belong to all parsimonious solutions for the cost scheme $(1,1)$ from all instances, and computed their robustness as defined in the previous sections. This set of ancestral adjacencies contains 87,019 adjacencies covering 106,903 ancestral genes, and participating in 7,566 syntenic conflicts. It can be observed that selecting such universally parsimonious ancestral adjacencies significantly reduced the number of syntenic conflicts, as almost all discarded ancestral adjacencies participated in syntenic conflicts. The robustness of these adjacencies is summarized in Fig. 2(B). It is interesting to observe that few adjacencies have a low robustness, while, conversely, a large majority of the universally parsimonious adjacencies are completely robust to a change of cost scheme (97,593 out of 106,639). This suggests that the DeCo model of parsimonious adjacency forests is robust, and infers highly supported ancestral adjacencies, which is reasonable given the relative sparsity of genome rearrangements in evolution compared to smaller scale evolutionary events. Considering syntenic conflicts, we can notice a positive result, i.e. that filtering by robustness re-

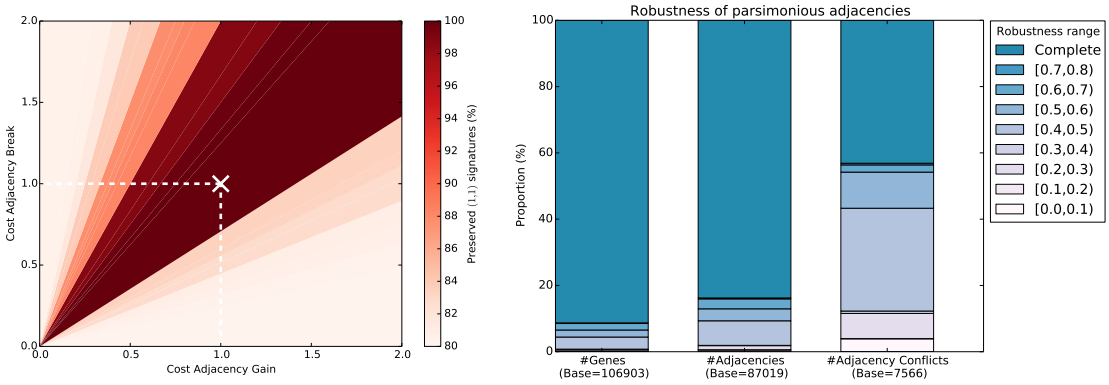

Fig. 2. (A) Average robustness of signatures predicted using the $(1,1)$ cost scheme. At each point (x,y), the colour indicates the proportion of signatures that are parsimonious, and therefore predicted, for the $(1,1)$ cost scheme, and remain parsimonious for the (x,y) cost scheme. (B) Universally parsimonious adjacencies and syntenic conflicts. (Left) Percentage of ancestral genes present in universally parsimonious adjacencies per level of minimum robustness of the adjacencies, expressed in radians. (Center) Percentage of universally parsimonious adjacencies per level of minimum robustness. (Right) Percentage of conserved conflicting ancestral adjacencies per level of minimum robustness.

sults in a significant decrease of the ratio of conflicting adjacencies. However its can be observed that even with robust universally parsimonious ancestral adjacencies, one can observe a significant number of adjacencies participating in syntenic conflicts. We discuss these observations in the next section.

5 Discussion

From an application point of view, the ability to exhaustively explore the parameter space allowed to observe that, on the considered instances, the `DeCo` model is extremely stable. Even taking parsimonious signatures that maximize the number of evolutionary syntenic events (i.e. considering cost schemes that leads to the maximum number of events) results in an average increase of roughly 33% events (1.25 to 1.66), and stays very low, much lower than gene specific events such as gene duplications. This is consistent with the fact that for rare evolutionary events such as genome rearrangements, a parsimony approach is relevant, especially when it can be complemented by efficient algorithms to explore slightly sub-optimal solutions, such as `DeClone`, and to explore the parameter space. In terms of direct applications of the method developed here and in [5], one could think about a gene-tree based reconstruction of ancestral gene orders based on considering the set of all ancestral adjacencies that are parsimonious for at least one cost scheme (that can be computed in polynomial time using the method described in the present paper), scored using a mixture of their Boltzmann probability (that can be computed efficiently using `DeClone`) and robustness to changes of the cost scheme. This set of ancestral adjacencies is likely to contain many syntenic conflicts, that could be cleared out independently and efficiently for each ancestral species using the algorithm of [10] for example.

An interesting observation is that even the set of ancestral adjacencies that are universally-parsimonious and robust to changes in the cost scheme contains a significant number of adjacencies participating in syntenic conflict. We believe the fact that this set of ancestral adjacencies is quite large supports the parsimony model of `DeCo` for studying the evolution of syntenic

characters, as discussed above. We conjecture that the main reason for syntenic conflicts is in the presence of a significant number of erroneous reconciled gene trees. This is supported by the observation that the ancestral species with the highest number of syntenic conflict are also species for which the reconciliation with the mammalian species tree resulted in a significantly larger number of genes than expected (data not shown). This points clearly to errors in either gene tree reconstruction and in the reconciliation with the mammalian species phylogeny, that tends to assign wrong gene duplications in some specific species, resulting an inflation of the number of genes, especially toward the more ancient species [7]. It would be interesting to apply the method we introduced here to corrected mammalian gene trees (see [8] and references there regarding gene tree correction).

From a methodological point of view, there exists another way to explore the parameter space of a dynamic programming phylogenetic algorithm. It consists in computing, rather than optimal signatures for classes of cost schemes, the *Pareto-front* of the input instance [9, 14]. A signature v is said to be *Pareto-optimal* if there is no other signature whose entries are equal or smaller than the corresponding entries in v, and is strictly smaller at at least one coordinate. The Pareto-front is the set of all Pareto-optimal signatures, and can be efficiently computed by dynamic programming [15, 14, 9]. In particular, [9] showed how, in the context of phylogenetic tree reconciliation, this approach can be extended to provide a support for specific evolutionary events. The Pareto-front differs from the approach we describe in the present work in several aspects. An advantage of the Pareto-front is that it is a notion irrespective of the type of cost function being used. This contrasts with the polytope propagation technique, which requires that the cost function be a linear combination of its terms. However, so far, the Pareto-approach has only been used to define a partition of the parameter space when the cost function is restricted to be linear/affine, and it remains to investigate the difference with the polytope approach in this case. It is also key to note that Pareto-optimal signatures might correspond to points that are inside the polytope defined by parsimonious signatures, i.e. signatures of solutions that are not parsimonious for any cost scheme, and the polytope approach effectively ignores them rather than computing them. This difference deserves further investigation, especially to characterize Pareto-optimal but non-parsimonious signatures.

References

1. M. S. Bansal, E. J. Alm, and M. Kellis. Reconciliation revisited: Handling multiple optima when reconciling with duplication, transfer, and loss. *Journal of Computational Biology*, 20(10):738–754, 2013.
2. S. Bérard, C. Gallien, B. Boussau, G. J. Szöllosi, V. Daubin, and E. Tannier. Evolution of gene neighborhoods within reconciled phylogenies. *Bioinformatics*, 28(18):382–388, 2012.
3. T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16:361–368, 1996.
4. C. Chauve, N. El-Mabrouk, L. Gueguen, M. Semeria, and E. Tannier. Duplication, rearrangement and reconciliation: A follow-up 13 years later. In *Models and Algorithms for Genome Evolution*, pages 47–62. Springer, 2013.
5. C. Chauve, Y. Ponty, and J. P. P. Zanetti. Evolution of genes neighborhood within reconciled phylogenies: An ensemble approach. In *BSB*, volume 8826 of *Lecture Notes in Computer Science*, pages 49–56. Springer, 2014.
6. D. Gusfield. Parametric combinatorial computing and a problem of program module distribution. *J. ACM*, 30(3):551–563, July 1983.
7. M. W. Hahn. Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution. *Genome Biology*, 8:R141, 2007.
8. M. Lafond, C. Chauve, R. Dondi, and N. El-Mabrouk. Polytomy refinement for the correction of dubious duplications in gene trees. *Bioinformatics*, 30(17):519–526, 2014.

9. R. Libeskind-Hadas, Y. Wu, M. S. Bansal, and M. Kellis. Pareto-optimal phylogenetic tree reconciliation. *Bioinformatics*, 30(12):87–95, 2014.
10. J. Manuch, M. Patterson, R. Wittler, C. Chauve, and E. Tannier. Linearization of ancestral multichromosomal genomes. *BMC Bioinformatics*, 13(S-19):S11, 2012.
11. L. Pachter and B. Sturmfels. Parametric inference for biological sequence analysis. *Proc Natl Acad Sci U S A*, 101(46):16138–16143, 2004.
12. L. Pachter and B. Sturmfels. Tropical geometry of statistical models. *Proc Natl Acad Sci U S A*, 101(46):16132–16137, 2004.
13. L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*. Cambridge University Press, 2005.
14. C. Saule and R. Giegerich. Observations on the feasibility of exact pareto optimization. In *Proceedings of the 1st Workshop on Computational Methods for Structural RNAs (CMSR 2014), Strasbourg, France, September 7, 2014.*, pages 43–56, 2014.
15. T. Schnattinger, U. Schöning, and H. A. Kestler. Structural RNA alignment by multi-objective optimization. *Bioinformatics*, 29(13):1607–1613, 2013.

Appendix

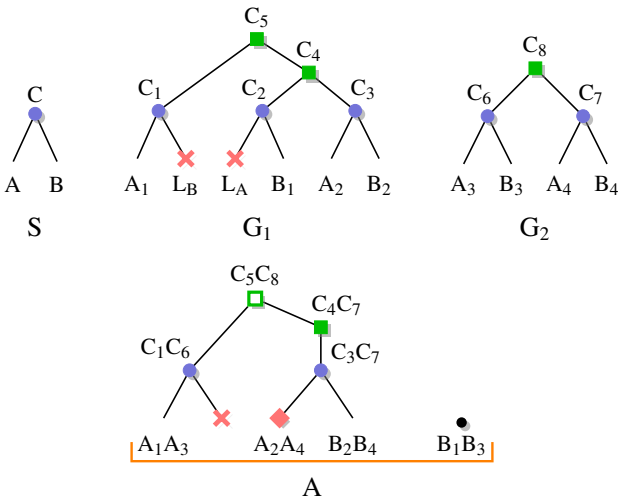

Fig. 3. A species tree S, with two extant species A and B and an ancestral species C. Two reconciled gene trees G_1 and G_2, with four extant genes in genome A, four extant genes in genome B and three ancestral genes in genome C. The set of extant gene adjacencies is (A_1A_3, B_1B_3, B_2B_4). An adjacency forest A composed of two adjacency trees. Blue dots represent speciation nodes. Leaves are extant species/genes/adjacencies, except the one labeled by a red cross (gene loss) or a red diamond (adjacency breaks). Green squares are (gene or adjacency) duplication nodes. Gene labels refer to the species they belong to. Every node of the adjacency tree is labeled by a couple of nodes from gene trees representing a gene adjacency. The signature the adjacency forest is $(1,1)$, and this adjacency forest is parsimonious for the cost scheme $(1,1)$. Figure adapted from [2].

1. If $e(g_1) = \mathsf{Extant}$ and $e(g_2) = \mathsf{Extant}$:

$$c_1(g_1, g_2) = \begin{cases} 0 & \text{if } g_1 g_2 \text{ is an extant adjacency} \\ \infty & \text{otherwise} \end{cases} \qquad c_0(g_1, g_2) = \begin{cases} 0 & \text{if } g_1 g_2 \text{ is not an extant adjacency} \\ \infty & \text{otherwise} \end{cases}$$

2. If $e(g_1) = \mathsf{GLoss}$ and $e(g_2) \in \{\mathsf{Extant}, \mathsf{Spec}, \mathsf{GDup}\}$: $c_1(g_1, g_2) = c_0(g_1, g_2) = 0$

3. If $e(g_1) \in \{\mathsf{Extant}, \mathsf{Spec}, \mathsf{GDup}\}$ and $e(g_2) = \mathsf{GLoss}$: $c_1(g_1, g_2) = c_0(g_1, g_2) = 0$

4. If $e(g_1) = \mathsf{GLoss}$ and $e(g_2) = \mathsf{GLoss}$: $c_1(g_1, g_2) = c_0(g_1, g_2) = 0$

5. If $e(g_1) \in \{\mathsf{Extant}, \mathsf{Spec}\}$ and $e(g_2) = \mathsf{GDup}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(g_1, b(g_2)) + c_0(g_1, a(g_2)), c_0(g_1, b(g_2)) + c_1(g_1, a(g_2)), \\ c_1(g_1, b(g_2)) + c_1(g_1, a(g_2)) + \mathsf{AGain}, c_0(g_1, b(g_2)) + c_0(g_1, a(g_2)) + \mathsf{ABreak} \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(g_1, b(g_2)) + c_0(g_1, a(g_2)), c_0(g_1, b(g_2)) + c_1(g_1, a(g_2)) + \mathsf{AGain}, \\ c_1(g_1, b(g_2)) + c_0(g_1, a(g_2)) + \mathsf{AGain}, c_1(g_1, b(g_2)) + c_1(g_1, a(g_2)) + 2\mathsf{AGain} \end{cases}$$

6. If $e(g_1) = \mathsf{GDup}$ and $e(g_2) \in \{\mathsf{Extant}, \mathsf{Spec}\}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(a(g_1), g_2) + c_0(b(g_1), g_2), c_0(a(g_1), g_2) + c_1(b(g_1), g_2), \\ c_1(a(g_1), g_2) + c_1(b(g_1), g_2) + \mathsf{AGain}, c_0(a(g_1), g_2) + c_0(b(g_1), g_2) + \mathsf{ABreak} \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(a(g_1), g_2) + c_0(b(g_1), g_2), c_0(a(g_1), g_2) + c_1(b(g_1), g_2) + \mathsf{AGain}, \\ c_1(a(g_1), g_2) + c_0(b(g_1), g_2) + \mathsf{AGain}, c_1(a(g_1), g_2) + c_1(b(g_1), g_2) + 2\mathsf{AGain} \end{cases}$$

7. If $e(g_1) = \mathsf{Spec}$ and $e(g_2) = \mathsf{Spec}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)), c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{ABreak}, \\ c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{ABreak}, \\ c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + 2\mathsf{ABreak}, \\ c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)), c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + \mathsf{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + \mathsf{ABreak}, c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + 2\mathsf{ABreak} \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)), c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{AGain}, \\ c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{AGain}, c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + 2\mathsf{AGain}, \\ c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)), c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + \mathsf{AGain}, \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + \mathsf{AGain}, c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + 2\mathsf{AGain} \end{cases}$$

8. If $e(g_1) = \mathsf{GDup}$ and $e(g_2) = \mathsf{GDup}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(a(g_1), g_2) + c_0(b(g_1), g_2), c_0(a(g_1), g_2) + c_1(b(g_1), g_2), \\ c_1(a(g_1), g_2) + c_1(b(g_1), g_2) + \mathsf{AGain}, c_0(a(g_1), g_2) + c_0(b(g_1), g_2) + \mathsf{ABreak}, \\ c_1(g_1, a(g_2)) + c_0(g_1, b(g_2)), c_0(g_1, a(g_2)) + c_1(g_1, b(g_2)), \\ c_1(g_1, a(g_2)) + c_1(g_1, b(g_2)) + \mathsf{AGain}, c_0(g_1, a(g_2)) + c_0(g_1, b(g_2)) + \mathsf{ABreak}, \\ c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)), \\ c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{AGain}, \\ c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{AGain}, \\ c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + 2\mathsf{AGain}, \\ c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{ABreak}, \\ c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{AGain} + \mathsf{ABreak}, \\ c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{AGain} + \mathsf{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{AGain} + \mathsf{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{AGain} + \mathsf{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)), \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{AGain}, \\ c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{AGain}, \\ c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \mathsf{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \mathsf{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + 2\mathsf{ABreak} \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(a(g_1), g_2) + c_0(b(g_1), g_2), c_0(a(g_1), g_2) + c_1(b(g_1), g_2) + \mathsf{AGain}, \\ c_1(a(g_1), g_2) + c_0(b(g_1), g_2) + \mathsf{AGain}, c_1(a(g_1), g_2) + c_1(b(g_1), g_2) + 2\mathsf{AGain}, \\ c_0(g_1, a(g_2)) + c_0(g_1, b(g_2)), c_0(g_1, a(g_2)) + c_1(g_1, b(g_2)) + \mathsf{AGain}, \\ c_1(g_1, a(g_2)) + c_0(g_1, b(g_2)) + \mathsf{AGain}, c_1(g_1, a(g_2)) + c_1(g_1, b(g_2)) + 2\mathsf{AGain} \end{cases}$$

Fig. 4. The DeCo dynamic programming equations, adapted from [2].

5.1 Boltzmann ensemble approach.

In [5], we showed how to extend the DeCo algorithm to an ensemble approach that allows one to explore the whole solution space of adjacency forests. Let $\mathscr{F}(G_1, G_2)$ be the set of all adjacency forests for G_1 and G_2, including both optimal and sub-optimal ones. The *partition function* associated to G_1 and G_2 is defined by

$$\mathscr{Z}(G_1, G_2) = \sum_{A \in \mathscr{F}(G_1, G_2)} e^{-\frac{S(A)}{kT}}$$

where kT is an arbitrary constant. The partition function implicitly defines a *Boltzmann probability distribution* over $\mathscr{F}(G_1, G_2)$, where the probability of an adjacency forest A is defined by:

$$P(A) = \frac{e^{-\frac{S(A)}{kT}}}{\mathscr{Z}(G_1, G_2)}.$$

The Boltzmann probability of an adjacency, or more generally of a feature that can be observed in an adjacency forest, is then defined as the ratio of the sum of the Boltzmann probabilities of the adjacency forests that contain this feature. Such probabilities can be computed efficiently using a variation of the dynamic programming algorithm of DeCo [5]. The impact of kT on the Boltzmann probability can be described as follows: when kT is small, the Boltzmann distribution probability is skewed toward parsimonious adjacency forests, while a high value of kT tends toward a more uniform probability distribution.