Académie de Montpellier
Université Montpellier II
Sciences et Techniques du Languedoc

MÉMOIRE DE STAGE DE

MASTER 2

effectué au CIRAD
UMR AMAP

Spécialité : **Professionnelle et Recherche unifiée en Informatique**

DÉTECTION DE CO-ÉVOLUTION DE GÈNES

par **Coralie GALLIEN**

Date de soutenance : **12 Septembre 2011**

Sous la direction de

Sèverine BÉRARD et **Éric TANNIER**

Madame	*Annie Chateau*	Rapporteur
Monsieur	*Miklós Molnár*	Rapporteur
Monsieur	*Rodolphe Giroudeau*	Examinateur

Sommaire

Table des figures

Liste des algorithmes

Remerciements

Je tiens à remercier particulièrement Sèverine Bérard et Éric Tannier qui m'ont co-encadrée pendant toute la durée de mon stage. Merci de vous être tant impliqués dans la réalisation de ce projet, de m'avoir soutenue tout au long de ce stage, et de m'avoir permis de vivre une réelle expérience de travail en équipe.

Je remercie également Brigitte Meyer-Berthaud et Mathieu Millan mes collègues de bureau pour leurs conseils avisés et le partage de leurs expériences.

Merci à Fabien Chevalot et à mes parents pour les relectures et corrections de ce mémoire.

Résumé

Ce stage de recherche s'inscrit dans la thématique de la bio-informatique, et plus précisément la phylogénie. Son objectif est, à partir de données sur des arbres phylogénétiques, d'inférer l'histoire évolutive d'adjacences entre gènes.

Pour y parvenir, nous avons utilisé le principe de programmation dynamique et nous nous sommes inspirés de l'algorithme de Fitch ([Fitch, 1971]).

Notre algorithme prend en entrée 2 arbres de gènes, un arbre d'espèce et une liste de gènes adjacents. A partir de ces données, l'algorithme réconcilie les 2 arbres de gènes avec l'arbre des espèces, sélectionne les adjacences dont les gènes qui la compose appartiennent chacun à un arbre de gènes différent, puis calcule l'histoire évolutive complète de coût minimum. Le coût de l'histoire est calculé en fonction des coûts des événements évolutifs mis en jeu.

Abstract

The research stage is about computational biology and more precisely about phylogeny. The goal is to infer the evolutionary history of adjacencies between genes from data of phylogenetics tree.

To achieve this goal, we have used the principle of dynamic programming and we were inspired by the Fitch algorithm.

The data of our algorithm are 2 genes trees, a species tree and a list of adjacent genes. From this data, the algorithm reconciles both of genes trees with the species tree, it selects adjacencies whith genes in eachgenes tree, and then it computes a complete evolutionary history with a minimum cost. The cost of this history is computed from the cost of each events that composes it.

Introduction

Au sein de la communauté scientifique, de nombreuses recherches sur l'**évolution** ont été menées afin de tenter d'expliquer l'histoire des espèces. De nos jours, on dispose d'informations sur le génome d'un nombre croissant d'organismes vivants. Un des problèmes majeurs de la bio-informatique est de retrouver, de façon cohérente, l'histoire évolutive des espèces actuelles et anciennes.

La bio-informatique trouve son utilité dans de nombreuses applications, notamment en médecine. En effet, grâce à cette science, il est possible de comparer à l'échelle de leur ADN (Acide DésoxyriboNucléique) 2 organismes vivants. Par exemple, si on compare l'ADN d'une bactérie pathogène à celui d'une proche cousine non pathogène, on peut alors essayer de repérer les gènes[1] impliqués dans la virulence de la souche infectieuse ([Dardel et Képès, 2002]).

Parmi les nombreuses méthodes de reconstruction de l'histoire évolutive, certaines d'entre elles se basent sur l'étude de la similarité entre gènes. Ces premières méthodes utilisent des distances entre gènes, basées le plus souvent sur l'alignement des séquences de ces gènes, pour inférer des arbres de gènes à partir desquels il est possible de reconstruire l'arbre des espèces. D'autres méthodes, à une échelle de temps moins fine, se basent sur la cartographie des gènes, c'est-à-dire leur ordre sur les chromosomes. Ces dernières vont inférer les "grands" réarrangements génomiques des espèces plus anciennes, comme par exemple les duplications entières de génomes.

Le sujet de ce stage vise à combiner ces 2 classes de méthodes en s'intéressant à l'histoire évolutive des adjacences de gènes. On dit que 2 gènes sont adjacents s'ils sont directement voisins sur le chromosome. Une telle approche, utilisant 2 types de données différentes (la similarité entre gènes et la proximité de ces gènes dans le génome) a pour but de tester la fiabilité des arbres de gènes reconstruits, présents dans les bases de données biologiques mais aussi de reconstituer des génomes ancestraux pour comprendre comment les génomes évoluent ([Bertrand *et al.*, 2010] et [Chauve *et al.*, 2010]).

Le but de ce stage est de retrouver l'histoire évolutive, de coût minimum, des adjacences en partant de données sur les adjacences de gènes et sur les arbres de gènes. Un des premiers points à traiter est de formaliser clairement les données et les résultats que l'algorithme doit gérer.

Après avoir présenté les contraintes et les limites imposées par la biologie, nous aborderons la problématique de ce stage de façon plus générique. Nous emprunterons des notions appartenant à la théorie des graphes, comme par exemple les arbres, qui nous permettront de modéliser les données. Le problème est résolu par un algorithme de complexité polynomiale basé sur un principe de programmation dynamique.

Ce mémoire s'articule en 5 parties principales. La première est un état des lieux, elle correspond à la présentation du lieu de travail et à l'organisation de celui-ci au cours de ces 5 mois. La deuxième permet de décrire le cadre biologique dans lequel nous allons travailler. C'est dans la troisième partie que nous modéliserons précisément le problème et que nous mettrons en évidence certaines propriétés nécessaires à sa résolution. L'algorithme ainsi que sa preuve seront exposés dans la quatrième partie. Et enfin, la cinquième partie permettra de montrer l'application de cet algorithme sur des données réelles.

[1] Un gène est une portion d'ADN, il est composé d'une séquence de nucléotides (Adénine, Thymine, Guanine, Cytosine) et sert à coder une protéine.

Chapitre 1

Présentation du contexte

La formation de Master 2 IFPRU (Informatique à Finalité Professionnelle et Recherche Unifié) de l'université Montpellier 2 propose à ses étudiants de compléter leur cursus théorique par un stage en entreprise ou en laboratoire de recherche. C'est dans ce cadre que s'inscrit ce stage de recherche. Il a été réalisé à l'UMR AMAP du CIRAD du 1^{er} janvier au 30 septembre 2011.

1.1 Le CIRAD, un centre de recherche national

Le **C**entre de **C**oopération **I**nternationale en **R**echerche **A**gronomique pour le **D**éveloppement (CIRAD) est un établissement public français. Il a pour mission de contribuer au développement économique et social des pays tropicaux et subtropicaux par : des recherches, des réalisations expérimentales, des actions de formation en France et à l'étranger, une information scientifique et technique, principalement dans les secteurs agricole, forestier et agro-alimentaire.

FIG. 1.1 – Logo du CIRAD

Le CIRAD est né en 1984 de la fusion d'institutions de recherches en sciences agronomiques, vétérinaires, forestières et agro-alimentaires des régions chaudes. Ses champs de recherche couvrent les sciences du vivant et les sciences sociales. Ils s'appliquent à l'agriculture, à la forêt, à l'élevage, à la gestion des ressources naturelles des pays du sud, à l'agro-alimentaire, aux écosystèmes et aux sociétés. Ils concernent les filières de production et les territoires. Ils s'appuient sur de grandes disciplines telles que l'économie, l'agronomie, l'amélioration des plantes, la protection des cultures, la modélisation ou la technologie agro-alimentaire.

1.2 UMR AMAP

FIG. 1.2 – Logo d'AMAP

L'**U**nité **M**ixte de **R**echerche « bot**A**nique et bio-infor**M**atique de l'**A**rchitecture des **P**lantes » (UMR AMAP) est une jeune structure de recherche, principalement orientée vers l'étude de la morphologie des plantes et de la structure du paysage, à l'aide d'outils informatiques et mathématiques. Au sein de cette UMR se côtoient de nombreuses disciplines (botanique, écologie, agronomie, mathématiques, informatique) mais également des chercheurs d'horizons variés. Ainsi, 5 organismes se partagent la tutelle d'AMAP : le CIRAD, le CNRS[1], l'INRA[2], l'IRD[3] et l'UM2[4]. La diversité de ces acteurs reflète la variété des thématiques abordées par l'UMR : de la recherche fondamentale à la recherche appliquée, les travaux d'AMAP concernent aussi bien les régions tempérées que tropicales.

1.3 Organisation du travail

Ce travail de recherche a été découpé en 7 phases :

1. appropriation du sujet avec recherche sur les différentes notions biologiques abordées, sur l'algorithme de Fitch et bibliographie
2. recherche et élaboration d'un algorithme
3. test de l'algorithme sur des exemples construits
4. complexité et preuve de l'algorithme
5. codage de l'algorithme
6. rédaction du rapport
7. préparation de l'oral
8. test de l'algorithme sur des données réelles

Diagramme de GANTT :

	janvier					février				mars		juillet			août					septembre			
	3	10	17	24	31	7	14	21	28	7	14	11	18	25	1	8	15	22	29	5	12	19	26
1																							
2																							
3																							
4																							
5																							
6																							
7																							
8																							

[1]CNRS : Centre National de Recherche Scientifique
[2]INRA : Institut National de de recherche Agronomique
[3]IRD : Institut de Recherche pour le Développement
[4]UM2 : Université Montpellier 2

Chapitre 2

Cadre biologique

2.1 Introduction

Depuis des siècles, la taxonomie[1] intéresse les hommes essentiellement pour des raisons utilitaires (plantes médicinales, élevage d'animaux, chasse, pêche...). Au fil des années les méthodes utilisées pour classer les espèces les unes par rapport aux autres se sont multipliées. Parmi ces diverses méthodes, on retrouve la **phylogénie**, qui aujourd'hui utilise largement la bio-informatique.

La **phylogénie** est l'étude de la parenté entre différents êtres vivants visant à comprendre leur évolution. On représente couramment une phylogénie par un arbre phylogénétique. Le nombre de nœuds entre les 2 nœuds des espèces étudiées représente autant d'ancêtres et il indique le degré de parenté entre ces espèces. Plus il y a de nœuds et donc d'ancêtres entre 2 espèces, plus leur parenté est éloignée, c'est-à-dire que leur plus proche ancêtre commun est ancien ([wiki/Phylogénie]).

Une **famille de gènes** est un ensemble de gènes unis par des liens de parenté. Ces gènes peuvent subir au cours du temps des événements mutationnels qui vont affecter leur structure et permettre à l'espèce qui les porte d'évoluer. Ces événements peuvent être :

- une perte (*cf.* 2.3.5) : une partie du gène ou le gène entier disparaît du génome de l'espèce
- une duplication (*cf.* 2.3.3) : le gène va se copier dans le génome de l'espèce
- une spéciation (*cf.* 2.3.2) : le gène va suivre l'évolution de l'espèce à laquelle il appartient, lorsque l'espèce va subir une spéciation, c'est-à-dire qu'elle va laisser sa place à ses 2 fils, une copie du gène sera alors présente dans chacun des fils. On parle alors de spéciation au niveau du gène.

Une **adjacence** correspond au fait que 2 gènes ne sont séparés par aucun autre gène sur un même chromosome et sont à une distance inférieure à une valeur fixée. Plusieurs chercheurs se sont penchés sur un même problème : celui de l'explication d'adjacences actuelles. Certains d'entre eux ont produit des algorithmes permettant d'expliquer ces adjacences mais sans tenir compte des duplications et des pertes de gène qui pouvaient avoir eu lieu ([Ma *et al.*, 2006]). Des améliorations ont été apportées par la suite à ce modèle pour lui permettre de prendre en compte les duplications mais sans modéliser explicitement l'évolution d'une adjacence ([Ma *et al.*, 2008]).

Afin d'illustrer le sujet de ce stage, prenons un exemple : la présence des gènes de l'insuline et de la kératine 8 chez de nombreuses espèces. Lors de l'évolution des différentes espèces, il a pu y avoir une adjacence entre 2 gènes de l'insuline issus d'une duplication par exemple. Ceci est un premier cas de création d'adjacence. Si, à un moment donné, les gènes de l'insuline et de la kératine 8 se trouvent côte à côte dans le génome d'une espèce (suite à un événement mutationnel) on aura alors un second cas de création d'adjacence.

Le but de ce stage (décrit plus précisément dans la section 3.1) est, à partir des adjacences actuelles de gènes et des arbres phylogénétiques des familles de gènes, de retrouver l'histoire évolutive de coût minimum de ces adjacences depuis leur(s) création(s).

[1] science des lois de la classification.

Tout d'abord, nous définirons ce qu'est un arbre phylogénétique ainsi que tous les concepts que nous utiliserons et qui y sont liés. Nous décrirons ensuite tous les événements évolutifs mis en jeu. Et enfin nous aborderons la notion de transmission d'adjacence, en fonction des événements sur les gènes impliqués dans l'adjacence.

2.2 Arbres phylogénétiques

Avant d'introduire la notion d'arbre phylogénétique, donnons quelques définitions concernant les gènes :

Définition 1 *Un **gène** est défini comme étant une séquence d'ADN. Il est situé à un endroit bien précis du chromosome le **locus**.*
*On parlera de **gènes ancestraux** lorsque l'on fera référence au génome d'une espèce ancestrale. Un gène appartient à une espèce. On le note d'une lettre (représentant l'espèce) indicée par un nombre (qui permet de différencier tous les gènes d'une même espèce). Par exemple, le gène 3 de l'espèce D se note D_3.*

Définition 2 *Soient un gène A_1 et un gène B_1 tels que A_1 et B_1 descendent d'un ancêtre commun C_1. On dit alors que les gènes A_1 et B_1 sont **homologues**.*

Définition 3 *Soit G un ensemble de gènes homologues. On dit que G est une **famille génique**.*

Définition 4 *La définition d'un **arbre phylogénétique** est similaire à celle d'un arbre de la théorie des graphes. C'est un graphe connexe non cyclique, il est orienté car toutes les arêtes partent d'un sommet "ancien" vers un sommet plus "récent". Le plus ancien de tous les sommets est la racine.*

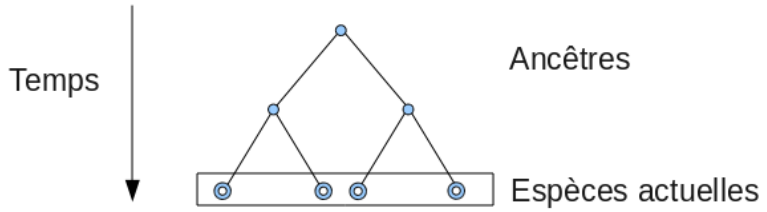

FIG. 2.1 – Représentation du temps sur un arbre phylogénétique

Par commodité, on ne représentera pas dans ce mémoire les flèches des arcs car elles sont toujours dans le même sens : de la racine vers les feuilles. Chaque sommet a un degré entrant de 1 (pour tous les nœuds sauf la racine) ou de 0 (pour la racine) et un degré sortant de 0 (pour une feuille) ou de 1 ou 2 (pour un nœud interne selon la nature du nœud). Pour nos besoins, nous allons préciser que les arbres phylogénétiques utilisés sont tous enracinés. Cependant il peut exister des arbres phylogénétiques non enracinés, c'est-à-dire sans racine, mais nous ne les étudions pas dans ce mémoire.

Définition 5 *Une **forêt** est un ensemble d'arbres.*

Les nœuds des arbres phylogénétiques peuvent être des gènes (définis ci-dessus), on parle alors d'**arbre de famille de gènes**, des espèces, on parle alors d'**arbre des espèces**, ou des adjacences, on parle alors d'**arbre d'adjacences**.
Les nœuds internes des arbres de famille de gènes sont étiquetés par des gènes et possèdent un type : Spéciation (*cf.* 2.3.2) ou Duplication (*cf.* 2.3.3). Les types des feuilles correspondent à des Gènes Actuels ou à des Pertes de Gène (*cf.* 2.3.5). En effet, un gène peut être amené à disparaître suite à un événement évolutif. Par abus de langage, on ne dira pas arbre de famille de gènes mais arbre de gènes.
Tous les arbres de gènes sont calculés avec un modèle d'évolution par substitution ([Felsenstein, 1996]). La **substitution** est la transformation d'un nucléotide en un autre par une mutation. Les arbres de gènes et d'espèces, sur lesquels nous travaillons, sont des arbres binaires (les degrés sortant des nœuds internes valent exactement 2).

Les arbres de gènes sur lesquels nous travaillons sont réconciliés, lors de la phase de pré-traitement, avec l'arbre des espèces (*cf.* Algorithme **??** en Annexe). Cette étape est nécessaire car ces arbres ne contiennent pas forcément des gènes de toutes les espèces. De plus, elle permet de faire apparaître les nœuds de duplication.

Définition 6 *La **réconciliation** est une fonction qui prend en entrée un arbre des espèces S et un arbre de gènes G. Elle a pour objectif d'inférer les événements évolutifs qui ont conduit à l'arbre de gènes observé. Le principe consiste à associer à chaque nœud de G une étiquette correspondant à un nœud, c'est-à-dire une espèce, de l'arbre S, ainsi qu'un type d'événement : Duplication ou Spéciation pour les nœuds internes, Perte ou Gène Actuel pour les feuilles.*

Le procédé de réconciliation permet de faire apparaître les duplications et les pertes de gène non codés explicitement dans les arbres de gènes. On ne traite pas le cas des transferts de gènes qui sont l'introduction dans le génome d'un gène provenant d'un autre organisme, ou du même organisme, par exemple en plusieurs exemplaires pour renforcer son expression ([Glossaire de l'INRA]).

Le principe de l'algorithme consiste à associer à chaque feuille de G une étiquette correspondant à une feuille de l'arbre S. En remontant vers la racine, on continu d'attribuer les étiquettes de S à G : si les 2 fils du nœud courant ont la même étiquette alors le père aura la même étiquette et il s'agira d'un nœud de duplication (*cf.* 2.3.3) sinon on récupère l'étiquette de S correspondante et il s'agit d'un nœud de spéciation(*cf.* 2.3.2). On va ensuite pouvoir vérifier que toutes les espèces sont bien représentées dans G ou si ce n'est pas le cas que leur absence soit modélisée par une perte (*cf.* 2.3.5).

La réconciliation a déjà été implémenté par plusieurs personnes, ici, on se sert de RAPgreen, ,développé en JAVA par un chercheur du CIRAD, Jean-François Dufayard. Le code est disponible à cette adresse : http ://code.google.com/p/rap-green/.

2.3 Événements évolutifs

Plusieurs événements permettent aux espèces d'évoluer, on les retrouve :
- au niveau des espèces :
  - spéciation (*cf.* 2.3.2)
- au niveau des gènes :
  - spéciation (*cf.* 2.3.2)
  - duplication de gène (*cf.* 2.3.3)
  - perte de gène (*cf.* 2.3.5)
- au niveau des adjacences :
  - création d'adjacence (*cf.* 2.3.1)
  - spéciation (*cf.* 2.3.2)
  - duplication de gène (*cf.* 2.3.3)
  - duplication d'adjacence (*cf.* 2.3.4)
  - perte de gène (*cf.* 2.3.5)
  - perte d'adjacence (*cf.* 2.3.6)
  - cassure d'adjacence (*cf.* 2.3.7)

Comme les gènes, les adjacences évoluent. Elles peuvent être affectées par des duplications ou pertes de gène, par des spéciations, ainsi que par des événements qui leur sont propres, duplications, pertes, créations ou cassures d'adjacence. On parle d'**événements évolutifs**.

Les adjacences étudiées sont considérées indépendamment les unes des autres. Ainsi, les événements ne seront pas les réarrangements eux-mêmes (inversion ou translocation) mais leurs conséquences sur les adjacences (création ou cassure). De nombreux événements sont à l'origine de la diversité des répartitions des adjacences entre gènes d'une même espèce. Dans les exemples que nous exposons dans ce mémoire, par souci de simplicité, tous les coûts des événements valent 1 sauf la spéciation qui a un coût de 0, mais dans notre modèle, les coûts des événements sont paramétrables. Ces coûts doivent cependant respecter certaines règles décrites ci-dessous. Voici la description des 7 événements que nous considérons :

2.3.1 Création d'adjacence

La **création d'adjacence** correspond à l'apparition d'une adjacence qui va expliquer une histoire évolutive possible. Son coût est $Cr \geq 0$.
Représentation :

FIG. 2.2 – Représentation graphique d'une création d'adjacence

2.3.2 Spéciation

La **spéciation** est le processus évolutif par lequel de nouvelles espèces apparaissent. La spéciation est à l'origine de la diversité biologique et constitue donc le point essentiel de la théorie de l'évolution. La spéciation peut suivre 2 voies : l'**anagénèse** et la **cladogénèse**. L'anagénèse est une accumulation de changements graduels au cours du temps qui transforment une espèce ancestrale en une nouvelle espèce, cette voie modifie les caractéristiques d'une espèce mais ne permet pas d'augmenter le nombre d'espèces. La cladogénèse est la scission d'un patrimoine génétique en au moins 2 patrimoines distincts, ce processus est à l'origine de la diversité biologique car il permet d'augmenter le nombre d'espèces. ([Lecointre et Guyader, 2006]) Ici, tous les nœuds de spéciation ont suivi la voie de la cladogénèse. Son coût est 0.
Représentation :

FIG. 2.3 – Représentation graphique d'une spéciation

2.3.3 Duplication d'un gène

Processus par lequel un gène est copié et transposé à un autre endroit dans le génome, possiblement à côté. Cet événement peut également affecter les adjacences de gènes. Son coût est $D_G \geq 0$.
Représentation :

FIG. 2.4 – Représentation graphique d'une duplication de gène

2.3.4 Duplication d'adjacence

La **duplication d'adjacence** vient de la factorisation de 2 duplications de gène. Son coût est $0 \leq D_A \leq 2 * D_G$. Les duplications d'adjacence et de gène sont des événements identiques mais le premier porte sur une portion d'ADN plus grande d'où son coût plus petit.
Représentation :

FIG. 2.5 – Représentation graphique d'une duplication d'adjacence

2.3.5 Perte d'un gène

Cet événement correspond à la disparition d'un gène qui peut être due à la transformation d'un gène en un pseudo-gène ou à la suppression du gène par réarrangement génomique. Cet événement peut également affecter une adjacence de gènes qui disparaît alors. Son coût est $P_G \geq 0$.
Représentation :

FIG. 2.6 – Représentation graphique d'une perte de gène

2.3.6 Perte d'adjacence

De même que pour la duplication d'adjacence, la **perte d'adjacence** vient de la factorisation de 2 pertes de gène. Son coût est $0 \leq P_A \leq P_G$. De la même manière que pour les duplications, les pertes d'adjacence et de gène sont des événements identiques mais le premier porte sur une portion d'ADN plus grande.
Représentation :

FIG. 2.7 – Représentation graphique d'une perte d'adjacence

2.3.7 Cassure

La **cassure** est une disparition de l'adjacence entre 2 gènes, à cause par exemple d'un réarrangement, les 2 gènes qui étaient adjacents se retrouvent éloignés l'un de l'autre. Son coût est $Ca \geq 0$.
Représentation :

FIG. 2.8 – Représentation graphique d'une cassure

2.4 Règles de transmission d'adjacence

Définition 7 *Nous avons vu qu'une* **adjacence** *de 2 gènes signifie qu'il n'existe pas d'autre gènes entre les 2 gènes considérés et qu'ils se situent à une distance inférieure à une certaine valeur.*
Une adjacence peut se **transmettre**, *par exemple, dans le cas d'une spéciation, si :*
- *2 gènes A_1 et A_2 sont adjacents,*
- *A_1 a pour fils le gène B_1,*
- *A_2 a pour fils le gène B_2,*

alors B_1 et B_2 sont adjacents. Par parallélisme avec les gènes, on peut donc dire que 2 adjacences sont **homologues** *si elles ont un ancêtre commun. On peut classer les adjacences en famille comme on classe les gènes en famille et donc utiliser les arbres phylogénétiques pour représenter ces familles d'adjacences de la même manière que l'on utilise les arbres de gènes.*
L'adjacence entre les gènes A_1 et A_2 se note $A_1 \sim A_2$ ou indifféremment $A_2 \sim A_1$. Si on souhaite exprimer uniquement l'adjacence sans parler des gènes qu'elle implique, on utilisera une lettre grecque pour éviter toute confusion avec les espèces, par exemple $\alpha = A_1 \sim A_2$.

On dit qu'une adjacence est **ancêtre** d'une autre si la seconde est issue de la première par une série de transmissions. Ainsi, 2 adjacences ont une **origine commune** si elles sont homologues, c'est-à-dire, si elles ont un ancêtre commun.

Règles de transmission des adjacences Nous recherchons les relations de parenté entre les adjacences et les événements qui expliquent leur présence ou absence dans les génomes actuels. Il s'agit donc de construire des arbres phylogénétiques d'adjacences.

Définition 8 *Un* ***arbre d'adjacences*** *est un arbre phylogénétique* **associé** *à un ou plusieurs arbres de gènes et à une liste d'adjacences actuelles. Tous les nœuds sont étiquetés par des adjacences entre nœuds des arbres de gènes associés. Et l'arbre respecte les propriétés énoncées dans la propriété 1. Les feuilles peuvent donc être des adjacences actuelles, des pertes de gène ou d'adjacence ou encore des cassures d'adjacence. Les nœuds internes sont soit des duplications de gène ou d'adjacence, soit des spéciations. La racine correspond à la création d'une adjacence, ce nœud racine porte aussi une des étiquettes précédemment évoquées.*
Un arbre d'adjacence peut être vide.

La figure 3.3 p.24 est un exemple d'arbre d'adjacences.

Définition 9 *Une* **forêt d'arbres d'adjacences** *est associée à un ou plusieurs arbres de gènes $\mathcal{G}_i$ et à une liste d'adjacences $\mathcal{L}$. Chacun des arbres de cette forêt est un arbre d'adjacences associé à un ou plusieurs sous-arbres des $\mathcal{G}_i$ et à une liste d'adjacences, sous-liste de $\mathcal{L}$.*
NB : l'union des sous-listes est égale à $\mathcal{L}$, leur intersection est vide.

Les règles de transmission des adjacences sont énoncées dans la propriété suivante.

Propriété 1 *Les nœuds internes d'un arbre d'adjacences $\mathcal{A}$ associé à un ou plusieurs arbres de gènes et à une liste d'adjacence, sont de 4 types différents et respectent les propriétés suivantes :*

1. *Nœud de Spéciation : Soit $A_1 \sim A_2$ un nœud de Spéciation de $\mathcal{A}$, alors A_1 et A_2 doivent être 2 nœuds de Spéciation dans leur arbre de gènes respectifs. Supposons que A_1 et A_2 aient 2 fils, respectivement B_1 et C_1 et B_2 et C_2. Alors les fils de $A_1 \sim A_2$ dans $\mathcal{A}$ sont exactement $B_1 \sim B_2$ et $C_1 \sim C_2$. Ceux-ci peuvent être étiquetés par tous les types d'événement (sauf la Création).*

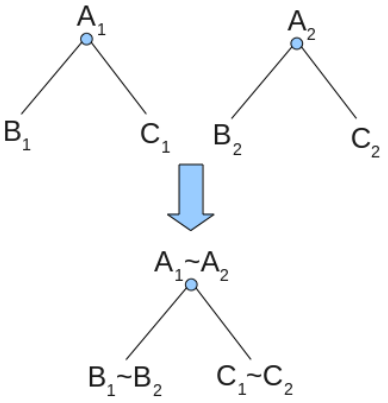

FIG. 2.9 – Nœud de Spéciation dans un arbre d'adjacences

2. *Nœud de Duplication d'Adjacence : Soit $A_1 \sim A_2$ un nœud de Duplication d'Adjacence de $\mathcal{A}$, alors A_1 et A_2 doivent être 2 nœuds de Duplication de Gène dans leur arbre de gènes respectifs. De plus, si A_1 et A_2 ont 2 fils, respectivement A_3 et A_4 et A_5 et A_6, alors $A_1 \sim A_2$ a exactement 2 fils dans $\mathcal{A}$ qui sont soit $A_3 \sim A_5$ et $A_4 \sim A_6$ soit $A_4 \sim A_5$ et $A_3 \sim A_6$. Ceux-ci peuvent être étiquetés par tous les types d'événement (sauf la Création).*

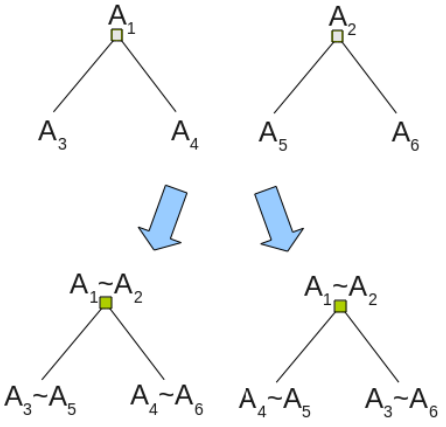

FIG. 2.10 – Nœud de Duplication d'Adjacence dans un arbre d'adjacences

3. *Nœud de Duplication de Gène : Soit $A_1 \sim A_2$ un nœud de Duplication de Gène de $\mathcal{A}$, alors A_1 ou A_2 (ou les 2) doit être un nœud de Duplication de Gène dans son arbre de gènes. Supposons que A_1 soit un*

nœud de Duplication de Gène avec 2 fils A_3 et A_4. Alors $A_1 \sim A_2$ a exactement un fils dans $\mathcal{A}$ qui est soit $A_3 \sim A_2$ soit $A_4 \sim A_2$. Celui-ci peut être étiqueté par tous les types d'événement (sauf la Création).

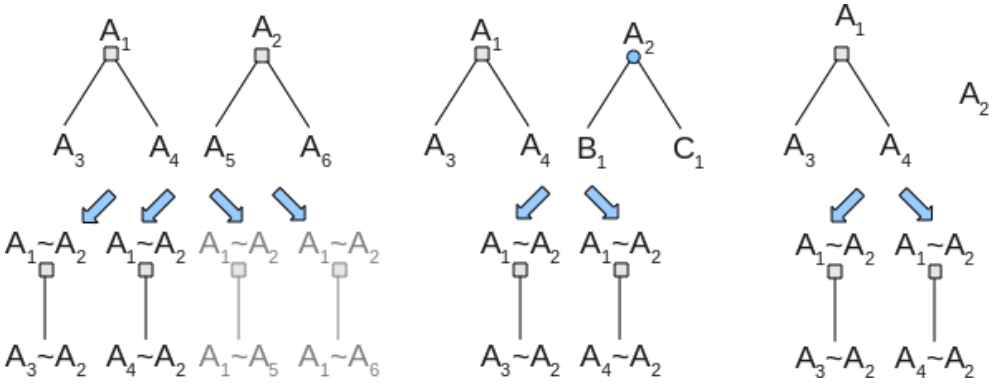

FIG. 2.11 – Nœud de Duplication de Gène dans un arbre d'adjacences

Remarque *Sur cette figure, pour le cas où les nœuds A_1 et A_2 sont de type Duplication, on voit qu'il y a 4 possibilités de transmission d'adjacence. Ceci est dû au fait qu'il y a 2 nœuds de Duplication, et que la transmission d'adjacence est symétrique. On peut donc soit avoir le cas où A_1 se duplique avant A_2 et on a l' adjacence $A_3 \sim A_2$ ou $A_4 \sim A_2$, soit le cas où A_2 se duplique avant A_1 et on a l'adjacence $A_1 \sim A_5$ ou $A_1 \sim A_6$.*

4. *<u>Nœud de création :</u> Soit $A_1 \sim A_2$ la racine de $\mathcal{A}$, A_1 et A_2 peuvent être de n'importe quel type (sauf le couple Spéciation/Gène Actuel qui n'existe pas) dans leur arbre de gènes respectifs. Dans tous les cas, $A_1 \sim A_2$ a une deuxième étiquette en plus de Création d'adjacence.*

NB : Dans aucun des cas énumérés ci-dessus nous ne rencontrons une adjacence entre un nœud de Spéciation et un Gène Actuel. En effet, soit un gène A_1 étiqueté Spéciation de l'espèce A qui a 2 fils : B_1 et C_1 et soit D_1 un Gène Actuel de l'espèce D. Les gènes A_1 et D_1 ne peuvent pas être adjacents car ils sont d'époques différentes. A_1 n'étant pas une feuille, A n'est pas une espèce actuelle alors que D en est une. A et D ne peuvent donc pas être contemporaines. Ce type d'adjacence n'existe donc pas.

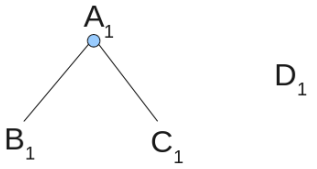

FIG. 2.12 – Cas : GèneActuel/Spéciation : adjacence impossible

Les feuilles de l'arbre d'adjacences $\mathcal{A}$, sont de 4 types différents et respectent les propriétés suivantes :

1. *<u>Adjacence actuelle :</u> Soit $A_1 \sim A_2$ une adjacence actuelle de $\mathcal{A}$, alors A_1 et A_2 doivent être 2 Gènes Actuels dans leur arbre de gènes respectifs.*

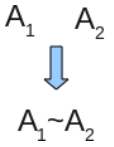

FIG. 2.13 – Adjacence entre 2 Gènes Actuels dans un arbre d'adjacences

2. *Perte d'adjacence : Soit $A_1 \sim A_2$ une Perte d'Adjacence de $\mathcal{A}$, alors A_1 et A_2 doivent être 2 Perte de Gènes dans leur arbre de gènes respectifs.*

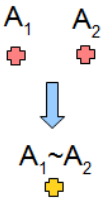

FIG. 2.14 – Feuille/Perte d'Adjacence

3. *Perte de Gène : Soit $A_1 \sim A_2$ une perte de gène de $\mathcal{A}$, alors A_1 ou bien A_2 doit être une Perte de Gène dans son arbre de gènes.*

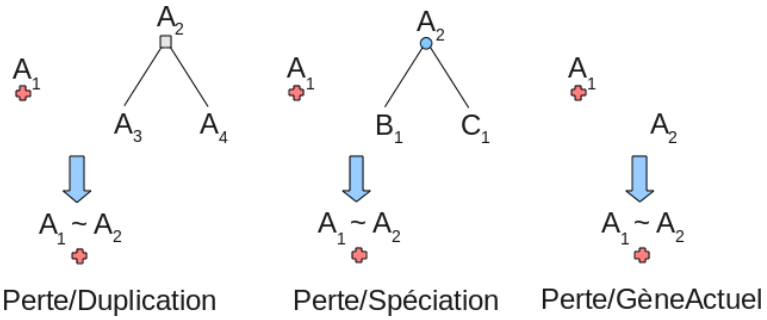

FIG. 2.15 – Feuille/Perte de Gène

4. *Cassure d'adjacence : Soit $A_1 \sim A_2$ une Cassure d'adjacence de $\mathcal{A}$, alors A_1 et A_2 peuvent être de n'importe quel type.*

Chapitre 3

Modélisation et propriétés

Le sujet d'étude est la reconstitution de l'histoire des adjacences entre gènes basée sur l'histoire évolutive de ces gènes et des espèces auxquelles ils appartiennent ainsi que sur les adjacences entre ces gènes. Pour commencer, nous travaillons uniquement avec 2 arbres de gènes. Lors de la résolution de ce problème, on devra tenir compte du fait que la reconstitution de l'histoire des adjacences implique plusieurs événements dont on cherche à minimiser la somme des coûts, on applique donc ici le principe de parcimonie très classique en phylogénie. Dans un premier temps, nous aborderons la problématique et la formalisation du résultat attendu, puis nous donnerons quelques définitions sur les différents coûts utilisés ici. Enfin, nous étudierons les propriétés mises en évidence au cours de ce stage.

Afin d'illustrer les définitions de cette section nous nous baserons sur l'exemple suivant auquel nous ferons régulièrement référence :

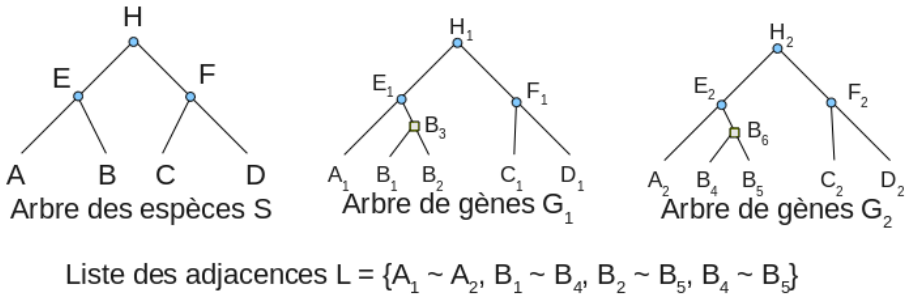

FIG. 3.1 – Exemple de référence

Sur cet exemple, on a un arbre des espèces $\mathcal{S}$ qui a 4 feuilles (ou espèces actuelles) : A, B, C et D et 3 nœuds internes E, F et H. Les nœuds internes sont des nœuds de spéciation. On a également 2 arbres de gènes $\mathcal{G}_1$ et $\mathcal{G}_2$ qui ont chacun 5 feuilles respectivement : A_1, B_1, B_2, C_1, D_1 et A_2, B_4, B_5, C_2, D_2, et des nœuds internes : B_3, E_1, F_1 et H_1 pour l'arbre $\mathcal{G}_1$ et B_6, E_2, F_2 et H_2 pour l'arbre $\mathcal{G}_2$. Les nœuds H_1, E_1, F_1, H_2, E_2 et F_2 sont des nœuds de spéciation alors que les nœuds B_3 et B_6 sont des nœuds de duplication de gène. La liste des adjacences fournies avec ces arbres est la suivante : $A_1 \sim A_2$, $B_1 \sim B_4$, $B_2 \sim B_5$ et $B_4 \sim B_5$.

Représentation des adjacences Au début du projet, une représentation particulière des 2 arbres de gènes et des adjacences s'est imposée : les arbres de gènes sont placés l'un en dessous de l'autre et de sorte que le premier ait la racine en haut, que le second ait la racine en bas et qu'ils aient les feuilles au centre de la représentation permettant ainsi de bien visualiser les adjacences. La figure suivante illustre cette représentation.

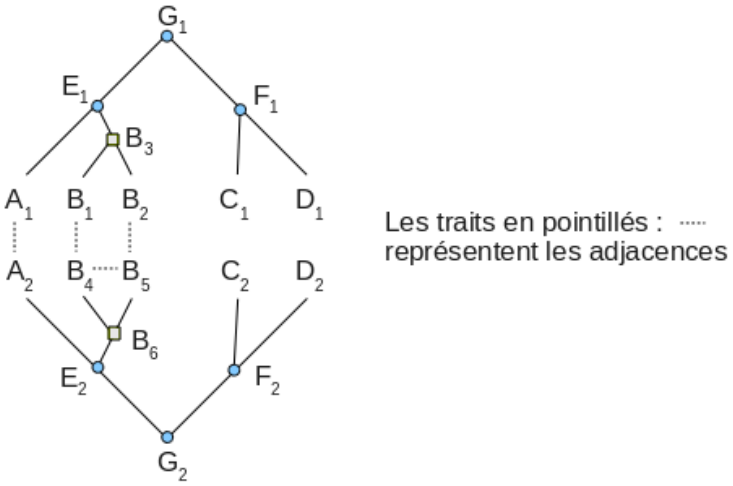

FIG. 3.2 – Représentation des arbres de gènes de l'exemple 3.1 face à face

3.1 Problématique et formalisation du résultat attendu

Limitation Étant limitée par le temps, certains choix ont été fait afin de pouvoir présenter un résultat cohérent à la fin de ce stage. Ainsi, nous nous limitons à seulement 2 arbres de gènes en entrée de l'algorithme et à l'étude des adjacences dont chacun des nœuds qui les composent appartiennent à un arbre de gènes différent. Dans l'exemple de la figure 3.1, ça veut dire qu'on ne traite que les adjacences $A_1 \sim A_2$, $B_1 \sim B_4$, $B_2 \sim B_5$ représentées verticalement dans la figure 3.2. Les 2 arbres de gènes auront la même espèce à leur racine après réconciliation.

Remarque : Les adjacences entre gènes d'un même arbre et entre gènes d'arbre différent ne peuvent pas avoir d'origine commune (*cf.* Propriétés 3 et 4).

Problématique On cherche à construire un algorithme qui prenne en entrée :
- 2 arbres de gènes $\mathcal{G}_1$ et $\mathcal{G}_2$
- une liste d'adjacence $\mathcal{L}$ (des adjacences entre gènes de $\mathcal{G}_1$ et $\mathcal{G}_2$)
- un arbre d'espèces $\mathcal{S}$

et qui calcule une forêt d'arbres d'adjacences, associée à $\mathcal{G}_1$, $\mathcal{G}_2$ et $\mathcal{L}$, de coût différentiel minimum (*cf.* définition 14 p.23).

Formalisation du résultat Un arbre d'adjacence est, comme la définition 8 le décrit page 17, un arbre phylogénétique, dont la racine correspond à la création d'une adjacence, les feuilles sont les Adjacences Actuelles ou des Pertes ou encore des Cassures et les nœuds internes sont l'histoire évolutive de ces adjacences. Chacun des arbres de cette forêt correspond à une histoire évolutive qui expliquera les adjacences présentes dans la liste $\mathcal{L}$ passée en paramètre. La multiplicité d'arbres d'adjacences est due au fait que ces histoires sont indépendantes les unes des autres. C'est à dire que, si pour expliquer les adjacences actuelles, il est nécessaire d'avoir plusieurs Créations d'adjacence alors on aura autant d'arbres d'adjacences. D'ailleurs, un arbre d'adjacences peut être constitué uniquement d'une feuille, cas d'une Création d'adjacence au dernier moment.

Définition 10 *Une* **solution** *est une forêt d'arbres d'adjacences associée à $\mathcal{G}_1$, $\mathcal{G}_2$ et $\mathcal{L}$. C'est-à-dire que les nœuds sont des adjacences de gènes dont l'un provient de $\mathcal{G}_1$, l'autre de $\mathcal{G}_2$. Les feuilles peuvent être des adjacences actuelles (uniquement celles de la liste $\mathcal{L}$), des Pertes d'Adjacence ou de Gène ou des Cassures d'adjacence. Toutes les adjacences de $\mathcal{L}$ sont des feuilles dans cette forêt. Les nœuds internes sont associés à des événements (Spéciation ou Duplication d'Adjacence ou de Gène).*

3.2 Les coûts

Dans cette section, nous définissons les coûts que nous utilisons dans la suite du rapport.

Définition 11 *Le* **coût d'un arbre de gènes** *(resp. le* **coût d'un arbre d'adjacences***) est la somme des coûts de chacun des nœuds de l'arbre : Duplication de Gène et Perte de Gène (resp. Création, Cassure, Duplication et Perte d'Adjacence, Duplication et Perte de Gène, Spéciation).*

Sur l'exemple illustré dans la figure 3.1, l'arbre de gène $\mathcal{G}_1$ a un nœud de Duplication : B_3 ce qui lui fait un coût de $D_G = 1$, l'arbre de gènes $\mathcal{G}_2$ a un nœud de Duplication : B_6 ce qui lui fait également un coût de 1. Sur l'exemple illustré dans la figure 3.3, l'arbre d'adjacence a une Création : Cr et un nœud de Duplication d'Adjacence : D_A ce que fait un coût total de $Cr + D_A = 2$.

Définition 12 *Le* **coût d'une forêt** *est la somme des coûts de tous les arbres de cette forêt.*

L'exemple de la figure 3.1 donne une liste de 4 adjacences. Si on imagine que ces 4 adjacences n'ont pas d'origine commune, on a alors 4 arbres d'adjacences composés chacun d'une seule feuille. Le coût de cette forêt d'arbres d'adjacences est donc de $4 * Cr = 4$.

Définition 13 *Le* ***coût maximum*** *désigne le coût d'une histoire évolutive triviale où toutes les adjacences ont été créées au dernier moment. Il prend en compte les événements sur les gènes et les adjacences. Il correspond à la somme des coûts des 2 arbres de gènes et du coût de la forêt d'arbres d'adjacences composés chacun d'un seul sommet : une adjacence de la liste $\mathcal{L}$.*
coûtMax = coût de l'arbre de gène $\mathcal{G}_1$ + coût de l'arbre de gène $\mathcal{G}_2$ + coût de la forêt d'arbres d'adjacences réduits à une seule adjacence

Sur cet exemple, on a donc un coût maximum de $2 * D_G + 4 * Cr = 6$.
La solution de notre problème est une forêt d'arbres d'adjacences. Or, le coût d'une telle forêt ne tient pas compte des coûts des événements survenus sur les gènes non impliqués dans les adjacences nœuds des arbres solutions. C'est pourquoi nous calculons un coût différentiel, qui lui, prendra en compte le coût de tous les événements sur les gènes et sur les adjacences.

Définition 14 *Le* ***coût différentiel*** *d'un arbre d'adjacences se calcule toujours par rapport au coût maximum. Chaque nœud de l'arbre a un coût différentiel :*

- *Adjacence Actuelle : $-Cr$ car on explique une adjacence, dont le coût était initialement compté dans le coût maximum, par une Création qui a lieu plus haut dans l'arbre*
- *Spéciation : 0 toutes les Spéciations ont déjà été comptées lors du calcul du coût maximum, de plus la Spéciation a un coût nul,*
- *Duplication de Gène : 0 toutes les Duplications de Gène ont déjà été comptées lors du calcul du coût maximum,*
- *Duplication d'Adjacence : $-2 * D_G + D_A$ car elle correspond à la factorisation de 2 Duplications de Gène, on enlève au coût maximum le coût de 2 Duplications de Gène et on lui ajoute le coût d'une Duplication d'Adjacence,*
- *Perte d'un Gène : 0 toutes les Pertes de Gène ont déjà été comptées lors du calcul du coût maximum,*
- *Perte d'Adjacence : $-2 * P_G + P_A$ car elle correspond à la factorisation de 2 Pertes de Gène, on enlève au coût maximum le coût de 2 Pertes de Gène et on lui ajoute le coût d'une Perte d'Adjacence,*
- *Cassure : $+Ca$ car on rajoute cet événement à l'histoire triviale qui donne le coût maximum,*
- *Création d'adjacence : $+Cr$ car on rajoute cet événement à l'histoire triviale qui donne le coût maximum. Le nœud Création d'adjacence est également d'un des types précédemment cités, les 2 coûts de ces nœuds s'ajoutent.*

Le coût différentiel de l'arbre d'adjacences est la somme des coûts différentiels de ses nœuds.

Définition 15 *Le* **coût de la solution** *est la somme des coûts différentiels des arbres d'adjacences qui la composent plus le coût maximum. Le coût de la solution contient donc bien tous les coûts des événements sur les gènes et sur les adjacences.*

Une solution de l'exemple de la figure 3.1 consiste à créer une adjacence entre E_1 et E_2, qui se transmettra entre A_1 et A_2 et entre B_3 et B_6. Ensuite, l'adjacence $B_3 \sim B_6$ se duplique en $B_1 \sim B_4$ et en $B_2 \sim B_5$. On peut représenter cette solution par l'arbre d'adjacences de la figure 3.3 :

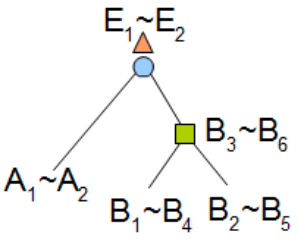

FIG. 3.3 – Arbre d'adjacence solution de l'exemple 3.1

Le coût maximum de l'exemple est 6. Ici, la solution ne contient qu'un seul arbre d'adjacence. Ses nœuds sont :

- une création d'adjacence entre les sommets E_1 et E_2, également nœuds de spéciation : $Cr + 0$
- une duplication d'adjacence : $-2 * D_G + D_A$
- 3 feuilles adjacences actuelles : $-3 * Cr$

Ceci nous amène à un coût de la solution total de :
6 (coût maximum) $+Cr - 2 * D_G + D_A - 3 * Cr = 2$.

3.3 Propriétés mises en évidence

Au cours de la recherche de solution à la problématique, plusieurs propriétés ont été mises en évidence.

Rappel : 2 adjacences peuvent avoir une origine commune s'il existe une solution où elles sont dans le même arbre.

Propriété 2 *Soient A_1 et A_2 de l'espèce A, B_1 et B_2 de l'espèce B, 4 gènes. Deux adjacences $\alpha(A_1 \sim A_2)$ et $\beta(B_1 \sim B_2)$ ne peuvent avoir d'origine commune que si les gènes ($(A_1$ et $B_1)$ et $(A_2$ et $B_2))$ ou $((A_1$ et $B_2)$ et $(A_2$ et $B_1))$ ont une origine commune.*

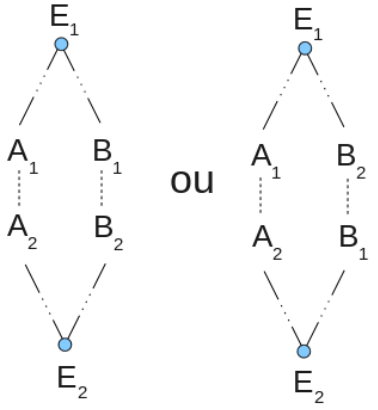

FIG. 3.4 – Représentation des possibilités d'origine commune des gènes

Idée de preuve Si les gènes n'ont pas d'origine commune alors on ne peut pas créer l'adjacence ancestrale.
NB : On ne peut pas créer d'adjacence entre 2 gènes d'espèces différentes.

Corollaire 1 *Soit un arbre d'adjacences dont $\alpha(A_1 \sim A_2)$ et $\beta(B_1 \sim B_2)$ sont des feuilles. Si A_1 et B_1 sont homologues (soit N_1 leur plus proche ancêtre commun) ainsi que A_2 et B_2 (N_2 leur plus proche ancêtre commun) alors $N_1 \sim N_2$ est le dernier ancêtre comme de α et β.*

Propriété 3 *Soient A_1, A_2, A_3 et A_4 4 gènes. A_1, A_2 et $A_3 \in G_1$ et $A_4 \in G_2$. On a les adjacences $\alpha(A_1 \sim A_2)$ et $\beta(A_3 \sim A_4)$. Il n'existe pas d'histoire commune aux adjacences α et β.*

Preuve : A_4 ne partage aucun ancêtre commun avec les sommets A_1 et A_2. Donc les 2 adjacences α et β n'ont donc aucune histoire évolutive commune d'après la propriété 2.

Propriété 4 *Soient A_1, A_2, A_3 et A_4 4 gènes. A_1 et $A_2 \in G_1$ et A_3 et $A_4 \in G_2$. On a les adjacences $\alpha(A_1 \sim A_2)$ et $\beta(A_3 \sim A_4)$. Il n'existe pas d'histoire commune aux adjacences α et β.*

Preuve : A_1 ne partage aucun ancêtre commun avec les sommets A_3 et A_4. Donc, d'après la propriété 2, les 2 adjacences α et β n'ont donc aucune histoire évolutive commune.

Propriété 5 *Soient $\alpha(A_1 \sim A_2)$ et $\beta(B_1 \sim B_2)$ 2 adjacences.*
Supposons que N_1 soit le plus proche ancêtre commun de A_1 et B_1 et que N_2 soit le plus proche ancêtre commun de A_2 et B_2.
Si N_1 et N_2 ne sont pas du même type (spéciation ou duplication), alors α et β n'ont pas d'histoire commune.

Preuve : Supposons que N_1 soit de type spéciation et que N_2 soit de type duplication. N_2 aurait alors 2 fils de la même espèce N_3 et N_4. Supposons aussi que N_3 soit sur le chemin de N_2 à A_2 et N_4 sur celui de N_2 à B_2. D'après les règles de transmission des adjacences, $N_1 \sim N_2$ se transmet soit à $N_1 \sim N_3$ soit à $N_1 \sim N_4$. $N_1 \sim N_3$ peut être ancêtre de $A_1 \sim A_2$ mais pas de $B_1 \sim B_2$ car N_3 n'est pas ancêtre de B_2. Symétriquement, $N_1 \sim N_4$ ne peut être ancêtre de $A_1 \sim A_2$. $A_1 \sim A_2$ et $B_1 \sim B_2$ n'ont donc pas d'ancêtre commun.

Chapitre 4

Algorithme mis en place

Cette partie est construite en suivant le modèle de l'algorithme principal DéCo mis en place : tout d'abord nous étudierons la phase de pré-traitement des données puis nous verrons plus en détail chacun des algorithmes permettant de calculer le coût en fonction du type de nœud étudié, ensuite nous prouverons que ces algorithmes renvoient le coût minimum d'une forêt d'adjacence. Nous donnerons l'algorithme DéCo, sa preuve et sa complexité avant de finir par des tests sur des exemples construits.

4.1 Pré-traitement

Avant de pouvoir appliquer l'algorithme DéCo sur les données passées en paramètre, une phase de pré-traitement est nécessaire. Elle consiste en :

1) Parser le fichier de données Cette étape consiste à construire en mémoire l'arbre des espèces $\mathcal{S}$, les arbres de gènes $\mathcal{G}_1$ et $\mathcal{G}_2$, et la liste des adjacences $\mathcal{L}$ dont les couples de nœuds (N_1, N_2) qui les composent sont tels que $N_1 \in G_1$ et $N_2 \in G_2$ ou $N_1 \in G_2$ et $N_2 \in G_1$.

2) Réconcilier les arbres de gènes avec l'arbre des espèces On va devoir ensuite, pour les 2 arbres de gènes $\mathcal{G}_1$ et $\mathcal{G}_2$, reconstruire les pertes et les duplications en s'appuyant sur l'arbre des espèces $\mathcal{S}$. Il s'agit de la réconciliation des arbres de gènes avec l'arbre des espèces. L'algorithme est donné en annexe : Algorithme **??** p.**??**.

3) Calculer le coût maximum Enfin, on va calculer, à partir du fichier passé en entrée, le coût maximum. Ce coût est important car, c'est à partir de celui-ci que l'on calculera par la suite le coût de la solution. Il correspond, comme il a été dit précédemment, à la somme des coûts des 2 arbres de gènes $\mathcal{G}_1$ et $\mathcal{G}_2$ et du coût de la forêt d'arbres d'adjacences réduits à une seule adjacence : $|L| * Cr$.

4.2 Algorithmes de calcul de coût

Pour toute cette section, on utilisera une fonction $\mathcal{G}_1(N_1)$ (resp. $\mathcal{G}_2(N_2)$) qui renvoie le sous-arbre de $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) dont la racine est $N_1 \in G_1$ (resp. $N_2 \in G_2$). On définit aussi une sous-liste de $\mathcal{L}$ composée des adjacences entre descendants de N_1 et N_2 : $\mathcal{L}(N_1, N_2)$.

Sauf indication contraire, dans la suite de ce mémoire, on considère les nœuds N_1 et N_2 tels que $N_1 \in G_1$ et $N_2 \in G_2$.

Le principe de l'algorithme DéCo utilisé pour résoudre l'histoire évolutive des adjacences est similaire à celui de l'extension de Sankoff de l'algorithme de Fitch ([Fitch, 1971]). L'algorithme de Sankoff ([Sankoff, 1975]) est un algorithme assez classique, largement utilisé en phylogénie. DéCo se base sur la version binaire de cet algorithme (alphabet à deux éléments au lieu de 4, cas traité par [Tang and Wang, 2005]), et la généralise (ici,

on traite le cas des duplications et des pertes).

Cet algorithme repose également sur le principe de l'alignement des arbres puisqu'on construit des relations entre les nœuds des arbres. Cependant, on autorise le fait de ne pas avoir de relations entre certains nœuds, on peut donc parler d'alignement local d'arbres.

On va calculer 2 coûts : le "coût 1" avec la fonction c_1 et le "coût 0" avec la fonction c_0 pour chaque couple de nœuds des arbres de gènes. $c_1(N_1, N_2)$ calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée à $\mathcal{G}_1(N_1)$, $\mathcal{G}_2(N_2)$ et à la liste $\mathcal{L}(N_1, N_2)$, forêt dans laquelle il existe le nœud de création $N_1 \sim N_2$ (exceptionnellement de coût 0 car la création est comptée plus haut). $c_0(N_1, N_2)$ calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée à $\mathcal{G}_1(N_1)$, $\mathcal{G}_2(N_2)$ et à la liste $\mathcal{L}(N_1, N_2)$, forêt dans laquelle il n'existe pas le nœud de création $N_1 \sim N_2$ (sauf si $N_1 \sim N_2 \in L$). Ces fonctions sont récursives et basées sur le principe de la programmation dynamique qui stocke les résultats. On va examiner tous les couples de nœuds entre nœud du premier et du second arbre de gènes. On peut résumer dans un tableau les différents types de couples possibles selon le type des nœuds du couple, les cases de ce tableau indiquent les sections auxquelles se référer pour en obtenir la description :

	GèneActuel	Perte	Duplication	Spéciation
GèneActuel	A : 4.2.1	C : 4.2.3	D : 4.2.4	X
Perte	C : 4.2.3	B : 4.2.2	C : 4.2.3	C : 4.2.3
Duplication	D : 4.2.4	C : 4.2.3	G : 4.2.7	F : 4.2.6
Spéciation	X	C : 4.2.3	F : 4.2.6	E : 4.2.5

NB : le X dans le tableau correspond à un cas impossible.

Le tableau est symétrique puisque dans une adjacence, les gènes ne sont pas ordonnés. Les 2 premiers algorithmes c_1 et c_0 prennent en entrée un couple de nœuds à examiner et vont faire appel à des algorithmes plus spécialisés en fonction du type des nœuds passés en paramètre (*cf.* 4.2.1 p.28). L'algorithme de la fonction c_0, similaire à celui de la fonction c_1, est disponible en annexe p.55. La fonction type appliquée à un nœud d'arbre de gènes renvoie son type. On a 4 possibilités : Gène Actuel, Perte de Gène, Duplication de Gène ou Spéciation.

Algorithme 1 $c_1(n_1, n_2)$

Données : n_1 et n_2 2 nœuds quelconques, l'un $\in G_1$, l'autre $\in G_2$
Sorties : renvoie $c_1(n_1, n_2)$

```
Switch (type(n1))
  Case GèneActuel
    Switch (type(n2))
      Case GèneActuel
        c1GèneActuelGèneActuel(n1, n2) //Cas A
      Case Perte
        c1PerteGDS(n2, n1) //Cas C
      Case Duplication
        c1GèneActuelDuplication(n1, n2) //Cas D
  Case Perte
    Switch (type(n2))
      Case GèneActuel, Duplication ou Spéciation
        c1PerteGDS(n1, n2) //Cas C
      Case Perte
        c1PertePerte(n1, n2) //Cas B
  Case Duplication
    Switch (type(n2))
      Case GèneActuel
        c1GèneActuelDuplication(n2, n1) //Cas D
      Case Perte
        c1PerteGDS(n2, n1) //Cas C
      Case Duplication
        c1DuplicationDuplication(n1, n2) //Cas G
      Case Spéciation
        c1SpéciationDuplication(n2, n1) //Cas F
  Case Spéciation
    Switch (type(n2))
      Case Perte
        c1PerteGDS(n2, n1) //Cas C
      Case Duplication
        c1SpéciationDuplication(n1, n2) //Cas F
      Case Spéciation
        c1SpéciationSpéciation(n1, n2) //Cas E
```

Les nœuds n_1 et n_2, que les algorithmes c_1 et c_0 traitent, sont des nœuds de même espèce. Par conséquent, on ne rencontrera jamais le cas où n_1 est un gène actuel et n_2 un nœud de spéciation ou son symétrique (*cf. NB* p.19).

Tous les algorithmes utilisés dans c_1 et c_0 sont décrits dans les sections suivantes. On y utilise les fonctions fg et fd.
$fg(n)$ renvoie le fils gauche du nœud n passé en paramètre.
$fd(n)$ renvoie le fils droit du nœud n passé en paramètre.

4.2.1 Cas A : GèneActuel/GèneActuel

Les 2 nœuds sont des Gènes Actuels. Si l'adjacence entre ces Gènes Actuels appartient à $\mathcal{L}$ (alors elle appartient à $\mathcal{L}(N_1, N_2)$), alors pour c_1 on calcule le coût différentiel d'un arbre réduit à cette Adjacence Actuelle auquel on retire le coût de la Création ($Cr - Cr - Cr$), pour c_0 on calcule le coût différentiel de ce même arbre réduit à une feuille ($Cr - Cr$). Si l'adjacence entre ces Gènes Actuels n'appartient pas à $\mathcal{L}$ alors pour c_1, on a le coût différentiel d'un arbre réduit à une Cassure auquel on retire le coût d'une Création ($Cr + Ca - Cr$), pour c_0 il n'y a rien à faire puisqu'elle n'existe pas.

Algorithme 2 c_1GèneActuelGèneActuel(n_1, n_2)

Données : n_1 et n_2 2 gènes actuels
Sorties : renvoie $c_1(n_1, n_2)$
  si $n_1 \sim n_2 \in L$ **alors**
    renvoyer $Cr - Cr - Cr$
  sinon
    renvoyer Ca
  fin si

Algorithme 3 c_0GèneActuelGèneActuel(n_1, n_2)

Données : n_1 et n_2 2 gènes actuels
Sorties : renvoie $c_0(n_1, n_2)$
  si $n_1 \sim n_2 \in L$ **alors**
    renvoyer $+Cr - Cr)$
  sinon
    renvoyer 0
  fin si

4.2.2 Cas B : Perte/Perte

Algorithme 4 c_1PertePerte(n_1, n_2)

Données : n_1 et n_2 2 pertes
Sorties : renvoie $c_1(n_1, n_2)$
  /*coût différentiel d'un arbre réduit à une perte d'adjacence*/
  renvoyer $P_A - 2 * P_G$

Algorithme 5 c_0PertePerte(n_1, n_2)

Données : n_1 et n_2 2 pertes
Sorties : renvoie $c_0(n_1, n_2)$
  renvoyer 0

Pour ces 2 algorithmes, l'adjacence $n_1 \sim n_2 \notin L$ puisque ni n_1 ni n_2 ne sont des gènes actuels.

4.2.3 Cas C : Perte/(GèneActuel ou Duplication ou Spéciation)

Aucune adjacence entre n_1 et un autre gène ne peut exister. Ceci implique que $\mathcal{L}(n_1, n_2)$ est vide.

Algorithme 6 c_1PerteGDS(n_1, n_2)

Données : n_1 une Perte et n_2 un Gène Actuel ou un nœud de Duplication ou un nœud de Spéciation
Sorties : renvoie $c_1(n_1, n_2)$
  renvoyer 0

Algorithme 7 c_0PerteGDS(n_1, n_2)

Données : n_1 une Perte et n_2 un Gène Actuel ou un nœud de Duplication ou un nœud de Spéciation
Sorties : renvoie $c_0(n_1, n_2)$
  renvoyer 0

4.2.4 Cas D : GèneActuel/Duplication

Algorithme 8 c_1GèneActuelDuplication(n_1, n_2)

Données : n_1 une gène actuel et n_2 un nœud de duplication
Sorties : renvoie $c_1(n_1, n_2)$

$$\text{renvoyer min} \begin{cases} c_1(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) & (1) \\ c_0(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) & (2) \\ c_1(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) + Cr & (3) \\ c_0(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) + Ca & (4) \end{cases}$$

À la ligne (1), l'adjacence $n_1 \sim n_2$ se transmet au fils gauche de n_2 d'où $c_1(n_1, fg(n_2))$ et donc pas au fils droit de n_2 d'où $c_0(n_1, fd(n_2))$.
À la ligne (2), l'adjacence $n_1 \sim n_2$ se transmet au fils droit de n_2 d'où $c_1(n_1, fd(n_2))$ et donc pas au fils gauche de n_2 d'où $c_0(n_1, fg(n_2))$.
À la ligne (3), l'adjacence $n_1 \sim n_2$ se transmet à un des 2 fils et on rajoute une création pour le deuxième d'où les 2 c_1 et la création.
À la ligne (4), l'adjacence $n_1 \sim n_2$ se transmet à un des 2 fils et on rajoute une cassure pour le deuxième d'où les 2 c_0 et la cassure.

Algorithme 9 c_0GèneActuelDuplication(n_1, n_2)

Données : n_1 un gène actuel et n_2 un nœud de duplication
Sorties : renvoie $c_0(n_1, n_2)$

$$\text{renvoyer min} \begin{cases} c_0(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) & (1) \\ c_1(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) + Cr & (2) \\ c_0(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) + Cr & (3) \\ c_1(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) + 2 * Cr & (4) \end{cases}$$

Dans ce cas, il n'y a pas d'adjacence entre n_1 et n_2.
À la ligne (1), aucune adjacence n'est transmise.
À la ligne (2), on crée une adjacence entre n_1 et le fils gauche de n_2.
À la ligne (3), on crée une adjacence entre n_1 et le fils droit de n_2.
À la ligne (4), on crée 2 adjacences entre n_1 et les 2 fils de n_2.

4.2.5 Cas E : Spéciation/Spéciation

Algorithme 10 c_1SpéciationSpéciation(n_1, n_2)

Données : n_1 et n_2 2 nœuds de spéciation
Sorties : renvoie $c_1(n_1, n_2)$

$$\text{renvoyer min} \begin{cases} c_1(fg(n_1), fg(n_2)) + c_1(fd(n_1), fd(n_2)) & (1) \\ c_1(fg(n_1), fg(n_2)) + c_0(fd(n_1), fd(n_2)) + Ca & (2) \\ c_0(fg(n_1), fg(n_2)) + c_1(fd(n_1), fd(n_2)) + Ca & (3) \\ c_0(fg(n_1), fg(n_2)) + c_0(fd(n_1), fd(n_2)) + 2 * Ca & (4) \end{cases}$$

L'adjacence se transmet au fils gauche et au fils droit dans tous les cas.
À la ligne (1), transmission d'adjacences.
À la ligne (2), on casse l'adjacence du fils droit
À la ligne (3), on casse l'adjacence du fils gauche
À la ligne (4), on casse les 2 adjacences

Algorithme 11 c_0SpéciationSpéciation(n_1, n_2)

Données : n_1 et n_2 2 nœuds de spéciation
Sorties : renvoie $c_0(n_1, n_2)$

$$\text{renvoyer min} \begin{cases} c_0(fg(n_1), fg(n_2)) + c_0(fd(n_1), fd(n_2)) & (1) \\ c_1(fg(n_1), fg(n_2)) + c_0(fd(n_1), fd(n_2)) + Cr & (2) \\ c_0(fg(n_1), fg(n_2)) + c_1(fd(n_1), fd(n_2)) + Cr & (3) \\ c_1(fg(n_1), fg(n_2)) + c_1(fd(n_1), fd(n_2)) + 2 * Cr & (4) \end{cases}$$

À la ligne (1), pas d'adjacence transmise.
À la ligne (2), on crée l'adjacence au fils gauche.
À la ligne (3), on crée l'adjacence au fils droit.
À la ligne (4), on crée une adjacence à chacun des fils.

4.2.6 Cas F : Spéciation/Duplication

Dans ce cas, la Duplication doit se faire avant la Spéciation. Soit le gène A_1 de l'espèce A qui va subir une Spéciation et donner 2 fils : B_1 et C_1 respectivement de l'espèce B et de l'espèce C. Soit le gène A_2 de l'espèce A qui va subir une Duplication et donner 2 fils A_3 et A_4 de l'espèce A. En effet, A_1, A_2, A_3 et A_4 sont de la même espèce. A_2 peut être contemporain à A_1, A_3 et A_4 mais A_3 et A_4 ne peuvent être contemporains à B_1 ou C_1 car ils appartiennent au génome de leur ancêtre.

Algorithme 12 c_1SpéciationDuplication(n_1, n_2)

Données : n_1 un nœud de spéciation et n_2 un nœud de duplication
Sorties : renvoie $c_1(n_1, n_2)$

$$\text{renvoyer min} \begin{cases} c_1(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) & (1) \\ c_0(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) & (2) \\ c_1(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) + Cr & (3) \\ c_0(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) + Ca & (4) \end{cases}$$

À la ligne (1), l'adjacence se transmet au fils gauche de n_2 et donc pas à son fils droit.
À la ligne (2), l'adjacence se transmet au fils droit de n_2 et donc pas à son fils gauche.
À la ligne (3), l'adjacence se transmet à un des 2 fils de n_2 et on rajoute une création pour le deuxième.
À la ligne (4), l'adjacence se transmet à un des 2 fils de n_2 mais on la casse.

Algorithme 13 c_0SpéciationDuplication(n_1, n_2)

Données : n_1 un nœud de spéciation et n_2 un nœud de duplication
Sorties : renvoie $c_0(n_1, n_2)$

$$\text{renvoyer min} \begin{cases} c_0(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) & (1) \\ c_1(n_1, fg(n_2)) + c_0(n_1, fd(n_2)) + Cr & (2) \\ c_0(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) + Cr & (3) \\ c_1(n_1, fg(n_2)) + c_1(n_1, fd(n_2)) + 2 * Cr & (4) \end{cases}$$

À la ligne (1), pas d'adjacence transmise.
À la ligne (2), on crée l'adjacence entre n_1 et le fils gauche de n_2.
À la ligne (3), on crée l'adjacence entre n_1 et le fils droit de n_2.
À la ligne (4), on crée 2 adjacences entre n_1 et chacun des 2 fils de n_2.

4.2.7 Cas G : Duplication/Duplication

Algorithme 14 c_1DuplicationDuplication(n_1, n_2)

Données : n_1 et n_2 2 nœuds de duplication

Sorties : renvoie $c_1(n_1, n_2)$

renvoyer min

$$
\left\{
\begin{array}{llllllllllllllll}
c_1(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & & & & & & & (1)\\
c_1(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Ca & & & & & (2)\\
c_0(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Ca & & & & & (3)\\
c_0(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & 2*Ca & & & & & (4)\\
c_0(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & & & & & & & (5)\\
c_0(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Ca & & & & & (6)\\
c_0(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Ca & & & & & (7)\\
c_1(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & 2*Cr & & & & & (8)\\
c_0(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & & & & & (9)\\
c_1(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & & & & & (10)\\
c_1(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & & & & & (11)\\
c_1(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & & & & & (12)\\
c_1(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & + & Ca & & & (13)\\
c_0(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_0(fg(n_1),fd(n_2)) & + & c_1(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & + & Ca & & & (14)\\
c_0(fg(n_1),fg(n_2)) & + & c_1(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & + & Ca & & & (15)\\
c_1(fg(n_1),fg(n_2)) & + & c_0(fd(n_1),fd(n_2)) & + & c_1(fg(n_1),fd(n_2)) & + & c_0(fd(n_1),fg(n_2)) & + & D_A-2*D_G & + & Cr & + & Ca & & & (16)\\
c_1(n_1,fg(n_2)) & + & c_0(n_1,fd(n_2)) & & & & & & & & & & & & & (17)\\
c_0(n_1,fg(n_2)) & + & c_1(n_1,fd(n_2)) & & & & & & & & & & & & & (18)\\
c_1(n_1,fg(n_2)) & + & c_1(n_1,fd(n_2)) & + & Cr & & & & & & & & & & & (19)\\
c_0(n_1,fg(n_2)) & + & c_0(n_1,fd(n_2)) & + & Ca & & & & & & & & & & & (20)\\
c_1(fg(n_1),n_2) & + & c_0(fd(n_1),n_2) & & & & & & & & & & & & & (21)\\
c_0(fg(n_1),n_2) & + & c_1(fd(n_1),n_2) & & & & & & & & & & & & & (22)\\
c_1(fg(n_1),n_2) & + & c_1(fd(n_1),n_2) & + & Cr & & & & & & & & & & & (23)\\
c_0(fg(n_1),n_2) & + & c_0(fd(n_1),n_2) & + & Ca & & & & & & & & & & & (24)
\end{array}
\right.
$$

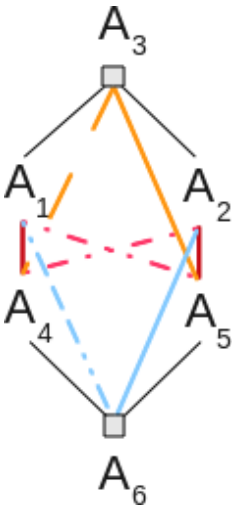

FIG. 4.1 – Adjacence entre 2 nœuds de duplication

Sur la figure 4.1, les adjacences potentielles ont été représentées en pointillés de couleur. Les adjacences rouges correspondent aux lignes (1), (2), (3) et (4). Les adjacences vertes correspondent aux lignes (4), (5), (6) et (7). Il y a 2 adjacences rouges, elles peuvent être cassées soit toutes les 2, soit une seule, soit aucune, il en va de même pour les adjacences vertes. Les adjacences rouges et vertes correspondent chacune à la duplication en simultanée des gènes A_3 et A_6. Ici, une adjacence va en engendrer 2.
On voit que la ligne (4) apparaît une fois dans chaque groupe d'adjacence, c'est parce qu'il s'agit du cas où les adjacences transmises sont cassées.
Comme ces groupes d'adjacences rouges et vertes sont symétriques, on va pouvoir étudier ces lignes simultanément.
Aux lignes (1) et (5), on transmet les adjacences aux 2 couples d'enfants.
Aux lignes (2), (3), (6) et (7), on transmet l'adjacence aux 2 couples d'enfants mais on casse une des 2 adjacences, d'où la cassure.
À la ligne (4), on casse les 2 adjacences, d'où les 2 cassures.
Les lignes (8) à (16) sont toutes les combinaisons qu'on peut avoir entre les groupes d'adjacences rouges et vertes que l'on n'a pas encore étudiées, par exemple : transmission des rouges et création d'une verte.
Les adjacences rouges, vertes, jaunes et bleues ne peuvent pas être issues de la même histoire évolutive. On dit qu'elles sont incompatibles.
Les adjacences jaunes correspondent aux lignes (17) à (20) et alors que les bleues aux lignes (21) à (24). Les adjacences jaunes représente la duplication du gène A_3 en premier et les bleues celle du gène A_6 en premier. Dans ces 2 cas, une adjacence va en donner une. Elles sont symétriques, on va donc pouvoir les étudier simultanément :
Aux lignes (17) et (21), on transmet l'adjacence entre un nœud de duplication et le fils gauche de l'autre nœud.
Aux lignes (18) et (22), on transmet l'adjacence entre un nœud de duplication et le fils droit de l'autre nœud.
Aux lignes (19) et (23), on transmet l'adjacence entre un nœud de duplication et un des 2 fils puis on crée une adjacence entre ce même nœud de duplication et l'autre fils.
Aux lignes (19) et (23), on transmet l'adjacence entre un nœud de duplication et un des 2 fils puis on casse cette adjacence, d'où la cassure.

Pour le coût 0, c'est plus simple, on reprend les mêmes cas que pour c_1DuplicationDuplication(n_1, n_2) et pour chaque ligne, puisqu'on est dans le cas où on ne transmet pas d'adjacence, on aura autant de création qu'il y a de fonction c_1. Il n'y aura pas de cassure.

Algorithme 15 c_0DuplicationDuplication(n_1, n_2)

Données : n_1 et n_2 2 nœuds de duplication

Sorties : renvoie $c_0(n_1, n_2)$

renvoyer min $\left\{ \begin{array}{llllllllll}
c_1(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & + & 2 * Cr & (1) \\
c_1(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & + & Cr & (2) \\
c_0(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & + & Cr & (3) \\
c_0(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & & & (4) \\
c_0(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & 2 * Cr & (5) \\
c_0(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & + & Cr & (6) \\
c_0(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & Cr & (7) \\
c_1(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & 4 * Cr & (8) \\
c_0(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & 3 * Cr & (9) \\
c_1(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & 3 * Cr & (10) \\
c_1(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & 3 * Cr & (11) \\
c_1(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & + & 3 * Cr & (12) \\
c_1(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & 2 * Cr & (13) \\
c_0(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_0(fg(n_1), fd(n_2)) & + & c_1(fd(n_1), fg(n_2)) & + & 2 * Cr & (14) \\
c_0(fg(n_1), fg(n_2)) & + & c_1(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & + & 2 * Cr & (15) \\
c_1(fg(n_1), fg(n_2)) & + & c_0(fd(n_1), fd(n_2)) & + & c_1(fg(n_1), fd(n_2)) & + & c_0(fd(n_1), fg(n_2)) & + & 2 * Cr & (16) \\
c_1(n_1, fg(n_2)) & + & c_0(n_1, fd(n_2)) & + & Cr & & & & & (17) \\
c_0(n_1, fg(n_2)) & + & c_1(n_1, fd(n_2)) & + & Cr & & & & & (18) \\
c_1(n_1, fg(n_2)) & + & c_1(n_1, fd(n_2)) & + & 2 * Cr & & & & & (19) \\
c_0(n_1, fg(n_2)) & + & c_0(n_1, fd(n_2)) & & & & & & & (20) \\
c_1(fg(n_1), n_2) & + & c_0(fd(n_1), n_2) & + & Cr & & & & & (21) \\
c_0(fg(n_1), n_2) & + & c_1(fd(n_1), n_2) & + & Cr & & & & & (22) \\
c_1(fg(n_1), n_2) & + & c_1(fd(n_1), n_2) & + & 2 * Cr & & & & & (23) \\
c_0(fg(n_1), n_2) & + & c_0(fd(n_1), n_2) & & & & & & & (24)
\end{array} \right.$

4.3 Preuves d'arrêt et d'optimalité des algorithmes de coût

Nous avons 7 cas à étudier rappelés dans le tableau de la figure 4.2.

Cas A : GèneActuel/GèneActuel	Algorithme 2	c_1GèneActuelGèneActuel
	Algorithme 3	c_0GèneActuelGèneActuel
Cas B : Perte/Perte	Algorithme 4	c_1PertePerte
	Algorithme 5	c_0PertePerte
Cas C : Perte/GDS	Algorithme 6	c_1PerteGDS
	Algorithme 7	c_0PerteGDS
Cas D : GèneActuel/Duplication	Algorithme 8	c_1GèneActuelDuplication
	Algorithme 9	c_0GèneActuelDuplication
Cas E : Spéciation/Spéciation	Algorithme 10	c_1SpéciationSpéciation
	Algorithme 11	c_0SpéciationSpéciation
Cas F : Spéciation/Duplication	Algorithme 12	c_1SpéciationDuplication
	Algorithme 13	c_0SpéciationDuplication
Cas G : Duplication/Duplication	Algorithme 14	c_1DuplicationDuplication
	Algorithme 15	c_0DuplicationDuplication

FIG. 4.2 – Mémo de correspondance entre les cas étudiés et les algorithmes :
Les cellules grisées correspondent aux algorithmes d'arrêt qui renvoient directement un coût sans rappeler c_1 ou c_0

Les preuves d'optimalité de ces 7 paires d'algorithmes sont regroupées en 3 lots :

1. le premier lot contient les cas A, B et C : ce sont des cas d'arrêt car ils ne rappellent pas les algorithmes c_1 et c_0
2. le deuxième lot ne contient que le cas D que l'on prouve par récurrence
3. le troisième lot est constitué des cas E, F et G, ce sont aussi 3 cas récursifs que l'on prouvera ensemble.

Après avoir donné une preuve d'arrêt des algorithmes c_1 et c_0, nous prouverons l'optimalité de ces 14 algorithmes.

4.3.1 Preuve d'arrêt

Les algorithmes c_1 et c_0 appellent un des algorithmes de 2 à 15 en fonction du type de nœuds passés en paramètre. Un bref rappel des algorithmes utilisés est donné dans la figure 4.2.

Propriété 6 *Pour tous couples de nœuds $n_1 \in G_1$ et $n_2 \in G_2$, les algorithmes c_1 et c_0 s'arrêtent.*

Preuve Si c_1 ou c_0 fait appel à un des algorithmes de 2 à 7, alors ils s'arrêtent car les cas A, B et C auxquels ces algorithmes correspondent renvoient directement une valeur. Sinon, ils appellent un des algorithmes de 8 à 15 correspondants aux cas D, E, F et G. Ces algorithmes rappellent les algorithmes c_1 et c_0 sur un des couples de nœuds suivants : (n_1, $fg(n_2)$ ou $fd(n_2)$), (n_2, $fg(n_1)$ ou $fd(n_1)$) ou ($fg(n_2)$ ou $fd(n_2)$, $fg(n_1)$ ou $fd(n_1)$). On voit que chacun de ces 8 appels se fait sur des paires de nœuds dont au moins l'un des 2 est fils des nœuds initiaux. Dans ces cas là, on descend en direction des feuilles. Comme on travaille avec des arbres binaires, à force de descendre, on finira forcément par arriver aux feuilles et donc à un des cas d'arrêt.

4.3.2 Preuve d'optimalité des algorithmes 2 à 15

Les algorithmes 2 à 15 calculent le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ et à la liste L(n_1,n_2) avec (n_1, n_2) la paire de nœuds qui leur est passée en paramètre. Cette forêt est contrainte par les arbres $\mathcal{G}_1$ et $\mathcal{G}_2$ et les adjacences actuelles $\mathcal{L}(n_1,n_2)$ qui, pour rappel, ont servis à calculer le coût maximum. Dans le cas des algorithmes c_1, cette forêt doit contenir le nœud $n_1 \sim n_2$ mais le coût de création de cette adjacence est soustrait. Dans le cas des algorithmes c_0, cette forêt ne doit pas contenir le nœud $n_1 \sim n_2$ sauf si $n_1 \sim n_2 \in L$.

Tous ces algorithmes permettent d'avoir des créations d'adjacence plus bas, c.-à-d. entre descendants de n_1 et n_2.

On rappelle qu'on ne peut créer des adjacences qu'entre 2 nœuds de même espèce.

On n'examine pas les solutions qui sont d'emblée non optimales comme 2 événements consécutifs qui s'annulent : création et cassure d'adjacence.

Cas : A, B et C

Cas A : GèneActuel/GèneActuel

Propriété 7 *$\forall n_1 \in G_1$ une feuille de type Gène Actuel, $\forall n_2 \in G_2$ une feuille de type Gène Actuel, n_1 et n_2 appartenant à la même espèce, c_1GèneActuelGèneActuel(n_1, n_2) (resp. c_0GèneActuelGèneActuel(n_1, n_2)) calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}$).*

Preuve Les algorithmes c_1GèneActuelGèneActuel et c_0GèneActuelGèneActuel prennent en paramètre 2 nœuds n_1 et n_2 qui sont des Gènes Actuels. $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ sont donc réduits à un seul sommet. Les seuls événements possibles entre 2 Gènes Actuels sont la cassure d'adjacence si elle existe, la création d'adjacence si elle n'existe pas, ou aucun événement sur les adjacences.

Pour le calcul de c_1GèneActuelGèneActuel, on doit considérer deux cas :
- soit $n_1 \sim n_2 \in L$ et donc $n_1 \sim n_2 \in L(n_1, n_2)$: la meilleure solution est un arbre d'adjacences ne contenant que la feuille $n_1 \sim n_2$ de type Adjacence Actuelle. Le coût différentiel de cet arbre est 0 mais c_1GèneActuelGèneActuel y soustrait de coût de la création de $n_1 \sim n_2$, d'où $-Cr$.
- soit $n_1 \sim n_2 \notin L$ et donc $n_1 \sim n_2 \notin L(n_1, n_2)$: la forêt résultat doit contenir le nœud $n_1 \sim n_2$ mais cette adjacence n'est pas dans $\mathcal{L}(n_1, n_2)$, la meilleure solution est donc un arbre d'adjacences de contenant que la feuille $n_1 \sim n_2$ de type Cassure, d'où Ca.

Pour le calcul de c_0GèneActuelGèneActuel, on doit aussi considérer deux cas :
- soit $n_1 \sim n_2 \in L$ et donc $n_1 \sim n_2 \in L(n_1, n_2)$: la meilleure solution est un arbre d'adjacences ne contenant que la feuille $n_1 \sim n_2$ de type Adjacence Actuelle, d'où 0.
- soit $n_1 \sim n_2 \notin L$ et donc $n_1 \sim n_2 \notin L(n_1, n_2)$: la meilleure solution est un arbre d'adjacences vide (ne contenant aucun événement), d'où 0.

Cas B : Perte/Perte

Propriété 8 *$\forall n_1 \in G_1$ une feuille de type Perte, $\forall n_2 \in G_2$ une feuille de type Perte, n_1 et n_2 appartenant à la même espèce, c_1PertePerte(n_1, n_2) (resp. c_0PertePerte(n_1, n_2)) calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}$).*

Preuve Les algorithmes c_1PertePerte et c_0PertePerte prennent en paramètre 2 nœuds n_1 et n_2 qui sont des pertes de gène. $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ sont donc réduits à un seul sommet et $\mathcal{L}(n_1, n_2)$ est vide. Les seuls événements possibles entre 2 pertes sont la création d'adjacence, la perte d'adjacence, la cassure d'adjacence ou aucun événement sur les adjacences.

Pour c_1PertePerte, on a 3 solutions possibles :
- arbre réduit à la feuille $n_1 \sim n_2$ de type Perte de Gène : coût différentiel de cet arbre Cr
- arbre réduit à la feuille $n_1 \sim n_2$ de type Perte d'Adjacence, cela correspond à factoriser les 2 pertes de gène en une perte d'adjacence : coût différentiel de cet arbre $Cr + P_A - 2 * P_G$
- arbre réduit à la feuille $n_1 \sim n_2$ de type Cassure : coût différentiel de cet arbre $Cr + Ca$

c_1PertePerte soustrait le coût de la création de l'adjacence $n_1 \sim n_2$ à chacune de ces solutions et renvoie le coût de la deuxième qui est le coût minimum parmi les 3.

Pour c_0PertePerte, on a qu'une solution possible, c'est un arbre vide de coût différentiel 0.

Cas C : Perte/GDS

Propriété 9 *$\forall n_1 \in G_1$ une feuille de type Perte, $\forall n_2 \in G_2$ une feuille de type Gène Actuel ou un nœud de Duplication ou de Spéciation (abrégé GDS), n_1 et n_2 appartenant à la même espèce, c_1PerteGDS(n_1, n_2) (resp. c_0PerteGDS(n_1, n_2)) calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}$).*

Preuve Les algorithmes c_1PerteGDS et c_0PerteGDS prennent en paramètre un nœud n_1 qui est une Perte et un nœud n_2 qui peut être un Gène Actuel ou une Duplication ou une Spéciation. $\mathcal{G}_1(n_1)$ est réduit à un seul sommet mais $\mathcal{G}_2(n_2)$ peut-être de taille quelconque ; $\mathcal{L}(n_1, n_2)$ est vide car n_1 n'a pas de descendant. Les seuls événements possibles entre ces types de nœud sont les création ou cassure d'adjacence ou aucun événement sur les adjacences.

Pour c_1PerteGDS, on a 2 solutions possibles :
- arbre réduit à la feuille $n_1 \sim n_2$ de type Perte de Gène : coût différentiel de cet arbre Cr
- arbre réduit à la feuille $n_1 \sim n_2$ de type Cassure : coût différentiel de cet arbre $Cr + Ca$

c_1PerteGDS soustrait le coût de la création de l'adjacence $n_1 \sim n_2$ à chacune de ces solutions et renvoie le coût de la première qui est minimale et devient 0.

Pour c_0PerteGDS, on a qu'une solution possible, c'est un arbre vide de coût différentiel 0.

Cas D : GèneActuel/Duplication

Propriété 10 *$\forall n_1 \in G_1$ une feuille de type Gène Actuel, $\forall n_2 \in G_2$ un nœud de Duplication, n_1 et n_2 appartenant à la même espèce, c_1GèneActuelDuplication(n_1, n_2) (resp. c_0GèneActuelDuplication(n_1, n_2)) calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}$).*

Remarque Si une adjacence est possible entre un Gène Actuel n_1 et un nœud de Duplication n_2, c'est parce que n_1 et n_2 sont de la même espèce N et donc tous les nœuds internes descendant de n_2 sont également des nœuds de Duplication de cette même espèce N. Les feuilles descendant de n_2 sont soit des Gènes Actuels, soit des Pertes. Cette configuration particulière fait que l'appel à c_1GèneActuelDuplication ou à c_0GèneActuelDuplication va rappeler c_1 et c_0 avec comme paramètres 2 nœuds : n_1 et un fils de n_2 : n, tel que n est un nœud de Duplication ou une feuille (Gène Actuel ou Perte). Ce qui fait que c_1 et c_0 vont rappeler c_1GèneActuelDuplication et c_0GèneActuelDuplication ou les cas d'arrêt A(GèneActuel/GèneActuel) ou C(GèneActuel/Perte).

On se permet donc de prouver ce cas D avec une preuve par récurrence.

Preuve par récurrence Nous allons prouver la propriété 10 par récurrence sur h la hauteur[1] de $\mathcal{G}_2(n_2)$ dont n_2 est racine.

$P(h)$: $\forall n_1 \in G_1$ une feuille de type Gène Actuel, $\forall n_2 \in G_2$ un nœud de Duplication, n_1 et n_2 appartenant à la même espèce, c_1GèneActuelDuplication(n_1, n_2) (resp. c_0GèneActuelDuplication(n_1, n_2)) calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ de hauteur h et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}(n_1, n_2)$).

Remarque : h ne peut pas être égal à 0 car un nœud de duplication a toujours 2 fils.

Base : $h = 1$ n_2 est le père de 2 feuilles (Gène Actuel ou Perte) $n_3 = fg(n_2)$ et $n_4 = fd(n_2)$. On a donc 3 cas de figure :

1. n_3 et n_4 sont des Gènes Actuels
2. n_3 et n_4 sont des Pertes

[1] On définit la hauteur d'un arbre comme la longueur du plus long chemin allant de la racine à une feuille, c.-à-d le nombre d'arêtes maximum sur un tel chemin.

3. n_3 est un Gène Actuel et n_4 est une Perte ou l'inverse

Nous allons montrez que $P(1)$ est vraie pour le premier cas, les preuves pour les 2 autres cas sont fortement similaires.

Pour c_1GèneActuelDuplication, le nœud $n_1 \sim n_2$ présent dans la solution est un nœud de Duplication de gène avec exactement 1 fils (*cf.* Prop 1, p. 18), ce fils, que nous appelons α, étant soit $n_1 \sim n_3$, soit $n_1 \sim n_4$. Appelons β l'adjacence non fille de $n_1 \sim n_2$.

Les événements possibles sont alors la création de β ou la cassure de α, ou aucun événement sur les adjacences.

La solution se présente donc sous forme d'une forêt dans laquelle un arbre à pour racine $n_1 \sim n_2$ et pour seul fils la feuille α de type Adjacence Actuelle (AA) ou Cassure (Ca), selon que $\alpha \in L(n_1, n_2)$ ou non ; à c_0 et c_1 et éventuellement un 2^e arbre réduit à une feuille de type Adjacence Actuelle β si $\beta \in L(n_1, n_2)$.

Cela conduit à l'examen de huit cas (le coût indiqué tient compte du retrait du coût de création de $n_1 \sim n_2$) :

	fils= $n_1 \sim n_3$	$n_1 \sim n_3$ $\in L(n_1, n_2)$	$n_1 \sim n_4$ $\in L(n_1, n_2)$	Solution
1	oui	oui	oui	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ AA ; arbre 2 : feuille $n_1 \sim n_4$ AA ; coût $= (-Cr) + (Cr - Cr)$
2	oui	oui	non	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ AA ; pas d'arbre 2 ; coût $= (-Cr) + (0)$
3	oui	non	oui	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ Ca ; arbre 2 : feuille $n_1 \sim n_4$ AA ; coût $= (Ca) + (Cr - Cr)$
4	oui	non	non	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ Ca ; pas d'arbre 2 ; coût $= (Ca) + (0)$
5	non	oui	oui	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ AA ; arbre 2 : feuille $n_1 \sim n_3$ AA ; coût $= (-Cr) + (Cr - Cr)$
6	non	oui	non	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ Ca ; arbre 2 : feuille $n_1 \sim n_3$ AA ; coût $= (Ca) + (Cr - Cr)$
7	non	non	oui	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ AA ; pas d'arbre 2 ; coût $= (-Cr) + (0)$
8	non	non	non	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ Ca ; pas d'arbre 2 ; coût $= (Ca) + (0)$

Remarque 1 : Il y a une symétrie par rapport aux fils de $n_1 \sim n_2$: les cas 1 et 5 sont symétriques, ainsi que 2 et 7, 3 et 6, 4 et 8.

Remarque 2 : On remarque aussi que comme n_3 est un Gène Actuel, alors :

– si $n_1 \sim n_3 \in L(n_1, n_2)$ alors c_0GèneActuelGèneActuel(n_1,n_3)=c_1GèneActuelGèneActuel(n_1,n_3)+Cr

– et si $n_1 \sim n_3 \notin L(n_1, n_2)$ alors c_1GèneActuelGèneActuel(n_1,n_3)=c_0GèneActuelGèneActuel(n_1,n_3)+Ca

mêmes égalités pour n_4.

La solution optimale parmi les 8 solutions présentées dans le tableau ci-dessus, est celle de coût minimum et qui respecte les appartenances de $n_1 \sim n_3$ et $n_1 \sim n_4$ à $\mathcal{L}(n_1, n_2)$:

– Le coût des solutions de 1 à 4 est calculé par l'expression :
c_1GèneActuelGèneActuel(n_1,n_3) + c_0GèneActuelGèneActuel(n_1,n_4)
Le premier terme valant $-Cr$ ou Ca selon que $n_1 \sim n_3 \in L(n_1, n_2)$, le second $Cr - Cr$ ou 0 selon que $n_1 \sim n_4 \in L(n_1, n_2)$ (*cf.* preuve du cas A : GèneActuelGèneActuel).
Cette expression est calculée par l'appel $c_1(n_1, fg(n_2))$ + $c_0(n_1, fd(n_2))$ **ligne (1)** de l'algo 8 car $fg(n_2) = n_3$ est un Gène Actuel et $fd(n_2) = n_4$ également.

– Le coût des solutions de 5 à 8 symétriques sont calculées par l'expression
c_0GèneActuelGèneActuel(n_1,n_3) + c_1GèneActuelGèneActuel(n_1,n_4).
Cette expression est calculée par l'appel $c_0(n_1, fg(n_2))$ + $c_1(n_1, fd(n_2))$ **ligne (2)** de l'algo 8 car $fg(n_2) = n_3$ est un Gène Actuel et $fd(n_2) = n_4$ également.

– Au vu de la **Remarque 2**, la **ligne (3)** de l'algo 8 devient équivalente à la ligne (1) si $n_1 \sim n_3$ est fille de $n_1 \sim n_2$ et à la ligne (2) si $n_1 \sim n_3$ est fille de $n_1 \sim n_2$, elle calcule donc le coût des cas 1, 3, 5 et 6 ; et la **ligne (4)** le coût des cas 2, 4, 7 et 8.

L'algorithme 8 c_1GèneActuelDuplication prend bien le minimum entre ces 4 lignes.
On prouverait de manière similaire que l'algorithme c_0GèneActuelDuplication est optimal.

Donc $P(1)$ est vraie.

Supposition au rang h Supposons que $P(h)$ est vraie pour $h \geq 1$, c.-à-d que $\forall n_1 \in G_1$ une feuille de type Gène Actuel, $\forall n_2 \in G_2$ un nœud de Duplication, n_1 et n_2 appartenant à la même espèce, c_1GèneActuelDuplication(n_1, n_2) (resp. c_0GèneActuelDuplication(n_1, n_2)) calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ de hauteur $h \geq 1$ et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}(n_1, n_2)$).

Preuve de $P(h+1)$ Soit n_1 un Gène Actuel et n_2 un nœud de Duplication tel que $\mathcal{G}_2(n_2)$ soit de hauteur $h+1$. n_2 a 2 fils : $n_3 = fg(n_2)$ et $n_4 = fd(n_2)$ on a donc $\mathcal{G}_2(n_3) \leq h$ et $\mathcal{G}_2(n_4) \leq h$.

Montrons que c_1GèneActuelDuplication(n_1, n_2) renvoie le coût d'une solution optimale :

La forêt solution contient toujours au moins 1 ou 2 arbres. L'un ayant pour racine $n_1 \sim n_2$ et pour fils $n_1 \sim n_3$ ou $n_1 \sim n_4$, que nous appelons α, l'autre adjacence étant nommée β. L'autre arbre, si il est présent, à pour racine β. La hauteur de ces arbres n'est pas connue à l'avance. Ils ont pour feuilles des Cassures ou des Adjacences Actuelles.

Remarque : la forêt solution peut contenir plus que deux arbres si les appels récursifs crée des adjacences entre n_1 et des descendants de n_3 et n_4.

Nous pouvons reprendre les 8 cas examinés lors de la preuve de la base, et pour garder la cohérence des cas examinés, remplacer les titres des colonnes 3 et 4 de la manière suivante :

- $n_1 \sim n_3 \in L(n_1, n_2)$ devient type($n_1 \sim n_3$)
- $n_1 \sim n_4 \in L(n_1, n_2)$ devient type($n_1 \sim n_4$)

Sachant que le type des adjacences $n_1 \sim n_3$ et $n_1 \sim n_4$ considérés sont Cassure ou pas Cassure si elles sont filles de $n_1 \sim n_2$ et Création (racine du 2^e arbre) ou absence sinon, ce qui maintient 8 cas à examiner :

	fils= $n_1 \sim n_3$	type $n_1 \sim n_3$	type $n_1 \sim n_4$	Solution
1	oui	$\neq$ Ca	Cr	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ de type $\neq$ Ca ; arbre 2 : racine $n_1 \sim n_4$; coût = $(c_1(n_1, n_3)) + (c_1(n_1, n_4) + Cr)$
2	oui	$\neq$ Ca	abs	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ de type $\neq$ Ca ; pas d'arbre de racine $n_1 \sim n_4$; coût = $(c_1(n_1, n_3)) + (c_0(n_1, n_4))$
3	oui	Ca	Cr	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ de type Ca ; arbre 2 : racine $n_1 \sim n_4$; coût = $(c_0(n_1, n_3) + Ca) + (c_1(n_1, n_4) + Cr)$
4	oui	Ca	abs	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_3$ de type Ca ; pas d'arbre de racine $n_1 \sim n_4$; coût = $(c_0(n_1, n_3) + Ca) + (c_0(n_1, n_4))$
5	non	Cr	$\neq$ Ca	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ de type $\neq$ Ca ; arbre 2 : racine $n_1 \sim n_3$; coût = $(c_1(n_1, n_4)) + (c_1(n_1, n_3) + Cr)$
6	non	Cr	Ca	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ de type Ca ; arbre 2 : racine $n_1 \sim n_3$; coût = $(c_0(n_1, n_4) + Ca) + (c_1(n_1, n_3) + Cr)$
7	non	abs	$\neq$ Ca	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ de type $\neq$ Ca ; pas d'arbre de racine $n_1 \sim n_3$; coût = $(c_1(n_1, n_4)) + (c_0(n_1, n_3))$
8	non	abs	Ca	arbre 1 : racine $n_1 \sim n_2$ fils $n_1 \sim n_4$ de type Ca ; pas d'arbre de racine $n_1 \sim n_3$; coût = $(c_0(n_1, n_4) + Ca) + (c_0(n_1, n_3))$

On remarque que les solutions 1 et 5 ont le même coût ainsi que les solutions 4 et 8.

On rappelle que vu que n_1 est un Gène Actuel et n_2 un nœud de Duplication, l'appel des fonctions c_1 et c_0 sur (n_1, n_3) et (n_1, n_4) rappellent les fonctions c_1GèneActuelDuplication, c_0GèneActuelDuplication ou les cas d'arrêt A ou C. Les appels à c_1GèneActuelDuplication, c_0GèneActuelDuplication se font sur n_3 ou n_4 comme second paramètre, et $\mathcal{G}_2(n_3)$ et $G_2(n_4)$ sont de hauteur $\leq h$, donc par application de l'hypothèse de récurrence ces fonctions renvoient bien le résultat optimal. Les appels aux cas d'arrêt A et C renvoient eux aussi un résultat

optimal comme démontré précédemment.

La solution optimale parmi les 8 solutions présentées dans le tableau ci-dessus, est celle de coût minimum :
- Le coût des solutions 1 et 5 est calculé par la **ligne (3)** de l'algo 8
- Le coût de la solution 2 est calculé par la **ligne (1)** de l'algo 8
- Le coût de la solution 3 n'est jamais optimal car supérieur à celui de la solution 7
- Le coût des solutions 4 et 8 est calculé par la **ligne (4)** de l'algo 8
- Le coût de la solution 6 n'est jamais optimal car supérieur à celui de la solution 2
- Le coût de la solution 8 est calculé par la **ligne (2)** de l'algo 8

L'algorithme 8 c_1GèneActuelDuplication prend bien le minimum entre les 4 lignes évoquées.
On montrerai de manière similaire que l'algorithme 9 c_0GèneActuelDuplication est optimal.

Donc $P(h+1)$ est vraie.

Conclusion $P(h)$ est vraie quelque soit $h \geq 1$.

Cas E : Spéciation/Spéciation, F : Spéciation/Duplication, G : Duplication/Duplication

Propriété 11 *Les algorithmes c_1SpéciationSpéciation, c_0SpéciationSpéciation, c_1SpéciationDuplication, c_0SpéciationDuplication, c_1DuplicationDuplication et c_0DuplicationDuplication appliqués aux nœuds n_1 et n_2 calculent le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}$).*

Cette propriété peut se démontrer par récurrence de manière similaire mais un peu plus complexe que le cas D. Cette preuve sera intégrée au rapport ultérieurement.

4.4 Algorithme général

L'algorithme général est appelé DéCo pour Détection de Co-évolution. Il prend en entrée un fichier qui contient les données nécessaires décrit dans le chapitre suivant. Il commence par parser ce fichier et en extraire un arbre des espèces S, 2 arbres de gènes $\mathcal{G}_1$ et $\mathcal{G}_2$ et une liste d'adjacences $\mathcal{L}$. Ensuite, l'algorithme réconcilie chacun des 2 arbres de gènes avec $\mathcal{S}$. Et enfin, il calcule le minimum entre c_1 et c_0 calculés entre les 2 racines de $\mathcal{G}_1$ et $\mathcal{G}_2$ pour renvoyer le coût de la solution.

Algorithme 16 DéCo(*fichier.txt*)

parser(*fichier.txt*)
reconciliation(G_1, $\mathcal{S}$)
reconciliation(G_2, $\mathcal{S}$)
renvoyer min(c_1(racine(G_1), racine(G_2)), c_0(racine(G_1), racine(G_2))) + coûtMax

4.5 Preuve de l'algorithme général

Dans cette section, nous mettrons d'ici la fin du stage les preuves des propriétés suivantes :

Propriété 12 *L'algorithme DéCo renvoie le coût optimal.*

Propriété 13 *La solution optimale est le minimum entre c_1 et c_0.*

Propriété 14 *L'algorithme DéCo appliqués aux nœuds n_1 et n_2 calcule le coût différentiel minimum d'une forêt d'arbres d'adjacences associée aux arbres $\mathcal{G}_1(n_1)$ et $\mathcal{G}_2(n_2)$ et à la liste $\mathcal{L}(n_1, n_2)$, forêt dans laquelle le nœud $n_1 \sim n_2$ existe et à laquelle on retire son coût de création (resp. dans laquelle le nœud $n_1 \sim n_2$ n'existe pas sauf s'il appartient à $\mathcal{L}$).*

Propriété 15 *L'algorithme calcule tous les coûts pour tous les couples de nœuds de même espèce.*

Preuve Nous avons prouvé précédemment que les algorithmes de 2 à 15 calculaient bien les coûts minimum pour tous les cas possibles. Il en va de même pour les algorithmes c_1 et c_0. Ainsi, le fait de prendre les racines des 2 arbres de gènes permet de garantir que les coûts de tous les couples de nœuds de même espèce sont calculés.

Propriété 16 *Un appel à c_0 ou à c_1 sur une paire de nœuds de même espèce ne génère que des appels à c_0 et c_1 sur des paires de nœuds de même espèce.*

4.6 Complexité

Le principe sur lequel repose l'algorithme DéCo est celui de la programmation dynamique. Donc tous les calculs de coût, pour chacun des couples de nœuds de même espèce et d'arbres de gènes différents, ne sont effectués qu'une seule fois. Chacun des algorithmes de 2 à 15 ont une complexité de l'ordre de la taille d'un arbre de gènes multiplié par la taille de l'autre arbre. Les algorithmes c_1 et c_0 sont quadratiques car ils font un nombre de tests limités.
L'étape de l'algorithme ayant la complexité la plus important est celle qui calcule chacun des coûts c_1 et c_0 pour tous les nœuds des 2 arbres de gènes. Soient m_1 le nombre de nœuds de l'arbre de gènes $\mathcal{G}_1$ et m_2 le nombre de nœuds de l'arbre de gènes $\mathcal{G}_2$. Nous avons donc une complexité de $m_1 * m_2$. Nous écrirons la preuve plus proprement ultérieurement.

4.7 Tests sur des exemples construits

Afin de réaliser ce stage, plusieurs exemples ont été nécessaires. En voici quelques-uns qui nous permettront de mieux appréhender l'algorithme mis en place.

4.7.1 Exemple 1

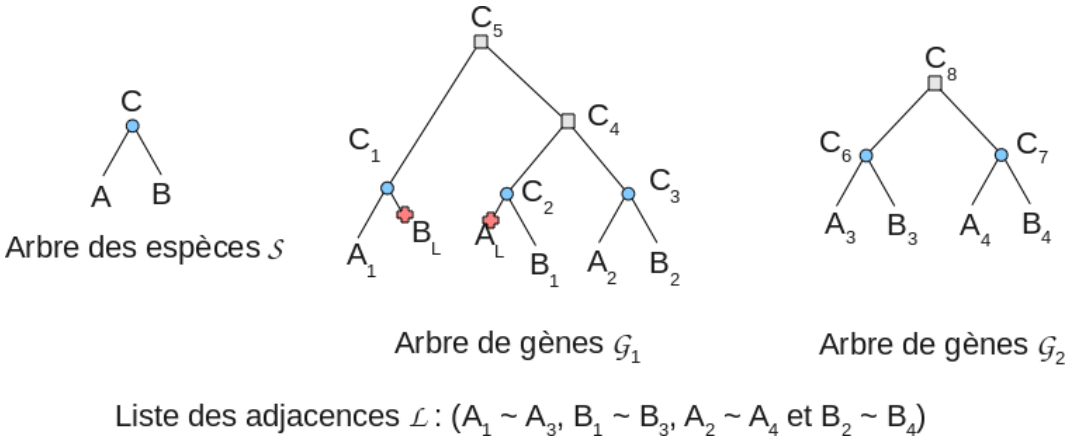

FIG. 4.3 – Données de l'algorithme : $\mathcal{S}$, $\mathcal{L}$, $\mathcal{G}_1$ et $\mathcal{G}_2$

On a donc un arbre des espèces $\mathcal{S}$, une liste de 4 adjacences $\mathcal{L}$ et 2 arbres de gènes réconciliés $\mathcal{G}_1$ et $\mathcal{G}_2$. À partir de ces données, on trace le graphe reliant les 2 arbres de gènes par les adjacences.

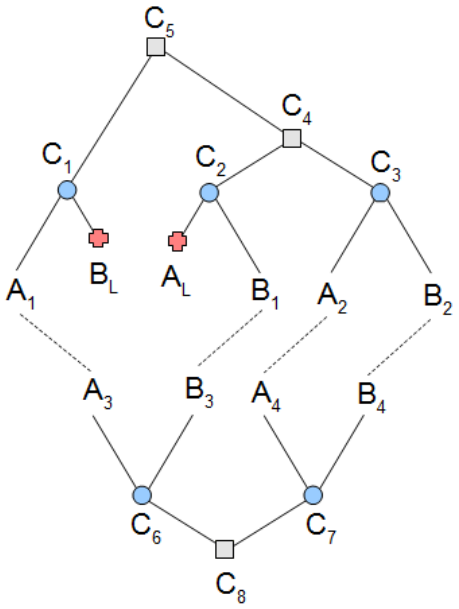

FIG. 4.4 – Graphe de l'exemple 1

L'algorithme va donc calculer les fonctions c_1 et c_0 pour chaque couple de nœuds de la même espèce. Dans cet exemple, les coûts de tous les événements sont 1 (sauf la spéciation qui vaut 0). On a donc les coûts c_1 et c_0 suivants pour les gènes actuels :

$$
\begin{array}{llll}
c_1(A_1, A_3) = & \begin{matrix} -Cr \\ -1 \end{matrix} & c_0(A_1,\, A_3) = & \begin{matrix} +Cr - Cr \\ 0 \end{matrix} \\
c_1(A_1, A_4) = & \begin{matrix} +Ca \\ 1 \end{matrix} & c_0(A_1,\, A_4) = & 0 \\
c_1(B_L, B_3) = & 0 & c_0(B_L,\, B_3) = & 0 \\
c_1(B_L, B_4) = & 0 & c_0(B_L,\, B_4) = & 0 \\
c_1(A_L, A_3) = & 0 & c_0(A_L,\, A_3) = & 0 \\
c_1(A_L, A_4) = & 0 & c_0(A_L,\, A_4) = & 0 \\
c_1(B_1, B_3) = & \begin{matrix} -Cr \\ -1 \end{matrix} & c_0(B_1,\, B_3) = & \begin{matrix} +Cr - Cr \\ 0 \end{matrix} \\
c_1(B_1, B_4) = & \begin{matrix} +Ca \\ 1 \end{matrix} & c_0(B_1,\, B_4) = & 0 \\
c_1(A_2, A_3) = & \begin{matrix} +Ca \\ 1 \end{matrix} & c_0(A_2,\, A_3) = & 0 \\
c_1(A_2, A_4) = & \begin{matrix} -Cr \\ 1 \end{matrix} & c_0(A_2,\, A_4) = & \begin{matrix} +Cr - Cr \\ 0 \end{matrix} \\
c_1(B_2, B_3) = & \begin{matrix} +Ca \\ 1 \end{matrix} & c_0(B_2,\, B_3) = & 0 \\
c_1(B_2, B_4) = & \begin{matrix} -Cr \\ -1 \end{matrix} & c_0(B_2,\, B_4) = & 0
\end{array}
$$

Pour les nœuds internes on a :

$$
\begin{aligned}
c_1(C_1,C_6) &= min(\quad c_1(A_1,A_3)+c_1(B_L,B_3),\\
&\qquad\quad c_1(A_1,A_3)+c_0(B_L,B_3)+Ca,\\
&\qquad\quad c_0(A_1,A_3)+c_1(B_L,B_3)+Ca,\\
&\qquad\quad c_0(A_1,A_3)+c_0(B_L,B_3)+2*Ca)\\
&= min(-1+0,-1+0+1,0+0+1,0+0+2*1)\\
&= min(-1,0,1,2)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_1,C_7) &= min(\quad c_1(A_1,A_4)+c_1(B_L,B_4),\\
&\qquad\quad c_1(A_1,A_4)+c_0(B_L,B_4)+Ca,\\
&\qquad\quad c_0(A_1,A_4)+c_1(B_L,B_4)+Ca,\\
&\qquad\quad c_0(A_1,A_4)+c_0(B_L,B_4)+2*Ca)\\
&= min(1+0,1+0+1,0+0+1,0+0+2*1)\\
&= min(1,2,1,2)\\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_2,C_6) &= min(\quad c_1(A_L,A_3)+c_1(B_1,B_3),\\
&\qquad\quad c_1(A_L,A_3)+c_0(B_1,B_3)+Ca,\\
&\qquad\quad c_0(A_L,A_3)+c_1(B_1,B_3)+Ca,\\
&\qquad\quad c_0(A_L,A_3)+c_0(B_1,B_3)+2*Ca)\\
&= min(0-1,0+0+1,0-1+1,0+0+2*1)\\
&= min(-1,1,0,2)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_2,C_7) &= min(\quad c_1(A_L,A_4)+c_1(B_1,B_4),\\
&\qquad\quad c_1(A_L,A_4)+c_0(B_1,B_4)+Ca,\\
&\qquad\quad c_0(A_L,A_4)+c_1(B_1,B_4)+Ca,\\
&\qquad\quad c_0(A_L,A_4)+c_0(B_1,B_4)+2*Ca)\\
&= min(0+1,0+0+1,0+1+1,0+0+2*1)\\
&= min(1,1,2,2)\\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_3,C_6) &= min(\quad c_1(A_2,A_3)+c_1(B_2,B_3),\\
&\qquad\quad c_1(A_2,A_3)+c_0(B_2,B_3)+Ca,\\
&\qquad\quad c_0(A_2,A_3)+c_1(B_2,B_3)+Ca,\\
&\qquad\quad c_0(A_2,A_3)+c_0(B_2,B_3)+2*Ca)\\
&= min(1+1,1+0+1,0+1+1,0+0+2)\\
&= min(2,2,2,2)\\
&= 2
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_3,C_7) &= min(\quad c_1(A_2,A_4)+c_1(B_2,B_4),\\
&\qquad\quad c_1(A_2,A_4)+c_0(B_2,B_4)+Ca,\\
&\qquad\quad c_0(A_2,A_4)+c_1(B_2,B_4)+Ca,\\
&\qquad\quad c_0(A_2,A_4)+c_0(B_2,B_4)+2*Ca)\\
&= min(-1-1,-1+0+1,0-1+1,0+0+2*1)\\
&= min(-2,0,0,2)\\
&= -2
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_6,C_4) &= min(\quad c_1(C_6,C_2)+c_0(C_6,C_3),\\
&\qquad\quad c_0(C_6,C_2)+c_1(C_6,C_3),\\
&\qquad\quad c_1(C_6,C_2)+c_1(C_6,C_3)+Cr,\\
&\qquad\quad c_0(C_6,C_2)+c_0(C_6,C_3)+Ca)\\
&= min(-1+0,0+2,-1+2+1,0+0+1)\\
&= min(-1,2,2,1)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_1,C_6) &= min(\quad c_0(A_1,A_3)+c_0(B_L,B_3),\\
&\qquad\quad c_1(A_1,A_3)+c_0(B_L,B_3)+Cr,\\
&\qquad\quad c_0(A_1,A_3)+c_1(B_L,B_3)+Cr,\\
&\qquad\quad c_1(A_1,A_3)+c_1(B_L,B_3)+2*Cr)\\
&= min(0+0,-1+0+1,0+0+1,-1+0\\
&\quad +2*1)\\
&= min(0,0,1,1)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_1,C_7) &= min(\quad c_0(A_1,A_4)+c_0(B_L,B_4),\\
&\qquad\quad c_1(A_1,A_4)+c_0(B_L,B_4)+Cr,\\
&\qquad\quad c_0(A_1,A_4)+c_1(B_L,B_4)+Cr,\\
&\qquad\quad c_1(A_1,A_4)+c_1(B_L,B_4)+2*Cr)\\
&= min(0+0,1+0+1,0+0+1,1+0+\\
&\quad 2*1)\\
&= min(0,2,1,3)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_2,C_6) &= min(\quad c_0(A_L,A_3)+c_0(B_1,B_3),\\
&\qquad\quad c_1(A_L,A_3)+c_0(B_1,B_3)+Ca,\\
&\qquad\quad c_0(A_L,A_3)+c_1(B_1,B_3)+Ca,\\
&\qquad\quad c_1(A_L,A_3)+c_1(B_1,B_3)+2*Ca)\\
&= min(0+0,0+0+1,0-1+1,0-1\\
&\quad +2*1)\\
&= min(0,1,0,1)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_2,C_7) &= min(\quad c_0(A_L,A_4)+c_0(B_1,B_4),\\
&\qquad\quad c_1(A_L,A_4)+c_0(B_1,B_4)+Cr,\\
&\qquad\quad c_0(A_L,A_4)+c_1(B_1,B_4)+Cr,\\
&\qquad\quad c_1(A_L,A_4)+c_1(B_1,B_4)+2*Cr)\\
&= min(0+0,0+0+1,0+1+1,0+1\\
&\quad +2*1)\\
&= min(0,1,2,3)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_3,C_6) &= min(\quad c_0(A_2,A_3)+c_0(B_2,B_3),\\
&\qquad\quad c_1(A_2,A_3)+c_0(B_2,B_3)+Cr,\\
&\qquad\quad c_0(A_2,A_3)+c_1(B_2,B_3)+Cr,\\
&\qquad\quad c_1(A_2,A_3)+c_1(B_2,B_3)+2*Cr)\\
&= min(0+0,1+0+1,0+1+1,1+1+2*1)\\
&= min(0,2,2,4)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_3,C_7) &= min(\quad c_0(A_2,A_4)+c_0(B_2,B_4),\\
&\qquad\quad c_1(A_2,A_4)+c_0(B_2,B_4)+Cr,\\
&\qquad\quad c_0(A_2,A_4)+c_1(B_2,B_4)+Cr,\\
&\qquad\quad c_1(A_2,A_4)+c_1(B_2,B_4)+2*Cr)\\
&= min(0+0,-1+0+1,0-1+1,-1-1\\
&\quad +2*1)\\
&= min(0,0,0,0)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_6,C_4) &= min(\quad c_0(C_6,C_2)+c_0(C_6,C_3),\\
&\qquad\quad c_1(C_6,C_2)+c_0(C_6,C_3)+\mathrm{Cr},\\
&\qquad\quad c_0(C_6,C_2)+c_1(C_6,C_3)+Cr,\\
&\qquad\quad c_1(C_6,C_2)+c_1(C_6,C_3)+2*Cr)\\
&= min(0+0,-1+0+1,0+2+1,-1+2+2*1)\\
&= min(0,0,3,3)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_7,C_4) &= min(\quad c_1(C_7,C_2)+c_0(C_7,C_3),\\
&\qquad\quad c_0(C_7,C_2)+c_1(C_7,C_3),\\
&\qquad\quad c_1(C_7,C_2)+c_1(C_7,C_3)+Cr,\\
&\qquad\quad c_0(C_7,C_2)+c_0(C_7,C_3)+Ca)\\
&= min(1+0, 0-2, 1-2+1, 0+0+1)\\
&= min(1,-2,0,1)\\
&= -2
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_7,C_4) &= min(\quad c_0(C_7,C_2)+c_0(C_7,C_3),\\
&\qquad\quad c_0(C_7,C_2)+c_1(C_7,C_3)+Cr,\\
&\qquad\quad c_1(C_7,C_2)+c_1(C_7,C_3)+Cr,\\
&\qquad\quad c_0(C_7,C_2)+c_0(C_7,C_3)+2*Cr)\\
&= min(0+0, 1+0+1, 0-2+1, 1-2+2*1)\\
&= min(0,2,-1,1)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_6,C_5) &= min(\quad c_1(C_6,C_1)+c_0(C_6,C_4),\\
&\qquad\quad c_0(C_6,C_1)+c_1(C_6,C_4),\\
&\qquad\quad c_1(C_6,C_1)+c_1(C_6,C_4)+Cr,\\
&\qquad\quad c_0(C_6,C_1)+c_0(C_6,C_4)+Ca)\\
&= min(-1+0, 0-1, -1-1+1, 0+0+1)\\
&= min(-1,-1,-1,1)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_6,C_5) &= min(\quad c_0(C_6,C_1)+c_0(C_6,C_4),\\
&\qquad\quad c_1(C_6,C_1)+c_0(C_6,C_4)+Cr,\\
&\qquad\quad c_0(C_6,C_1)+c_1(C_6,C_4)+Cr,\\
&\qquad\quad c_1(C_6,C_1)+c_1(C_6,C_4)+2*Cr)\\
&= min(0+0, -1+0+1, 0-1+1, -1-1+2*1)\\
&= min(0,0,0,0)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_7,C_5) &= min(\quad c_1(C_7,C_1)+c_0(C_7,C_4),\\
&\qquad\quad c_0(C_7,C_1)+c_1(C_7,C_4),\\
&\qquad\quad c_1(C_7,C_1)+c_1(C_7,C_4)+Cr,\\
&\qquad\quad c_0(C_7,C_1)+c_0(C_7,C_4)+Ca)\\
&= min(1-1, 0-2, 1-2+1, 0-1+1)\\
&= min(0,-2,0,0)\\
&= -2
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_7,C_5) &= min(\quad c_0(C_7,C_1)+c_0(C_7,C_4),\\
&\qquad\quad c_1(C_7,C_1)+c_0(C_7,C_4)+Cr,\\
&\qquad\quad c_0(C_7,C_1)+c_1(C_7,C_4)+Cr,\\
&\qquad\quad c_1(C_7,C_1)+c_1(C_7,C_4)+2*Cr)\\
&= min(0-1, 1-1+1, 0-2+1, 1-2+2*1)\\
&= min(-1,1,-1,1)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_1,C_8) &= min(\quad c_1(C_1,C_6)+c_0(C_1,C_7),\\
&\qquad\quad c_0(C_1,C_6)+c_1(C_1,C_7),\\
&\qquad\quad c_1(C_1,C_6)+c_1(C_1,C_7)+Cr,\\
&\qquad\quad c_0(C_1,C_6)+c_0(C_1,C_7)+Ca)\\
&= min(-1+0, 0+1, -1+1+1, 0+0+1)\\
&= min(-1,1,1,1)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_1,C_8) &= min(\quad c_0(C_1,C_6)+c_0(C_1,C_7),\\
&\qquad\quad c_1(C_1,C_6)+c_0(C_1,C_7)+Cr,\\
&\qquad\quad c_0(C_1,C_6)+c_1(C_1,C_7)+Cr,\\
&\qquad\quad c_1(C_1,C_6)+c_1(C_1,C_7)+2*Cr)\\
&= min(0+0, -1+0+1, 0+1+1, -1+1+2*1)\\
&= min(0,0,2,2)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_2,C_8) &= min(\quad c_1(C_2,C_6)+c_0(C_2,C_7),\\
&\qquad\quad c_0(C_2,C_6)+c_1(C_2,C_7),\\
&\qquad\quad c_1(C_2,C_6)+c_1(C_2,C_7)+Cr,\\
&\qquad\quad c_0(C_2,C_6)+c_0(C_2,C_7)+Ca)\\
&= min(-1+0, 0+1, -1+1+1, 0+0+1)\\
&= min(-1,1,1,1)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_2,C_8) &= min(\quad c_0(C_2,C_6)+c_0(C_2,C_7),\\
&\qquad\quad c_1(C_2,C_6)+c_0(C_2,C_7)+Cr,\\
&\qquad\quad c_0(C_2,C_6)+c_1(C_2,C_7)+Cr,\\
&\qquad\quad c_1(C_2,C_6)+c_1(C_2,C_7)+2*Cr)\\
&= min(0+0, -1+0+1, 0+1+1, -1+1+2*1)\\
&= min(0,0,2,2)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_3,C_8) &= min(\quad c_1(C_3,C_6)+c_0(C_3,C_7),\\
&\qquad\quad c_0(C_3,C_6)+c_1(C_3,C_7),\\
&\qquad\quad c_1(C_3,C_6)+c_1(C_3,C_7)+Cr,\\
&\qquad\quad c_0(C_3,C_6)+c_0(C_3,C_7)+Ca)\\
&= min(2+0, 0-2, 2-2+1, 0+0+1)\\
&= min(2,-2,1,1)\\
&= -2
\end{aligned}
$$

$$
\begin{aligned}
c_0(C_3,C_8) &= min(\quad c_0(C_3,C_6)+c_0(C_3,C_7),\\
&\qquad\quad c_1(C_3,C_6)+c_0(C_3,C_7)+Cr,\\
&\qquad\quad c_0(C_3,C_6)+c_1(C_3,C_7)+Cr,\\
&\qquad\quad c_1(C_3,C_6)+c_1(C_3,C_7)+2*Cr)\\
&= min(0+0, 2+0+1, 0-2+1, 2-2+2*1)\\
&= min(0,3,-1,2)\\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
c_1(C_4,C_8) &= min(\quad c_1(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G+Ca,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G+Ca,\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G+2*Ca,\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G,\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G+Ca,\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G+Ca,\\
&\qquad c_1(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G+2*Cr,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_1(C_4,C_6)+c_0(C_4,C_7),\\
&\qquad c_0(C_4,C_6)+c_1(C_4,C_7),\\
&\qquad c_1(C_4,C_6)+c_1(C_4,C_7)+Cr,\\
&\qquad c_0(C_4,C_6)+c_0(C_4,C_7)+Ca,\\
&\qquad c_1(C_2,C_8)+c_0(C_3,C_8),\\
&\qquad c_0(C_2,C_8)+c_1(C_3,C_8),\\
&\qquad c_1(C_2,C_8)+c_1(C_3,C_8)+Cr,\\
&\qquad c_0(C_2,C_8)+c_0(C_3,C_8)+Ca)\\
&= min(-4,-1,-2,1,2,1,2,1,1,2,-1,-2,2,1,0,1,-2,-2,-2,0)\\
&= -4\\
c_0(C_4,C_8) &= min(\quad c_1(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6)+2Cr,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6)+Cr,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6)+Cr,\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_0(C_3,C_6),\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+2Cr,\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+Cr,\\
&\qquad c_0(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+Cr,\\
&\qquad c_1(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+4*Cr,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+3*Cr,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_1(C_3,C_6)+3*Cr,\\
&\qquad c_1(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+3*Cr,\\
&\qquad c_1(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+3*Cr,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+2*Cr,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_0(C_2,C_7)+c_1(C_3,C_6)+2*Cr,\\
&\qquad c_0(C_2,C_6)+c_1(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+2*Cr,\\
&\qquad c_1(C_2,C_6)+c_0(C_3,C_7)+c_1(C_2,C_7)+c_0(C_3,C_6)+2*Cr,\\
&\qquad c_1(C_4,C_6)+c_0(C_4,C_7),\\
&\qquad c_0(C_4,C_6)+c_1(C_4,C_7),\\
&\qquad c_1(C_4,C_6)+c_1(C_4,C_7)+Cr,\\
&\qquad c_0(C_4,C_6)+c_0(C_4,C_7)+Ca,\\
&\qquad c_1(C_2,C_8)+c_0(C_3,C_8),\\
&\qquad c_0(C_2,C_8)+c_1(C_3,C_8),\\
&\qquad c_1(C_2,C_8)+c_1(C_3,C_8)+Cr,\\
&\qquad c_0(C_2,C_8)+c_0(C_3,C_8)+Ca)\\
&= min(1,0,-1,0,5,2,3,4,4,5,2,1,3,2,1,2,0,-1,-1,0,0,-1,-1,0)\\
&= -1
\end{aligned}
$$

Calcul des coûts c_1 et c_0 entre les 2 racines :

$$
\begin{aligned}
c_1(C_5,C_8) &= min(\quad c_1(C_1,C_6)+c_1(C_4,C_7)+c_0(C_1,C_7)+c_0(C_4,C_6)+D_A-2D_G,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_0(C_4,C_6)+D_A-2D_G+Ca,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_0(C_4,C_7)+c_0(C_4,C_6)+D_A-2D_G+Ca,\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_0(C_4,C_6)+D_A-2D_G+2Ca,\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G,\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+D_A-2D_G+Ca,\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G+Ca,\\
&\qquad c_1(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G+2*Cr,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_1,C_6)+c_1(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+D_A-2D_G+Cr,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+D_A-2D_G+Cr+Ca,\\
&\qquad c_1(C_5,C_6)+c_0(C_5,C_7),\\
&\qquad c_0(C_5,C_6)+c_1(C_5,C_7),\\
&\qquad c_1(C_5,C_6)+c_1(C_5,C_7)+Cr,\\
&\qquad c_0(C_5,C_6)+c_0(C_5,C_7)+Ca,\\
&\qquad c_1(C_1,C_8)+c_0(C_4,C_8),\\
&\qquad c_0(C_1,C_8)+c_1(C_4,C_8),\\
&\qquad c_1(C_1,C_8)+c_1(C_4,C_8)+Cr,\\
&\qquad c_0(C_1,C_8)+c_0(C_4,C_8)+Ca)\\
&= min(-4,-2,-2,0,-2,0,-2,-2,-,-2,-4,-2,-2,-2,0,0,-2,-2,-2,0,-2,-4,-4,0)\\
&= -4\\
c_0(C_5,C_8) &= min(\quad c_1(C_1,C_6)+c_1(C_4,C_7)+c_0(C_1,C_7)+c_0(C_4,C_6)+2Cr,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_0(C_4,C_6)+Cr,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_0(C_4,C_7)+c_0(C_4,C_6)+Cr,\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_0(C_4,C_6),\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+2*Cr,\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+Cr,\\
&\qquad c_0(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+Cr,\\
&\qquad c_1(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+4*Cr,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+3*Cr,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_1(C_4,C_6)+3*Cr,\\
&\qquad c_1(C_1,C_6)+c_1(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+3*Cr,\\
&\qquad c_1(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+3*Cr,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+2*Cr,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_0(C_4,C_7)+c_1(C_4,C_6)+2*Cr,\\
&\qquad c_0(C_1,C_6)+c_1(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+2*Cr,\\
&\qquad c_1(C_1,C_6)+c_0(C_4,C_7)+c_1(C_4,C_7)+c_0(C_4,C_6)+2*Cr,\\
&\qquad c_1(C_5,C_6)+c_0(C_5,C_7)+Cr,\\
&\qquad c_0(C_5,C_6)+c_1(C_5,C_7)+Cr,\\
&\qquad c_1(C_5,C_6)+c_1(C_5,C_7)+2*Cr,\\
&\qquad c_0(C_5,C_6)+c_0(C_5,C_7),\\
&\qquad c_1(C_1,C_8)+c_0(C_4,C_8)+Cr,\\
&\qquad c_0(C_1,C_8)+c_1(C_4,C_8)+Cr,\\
&\qquad c_1(C_1,C_8)+c_1(C_4,C_8)+2*Cr,\\
&\qquad c_0(C_1,C_8)+c_0(C_4,C_8))\\
&= min(0,-1,-1,-1,1,1,-1,1,1,1,-1,1,-1,-1,1,1,-1,-1,-1,-1,-1,-3,-3,-1)\\
&= -3
\end{aligned}
$$

Le minimum entre $c_1(C_5,C_8)$ et $c_0(C_5,C_8)$ est $c_1(C_5,C_8) = -4$. On crée donc l'adjacence entre les racines.

Le coût -4 vient de la première ligne de $c_1(C_5,C_8)$, c'est-à-dire qu'on a une Création d'adjacence entre les nœuds C_1 et C_6 et les nœuds C_4 et C_7 et pas de Création entre C_1 et C_7 ni entre C_4 et C_6. D'ailleurs,

$c_0(C_1, C_7) = 0$, la ligne qui correspond à ce coût indique qu'il ne faut pas créer d'adjacence entre les Gènes actuels A_1 et A_4 ni entre B_L et B_4, $c_0(C_4, C_6) = 0$, la ligne qui correspond à ce coût indique qu'il ne faut pas créer d'adjacence entre les nœuds de Spéciation C_6 et C_2 ni entre les nœuds de Spéciation C_6 et C_3.

$c_1(C_1, C_6) = -1$, ce coût minimum vient de la transmission d'adjacence à $A_1 \sim A_3$ et à $B_L \sim B_3$. A_1, A_3 et B_3 sont des Gènes Actuels et B_L est une perte. $c_1(C_4, C_7) = -2$, ce coût minimum vient de la transmission d'adjacence à $C_7 \sim C_3$. C_3 et C_7 sont des nœuds de Spécation. L'adjacence $C_3 \sim C_7$ se transmet aux 2 couples de Gènes Actuels (A_2 et A_4) et (B_2 et B_4).

On a donc l'arbre d'adjacences suivant :

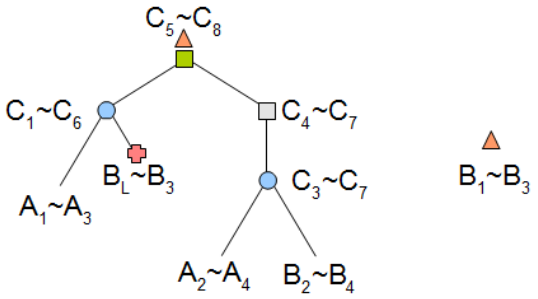

FIG. 4.5 – Arbre d'adjacences solution de l'exemple 1

4.7.2 Exemple 2

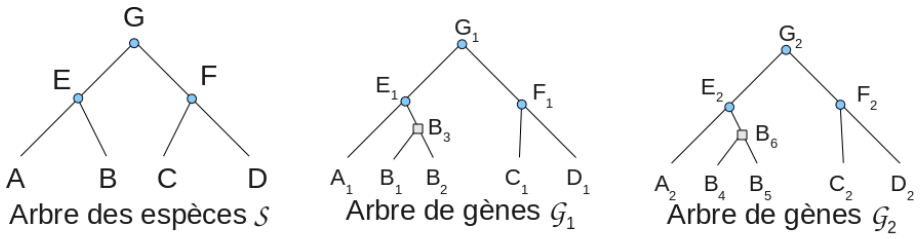

FIG. 4.6 – Données de l'algorithme : $\mathcal{S}$, $\mathcal{L}$, $\mathcal{G}_1$ et $\mathcal{G}_2$

On a donc un arbre des espèces $\mathcal{S}$, une liste de 4 adjacences $\mathcal{L}$ et 2 arbres de gènes réconciliés $\mathcal{G}_1$ et $\mathcal{G}_2$. À partir de ces données, on trace le graphe reliant les 2 arbres de gènes par les adjacences.

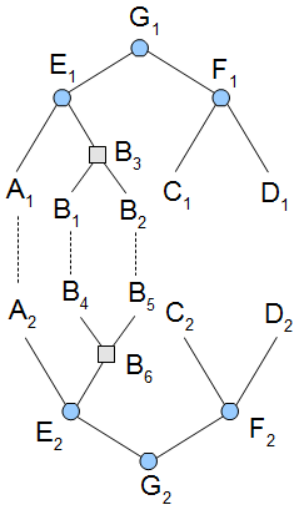

FIG. 4.7 – Graphe de l'exemple 2

L'algorithme va donc calculer les fonctions c_1 et c_0 pour chaque couple de nœuds de la même espèce. Dans cet exemple, les coûts de tous les événements sont 1 (sauf la Spéciation qui vaut 0).

On a donc les c_1 et c_0 suivant pour les couples de Gènes Actuels :

$$\begin{array}{lll} c_1(A_1,A_2) & = -Cr & c_0(A_1,A_2) = +Cr - Cr \\ & = -1 & = 0 \\ c_1(B_1,B_4) & = -Cr & c_0(B_1,B_4) = +Cr - Cr \\ & = -1 & = 0 \\ c_1(B_1,B_5) & = +Ca & c_0(B_1,B_5) = 0 \\ & = 1 & \\ c_1(B_2,B_4) & = +Ca & c_0(B_2,B_4) = 0 \\ & = 1 & \\ c_1(B_2,B_5) & = -Cr & c_0(B_2,B_5) = +Cr - Cr \\ & = -1 & = 0 \\ c_1(C_1,C_2) & = +Ca & c_0(C_1,C_2) = 0 \\ & = 1 & \\ c_1(D_1,D_2) & = +Ca & c_0(D_1,D_2) = 0 \\ & = 1 & \end{array}$$

Pour les couples impliquant un Gène Actuel et un nœud de Duplication on a :

$$\begin{array}{lll} c_1(B_4,B_3) & = min(& c_1(B_4,B_1) + c_0(B_4,B_2), \\ & & c_0(B_4,B_1) + c_1(B_4,B_2), \\ & & c_1(B_4,B_1) + c_1(B_4,B_2) + Cr, \\ & & c_0(B_4,B_1) + c_0(B_4,B_2) + Ca, \\ & = min(-1+0, 0+1, -1+1+1, 0+0+1) & \\ & = min(-1,1,1,1) & \\ & = -1 & \end{array}$$

$$\begin{array}{lll} c_0(B_4,B_3) & = min(& c_0(B_4,B_1) + c_0(B_4,B_2), \\ & & c_1(B_4,B_1) + c_0(B_4,B_2) + Cr, \\ & & c_0(B_4,B_1) + c_1(B_4,B_2) + Cr, \\ & & c_1(B_4,B_1) + c_1(B_4,B_2) + 2*Cr, \\ & = min(0+0, -1+0+1, 0+1+1, -1+1+2*1) & \\ & = min(0,0,2,2) & \\ & = 0 & \end{array}$$

$$\begin{array}{lll} c_1(B_5,B_3) & = min(& c_1(B_5,B_1) + c_0(B_5,B_2), \\ & & c_0(B_5,B_1) + c_1(B_5,B_2), \\ & & c_1(B_5,B_1) + c_1(B_5,B_2) + Cr, \\ & & c_0(B_5,B_1) + c_0(B_5,B_2) + Ca, \\ & = min(1+0, 0-1, 1-1+1, 0+0+1) & \\ & = min(1,-1,1,1) & \\ & = -1 & \end{array}$$

$$\begin{array}{lll} c_0(B_5,B_3) & = min(& c_0(B_5,B_1) + c_0(B_5,B_2), \\ & & c_1(B_5,B_1) + c_0(B_5,B_2) + Cr, \\ & & c_0(B_5,B_1) + c_1(B_5,B_2) + Cr, \\ & & c_1(B_5,B_1) + c_1(B_5,B_2) + 2*Cr, \\ & = min(0+0, 1+0+1, 0-1+1, 1-1+2*1) & \\ & = min(0,2,0,2) & \\ & = 0 & \end{array}$$

$$
\begin{aligned}
c_1(B_1,B_6) &= min(\quad c_1(B_1,B_4)+c_0(B_1,B_5),\\
&\qquad\quad c_0(B_1,B_4)+c_1(B_1,B_5),\\
&\qquad\quad c_1(B_1,B_4)+c_1(B_1,B_5)+Cr,\\
&\qquad\quad c_0(B_1,B_4)+c_0(B_1,B_5)+Ca,\\
&= min(-1+0, 0+1, -1+1+1, 0+0+1)\\
&= min(-1,1,1,1)\\
&= -1
\end{aligned}
\qquad
\begin{aligned}
c_0(B_1,B_6) &= min(\quad c_0(B_1,B_4)+c_0(B_1,B_5),\\
&\qquad\quad c_1(B_1,B_4)+c_0(B_1,B_5)+Cr,\\
&\qquad\quad c_0(B_1,B_4)+c_1(B_1,B_5)+Cr,\\
&\qquad\quad c_1(B_1,B_4)+c_1(B_1,B_5)+2*Cr,\\
&= min(0+0, -1+0+1, 0+1+1, -1+1+2*1)\\
&= min(0,0,2,2)\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
c_1(B_2,B_6) &= min(\quad c_1(B_2,B_4)+c_0(B_2,B_5),\\
&\qquad\quad c_0(B_2,B_4)+c_1(B_2,B_5),\\
&\qquad\quad c_1(B_2,B_4)+c_1(B_2,B_5)+Cr,\\
&\qquad\quad c_0(B_2,B_4)+c_0(B_2,B_5)+Ca,\\
&= min(1+0, 0-1, 1-1+1, 0+0+1)\\
&= min(1,-1,1,1)\\
&= -1
\end{aligned}
\qquad
\begin{aligned}
c_0(B_2,B_6) &= min(\quad c_0(B_2,B_4)+c_0(B_2,B_5),\\
&\qquad\quad c_1(B_2,B_4)+c_0(B_2,B_5)+Cr,\\
&\qquad\quad c_0(B_2,B_4)+c_1(B_2,B_5)+Cr,\\
&\qquad\quad c_1(B_2,B_4)+c_1(B_2,B_5)+2*Cr,\\
&= min(0+0, 1+0+1, 0-1+1, 1-1+2*1)\\
&= min(0,2,0,2)\\
&= 0
\end{aligned}
$$

Pour les couples impliquant 2 nœuds de Duplication on a :

$$
\begin{aligned}
c_1(B_3,B_6) &= min(\quad c_1(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G,\\
&\qquad\quad c_1(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G+Ca,\\
&\qquad\quad c_0(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G+Ca,\\
&\qquad\quad c_0(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G+2*Ca,\\
&\qquad\quad c_0(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G,\\
&\qquad\quad c_0(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G+Ca,\\
&\qquad\quad c_0(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G+Ca,\\
&\qquad\quad c_1(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G+2*Cr,\\
&\qquad\quad c_0(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G+Cr,\\
&\qquad\quad c_1(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G+Cr,\\
&\qquad\quad c_1(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G+Cr,\\
&\qquad\quad c_1(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G+Cr,\\
&\qquad\quad c_1(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G+Cr+Ca,\\
&\qquad\quad c_0(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+D_A-2*D_G+Cr+Ca,\\
&\qquad\quad c_0(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G+Cr+Ca,\\
&\qquad\quad c_1(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+D_A-2*D_G+Cr+Ca,\\
&\qquad\quad c_1(B_3,B_4)+c_0(B_3,B_5),\\
&\qquad\quad c_0(B_3,B_4)+c_1(B_3,B_5),\\
&\qquad\quad c_1(B_3,B_4)+c_1(B_3,B_5)+Cr,\\
&\qquad\quad c_0(B_3,B_4)+c_0(B_3,B_5)+Ca,\\
&\qquad\quad c_1(B_1,B_6)+c_0(B_2,B_6),\\
&\qquad\quad c_0(B_1,B_6)+c_1(B_2,B_6),\\
&\qquad\quad c_1(B_1,B_6)+c_1(B_2,B_6)+Cr,\\
&\qquad\quad c_0(B_1,B_6)+c_0(B_2,B_6)+Ca)\\
&= min(-3,-1,-1,1,1,1,1,1,1,1,-1,-1,1,1,1,1,-1,-1,-1,1,-1,-1,-1,1)\\
&= -3
\end{aligned}
$$

$$
\begin{aligned}
c_0(B_3,B_6) &= min(\quad c_1(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4)+2*Cr,\\
&\qquad c_1(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4)+Cr,\\
&\qquad c_0(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4)+Cr,\\
&\qquad c_0(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_0(B_2,B_4),\\
&\qquad c_0(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4),\\
&\qquad c_0(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+Ca,\\
&\qquad c_0(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+Ca,\\
&\qquad c_1(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4)+2*Cr,\\
&\qquad c_0(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4)+Cr,\\
&\qquad c_1(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_1(B_2,B_4)+Cr,\\
&\qquad c_1(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+Cr,\\
&\qquad c_1(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+Cr,\\
&\qquad c_1(B_1,B_4)+c_0(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+Cr,\\
&\qquad c_0(B_1,B_4)+c_1(B_2,B_5)+c_0(B_1,B_5)+c_1(B_2,B_4)+Cr,\\
&\qquad c_0(B_1,B_4)+c_1(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+Cr,\\
&\qquad c_1(B_1,B_4)+c_0(B_2,B_5)+c_1(B_1,B_5)+c_0(B_2,B_4)+Cr,\\
&\qquad c_1(B_3,B_4)+c_0(B_3,B_5),\\
&\qquad c_0(B_3,B_4)+c_1(B_3,B_5),\\
&\qquad c_1(B_3,B_4)+c_1(B_3,B_5)+Cr,\\
&\qquad c_0(B_3,B_4)+c_0(B_3,B_5)+Ca,\\
&\qquad c_1(B_1,B_6)+c_0(B_2,B_6),\\
&\qquad c_0(B_1,B_6)+c_1(B_2,B_6),\\
&\qquad c_1(B_1,B_6)+c_1(B_2,B_6)+Cr,\\
&\qquad c_0(B_1,B_6)+c_0(B_2,B_6)+Ca)\\
&= min(0,0,0,0,4,2,2,4,4,4,2,2,2,2,2,2,0,0,0,0,0,0,0,0)\\
&= 0
\end{aligned}
$$

Pour les couples impliquant 2 nœuds de Spéciation on a :

$$
\begin{aligned}
c_1(E_1,E_2) &= min(\quad c_1(A_1,A_2)+c_1(B_3,B_6),\\
&\qquad c_1(A_1,A_2)+c_0(B_3,B_6)+Ca,\\
&\qquad c_0(A_1,A_2)+c_1(B_3,B_6)+Ca,\\
&\qquad c_0(A_1,A_2)+c_0(B_3,B_6)+2*Ca,\\
&= min(-1-3,-1+0+1,0-3+1,0+0+2*1)\\
&= min(-4,0,-2,2)\\
&= -4
\end{aligned}
$$

$$
\begin{aligned}
c_0(E_1,E_2) &= min(\quad c_0(A_1,A_2)+c_0(B_3,B_6),\\
&\qquad c_1(A_1,A_2)+c_0(B_3,B_6)+Cr,\\
&\qquad c_0(A_1,A_2)+c_1(B_3,B_6)+Cr,\\
&\qquad c_1(A_1,A_2)+c_1(B_3,B_6)+2*Cr,\\
&= min(0+0,-1+0+1,0-3+1,-1-3\\
&+2*1)\\
&= min(0,2,-2,-2)\\
&= -2
\end{aligned}
$$

$$
\begin{aligned}
c_1(F_1,F_2) &= min(\quad c_1(C_1,C_2)+c_1(D_1,D_2),\\
&\qquad c_1(C_1,C_2)+c_0(D_1,D_2)+Ca,\\
&\qquad c_0(C_1,C_2)+c_1(D_1,D_2)+Ca,\\
&\qquad c_0(C_1,C_2)+c_0(D_1,D_2)+2*Ca,\\
&= min(1+1,1+0+1,0+1+1,0+0+2*1)\\
&= min(2,2,2,2)\\
&= 2
\end{aligned}
$$

$$
\begin{aligned}
c_0(F_1,F_2) &= min(\quad c_0(C_1,C_2)+c_0(D_1,D_2),\\
&\qquad c_1(C_1,C_2)+c_0(D_1,D_2)+Cr,\\
&\qquad c_0(C_1,C_2)+c_1(D_1,D_2)+Cr,\\
&\qquad c_1(C_1,C_2)+c_1(D_1,D_2)+2*Cr,\\
&= min(0+0,1+0+1,0+1+1,1+1+2*1)\\
&= min(0,2,2,4)\\
&= 0
\end{aligned}
$$

Calcul des coûts c_1 et c_0 entre les 2 racines :

$$
\begin{aligned}
c_1(G_1,G_2) &= min(\quad c_1(E_1,E_2)+c_1(F_1,F_2),\\
&\qquad c_1(E_1,E_2)+c_0(F_1,F_2)+Ca,\\
&\qquad c_0(E_1,E_2)+c_1(F_1,F_2)+Ca,\\
&\qquad c_0(E_1,E_2)+c_0(F_1,F_2)+2*Ca,\\
&= min(-4+2,-4+0+1,0+2+1,0+0+2*1)\\
&= min()\\
&= 2
\end{aligned}
$$

$$
\begin{aligned}
c_0(G_1,G_2) &= min(\quad c_0(E_1,E_2)+c_0(F_1,F_2),\\
&\qquad c_1(E_1,E_2)+c_0(F_1,F_2)+Cr,\\
&\qquad c_0(E_1,E_2)+c_1(F_1,F_2)+Cr,\\
&\qquad c_1(E_1,E_2)+c_1(F_1,F_2)+2*Cr,\\
&= min(0+0,-4+0+1,0+2+1,-4+2\\
&+2*1)\\
&= min(0,-3,3,0)\\
&= -3
\end{aligned}
$$

Le minimum entre $c_1(G_1,G_2)$ et $c_0(G_1,G_2)$ est $c_0(G_1,G_2) = -3$. On ne crée donc pas l'adjacence entre les racines.

Le coût -3 vient de la deuxième ligne de $c_0(G_1, G_2)$, c'est-à-dire qu'on a une Création d'adjacence entre les nœuds E_1 et E_2 et pas de Création entre F_1 et F_2. D'ailleurs, $c_0(F_1, F_2) = 0$, la ligne qui correspond à ce coût indique qu'il ne faut pas créer d'adjacence entre les Gènes actuels C_1 et C_2 ni entre D_1 et D_2.

$c_1(E_1, E_2) = -4$, ce coût minimum vient de la transmission d'adjacence à $A_1 \sim A_2$ et à $B_3 \sim B_6$. A_1 et A_2 sont des Gènes Actuels, et B_3 et B_6 sont un couple de nœuds de Duplication. L'adjacence $B_3 \sim B_6$ se transmet à $B_1 \sim B_4$ et à $B_2 \sim B_5$.

On a donc l'arbre d'adjacences suivant :

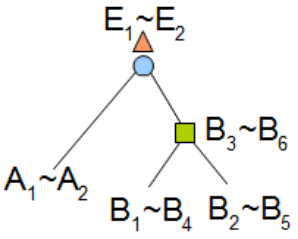

FIG. 4.8 – Arbre d'adjacences solution de l'exemple 2

Chapitre 5

Applications sur des données réelles

À ce jour, cette partie n'a pas encore été réalisée. Elle le sera durant le dernier mois de stage. Les données à traiter sont fournies sous forme d'un fichier texte organisé de cette manière :
Les premières lignes commencent par la chaîne de caractère "SEQ", elles correspondent chacune à la description d'un gène. On retrouve sur chaque ligne :

- le nom de l'espèce à laquelle appartient le gène
- le nom du gène
- le nom du chromosome qui le porte
- la localisation du départ du gène sur le chromosome
- la localisation de la fin du gène sur le chromosome
- le sens de lecture du gène sur le chromosome

Ensuite, vient une ligne où apparaît la chaîne de caractère "DATA". Elle annonce le début de l'arbre de gènes à la ligne suivante. L'arbre contient les noms des gènes qui sont donc les feuilles. Ici, on utilise une notation parenthésée, c'est-à-dire que 2 nœuds qui ont le même ancêtre vont être à l'intérieur d'une paire de parenthèses et séparés par une virgule.

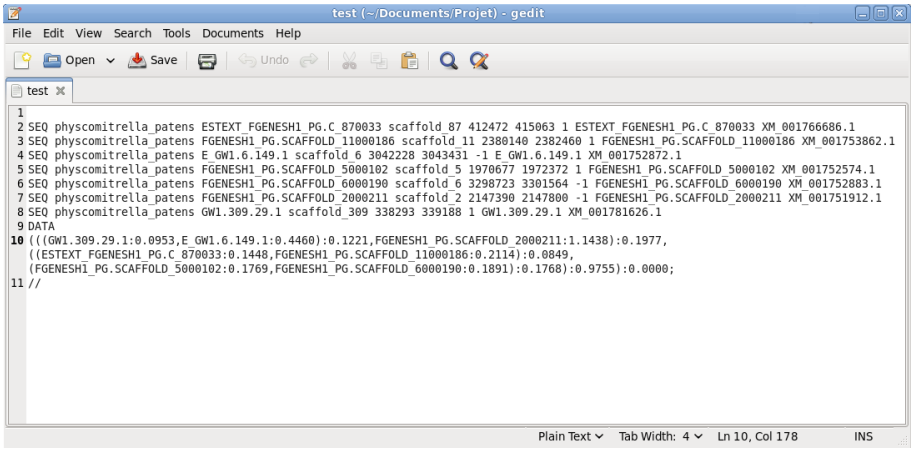

FIG. 5.1 – Exemple d'un jeu de données

Dans un même fichier, on peut avoir plusieurs arbres, ils seront séparés par une ligne contenant la chaîne "//". Ces types de fichiers sont disponibles sur le site : [NCBI Taxonomy]

Cette partie n'a pas pu être réalisée totalement pendant ce stage car l'élaboration de l'algorithme de réconciliation aura pris plus de temps que prévu. De plus, l'algorithme DéCo a été codé au fur et à mesure de l'avancement des recherches ce qui fait que le code n'est pas optimal. Il est donc relativement lent à l'exécution, et prend trop de place en mémoire. Les tests ne peuvent être donc fait que sur un nombre restreint d'arbres de gènes (une dizaine).

Cependant, sur les quelques exemples testés, on a pu remarqué que certains gènes ancestraux avaient plus de 2 gènes adjacents ce qui pourrait poser problème.

Conclusion et perspectives

Nous avons un algorithme qui prend en entrée un fichier de données. À l'aide d'un arbre des espèces codé dans un fichier au format newick, l'algorithme rapgreen réconcilie chacun des 2 arbres de gènes. L'algorithme DéCo calcule ensuite, pour chaque couple d'arbres de gènes et pour chaque couple de nœuds de la même espèce, les coûts des histoires possibles entre les nœuds de ce couple.

Les limites de cette approche, qui tente d'expliquer l'histoire des adjacences, reposent en partie sur une hypothèse faite au départ : l'indépendance des adjacences. Cependant, un modèle plus réaliste devrait prendre en compte l'interdépendance de certaines adjacences, en particulier celles reliant des paires de gènes proches sur le chromosome.

Par exemple : on a le gène A_1 adjacent au gène A_2 lui-même adjacent au gène A_3 dans une espèce A. Si le gène A_2 est perdu. On a alors une adjacence entre les gènes A_1 et A_3. Toutes ces adjacences sont considérées comme étant indépendantes alors qu'elles ne le sont pas.

Perspectives L'algorithme produit permet de résoudre l'histoire évolutive des adjacences, dans le cas où elles sont entre 2 arbres de gènes différents, il serait intéressant d'étendre ce procédé à un nombre illimité d'arbres de gènes.
Pour le moment, l'algorithme ne s'applique qu'à des adjacences entre 2 gènes appartenant à des arbres de gènes différents, il reste donc à étudier comment améliorer l'algorithme pour qu'il prenne en compte les adjacences entre 2 gènes appartenant à un même arbre de gènes.

Plus personnellement, ce stage m'a beaucoup apporté. Il m'a donné une réelle expérience du travail en équipe. Mais aussi, il m'a permis de me rendre compte de combien il était important de prouver un algorithme.

Annexes

Algorithme c_0

Algorithme 17 $c_0(n_1, n_2)$

Données : n_1 et n_2 2 nœuds quelconques
Sorties : renvoie $c_0(n_1, n_2)$

```
Switch (type(n1))
  Case GèneActuel
    Switch (type(n2))
      Case GèneActuel
        c0GèneActuelGèneActuel(n1, n2) //Cas A
      Case Perte
        c0PerteGDS(n2, n1)
      Case Duplication
        c0GèneActuelDuplication(n1, n2) //Cas D
  Case Perte
    Switch (type(n2))
      Case GèneActuel, Duplication ou Spéciation
        c0PerteGDS(n1, n2) //Cas C
      Case Perte
        c0PertePerte(n1, n2) //Cas B
  Case Duplication
    Switch (type(n2))
      Case GèneActuel
        c0GèneActuelDuplication(n2, n1) //Cas D
      Case Perte
        c0PerteGDS(n2, n1) //Cas C
      Case Duplication
        c0DuplicationDuplication(n1, n2) //Cas G
      Case Spéciation
        c0SpéciationDuplication(n2, n1) //Cas F
  Case Spéciation
    Switch (type(n2))
      Case Perte
        c0PerteGDS(n2, n1) //Cas C
      Case Duplication
        c0SpéciationDuplication(n1, n2) //Cas F
      Case Spéciation
        c0SpéciationSpéciation(n1, n2) //Cas E
```

Bibliographie

[Bertrand *et al.*, 2008] D. Bertrand, M. Lajoie and N. El-Mabrouk. **Inferring Ancestral Gene Orders for a Family of Tandemly Arrayed Genes** Journal of Computational Biology, Vol. 15, No.8, pp 1063-1077, 2008.

[Bertrand *et al.*, 2010] D. Bertrand, Y. Gagnon, M. Blanchette and N. El-Mabrouk **Reconstruction of Ancestral Genome subject to Whole Genome Duplication** Speciation, Rearrangement and Loss (preview) Proceedings of WABI 2010, LNCS/LNBI 6293, pp. 78- 89, 2010.

[Chauve *et al.*, 2010] Chauve C, Gavranovic H, Ouangraoua A, Tannier E, 2010 **Yeast Ancestral Genome Reconstructions : The Possibilities of Computational Methods II**, Journal of Computational Biology, vol.17(9) pp.1097-1112

[Cours de l'INSERM] D Thieffry, INSERM ERM206 Université de la Méditerranée, Marseille, 2006 **De la Bioinformatique à la Biologie des Systèmes**

[Cours de Master 2] Cours de bio-informatique du Master 2

[Dardel et Képès, 2002] Dardel F, Képès F, **Bioinformatique Génomique et post-génomique**

[Felsenstein, 1996] J. Felsenstein, **Inferring phylogenies from protein sequences by parsimony, distance, and likelihood methods**, 1996

[Fitch, 1971] W Fitch, 1971 **Towards defining the course of evolution : Minimum change for a specific tree topology** Systematic Zoology

[Glossaire de l'INRA] http ://www.inra.fr/internet/Directions/DIC/ACTUALITES/DOSSIERS/glossaire.html

[Lajoie *et al.*, 2010] M. Lajoie, D. Bertrand and N. El-Mabrouk **Inferring the Evolutionary History of Gene Clusters from Phylogenetic and Gene Order Data** (journal version ,software) Proceedings of Molecular Biology and Evolution, Vol.27, No. 4, pp. 761-772, 2010.

[Lecointre et Guyader, 2006] Classification phylogénétique du vivant, G. Lecointre, H. Le Guyader, 2006

[Ma *et al.*, 2006] Ma J, Zhang L, Suh BB, Raney BJ, Burhans RC, Kent WJ, Blanchette M, Haussler D, and Miller W. **Reconstructing contiguous regions of an ancestral genome** Genome Research, 16(12) :1557-1565, Dec 2006.

[Ma *et al.*, 2008] Ma J, Ratan A, Raney BJ, Suh BB, Zhang L, Miller W, and Haussler D. **DUPCAR : Reconstructing contiguous ancestral regions with duplications** Journal of Computational Biology, 15(8) :1007-1027, Oct 2008.

[NCBI Taxonomy] http ://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/

[Sankoff, 1975] Sankoff, 1975 **Minimal Mutation Trees of Sequences**, SIAM Journal on Applied Mathematics, 28, p.35.

[Tang and Wang, 2005] Tang, J. and Wang, L., **Improving Genome Rearrangement Phylogeny Using Sequence-Style Parsimony** Proc. 5th IEEE Conf. on Bioinformatics and Bioengineering (BIBE 2005), 137-144.

[Vernot *et al.*, 2007] B. Vernot, M. Stolzer, A. Goldman, D. Durand. 2007. **Reconciliation with Non-Binary Species Trees** In Computational Systems Bioinformatics : CSB2007 Conference Proceedings, Imperial College Press : 441-452

[wiki/Phylogénie] http ://fr.wikipedia.org/wiki/Phylogénie