Solving the Persistent Phylogeny Problem in polynomial time

Paola Bonizzoni* Gianluca Della Vedova* Gabriella Trucco †
Mauricio Soto Gomez ‡

Friday 7th May, 2021

Abstract

The notion of a Persistent Phylogeny generalizes the well-known Perfect phylogeny model that has been thoroughly investigated and is used to explain a wide range of evolutionary phenomena. More precisely, while the Perfect Phylogeny model allows each character to be acquired once in the entire evolutionary history while character losses are not allowed, the Persistent Phylogeny model allows each character to be both acquired and lost exactly once in the evolutionary history. The Persistent Phylogeny Problem (PPP) is the problem of reconstructing a Persistent phylogeny tree, if it exists, from a binary matrix where the rows represent the species (or the individuals) studied and the columns represent the characters that each species can have.

While the Perfect Phylogeny has a linear-time algorithm, the computational complexity of PPP has been posed, albeit in an equivalent formulation, 20 years ago. We settle the question by providing a polynomial time algorithm for the Persistent Phylogeny problem.

1 Introduction

The problem of reconstructing an evolutionary history from characters is a classical topic in computational biology. Two main models have been produced under the parsimony criteria, in the context of the evolution of taxa described by a set of characters: the Camin-Sokal and the Dollo model [6], [13]. In these two models characters assume commonly two distinct states 0 and 1 , where 0 models the absence and 1 the presence of a character in a given taxa. The two models aim to build a tree representing the evolution of a set of existing taxa and minimizing mutation events, where the events associated to each edge represent the change in state from 0 to 1 or from 1 to 0 of a given character during the evolution of the ancestral taxa. The Camin-Sokal model assumes that each character may change state from 0 to 1 multiple times in the tree, but no change from 1 to 0 state is allowed. Differently, in the Dollo model characters may change state from 0 to 1 only once in the tree, but they may change state from 1 to 0 multiple times. The Dollo model appears appropriate for reconstructing the evolution of genes in eukaryotic organisms, mainly the gain and loss of genes modeled as change to 1 and to 0 respectively in the taxa. Indeed, while multiple gains of the same gene in different lineages is improbable, multiple losses of a gene are more common [13]. The model recently gained a lot if interest in the context of reconstructing the clonal evolution in tumors due to mutation events taking into account copy number aberrations [10,11]. In this context characters are mutations that are gained or lost during the clonal evolution. It is assumed that a given mutation is acquired at most once in the tree, but it may be lost due to the loss of genes related to deletion events. The recent literature in tumor evolution from single cell and bulk data extend the classical infinite site assumption allowing

*Dipartimento di Informatica Sistemistica e Comunicazione, Univ. degli Studi di Milano–Bicocca. bonizzoni,dellavedova@disco.unimib.it

†Dipartimento di Informatica, Univ. degli Studi di Milano, gabriella.trucco@unimi.it

‡Dipartimento di Ingegneria Gestionale, Politecnico di Milano mauricioabel.soto@polimi.it

loses [5]. From the algorithmic point of view, Dollo model leads to Np-complete problems, while its most well studied restriction, the Perfect Phylogeny is easy since it has a simple linear time solution [8]. In the Perfect Phylogeny, characters are *perfect*: they can change state from 0 to 1 at most once during the evolution of taxa and never change state from one to zero. However, as discussed above, biological problems motivate the algorithmic investigation of the Dollo model or variants that may have efficient solutions. In this direction, the notion of a *persistent* character have been proposed in [12] and then introduced [1], [2] to reconstruct trees: a persistent character can change state from 0 to 1 at most once during the evolution then change state from one to zero, again at most once. The computational problem of the Persistent Perfect Phylogeny (PPP) or Persistent Phylogeny has been formulated and investigated in [1], while other variants have been considered in [4]. In [1], an exact algorithm for the PPP problem that is polynomial in the number of taxa, but exponential in the number of characters has been proposed. An integer linear programming solution has been explored in [9]. The ILP formulation uses a reformulation of the PPH problem as the problem of completing a matrix M_e obtained by doubling columns of the input matrix M of the PPP problem (i.e. adding a persistent copy of the character) and posing as unknown the state of characters that may be persistent, i.e. those characters that have entry 0 in matrix M. In [3] the PPP problem is restated as a colored-graph problem. In this paper we use this graph framework to design a polynomial time algorithm that solves the PPP problem.

2 Preliminaries

The input of the PPP problem is an $n \times m$ binary matrix M, where its set $C = \{c_1, \ldots, c_m\}$ of columns is associated to characters and its set $S = \{s_1, \ldots, s_n\}$ of rows is associated to taxa also called *species*. In the matrix M we have that $M[s, c] = 1$ if and only if the species s has character c, otherwise $M[s, c] = 0$. Then the PPP problem decides the existence of a rooted tree T whose edges are labeled with a disjoint sets of characters (positive or negated) and nodes are labeled with the union of the non negated edge-labels in their path from the root. Each character of M labels at most one edge as positive and may label an edge as negated if such edge follows the edge labeled by the positive character. The labeling of nodes must ensure that each species s can be associated to a node in the tree labeled by its character set. The labeling of nodes in the tree with characters is formalized by the notion of state. For each node x of T, we define the *state* of x as the m-dimensional binary vector such that $x[i]$ is equal to one if and only if c_i is in the label set of x or, in other words, if c is present in the label of an edge in the path from the root to x and c is never negated along such path.

We say that the character $c \in C$ is *gained* in the unique edge where its state goes from 0 to 1 or, more formally, in the edge (x, y) such that y is a child of x and c has state 0 in x and state 1 in y. In this case the label set of the edge (x, y) contains the label c^+. Conversely, c is *lost* in the edge (x, y) if y is a child of x and character c has state 1 in x and state 0 in y. In this case the label set of the edge (x, y) contains the label c^-.

Characters c^+ and d^- are called *signed* characters. We stress the fact that since edge labels are disjoint, then for each character c at most one edge contains the label c^+(c^-, respectively) [1,15].

We now introduce a more general definition of Persistent Phylogeny in which some characters can be already present in the root node.

Definition 1 (Persistent Phylogeny)**.** Let M be an $n \times m$ binary matrix over a set C of m characters and a set S of n species of M. Let A be a subset of the set C of characters in M: characters in A are called *active. Then a* persistent phylogeny*, in short* PPP*, for the pair* (M, A) *is a rooted tree* T *such that:*

1. *each node* x *of* T *is labeled by a vector* $l_x \in \{0, 1\}$ *of length* m*, where the value* $l_x[j] \in \{0, 1\}$ *represents the state of character* c_j *in the node* x*;*

2. *the root* r *of* T *is labeled by a vector* l_r *such that* $l_r(j) = 1$ *if and only if* $c_j \in A$*;*

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
s_1	0	0	0	1	0	0	0	1
s_2	0	0	1	1	1	1	0	0
s_3	0	1	1	0	0	0	0	0
s_4	1	1	0	0	0	0	0	0
s_5	1	1	1	0	1	0	1	0
s_6	0	1	1	1	1	0	0	0

Figure 1: Instance of the Persistent Persistent Phylogeny problem where $A = \{c_4\}$.

3. *each edge $e = (x, y)$ is labeled by the set of signed characters c_j (representing a change in the state of c_j), such that $l_x(j) \neq l_y(j)$; namely, if $l_x[j] = 0$ and $l_y[j] = 1$ then the label of e contains the positive character c_j^+ , otherwise it contains the negative character c_j^-*

4. *If c_j is in A, then there exists at most an edge $e = (x, y)$ labeled with c_j^-; if c_j is inactive there are at most one edge $e = (x, y)$ with the label c_j^+. In the later case, if there exists an edge $e' = (u, v)$ labeled with c_j^- then e, e' occur along the same path from the root to a leaf of T and e is closer to the root than e' along this path.*

5. *for each row s of M there exists a node x of T associated to row s and such that the vector l_x is equal to the row s.*

In this case we say that the pair (M, A) is solved by tree T, where A may be also empty, in which case we simply say that matrix M is solved by tree T.

Notice that the previous definition allows the root of the tree to be labeled by a non zero vector. This generalization is useful when considering subtrees of the phylogenies as recursive solutions of the same problem on sub-instances of the input. Indeed, notice that a subtree T' of the tree T solving an instance (M, A) is a solution of the instance $(M, A_{T'})$ where $A_{T'}$ are the characters in the label of T' root.

If there exists an edge of T labeled c^-, then the character c is called *persistent* in T. A special case, called *perfect phylogeny* [8], is when no character is lost in T.

Observe that the definition of persistent phylogeny allows the internal nodes of tree T to be labeled by species.

Definition 2 (standard form)**.** We will say that tree T is in its *standard form* if each node x of the tree is either labeled by a species of matrix M or otherwise it is a node with degree greater than two. **For simplicity we identify the label l_x with the node x, where x is also called *state* of the tree T.**

Notice that given a tree T we can set it in its standard form by contracting to a single edge every simple path whose nodes are not associated to any species of the input matrix and then labeling the new edge with the union of labels along the corresponding path. Given a signed positive character c^+ we will omit the plus + in denoting the character.

The following notions are based on the input matrix M. Let c be a character and let M be the matrix associated to an instance of the PPP problem. Then $S(c)$ is the set of species that have the character c, that is the set $\{s \in S : M[s, c] = 1\}$. Given two characters c_1 and c_2, we will say that c_1 *includes* c_2 if $S(c_1) \supseteq S(c_2)$.

Similarly, we define the set $C(s)$ of characters of s, for a species s of M as follows: $C(s) = \{c \in C : M[s, c] = 1\}$.

3 Red-black graphs and a characterization theorem

The polynomial time algorithm we propose to solve the PPP problem is based on a main characterization stated in [3] that relates the solution of the PPP problem to the one of finding a

sequence of graph operations on a colored graph associated to the instance (M, A), called *red-black graph*.

In the following we show that an instance (M, A) has a graph representation by a red-black graph and vice versa, which will justify our use of red-black graphs in place of binary matrices. A red-black graph is a bipartite graph with red and black edges that connect characters to species. Each species may have incident edges that are black and red, while characters have only black or red incident edges. Formally, given (M, A) an instance of PPP, the *red-black associated with* (M, A) is the graph $G_{RB} = (S \cup C, E)$ that has black edges given by set $E = \{(s, c) : s \in S, c \in C, M[s, c] = 1\}$ for c inactive, and red edges given by set $E = \{(s, c) : s \in S, c \in C, M[s, c] = 0\}$ for c active. In other words, active characters are characters of the graph with only incident red-edges, while inactive characters have only incident black edges.

Vice versa, given a red-black graph $G_{RB} = (S \cup C, E)$, the instance (M, A) *associated with* G_{RB} has species S and characters C and active set A given by the characters in C with incident red edges. Moreover $M[s, c] = 1$ iff (1) (s, c) is a black edge of G_{RB}, or (2) c is active and (s, c) is not an edge of G_{RB}.

Since red-black graphs are used in place of matrices of the PPP problem we associate a tree to a red-black G_{RB}, which is the tree T for the pair (M, A) associated with G_{RB}. We also say that G_{RB} is solved by tree T.

If T is a persistent phylogeny for G_{RB} we also say that G_{RB} is solved by tree T. In a previous paper [3] it has been proved that the tree T can be related to a sequence of graph operations or character realizations on graph G_{RB} which are described in the following definition. In particular, the main Proposition 6 states such a relation.

Let us recall that a connected component is called *nontrivial* if it has more than one vertex. We recall that, given a vertex v of a graph G, the *neighborhood* of v is the set of vertices of G that are adjacent to v, and it is denoted by $N(v)$ [7].

Definition 3 (realization of character c)**.** Let G_{RB} be a red-black graph, and let c be a character of G_{RB}. Let $D(c)$ be the set of species in the connected component of G_{RB} that contains c. The result of the *realization of* c *on* G_{RB}, which is defined only if c is inactive, is a red-black graph obtained from G_{RB} by adding a red edge between c and each species in $D(c) \setminus N(c)$, deleting all black edges incident on c, and finally deleting all isolated vertices. Then c is active in G_{RB}.

An active character c that is connected to all species of a graph G_{RB} is called *red-universal* in G_{RB}; then all the red-edges are deleted from G_{RB} as well as vertex c and it becomes an isolated vertex which is then removed from the graph. Thus as soon as a character c is red-universal it becomes an isolated node in the graph that is removed from the graph.

Definition 4 (realization of a species s)**.** The *realization of a species* s on graph G_{RB} is the realization of its set $C(s)$ of inactive characters in any order. We say that s can be realized in graph G_{RB}, if after its realization s becomes an isolated node of graph G_{RB}.

The following notions of *c-reduction* and *successful reduction* introduced in [1] are used to denote a sequence of character realizations that lead to the empty graph.

A c-reduction R is a sequence $\langle c_1, \ldots, c_k \rangle$ of positive characters: it is called *feasible* if the realization of each character in the sequence R one after the other is possible. Thus a c-reduction represents a sequence of graph operations on G_{RB}.

More precisely, the *application* of a feasible c-reduction $R = \langle c_1, \ldots, c_k \rangle$ to red-black graph G_{RB} is the graph G^k_{RB} obtained as follows: G^0_{RB} is the red-black graph G_{RB}, while, when $i > 0$, G^i_{RB} is obtained from G^{i-1}_{RB} by realizing the character c_i and by removing the characters c_l such that c_l is red-universal after the realization of c_i.

Definition 5 (Successful reduction)**.** A c-reduction R that is feasible for a graph G_{RB} is a *successful reduction* for G_{RB} if the application to G_{RB} makes the graph empty.

A *tree traversal* of T is a sequence of the characters labeling the edges entering nodes of T where each node appears exactly once [14] and has the additional properties that a node always precedes all of its descendants.

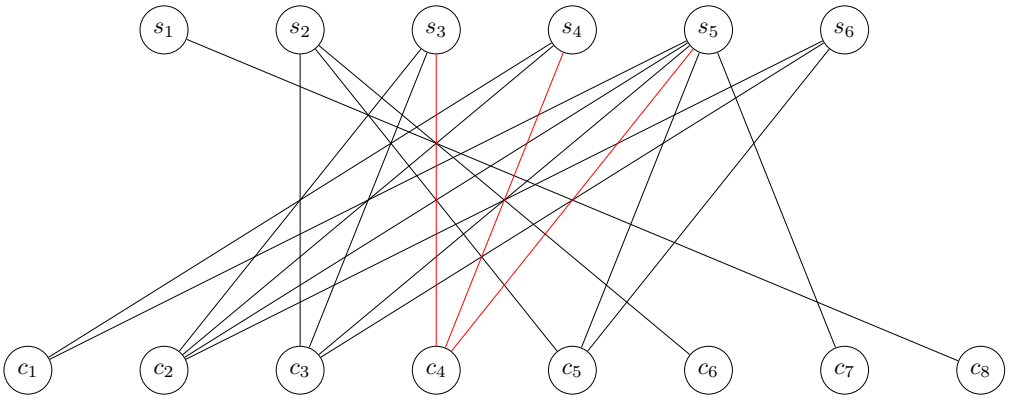

Figure 2: Red-black graph associated with the matrix in Figure 1 and with set $A = \{c_4\}$ of active characters, indeed all its incident edges are red.

The following fact that we use in the paper is stated and proved in [3].

Proposition 6 (Tree traversal)**.** *A tree traversal of the positive characters of a tree T solving a graph G_{RB} is a successful reduction. Vice versa, given a successful reduction of a red-black graph G_{RB} there exists a polynomial time algorithm that computes a tree T solving G_{RB}.*

In other words, we will make use of the following fact: **the PPP problem reduces to the one of finding a successful reduction R for the red-black graph associated to its instance and R gives a tree traversal of a tree solving the graph**. In appendix we gives an example of graph reduction associated to a tree traversal.

We observe that a successful reduction for a red-black graph G_{RB} can be characterized using the notion of red Σ-graph: this is a path of length four induced in G_{RB} by a pair c_1 and c_2 of active characters and by three species s_1 and s_2 and s_3. Observe that c_1 and c_2 cannot be red-universal in the red-black graph as they cannot be connected to all species of the red-black graph by red edges and thus the active characters c_1 and c_2 cannot be red-universal characters. Consequently, a red-black graph containing a red Σ-graph does not admit a successful reduction as it cannot be reduced to an empty graph by a c-reduction.

Definition 7 (conflicting characters)**.** Inactive characters c_1 and c_2 are a *conflicting pair* in a graph G_{RB} if they are overlapping: i.e., , $S(c_1) \cap S(c_2) \neq \emptyset$, $S(c_1) \not\subseteq S(c_2)$ and $S(c_2) \not\subseteq S(c_1)$ and there exists some species s which contains *neither* c_1 nor c_2. Note that this implies that that 4-gametes [8] $(0,0)$, $(0,1)$, $(1,1)$, $(1,0)$ configuration appears in the corresponding matrix.

Definition 8 (forbidden configuration)**.** Given a red-black graph G_{RB}, a species s has a *forbidden configuration* if the realization of species s in G_{RB} induces a red-Σ graph in G_{RB}. We also say that the forbidden configuration is induced by a conflicting pair if the realization of the pair in any order induces a red-Σ graph in G_{RB}.

The notion of species s having a forbidden configuration is the main tool used in the proofs of the basic Lemmas stated in the paper. Indeed, notice that after the realization of s the graph cannot be reduced (see figure 3 for the illustration of this notion).

3.1 Reducible red-black graphs

The following main observation [1, 3] and Lemma 12 will be used in the next sections and are crucial in the polynomial time algorithm for computing a tree solving a red-black graph.

Observation 9. *A red-black graph consisting of k distinct connected components has a successful reduction R if and only if each component G_{RB_i} has a successful reduction R_i. In this case, a successful reduction R can be obtained from any concatenation of the k sequences R_i.*

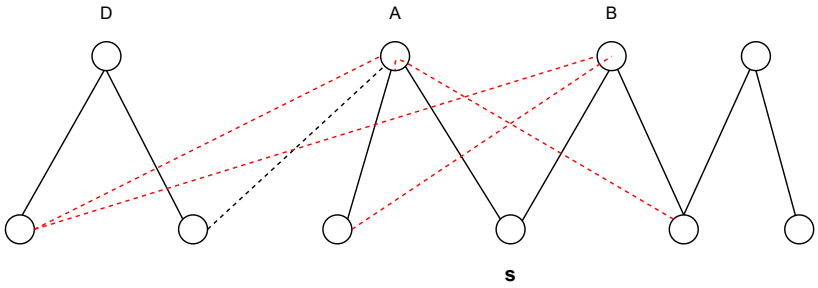

Figure 3: Example of species s that has a forbidden configuration since it contains a conflicting-pair and the realization of the pair of characters A and B (conflicting-pair) in any order induces a red-Σ graph in G_{RB}.

As a consequence of Observation 9, in the paper we consider reducible graphs.

Definition 10 (reducible graph)**.** A red-black graph G_{RB} is a *reducible* if and only if G_{RB} is connected and it admits a successful reduction.

We will assume that the instances of the PPP problem do not contain any red-universal , null, degree-one or universal characters (a null character is an isolated vertex of the red-black graph, while a universal character is adjacent to all species of the red-black graph by black edges -), or a null species (a species that possesses no characters). Notice that the removal of null characters does not modify the phylogeny, while a null species can only be associated to the root of the phylogeny. Removing a universal character trivially consists of fixing the first character of a c-reduction or, equivalently, determining the label of a topmost edge of the phylogeny. **Moreover, we will assume that the instances of the PPP problem do not have two identical columns or two identical species.**

Observation 11. *Let G_{RB} be a red-black graph. If a character c is universal or c has degree one then G_{RB} is reducible if and only if the graph G_{RB} without character c is reducible.*

Moreover, we will use the following relationship between species of the graph: given species s and s', we say that s' *includes* species s if the set of inactive characters of s are included in the set of inactive characters of s'.

Lemma 12. *Let G_{RB} be a red-black graph solved by T. If G_{RB} is connected then the root r of T has only one child x.*

As a consequence of Lemma 12 we can give the following definition of *initial state*.

Definition 13 (initial state)**.** Let G_{RB} be a connected red-black graph solved by a tree T. Then the state given by the unique child x of the root is called *initial state* in tree T.

Remark 14. *Observe that as a consequence of the standard form of a tree T, either the initial state x of tree T is a species of the graph G_{RB} or otherwise it is a node with at last two children.*

Moreover, observe that given a reducible graph G_{RB} then it must contain at least an inactive character, otherwise if all characters are active and not red-universal , it means that the graph contains a Σ-graph which is a contradiction with the fact that G_{RB} is reducible.

4 Maximal reducible graphs

Given a red-black graph G_{RB}, the *red-black graph induced in G_{RB} by a set C' of characters*, denoted as $G_{RB}|C'$, consists of the subgraph G'_{RB} of G_{RB} where the only characters of G'_{RB} are those in C' and the species of G_{RB} are restricted to have only characters in C'.

Definition 15 (maximal characters in the graph)**.** Given a red-black graph G_{RB}, a character c of G_{RB} is called *maximal* in G_{RB} if for any other character c', given S_c and $S_{c'}$ the set of species connected to c and c', respectively, in G_{RB}, then S_c is disjoint from set $S_{c'}$ or S_c includes $S_{c'}$.

The following notion of *maximal reducible graph* is used to denote a special class of graphs having inactive characters that are maximal ones.

Two inactive characters c, c' in the graph *overlap* if they share a common species but neither is included in the other one. In this case we also say that c, c' are not *comparable*.

Definition 16 (maximal reducible graph)**.** A *maximal reducible* graph is a reducible red-black graph such that the set of inactive characters consists only of maximal characters among the inactive ones.

In the following, we denote by:

- C_M, the set of inactive characters for a red-black graph G_{RB} that are maximal among the inactive ones,
- A, the active characters of graph G_{RB}.

5 Finding a reduction of a maximal reducible graph

In this section we give technical Lemmas that are used to find a c-reduction of a maximal reducible graph.

5.1 Maximal character tree

In the following, by a simple path we mean a path consisting of nodes with degree two, except for the root and an ending node of the path which is of degree one, called *leaf* node, or greater than two, in which case the end node is called *split* node.

Lemma 17 (Maximal character tree)**.** *Let G_{RB} be a maximal reducible graph and let T be a tree solving the graph G_{RB}. Then T is composed by a simple path from the root in which all characters in C_M are gained that we refer as* initial path*. Moreover the path finishes in a leaf or in a single split node of degree greater than two under which all edges contain character losses of characters in C_M (see Figure 4).*

We call branch-tree *a tree that includes a split node, also called branch-node. Otherwise, if no such node is given the tree is called* line-tree*.*

Proof. If the tree T is a path then the result is trivially true. Let assume that T is not a path, if two (maximal) characters c_1 and c_2 are gained in two independent branches of T then one of them must be contained in a character that labels an edge that is the minimum common ancestor of edges labeled by c_1 and c_2, contradicting their maximality. Therefore all characters are gained in a single path. Finally, if a character is lost in a branch beginning in an internal node of the initial path then it will contain every character c' activated under it, which contradicts the maximality of c'.

□

[• **paola:** quando si ha uno split node con un ramo di soli inattivi negati e rami con attivi negati, è un branch-tree? - si fare la figura – •]

In the following we will use the properties of maximal inactive characters in a graph G_{RB}:

Property 18 (ordering of negated characters)**.** *Let G_{RB} be a red-black graph and T a tree associated to a reduction R. Given c, c' maximal inactive characters then it holds that:*

1. *if c, c' label the same edge in T, then they are negated along distinct branches of the tree T, i.e. in two distinct connected components of the graph (i.e, in an arbitrary order),*

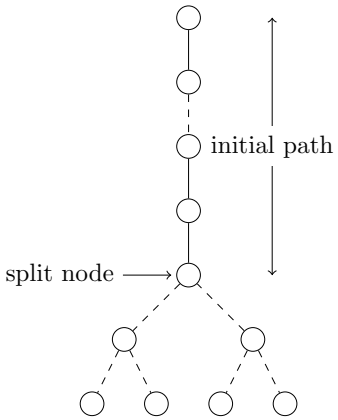

Figure 4: Structure of a tree solving a set of maximal characters, edges containing a gain are drawn with a continuous line. The tree is composed by an initial path containing all character gains, the initial path finishes in a split node under which only losses are allowed.

2. *if c, c' label distinct edges of T and they are negated in a same path of T from the root to a leaf, then they are removed from the graph in the same order in which they were realized in the reduction R.*

The following property is a direct consequence of the previous property 18.

Corollary 19. *Let T be a tree solving a maximal red-black graph G_{RB}. If the initial species of a tree T solving G_{RB} has at least two inactive maximal characters, then no maximal character is negated along the initial-path of tree T.*

5.2 Safe species

Let us recall that graph G_{RB} has species and character nodes. A species s in G_{RB} is *safe* if and only if its realization still produces a reducible graph, we also say that s can be safely removed from graph G_{RB}. Moreover the *degree* of a species s is the number of black and red edges incident to s in graph G_{RB}.

Definition 20 (pending species)**.** In a graph G_{RB} we call *pending species* a species node of G_{RB} having only one incident black edge.

Let us recall that while a character c has only black (i.e. c is inactive) or red incident edges (i.e. c is active) a species can have both red and black edges. Observe that if a species s contains an active character c, then c has no red edge incident to s, i.e. species s contains all active characters that do not have red edges incident to it.

Thus we give the following notion.

Definition 21 (active or quasi-active)**.** A species s is *active* if it has no red incoming edges in the graph and it is realized (i.e. removed) in the graph without producing a red Σ-graph. Otherwise, if s has red-incoming edges and can be realized (i.e. removed) without producing a red Σ-graph is called *quasi-active* species.

Observation 22 (safe removal of a species)**.** *If a species s can be safely removed in graph G_{RB}, then it must be active or quasi-active. Indeed, if the realization of the species produces a red Σ-graph then it is not possible to reduce the graph to the empty one by a c-reduction.*

Definition 23 (Neighbor)**.** The neighbor of a species s in the graph is a species s' that share some inactive character with s.

Definition 24 (Closest Neighbor)**.** The closest neighbor of a species s in the graph is a minimum size species s' of the graph whose set of inactive characters properly includes the inactive characters of s.

Intuitively, if s' is a closest neighbor of s then they are consecutive along a path of black edges of the graph but not necessarily two consecutive species in a path of black edges must be closest neighbors, but they are simply neighbors.

Definition 25 (active pair and p-active)**.** An active pair consists of the pair (s, s') of species such that s is active and there exists a closest neighbor s' of s such that the realization of s and then s' does not induce a Σ-graph. A species s is called p-active if (s, s') is an active pair.

Lemma 26. *Let s be an active pending species of G_{RB} such that s is an internal node of the initial path of tree T solving G_{RB}. Then s cannot be a p-active species.*

Proof. The proof consists in showing that for any closest neighbor s' of s, the pair (s, s') is not active as the species s' has a conflicting-pair of characters (c, c'), where $c \in C(s)$, $c' \in C(s')$ and there exists a configuration $(1, 0)$ for the pair (c, c') i.e. a species having c and not c' that is not removed from the graph when s and then s' are realized. Indeed, assuming that s' is a closest neighbor of s, it must be that s' has a character c' such that is not in s. Clearly, c, c' are not comparable being maximal characters, but they are also a conflicting pair since there exists a species s_0 without c, c'. Such a species s_0 is the one that occurs before s along the initial path of tree T if s' occurs below s in the tree T, indeed being s an internal node, which is a pending species having only character c, such a species s_0 exists in the tree (i.e. s_0 does not have c neither c'. Otherwise, if s' occurs before s along the initial path, s_0 occurs below s in the tree, and such a species exists as there exists a species below s having a new inactive character c'' not in s and c, c'' are not comparable, meaning that c must be negated after c' is also negated. Having proved that c, c' is a conflicting-pair let us show now that the realization of s and s' does not remove a configuration $(1, 0)$ for the pair (c, c') in the graph. Observe that also in this case we need to distinguish the case that s' occurs before s or below s in the tree T. If s' occurs below s in the tree, then there exists a species s_1 having c and not c' and is above s, as s is pending and an internal node, and s_1 is not removed from the graph after the realization of s, s' proving what required, i.e. s_1 induces the configuration $(1, 0)$, as required. The case of s' occurring above s is symmetric.

□

5.2.1 Graphs with a unique pending species

Lemma 27. *If s is the unique pending species in a graph G_{RB} then s can be safely removed from G_{RB}.*

Proof. By definition of pending species, since s has degree one, then it has a unique character $c \in C_M$ and it has all active characters of the graph. Assume that any successful reduction reduction does not start with s. Now, by Lemma 17, either the tree T solving the graph starts with all characters occurring along the initial-path and no species is in the initial-path, or otherwise the initial-path starts with a species s' which is eventually the split node. In the first case, clearly the successful reduction can start with s as it starts with all characters occuring along the initial-path. Thus assume that the initial-path starts with s'. Since s is the only pending species, s' includes at least two inactive maximal characters c_1 and c_2. Since c_1 and c_2 are not comparable characters, as they are maximal inactive characters and since in the graph there exists species s without c_1 and c_2 (otherwise s starts the successful reduction), it follows that s' has a forbidden configuration induced by c_1 and c_2 leading to a contradiction, since by definition the realization of s' leads to the red Σ-graph. It follows that there must exists a successful reduction starting with s, thus proving that s can be safely removed from G_{RB}.

□

5.2.2 Graphs with more pending species or no pending species

The following property is a consequence of the notion of graph G_{RB} as a connected one. Indeed, if we have two consecutive species along a path, if they do not share any inactive character they are in distinct connected components.

Property 28. *Let T be a tree that solves the graph G_{RB}. If s_1 and s_2 are consecutive in a path from the root to a leaf in T then s_1 and s_2 are neighbors.*

[• **paola:** nota bene che qua dobbiamo decidere se la nozione di branch o line si riferisce solo agli inattivi -vedi lemma sottostante •]

Lemma 29. *Let s be an active pending species of a graph. Then s cannot be a split node.*

Proof. By definition of pending species, s consists of a unique inactive character c. On the other hand by definition of split node s it is followed by at least two branches in which inactive characters are negated, which is not possible since the only inactive characters that still has to be negated is the one of s.

□

[• **PB:** esteso il Lemma per specie quasi-active •]

Definition 30 (degenerate)**.** A graph is *degenerate* if it consists of all species s_i having all inactive characters excepts for character c_i and it contains eventually the species s having all inactive characters.

In the following we show that either a graph is degenerate or otherwise a safe species s is given by one of the cases (Proposition 36). We will show how to solve degenerate graphs.

- s is the unique pending,
- s is a minimum size p-active species,
- s is the unique active species,
- s is any quasi-active species of the graph.

Lemma 31. *If the graph is not degenerate and has no p-active species, then it has a unique active species that is safe or the graph has no active species and any quasi-active species s is safe*

Proof. Assume first that the graph is solved by a line-tree or a branch-tree T having an initial-state s. We need to consider two cases: s is active or s is not active. Case i): assume that s is active, since it is not p-active the following cases are possible: i.1) s is not followed by any other species in the initial-path, or i.2) for any other species s' of the initial-path that follow s, s' does not include the inactive characters of s (indeed, if s is followed by a species s' including s, by construction (s, s') is an active pair as they are the first species to be realized in the successful reduction leading to the tree T. If case i.1) holds then it is immediate that the graph is degenerate which is not possible. If case i.2) holds, the observe that an active character is negated in s'. This fact it implies that all the other species in the tree are not active, which implies that s is the only active species in the tree, thus proving the Lemma in this case. Indeed, being s the initial-state of the tree T, the only character that can be negated is the active one (see property 18. Case ii): assume that s is not active which implies that all species in the tree are not active. Clearly s is quasi-active, i.e. s consists of at least an inactive character and the negation of an active character c and the realization of s does not induce red Σ-graphs. Clearly by construction s is safe since it is the first species of the traversal of T, i.e. of a successful reduction of the graph. Let us now show that any quasi-active species s' of the graph is safe. Observe that by construction the realization of the inactive characters of s must disconnect the graph into disjoint components thus allowing

to free active characters. Let $\mathcal{C}_1$ be the component consisting of the subgraph induced by the the active characters of s and their incident species. Let $\mathcal{C}_2$ be instead the component consisting of the subgraph not having species incident to the active characters of s. Then after the realization of s, the active characters of $\mathcal{C}_1$ must be free to remove them and then the species s from the graph. Thus it holds that the graph is decomposed into components $\mathcal{C}_1$ and $\mathcal{C}_2$ such that only one of the two components has quasi-species. Indeed, by the main property that a tree solving a maximal reducible graph does not have inactive characters below the split node, if after the realization of s we have distinct components representing two subtrees, then both of them cannot have inactive characters. It follows that the tree must be a line-tree and the realization of s produces exactly two components $\mathcal{C}_1$ and $\mathcal{C}_2$, the one having s and the other having the quasi-active species s'.

By the above argument a quasi-active species is a species such that its realization decompose the graph in disjoint components and it is removed from the graph, we can easily show that it cannot be the internal node of the line-tree. It follows that s' is a leaf of the tree and we can invert the tree and have s' has an initial state of the inverted tree. Hence it follows that also s' is safe, thus concluding the proof.

□

Lemma 32. *If the graph is not degenerate, then any pending species that is p-active is safe.*

Proof. Observe that the pending species are the species of minimum size and they must have all active characters, otherwise they have incident red edges. Since the graph is not degenerate, then it is solved by a tree such that has an initial state along the initial path of the tree. Let s be a p-active species of the graph that is pending and assume that is not the initial-state of any tree solving the graph. Assume first that s is the internal state of a path of the tree. By Lemma 26 s cannot be p-active.

Consequently, s must be a leaf of the tree or s is the split node of a branch-tree. This last case is not possible by Lemma 29. Since s is p-active there must exist a closest neighbor species s'' that includes s and such that the realization of s and s'' does not induce red Σ-graph. First assume that s'' is the species immediately above the leaf labeled s. Assume now that s consists of the only character c while s'' has the additional character c''. Now, being c, c'' maximal characters, they are not comparable and since the initial-state is distinct from s they are conflicting characters as the initial-state induces the $(0,0)$ configuration. Now, the following two cases are possible: 1) either c occurs in the tree before c'' or 2) c and c'' occur on the same edge. Let us consider case 1) first. It is immediate to have a species that induces the configuration $(1,0)$ and is distinct from s (indeed observe that s is a leaf of the tree), which implies that after the realization of s, s'' has a forbidden configuration induced by the characters c, c''. Hence (s, s'') is not an active pair. Consequently, the only possible case is that c'' occurs before c along a line-tree ending in s thus implying that there is an alternative line-tree starting from s and obtained by inverting the line-tree. Consider now case 2), i.e. c, c'' label the same edge of the initial-path. Now, let the edge labeled c, c'' be preceded by the edge labeled c'. Clearly, by property 18 c, c'' are negated in two distinct components of the graph (i.e. the negations of c, c'' occur along distinct paths of the graph) which implies that c or c'' are in conflict with c'. Now the realization of c, c'' does not remove the conflict with species having c' unless the edge labeled c' is the one incident to the initial-state of the tree, that is the configuration $(1,0)$ for the pair c', c'' or for the pair c', c is removed after the realization of c'. Otherwise if the initial-state is another $(0,0)$ configuration for the pair c', c'' or for the pair c', c, since there are two distinct species in the leaves one with c and one with c'' and both cannot be realized, it follows that the conflict c', c'' or c, c' cannot be removed. In other words this fact proves that only three characters are possible in the tree in order to have this case to hold. See figure 6 for this case. Then s can be the initial-state of a tree solving the graph, proving the Lemma.

Finally to conclude the proof, let us consider the case the closest neighbor of s is a species s' such that is the one below the initial species of the tree solving the graph, as the only character c such that c and the character c' of s are not conflicting is the the one introduced in the initial species of the tree. Indeed in this case c and c' are overlapping but they are not in conflict. This

case is only possible in trees that are isomorphic as illustrated in figure 5 by exchanging labels c with c' in the tree solving the graph. Moreover such trees have at most three characters. They are called *3-canonical* trees.

□

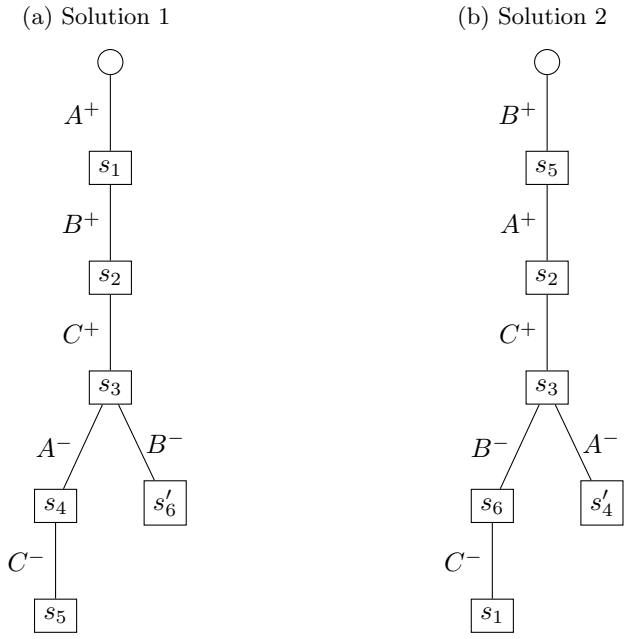

Figure 5: Two trees having two pending p-active species. In solution 1, we have species s_1 and in solution 2 we have species s_5.

Lemma 33. *If the graph is not degenerate and has no pending species, then it has at most two p-active species s_1, s_2 of minimum size that are safe and are called* sources *of the two trees T_1 and T_2 solving the graph. Then the following statement holds:*

1. *source s_1 differs from source s_2 by a single character A which is replaced by B in source s_2,*
2. *source s_2 is a leaf of tree T_2 and then T_2 is obtained by inverting the path of T_1 from s_1 to s_2.*

Proof. Let s_0 be the species labeling the initial state of a tree solving the graph. First observe that since s_0 is not pending it means that it has at least two non active characters. Let s be a p-active species of the graph that is of minimum size (it is not necessarily unique).

By contradiction, let us assume that species s is not contained in the first species of a reduction R associated to a tree T solving the graph. that is s is not contained in s_0. Then it must be that species s is an internal node of a path of tree T or otherwise it is a leaf of tree T.

However, if s is an internal node, since s_0 is not pending, by Corollary 19 no inactive character is negated along the initial path, it follows that s contains s_0, thus contradicting that s is the p-active species of the graph of minimum size. Indeed, s_0 is of size smaller than s and s_0 is by construction a p-active species of the graph. It follows that s must be a leaf.

Observe that by Corollary 19, each character of s_0 must be negated along a distinct path, implying that only one character of s_0 may be not present in s. Being s active, by definition it includes all active characters of s_0 and let A be the only inactive character of s_0 not present in s. Observe that s_0 and s are not comparable species as species s is of minimum size and s_0 cannot contain s. Thus assume that in s the character A is replaced by a character B. Observe that if another character D occurs in s, where D does not occur in s_0, then D and B are conflicting thus

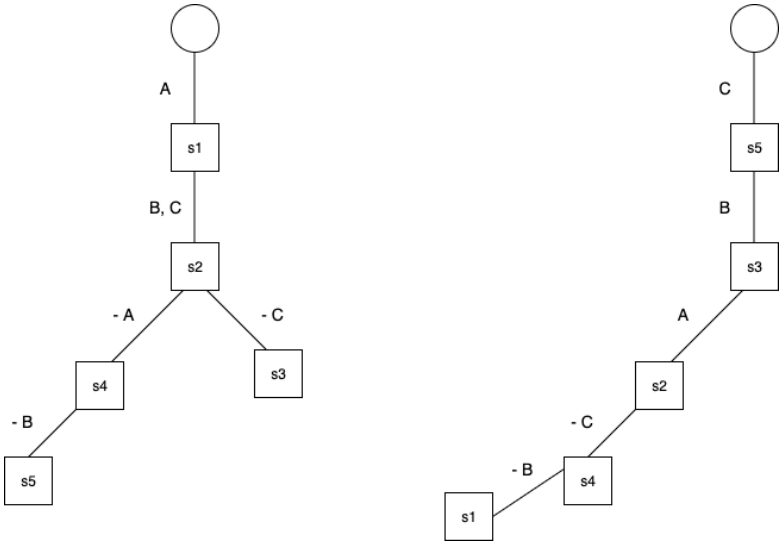

Figure 6: Example of two solutions where the species s_5 is a leaf and also the initial state of the tree.

implying that the realization of s induces a red σ-graph. Thus we need to assume that s, s_0 differs only by a single character. This proves the first statement of the Lemma.

We distinguish two cases: i) B does not label the last edge of the initial path ii) B is the last character introduced in the initial path. Assume first that case i) holds: then there is a path below the split node where all characters below B including A are negated. Assume that C is incident to the split node. Then it is easy to show that C, B are conflicting characters and thus in this case s cannot be a source of a tree solving the graph, as there are two species having B and not C and the species s_0 is a $(0, 0)$ configuration for the conflicting pair B, C. So this case is not possible (see Figure 7 Solution 1)).

Assume now that case ii) holds, that is B is the last character introduced in the initial path. It means that all the characters along the initial-path of tree T, and occurring after A are negated before the occurrence of species s and each one labels a single edge and they are negated in the same order they are introduced by Property 18. Indeed, if two characters label the same edge, then they are negated on two distinct paths, which implies that s will have also another character of the initial-path distinct from B, a contradiction. By inspection, it is easy to see that the tree having the path from s to s_0 inverted is a solution of the red-black graph and and thus there exists a reduction starting from the species s (see Figure 7 Solution 2). Indeed the two trees have the same set of species of the graph. This case proves the Lemma. □

Thus a main consequence of the previous Lemma is the following result.

Corollary 34. *If the graph has no pending species and is not degenerate, then the p-active species of minimum size is safe.*

Lemma 35. *If the graph is degenerate, then any inactive character can be realized in the graph producing a reducible graph.*

Proof. This is a consequence of the definition of degenerate by which the graph is solved by a tree where all inactive characters are introduced along the initial-path and then negated below the split node to realize its species.

□

A main consequence of the previous Lemmas is the following Proposition:

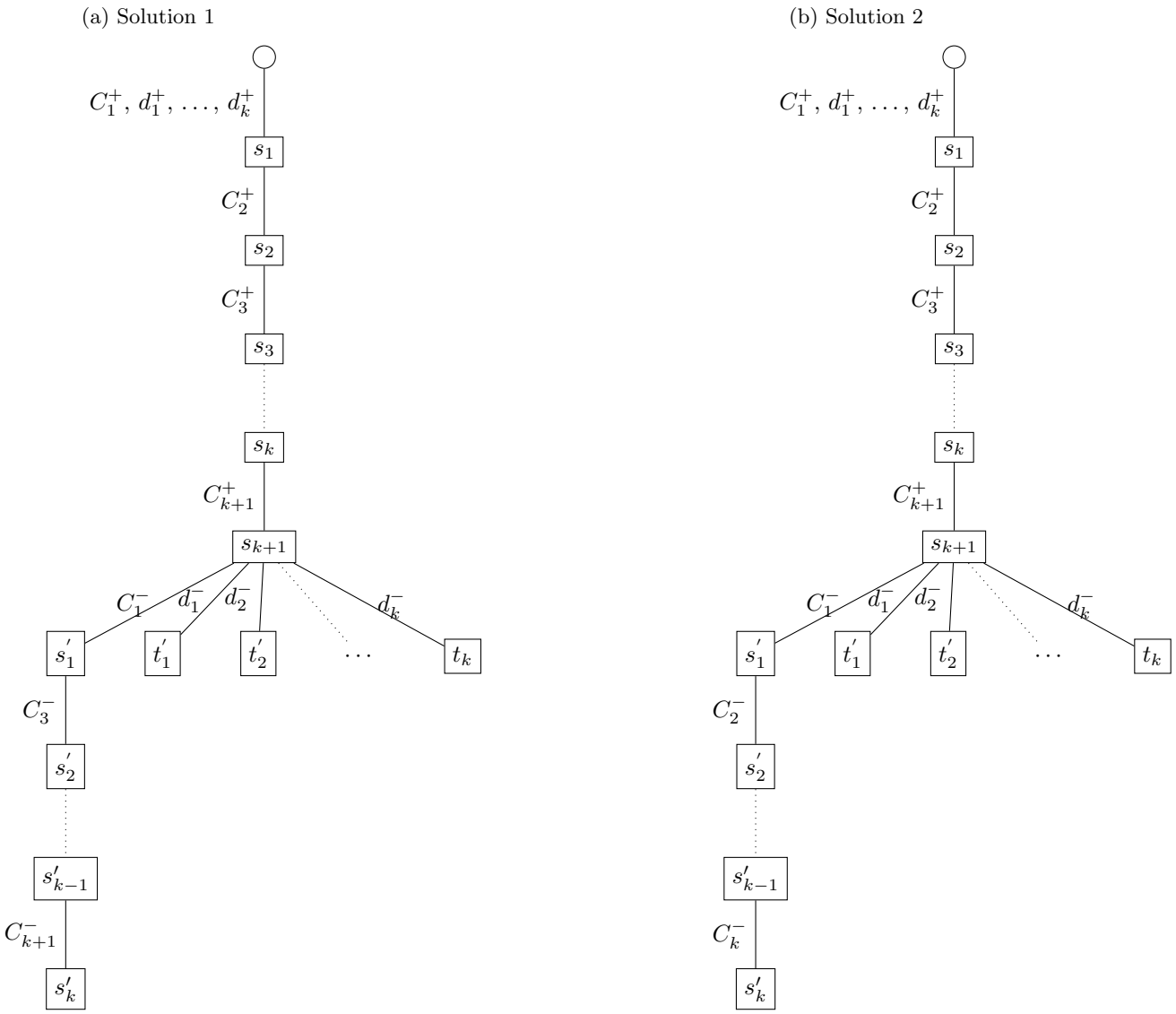

Figure 7: The figure illustrates case i) and case ii) of the proof of Lemma 33.

Proposition 36. *Given a non degenerate graph G_{RB} and s a safe species of G_{RB}, is at least one of the following statements holds:*

1. *s is an active pending species,*
2. *s is a p-active species of minimum size,*
3. *s is a quasi-active species and all species have an incoming red-edge,*
4. *s is a unique active species.*

Proof. Assume that in a tree solving the graph a species s of the initial-path is followed by a species s' including s, then it is immediate that s is a p-active specie, since the realization of s and s' does not induce red Σ-graphs. If the graph G_{RB} has no p-active species, then by Lemma 31 and Lemma 34 the Proposition follows. □

6 The polynomial algorithm for maximal reducible graphs

A polynomial algorithm for a maximal reducible graph directly follows from the previous results and consists of iterating in the graph the realization of a safe species till the graph is reduced: whenever such a safe species does not exist it mean that the graph is not reducible. The correctness of algorithm $PPP(G, \emptyset)$ is a direct consequence of Proposition 36.

Algorithm 1: PPP

Data: M binary matrix
Result: PPP(M)
initialization: Construct RB graph G;
$PPP(G, \emptyset)$;

Algorithm 2: PPP(G, A)

while *Graph is not void* **do**
  if *There is a unique pendant species s* **then**
    $r := s$
  else if *There is a p-active species s* **then**
    $r :=$ minimal p-active species
  else if *G is degenerate* **then**
    $r :=$ the set of all inactive characters
  else if *G has a unique active species s* **then**
    $r := s$
  else if *G has a quasi active species s and all the species have a red-incoming edge* **then**
    $r := s$
  realize r;
  update G and A;
  foreach *G_i connected component of G* **do**
    PPP(G_i,A);

6.1 Characterization of line-trees

Let T be a line-tree solving a graph G_{RB}. If the line-tree has active characters, then the graph has a unique active minimum species which is safe. Then the solution is unique. Otherwise the graph has at most two minimum size active species which correspond to the unique pending species of the graph: they are the initial and end species of the line-tree. Thus the graph is solve by two trees, the tree T and the tree obtained by reading the t tree T from the end leaf species.

6.2 Characterization of branch-trees

In this section we consider the number of trees solving maximal reducible graphs.

The solution is unique in the case the graph G_{RB} admits a tree such that the initial-path of the tree contains a negated character, that is the branch-node is not complete, i.e. does not have all maximal characters: the graph is called *1-solvable*.

Lemma 37. *Let G_{RB} be a graph such that is solved by a branch-tree whose split node is not complete. Then the graph is 1-solvable.*

Proof. The proof consists in showing that there exists no p-active species in the graph except for the initial-state s of the tree solving the graph. Indeed, since there is a negated active character along the initial-path it means that the initial-state consists only of a single character A and induces a $(0, 0)$ configuration for any pair of characters that are below the single character A. Any other species s' such that is active and (s', s_2) is a active pair is such that s consists of a single character or otherwise s' contains a conflicting pair in the graph for the previous argument.

If s' is a leaf species then since C occurs along the initial-path, it is immediate to show that s_2 has a conflicting pair (C, D) which cannot be removed in the graph by the realization of C since there are two species having C and without D, one is s' and the other occurs in the initial-path. Otherwise if s' occurs along the initial-path, then there exists a species with C and A and without the character in s_2. It follows that again s_2 has a conflicting pair (C, D) which cannot be removed in the graph by the realization of C since there are two species having C and without D, one is s' the other the species with A and C. □

The proof of the previous Lemma 34 provides a characterization of branch-trees solving graphs with no pending species and a minimum size p-active species. The characterization shows that a graph may admit at most two p-active species s.

Definition 38. A graph may have two or more p-active species s of minimum size that are safe and in this case is called:

- **2-solvable** : it is solved by two branch-trees that differs one from the other by inverting a path from the root to a leaf, called *inverted-path* - the minimum p-active species are called **sources** of the graph,
- **3-canonical**: it is solved by two branch-trees that are isomorphic by the renaming of two characters and consists of three characters D, B, A and species $\{\{D\}, \{B\}, \{B, D\}, \{B, D, A\}, \{A, B\}$,
- **m-solvable or degenerate**: it is solved by a branch-tree illustrated in figure .... da fare
- **line-2-solvable**: it is solved by two line-trees that differ one from the other by the by inverting a path from the root to a leaf, called *inverted-path* - the minimum p-active species are called **sources** of the graph.

Observe that a main property of a c-reduction for a graph is the following:
Property c-reduction: Any subsequence of a c-reduction over the set X of characters is a c-reduction for the subgraph induced by the set X of characters.

We need to consider also the case of graphs having no p-active species since the graph consists of species all having red-incident edges: we call such a graph hybrid-graph. Now, we know that a c-reduction R of an hybrid-graph includes as a sub-sequence the c-reduction R_s of the sub-graph consisting only of the inactive characters. Thus the characterization stated above applies also in this case, i.e. either we obtain a 2-solvable or a 3-canonical or m-solvable or degenerate graph.

7 Solving general graphs

In a general graph we distinguish inactive minimal characters from maximal ones, where a minimal character is a character of G that is not maximal, being a maximal character a character that is not included in another character of G.

By a g-skeleton of a graph G_{RB} we mean the subgraph induced only by the maximal inactive characters of G_{RB} and by the active characters of the graph. Given a graph G_{RB} having g-skeleton a graph G_M, by the Property c-reduction, the c-reduction of G_M must be a sub-sequence of a c-reduction of G_{RB}. Observe that the c-reduction starts with maximal inactive characters of the graph G_M. Indeed assume to the contrary that the c-reduction starts with inactive minimal characters of a species s of the graph. Given s' a species that follows s in then c-reduction, then it has inactive characters that are not in s, thus implying that characters of s' and not of s cannot include characters of s, contradicting the assumption that that s has only minimal characters. Indeed, the only way to have inactive minimal characters in s is that such characters are included in a maximal inactive character of s. It follows that s must have at least an inactive maximal character, and thus the the c-reduction starts with maximal inactive characters of the graph G_M, as stated above.

7.1 Minimal form for general graphs

In the rest of the section we will consider only general graphs that are in *minimal form* which is defined by using the notion of *Y-pending set of characters of graph G*: a set X of minimal characters of G is Y-pending, for Y a set of characters of G, if each character in X is contained in all characters of Y while it is **disjoint from characters in** $C(G) - \{X, Y\}$. Now, by this last property, given a minimal character x in X, then the set $S(x)$ of species incident in the graph to node x is included in the set $S(y)$ of the species of any character y in Y, while for any character z not in $Y \cup X$ it holds that set $S(z)$ is disjoint from $S(x)$. Let G_r be the subgraph of G such that is induced only by characters that are not C-pending w.r.t. any set C: we call such graph in *minimal form*. Let G_m be the subgraph of G induced by the set C of characters and by a set X which is C-pending (see Figure 8).

Clearly if G is reducible then subgraphs G_r and G_m must be reducible. Indeed by Proposition 6, a c-reduction of G consists of a traversal of the tree T solving G: such a traversal may be split into a traversal of tree T_r solving G_r and then by a traversal of the subtrees in the forest T_m corresponding to components of G_m. Indeed, T_m consists only of characters that do not overlap any characters of T_r and thus the traversal of T may consists of a separate traversal of T_r and then of T_m, i.e. there exists a c-reduction for G consisting first by a c-reduction of G_r and then a c-reduction of G_m, proving what stated above. Consequently, it is not restrictive to assume that the polynomial time algorithm for reducing general graphs applies to G_r first and then applies recursively to the connected components obtained by the c-reduction of G_r (See Figure 11).

Theorem 39. *Given G a reducible graph and G_r the subgraph in normal form and $\mathcal{G}$ the collection of subgraphs induced only by the characters not in G_r, then there exists a c-reduction of G that consists of a c-reduction of G_r followed in any order by a c-reduction of the graphs in $\mathcal{G}$.*

Proof. The proof of the Theorem is a consequence of the main observation that applying a c-reduction to G consisting only of characters of G_r the resulting graph graph consists of the collection $\mathcal{G}$, as indeed all species of the characters in $\mathcal{G}$ are properly contained or disjoint from the characters in the c-reduction. Consequently, the separate c-reductions of graphs in $\mathcal{G}$ can be applied to reduce the graph to the empty one. □

Observe that in the rest of the paper we assume to consider reducible graphs in minimal form since we can apply a recursive step. If the algorithm fails to produce a c-reduction it means that the graph is not reducible.

7.2 General graphs with a 2-solvable g-skeleton

Observe that in a 2-solvable graph there are two trees solving the graph: each tree starts with a different species which are called sources. When considering a general graph G whose g-skeleton is a 2-solvable graph then we need to understand which of the two sources are included in a species s' that starts a successful reduction of the graph G: let us recall that we say that s' is *safe* for G in this case.

Extension of a source In the following we define *extension of a source s* of a 2-solvable g-skeleton of G a minimum size species s^* of G including s such that the realization in G of s^* and of all minimal characters in overlap with the ones in s^* does not induce a red-Σ-graph in G.

Lemma 40. *Let G_M be a g-skeleton graph of a graph G such that G_M is 2-solvable and let s be a source of G_M. Under the assumption that G is in minimal form, there exists a unique extension s^* of s in G.*

Proof. Assume to the contrary that there is also another species of minimum size s' that includes the source s of the g-skeleton, while s^* is the initial state of a tree T solving G, i.e. s^* is an extension of s. It is not restrictive to assume that s' includes only the minimal character b while s^* includes only the minimal character a. In the following we show that the realization of s' in G and a and b produces a red-Σ-graph in G. By the assumption that G is in minimal form, it

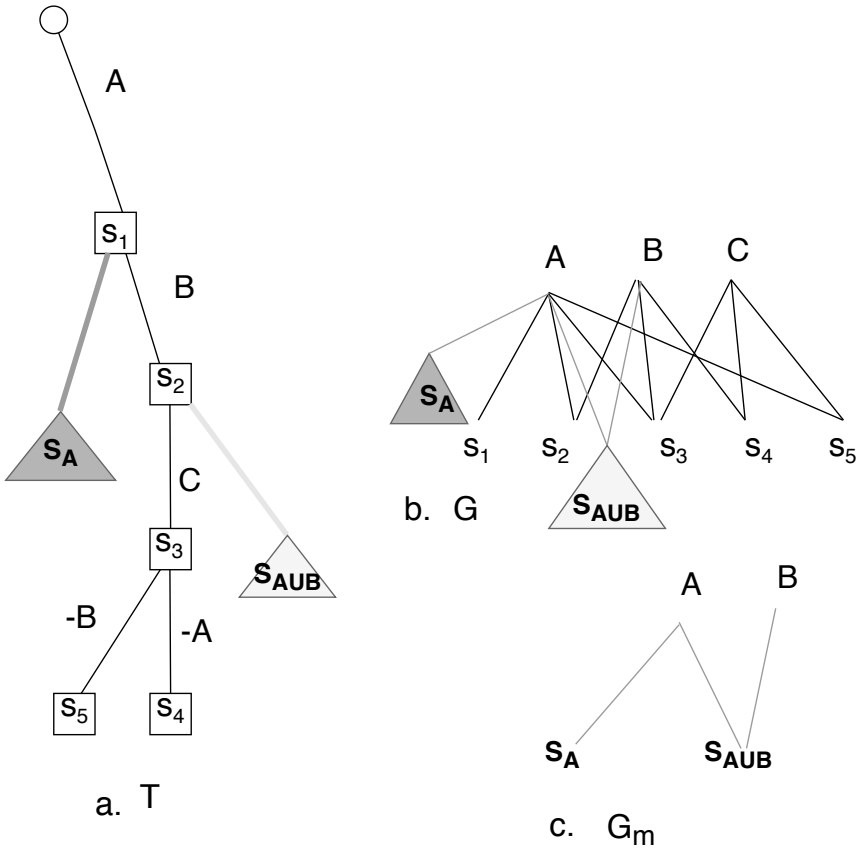

Figure 8: Figure a. represents a tree T solving the graph G of figure b. The tree has a forest T_m of subtrees corresponding to $\{A \cup B\}$-pending characters and A-pending characters; such a forest is represented by the grey substrees. Figure c. instead represent the subgraph G_m induces by the species of the grey forest.

must be that in tree T the species s' occur below the species s^* and moreover it must be that b is negated below s^* being a minimal character and thus (a, b) is a conflicting pair, as a is negated before b is negated. Clearly, b is overlapping a character distinct from a being the graph G in minimal form, otherwise it holds that a, b are Y-pending for Y the maximal characters of s. This fact implies that there are at least two species having b and not a and thus the realization of s'

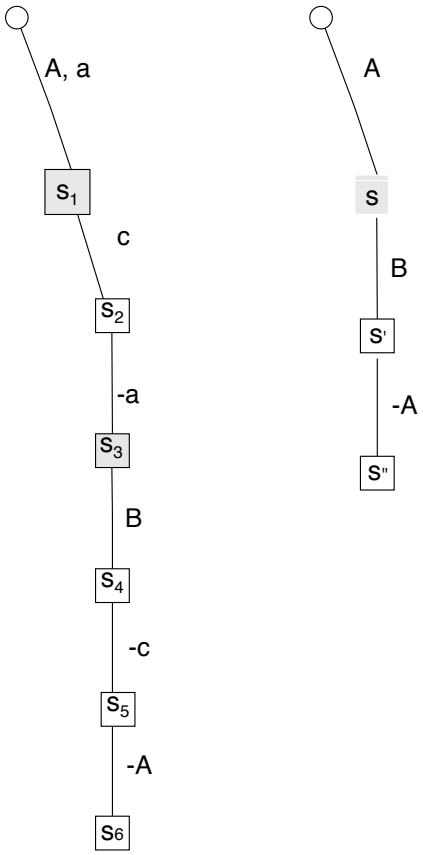

Figure 9: Figure illustrate two trees, on the left the one solving a graph G and on the right the tree solving the maximal induced graph of G. Note that s_1 and s_3 are two minimal species that include the species s consisting only of the maximal character A. However only s_1 is an extension of species s, for s the initial-state of the tree solving the maximal graph induced graph. On the other hand species s_3 is not an extension of species s.

and b and then a produces a red-Σ-graph in G as the conflict is not removed after the realization of b (see Figure 9), thus obtaining a contradiction with the definition of s' being an extension of s.

□

The following result is an easy consequence of the previous Lemma 40 and the definition of extension s of a source: indeed the realization of all minimal characters overlapping the ones in s can be done in polynomial time and the condition of Σ free graph is easily tested in polynomial time.

Corollary 41. *Given a graph G and G_M its g-skeleton graph and s a source of G_M, then the extension s^* of s in G can be computed in polynomial time.*

Let s be the extension of a source of a 2-solvable g-skeleton of G: then let us define the closure of s, $CL(s)$. Intuitively, the closure will consists of all the minimal characters of G that occur in an initial-path of a tree that starts with s and are negated along the initial-path.

$CL(s)$ is defined as follows: $a \in CL(s)$ if and only if a is minimal in G and moreover it is included in all maximal characters of s.

Lemma 42. *Let G be a graph whose g-skeleton is 2-solvable and such that s is safe for G, being s the extension of a source of the g-skeleton of G. For any minimal character a of G, it holds that $a \in CL(s)$ if and only if a is introduced and negated along the initial-path of the tree T with initial state s.*

Proof. Assume first that $a \in CL(s)$. Assume to the contrary that a is negated below the split node. This fact implies that a overlaps a maximal character C which is in s since C is negated along the branch distinct from the one where a occurs, a contradiction with the fact that it must be included in all maximal characters of s. Indeed, let us recall that first maximal characters to be negated in 2-solvable graphs are those in s since s includes at least two maximal characters. Assume now that a is introduced and negated along the initial-path of the tree T with initial state s. It is immediate that $a \in CL(s)$ being a included in all maximal characters of s. □

Lemma 43. *Let G be a graph whose g-skeleton is 2-solvable and let X be the set of minimal characters that are not in the closure of any of the two extensions of the two sources of the graph and let T be a tree solving the graph. Then X consists only of minimal characters that are introduced in the initial-path of T and then negated along the inverted-path below the split node of T or otherwise are negated along paths distinct from the inverted-one of T.*

Proof. The Lemma is a consequence of Lemma 42. □

Inverted-graph from s of a graph G having a g-skeleton that is 2-solvable Let G be a reducible graph.

Let s be a source of a g-skeleton of G that is 2-solvable. Let $CL(s)$ be the closure of s and consider the subgraph of G induced only by the maximal characters and the characters of $CL(s)$: we call such graph the *the s-graph*.

Then the *inverted-graph from s* of graph G is the subgraph of the s-graph consisting only of the species that all have the maximal characters in the intersection of $C(s_1)$ and $C(s_2)$, for s_1 and s_2 sources of the g-skeleton of G. Intuitively, the inverted-graph from s is solved by the inverted-path from species s.

example 1. By the characterization of 2-solvable g-skeleton of a graph G, it is immediate that an inverted-path consists of all the species of the g-skeleton that include the set consisting of the maximal characters that are common to the two sources of the g-skeleton, that we call *source-characters*. Then the inverted-graph from s is obtained by considering all species of the s-graph including all source-characters.

Definition 44 (l-source)**.** The extension s^* of a species s that is a source in the g-skeleton of G is an *l-source of G* if the inverted-graph from s is a line-tree.

Let us recall that by construction the inverted-graph from s is a line-tree since it consists of species of a tree T solving the graph G that lay along a path from the two sources of the g-skeleton.

Informally the extension s of a source is not safe if it is the leaf an inverted-path such that along such path occurs a minimal character b that is then negated along the branch distinct from the inverted-path and moreover b overlaps a minimal character of s (see Figure **??**) i.e. it is of type-1 according to the following notion.

Definition 45 (type-1)**.** Let s^* and s'^* be the extensions of the sources s, s' of a 2-solvable g-skeleton of a graph. Then s^* is of *type-1* if it includes a minimal character a such that the following conditions holds:

1. a is not in s'^*
2. a is in overlap with another minimal character b, b not in s'^* and not in $CL(s)$ and b is not in the leaf species of paths distinct from the inverted-path of a tree T solving the graph.

Observation 46. *Observe that by the condition 1 of the definition of type-1, in a tree having initial-state s^*, s^* is never of type-1. Indeed, since a and b are not in the extension s'^*, it means that given b not in $CL(s^*)$ it is negated along the inverted-path below the split node, otherwise by Lemma 42 it is in $CL(s^*)$ which is not possible. Moreover, being b in overlap with a, b is introduced along the initial-path. Indeed a is minimal and it is in s^* which implies that a must be negated along the initial-path as a is not in s'^* (indeed otherwise a is negated below the split node and along the inverted-path which contradicts that a is minimal). Thus either b is in s^* or otherwise b must be negated along the inverted-path which implies that b must be in the leaf of the path which is not the inverted one. The type-1 condition is used to characterize the extension s^* of a source that is leaf of the inverted-path of a tree solving G and which cannot be safe (see Figure...).*

[• **paola:** controllare che sia correttamente indicata s rispetto a s^* estensione •]

Lemma 47. *Let s^* be the extension of a source s of the g-skeleton of graph G. Assume that s^* is safe for G. Then s^* must be a l-source of G and it is not of type-1.*

Proof. Assume that T is a tree solving graph G with s^* the initial-state of T, being s^* a safe species of G. First by the previous observation 46, it holds that s^* cannot be of type-1, as required by the Lemma.

Assume to the contrary that s^* is not a l-source. Observe that by Lemma 42 it must be that for any character a in $CL(s^*)$, a must be introduced and negated along the initial-path of tree T. But this fact implies that the subgraph consisting of the inverted-path of the g-skeleton extended with $CL(s^*)$ is a line-tree, a contradiction with the initial assumption that s^* is not a l-source. Observation 46 concludes the proof. □

In the following we show that for a l-source being not of type-1 is a sufficient condition to be a safe species of the graph, thus having a necessary and sufficient condition that can be tested in polynomial time that allows to detect efficiently the safe species of graph G.

Lemma 48. *A l-source species s not of type-1 of a graph is safe. There exists a polynomial time algorithm to test if a species s is a l-source not of type-1.*

Proof. We prove the Lemma by showing the construction of a c-reduction for graph G starting from species s. Indeed, by assumption graph G is reducible. Now, assume to the contrary that the c-reduction only starts from an extension of the source s_1, where s_1 is distinct from the source s_2 of which s is the extension. Observe that the subgraph induced by $CL(s_1)$ and the species of the inverted path is a line-tree by lemma 47, where $CL(s_1)$ are characters that are not in $CL(s_2)$. More precisely, by Lemma 42 $CL(s_1)$ are characters that are introduced and negated along the initial-path of a tree T_1 starting from the extension ${s_1}^*$ of s_1. Now since s is also a l-source, it must be that the inverted-graph G' from s is a line-tree, which implies that also the subgraph of G obtained extending G' with characters in $CL({s_1}^*)$ is a line-tree. Now, let us consider the set X of the other minimal characters that are not in $CL(s_1)$. Let X' be the subset of X of all minimal characters that are introduced and negated along the inverted-path or are introduced and then negated along the branches which are not the inverted-path. Let X'' be the subset of X of all minimal characters that are negated along the path that is not the inverted-one and are introduced along the initial-path of a tree T_1. Now, if X'' is not empty it follows that either s or ${s_1}^*$ is of type-1, which is not possible by Lemma 47.

As a consequence only X' can be not empty. Now, it is immediate to verify that the tree T' obtained by inverting the inverted-path of tree T_1 is also a solution of the graph, thus proving that we have a c-reduction starting with s.

Let us now show that there exists a polynomial time algorithm to compute a species s that is a l-source not of type-1. By Corollary 41 the extension of the sources can be computed in polynomial time. Then by definition given the extension s, testing whether a graph is a line-tree can be done using the well known algorithm []...Observe that $CL(s$ is also easily computed in polynomial time as well as the property of s of not being of type-1, as defined in definition 45. □

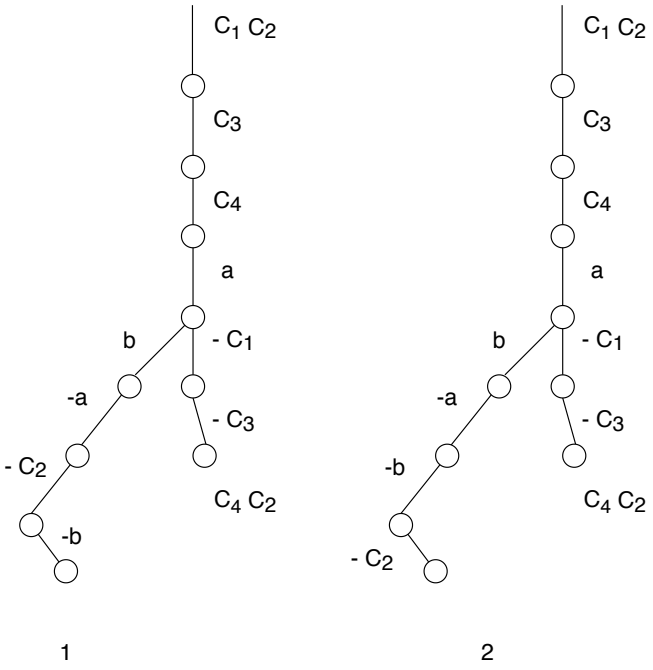

Figure 10: In tree 1, the species s having characters $\{C_1, C_2, a\}$ is of type-1, while in tree 2 s is not of type-1, since b is in $CL(s)$

7.3 The general algorithm for 2-solvable graphs

In the following let us detail the pseudo-code for the general algorithm for 2-solvable graphs.

Algorithm 3: PPR-general(G_i, A)

```
while Graph is not void do
    Compute the g-skeleton G_M;
    if G_M is 2-solvable then
        r := Source-2-solvable(G, A);
    else if G_M is 3-canonical then
        r := Source-3-canonical(G, A);
    else if G_M is m-solvable then
        r := Source-m-solvable(G, A);
    if r is not empty then
        realize r in G;
        A:= set of characters of r;
        update G and A;
        foreach G_i connected component of G do
            PPP-general(G,A);
    return fail;
```

The previous algorithm uses the procedures detailed below.

Algorithm 4: Source-2-solvable(G, A)

```
Compute the g-skeleton G_M;
if G_M has a unique source s then
    if s has a unique extension s* then
        r := s* ;
        return r;
else if G_M has two sources s_1, s_2 then
    else if s_1 and s_2 have a unique extension s_1* and s_2*, respectively then
        if s ∈ {s_1*, s_2*} and Test-l-source(s) and NOT Type-1(s) then
        r := s;
        return r;
return the empty set;
```

The following procedure tests in polynomial time whether the species s^* is a l-source according to Definition 44.

Algorithm 5: Test-l-source(s)

```
Let G' be the g-skeleton of G such that is 2-solvable;
C_s := maximal characters of s;
foreach a, minimal character of G do
    if a included in each character of C_s then
        CL(s):= {a} ∪ CL(s);
        % Compute CL(s) := the closure of s;
s-graph : = the subgraph of G induced by C_s ∪ CL(s);
% compute the the s-graph ;
G_s : = the subgraph of the s-graph induced by the species including all characters in
  C(s) ∩ C(s'), s' the other source % G_s is the inverted-graph from s;
if G_s is a line-tree then
    return true;
return false ;
```

The following procedure tests in polynomial time whether the species s^* is a type-1 according to Definition 45.

Let us recall that species of the inverted-path are only those having maximal characters in $C(s) \cap C(s')$ for s, s' the species that are the ends of the inverted-path of the g-skeleton of G.

Algorithm 6: Type-1(s^*)

Let s'^* be the extension of the other source of a 2-solvable g-skeleton of graph G;
foreach $a \in C(s^*)$ *that is a minimal character and* a *not in* $C(s'^*)$ **do**
  Set-B: $= \{b$: b is a minimal character not $\in C(s'^*) \cup CL(s)$ and b in overlap with $a\}$;
  Leaf-species-not-in-inverted-path:$= \{s_m : s_m$ is a minimum size species of G such that $C(s) \cap C(s')$ is not a subset of $C(s_m)\}$;
  foreach b *in Set-B* **do**
    if b *is not in any species of set Leaf-species-not-in-inverted-path* **then**
      return s^* true ;
return s^* false ;

7.4 General graphs with a 3-canonical g-skeleton

[• **paola:** manca la questione dei caratteri attivi •]

In this section we analyze the choice of the safe source in the case of graphs with a 3-canonical g-skeleton.

Lemma 49. *Let G_{RB} be a reducible graph such that it has a 3-canonical g-skeleton G_M. Assume that $\{A, B, D\}$ is the set of maximal characters of graph G_M. Let s_1, s_2 be the two sources of G_M where s_1 has maximal character B and s_2 maximal character D. Then the safe source is s_1 if and only if for every minimal character e the following two conditions do not hold:*

1. *e is disjoint from character A and e is not in the species with only maximal character D and maximal characters $\{B, D\}$,*
2. *e is conflicting with A and species whose set of maximal characters is $\{A, D\}$ and $\{B, D\}$ respectively, do not have character e and —-if species whose set of maximal character is $\{A, B\}$ has the character e, then there exists a species with only maximal character B having e.—*

Proof. Assume to the contrary that s_1 is the source and the condition 2) holds; then we show that we get a contradiction. The following cases must be considered.

Case 1. Assume that e occurs along the initial-path of the skeleton tree. Since species whose set of maximal characters is $\{A, D\}$ and $\{B, D\}$ respectively, do not have character e, then it must be that e occurs below the edge labeled A. Since e is conflicting with A the following two cases are possible: i) either e is negated after the edge labeled A^- or ii) e is negated before the edge D^- below the branch-node. If case i) and case ii) hold it means that there exists a species with only maximal characters $\{A, D\}$ which has character e, a contradiction. Thus this case is not possible.

Case 2. Assume that e occurs below the branch-node of the skeleton tree. Again since it must be that species whose set of maximal characters is $\{A, D\}$ do not have character e, e must occur either in the branch having D^- or it must occur below A^- and in both cases e is not conflicting with A, which is not possible by condition 2). Thus even this case is not possible.

Case 1 and 2 prove what required, that is if s_1 is a source then condition 2) does not hold. Assume to the contrary that s_1 is the source and the condition 1) holds; then we show that we obtain a contradiction.

Observe that it is not possible with source s_1 having both species with maximal characters $\{B, D\}$ and $\{D\}$ with e since they are on opposite sides of the same path and e is not universal. Moreover, if e is disjoint from A and it is not in species with $\{B.D\}$ it means that it must be negated before the occurrence of D which would imply that e is not in overlap with any maximal character nor it is in overlap with minimal characters that overlap maximal ones. In other words, e is not in the graph in minimal form. Thus condition 1) is not possible.

Vice versa, assume now that the stated conditions of the Lemma do not hold. Then we show that s_1 is a safe source.

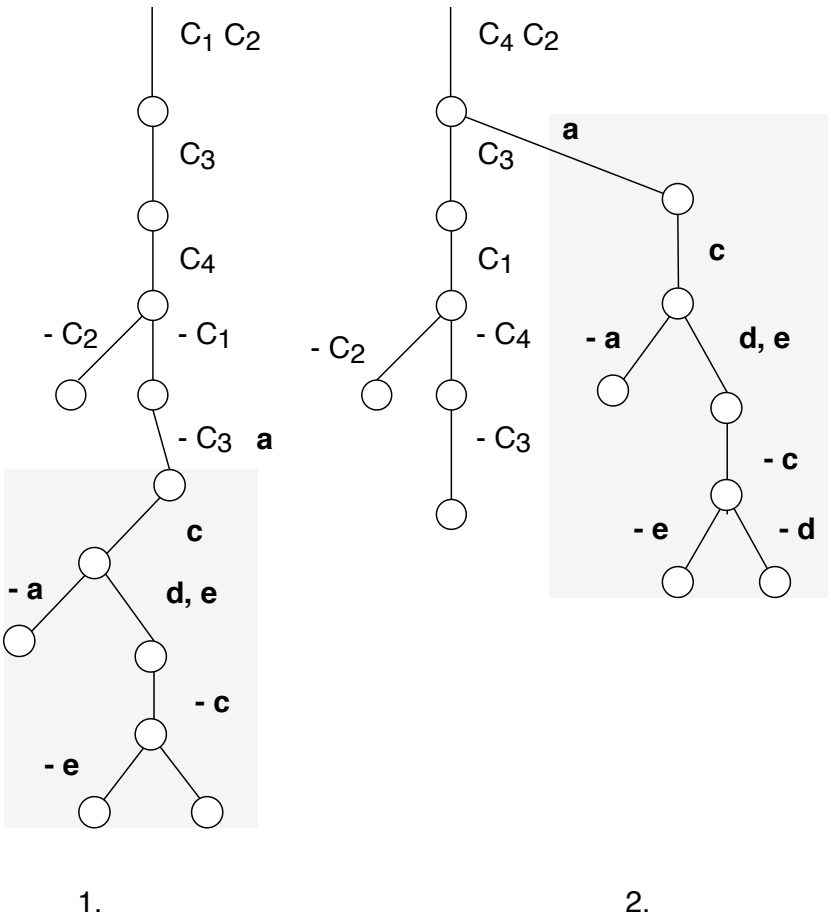

Figure 11: Figure 1. and 2. represents two trees solving the same graph G. The small letters are characters that are minimal in the graph and they induce the grey substree representing pending characters of the graph G. Such a grey subtree that may solved separately.

Assume first that e is disjoint from A or $e < A$ and e is not in the species having the only maximal character $\{D\}$. Indeed, it may be in the species with $\{D, B\}$, violating condition 1).

Case 1. If e occurs along the initial-path it must be that it is negated before A occurs. This clearly implies that the same species are present in the tree where we switch D with B, showing that also s_1 can be a safe source.

Case 2. Assume that e occurs below the split node. Similarly as above we can show that the

same species are present in the tree where we switch D with B, showing that also s_1 is a safe source.

Assume now that e is conflicting with A and the safe source is s_2. Now e must occur before the edge labeled A (indeed e is not maximal) or otherwise if it occurs below the split node, it is easy to observe that condition 2) holds, i.e. e is in the species with $\{A, B\}$ and $\{B\}$. By inspection of the different cases it holds that the same species are present in the tree where we switch D with B, showing that also s_1 can be a safe source.

□

7.4.1 General graphs with line 2-solvable g-skeleton

This case corresponds to the case of general graphs having g-skeleton solved by a line-tree. Observe that given a tree T solving such general graphs, if we consider the subtree of T, called T_M consisting only of the maximal characters of T, then it may be a branch-tree, with a special branch related to the negation of a maximal character soon after the introduction of the character (as detailed in figure ...). Thus this case is very similar to 2-solvable graphs. Indeed, we apply the same conditions stated in Lemma 48. The implementation of the conditions l-source and type-1 change, since the inverted-path consists of all species of the g-skeleton.

—— to be done——-

7.5 General graphs with a m-solvable or degenerate g-skeleton

In this section we consider graphs G_{RB} having degenerate g-skeleton. Let us recall that a degenerate graph is such that has a set $\{c_1, \cdots, c_m\}$ of m-characters and m species s_i where s_i consists of all characters except for a character c_i. Thus a degenerate graph is solved by m possible trees, each having the initial-state consisting of species s_i: for this reason we call degenerate graphs m-canonical.

Let G_{RB} be a m-canonical graph then we will use the following notions.

- *s_i-species of G_{RB}*: a species of G_{RB} having all maximal characters except for c_i,
- *c_i-minimal character of G_{RB}*: a minimal character of G_{RB} that does not overlap any maximal character except [• **paola:** controllare per eventually •] eventually for c_i,
- *closure of a s_i-species s' of G_{RB}*, in short *cl*(s'): the set of c_i-minimal characters d such that $d \in C(s')$ or d overlaps characters already in *cl*(s'),
- **s_i-closure of G_{RB}**: the union of all sets cl(s') for each s_i-species of G_{RB}.

In the following we consider properties of cl(s_i).

Lemma 50. *Let G_{RB} be a m-canonical-graph and let its skeleton tree be i-centered. Then given d a minimal character that occurs along the initial-path of T. Then if d is c_i-minimal, its negation occurs before the branch-node.*

Proof. Now, if d is negated below the branch-node, it must overlap at least a maximal character of the tree T which contradicts the fact that it is c_i-minimal, thus d must be negated along the initial-path. □

The first interesting properties of *cl*(s') is the following.

Lemma 51. *Let G_{RB} be a m-canonical graph and let its skeleton tree be i-centered. Then* cl(s') *for any s_i-species s' occurring along the initial path consists only of characters which are negated along the initial path. Otherwise, if a s_i-species occurs below the branch-node, then* cl(s') *consists only of characters occurring below the branch-node.*

Proof. Let s' be a s_i-species that occurs along the initial-path of tree T. By definition of $cl(s')$ if $d \in C(s')$, then d is c_i-minimal and then by lemma 50 it must be that d occurs negated along the initial-path. Assume that f is c_i-minimal and f is in s', for a s_i-species s' occurring below the branch-node.Now, if f occurs along the initial-path and is then negated below the branch-node it holds that f overlaps a maximal character distinct from c_i and this contradicts the definition of f as c_i-minimal. Now, if e is c_i-minimal and e is in $cl(s')$ but not in s', then if e occurs along the initial-path in virtue of Lemma 50 it must be negated along the initial-path, which implies that it cannot overlap characters already in the closure of s' and thus it cannot be in the closure of s'. Consequently the closure of s' consists of characters that occur below the branch-node. □

Lemma 52. *Let G_{RB} be a m-canonical-graph which is reduced with skeleton a tree T. Two s_i-species s' and s'' have disjoint closures if and only if one of species s' and s'' occurs along the initial path while the other species occurs below the branch-node.*

Proof. Consider s occurring before s' along the initial-path which are both s_i-species, it holds that there exists a minimal a in the closure of s such that a overlaps character c_i. Since both s, s' occurs before c_i, it holds that a is also in closure of s', proving that they cannot have disjoint closures. The same argument applies in the case of s_i-species occuring below the branch-node. This fact concludes the proof of the Lemma.

□

Observe that in a m-canonical tree T, there are k paths starting from the root and ending in a leaf without character c_i: each path consists only of s_i-species or species having all maximal characters.

We consider subgraphs induced by such paths as defined below.

- **the s_i-species path**, denoted s-path(s_i), consists of the subgraph of G_{RB} restricted only to s_i-species or species having all the maximal characters, where only minimal characters in the s_i-closure of G_{RB} are considered in the graph.

Definition 53. Let G_{RB} be a m-canonical-graph. Then species s_i is *unavoidable* if at least one of the two conditions hold:

1. the s-path(s_i) is not a line-tree and s_i is in the graph G_{RB}, or
2. there are two s_i-species of the t-skeleton named s' and s'' such that they have disjoint closures.

   [• **paola:** questa è la vera differenza •]

In the following we state the main characterization used to test **k-centered property** in polynomial time:

- *the skeleton is k-centered if s_k is unavoidable or none of the other species s_i is unavoidable.*

As a consequence of the fact that by definition, the unavoidable property can be tested in polynomial time, combining the following Lemmas 56 and 55 the choice of the source of the skeleton for a m-canonical tree is done in polynomial time, as required.

— da rivedere rispetto alla condizione che s_i is in the graph G_{RB}—-

Lemma 54. *Let G_{RB} be a m-canonical graph. If the s-path(s_i) is not a line-tree and s_i is in the graph G_{RB}, then it must be that s_i occurs along the initial-path of a tree solving G_{RB}.*

Proof. Assume to the contrary that s_i occurs below the branch-node. Since the s-path(s_i) is not a line-tree, it must be that there exists a minimal character that occur below the branch-node that is in overlap with c_i and any s_i-species below the branch-node have at least a minimal character, which implies that s_i occurs along the initial-path of a tree solving G_{RB}, a contradiction. Let us recall that the graph is in minimal form which implies that every minimal character is either in overlap with c_i or otherwise it is in overlap with minimal characters already in overlap with c_i [• **paola:** la forma minimal già implica la nozione di chiusura? •]. □

Lemma 55. *Let G_{RB} be a m-canonical graph. If s_k is unavoidable, then the skeleton must be k-centered.*

Proof. Assume that condition 1 holds for species s_k, that is s-path(s_k) is not a line-tree and s_k is in the graph G_{RB}. It follows that s_k occurs along the initial-path as a consequence of Lemma 54 and thus the skeleton must be k-centered. Now assume that condition 2 holds, i.e. there are two s_k-species s' and s'' of the t-skeleton such that they have disjoint closures in G_{RB}. Then by lemma 52 we must have s_k-species above and below the branch-node proving that the skeleton must be k-centered. □

Lemma 56. *Let G_{RB} be a m-canonical graph and let T be its skeleton tree. Then T is k-centered if no species s_i with i distinct from k is unavoidable.*

Proof. In the following we show that if the tree is necessarily i-centered, then s_i is unavoidable. Thus let the tree be only i-centered. Assume first that there are at least two s_i-species s, s' with not empty closure above and below the branch-node. By Lemma 52, it is immediate that s, s' must occur above or below the branch-node, but not below and above, otherwise the condition 2 for unavoidable holds. Thus assume first that s, s' occurs above the branch-node and no extension is below the branch-node. It is immediate to move the extensions s, s' below the branch-node, since the s-path(s_i) must be a line-tree and can be inverted. Indeed, if the s-path(s_i) is not a line-tree, it must be that s, s' occur only below the branch-node, in virtue of Lemma 52 and condition 1 of unavoidable. But this fact implies that s_i must occur in G_{RB} if the s-path(s_i) is not a line-tree, as in this case there are extensions only below the branch-node. Then condition 1 of unavoidable holds, again a contradiction. □

References

[1] P. Bonizzoni, C. Braghin, R. Dondi, and G. Trucco. The binary perfect phylogeny with persistent characters. *Theoretical computer science*, 454:51–63, 2012.

[2] P. Bonizzoni, A. P. Carrieri, G. Della Vedova, R. Dondi, and T. M. Przytycka. When and How the Perfect Phylogeny Model Explains Evolution. In N. Jonoska and M. Saito, editors, *Discrete and Topological Models in Molecular Biology*, Natural Computing Series, pages 67–83. Springer Berlin Heidelberg, Berlin, Germany, 2014.

[3] P. Bonizzoni, A. P. Carrieri, G. Della Vedova, R. Rizzi, and G. Trucco. A colored graph approach to perfect phylogeny with persistent characters. *Theoretical Computer Science*, pages –, 2016.

[4] P. Bonizzoni, A. P. Carrieri, G. Della Vedova, and G. Trucco. Explaining evolution via constrained persistent perfect phylogeny. *BMC Genomics*, 15(Suppl 6):S10, Oct. 2014.

[5] P. Bonizzoni, S. Ciccolella, G. Della Vedova, and M. Soto. Does relaxing the infinite sites assumption give better tumor phylogenies? an ilp-based comparative approach. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(5):1410–1423, 2018.

[6] J. H. Camin and R. R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, 19(3):pp. 311–326, 1965.

[7] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, third edition, 2005.

[8] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, pages 19–28, 1991.

[9] D. Gusfield. Persistent phylogeny: a galled-tree and integer linear programming approach. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 443–451. ACM, 2015.

[10] J. Kuipers, K. Jahn, B. J. Raphael, and N. Beerenwinkel. Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. *Genome Research*, 2017.

[11] A. McPherson, A. Roth, E. Laks, T. Masud, A. Bashashati, A. W. Zhang, G. Ha, J. Biele, D. Yap, A. Wan, L. M. Prentice, J. Khattra, M. A. Smith, C. B. Nielsen, S. C. Mullaly, S. Kalloger, A. Karnezis, K. Shumansky, C. Siu, J. Rosner, H. L. Chan, J. Ho, N. Melnyk, J. Senz, W. Yang, R. Moore, A. J. Mungall, M. A. Marra, A. Bouchard-Cote, C. B. Gilks, D. G. Huntsman, J. N. McAlpine, S. Aparicio, and S. P. Shah. Divergent modes of clonal spread and intraperitoneal mixing in high-grade serous ovarian cancer. *Nat Genet*, 48(7):758–767, 2016.

[12] T. Przytycka, G. Davis, N. Song, and D. Durand. Graph theoretical insights into evolution of multidomain proteins. *Journal of computational biology*, 13(2):351–363, 2006.

[13] I. B. Rogozin, Y. I. Wolf, V. N. Babenko, and E. V. Koonin. Dollo parsimony and the reconstruction of genome evolution. In I. V. A. Albert, editor, *Parsimony, Phylogeny, and Genomics*. Oxford University Press, 2006.

[14] R. Sedgewick. *Algorithms in C Parts 1-4: Fundamentals Data Structures Sorting Searching*. Algorithms in C. Addison-Wesley, 2001.

[15] J. Zheng, I. B. Rogozin, E. V. Koonin, and T. M. Przytycka. Support for the Coelomata Clade of Animals from a Rigorous Analysis of the Pattern of Intron Conservation. *Mol. Biol. Evol.*, 24(11):2583–2592, 2007.

8 Appendix

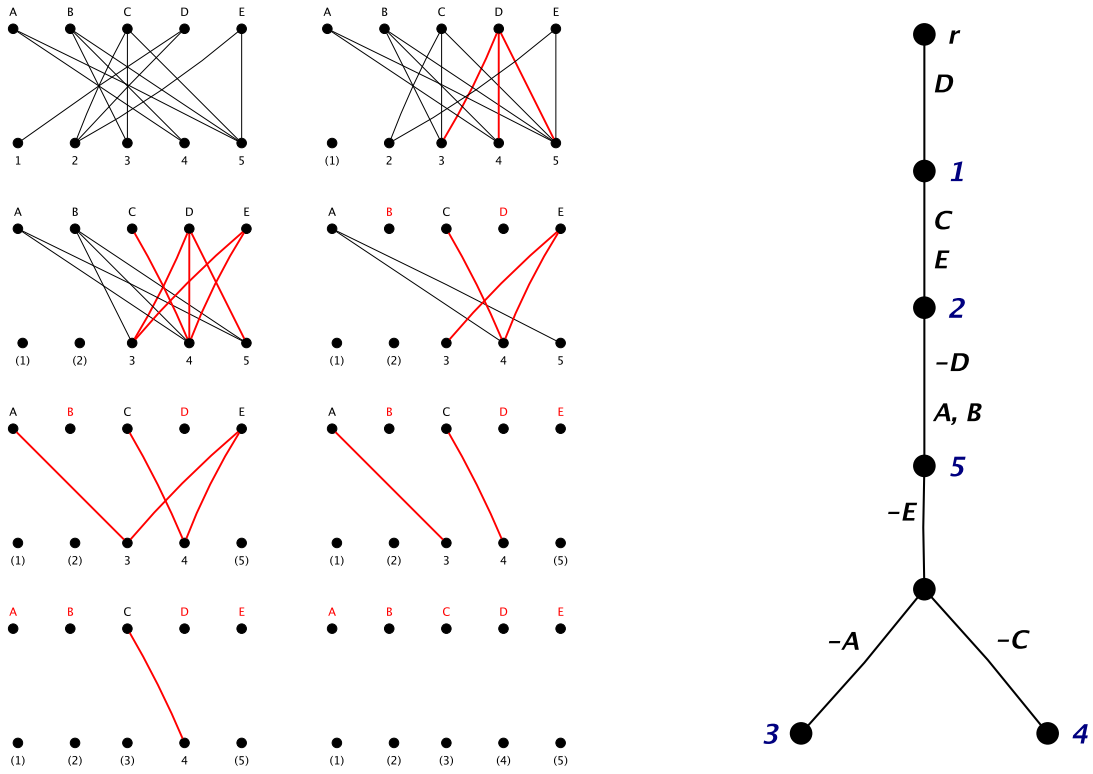

Figure 12: On the left the reduction steps of a red-black graph and on the right the traversal of a tree built from the reduction on the left.