Séminaire VAG – LIRMM – 13/11/2008

Décomposition et reconstruction de réseaux phylogénétiques de niveau k depuis des triplets

Philippe Gambette

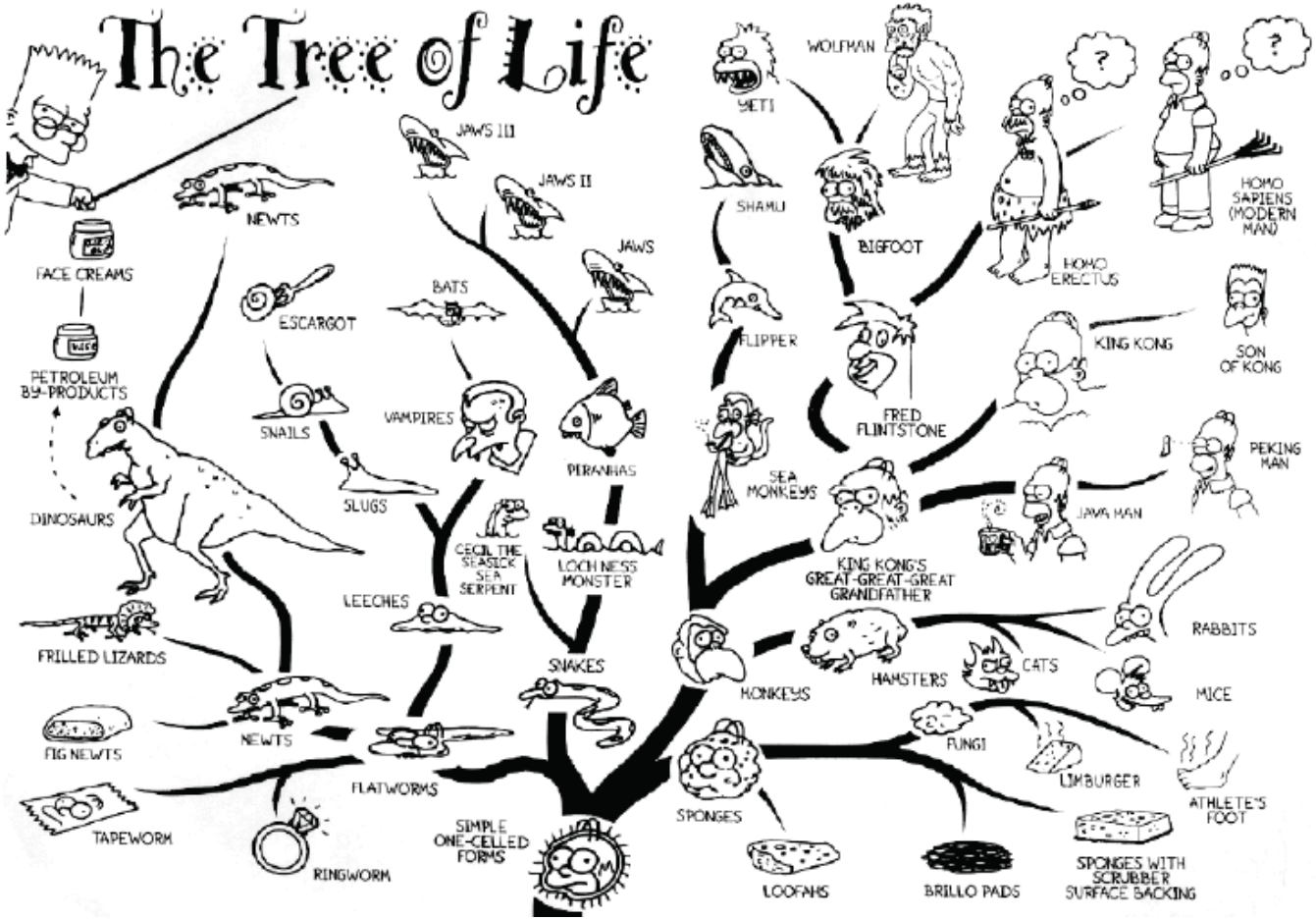
Plan

- Les réseaux phylogénétiques
- Décomposition des réseaux de niveau *k*
- Reconstruction de réseaux depuis des triplets
- L'approche par obstructions
- Codage des réseaux de niveau 1 par leurs triplets

Les arbres phylogénétiques

Arbre phylogénétique

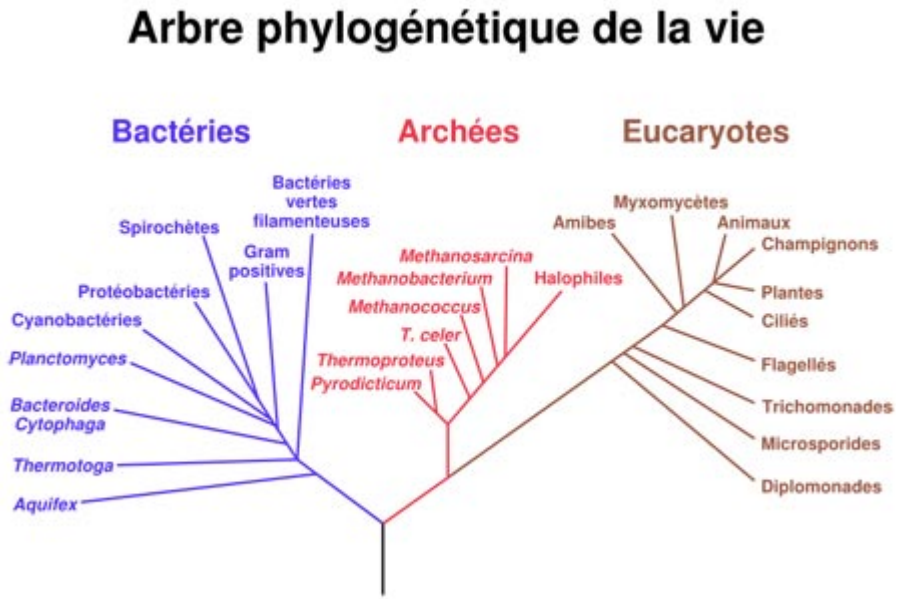
Un **arbre phylogénétique** est un arbre schématique qui montre les relations de parentés entre des entités supposées avoir un ancêtre commun.



Les arbres phylogénétiques

Arbre phylogénétique

Un **arbre phylogénétique** est un arbre schématique qui montre les relations de parentés entre des entités supposées avoir un ancêtre commun.

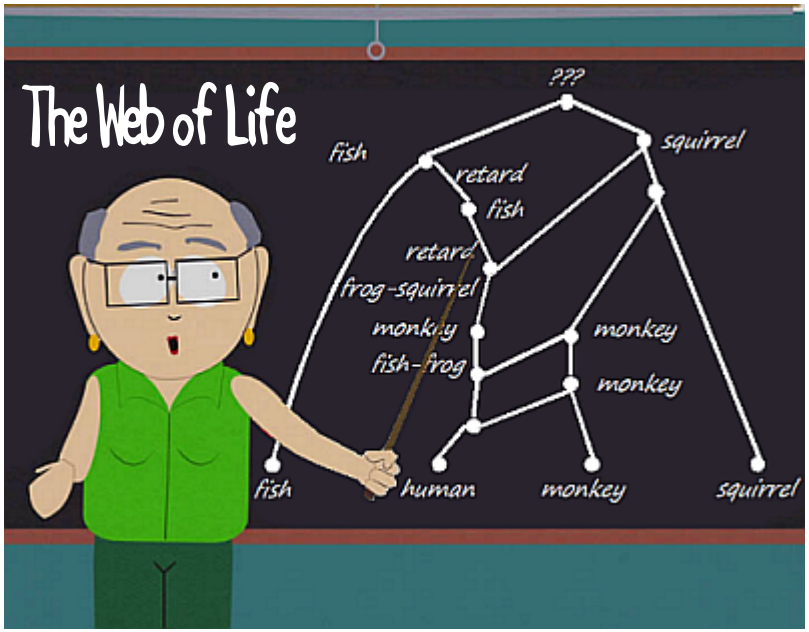


D'après Woese, Kandler, Wheelis : Towards a natural system of organisms: proposal for the domains Archaea, Bacteria, and Eucarya, Proceedings of the National Academy of Sciences, 87(12), 4576–4579 (1990)

Les réseaux phylogénétiques

Réseau phylogénétique

Un **réseau phylogénétique** désigne un graphe utilisé pour visualiser les relations liées à l'évolution entre des espèces ou des organismes. Il doit être employé quand interviennent des événements d'hybridations, de transferts horizontaux de gènes, ou de recombinaisons génétiques.

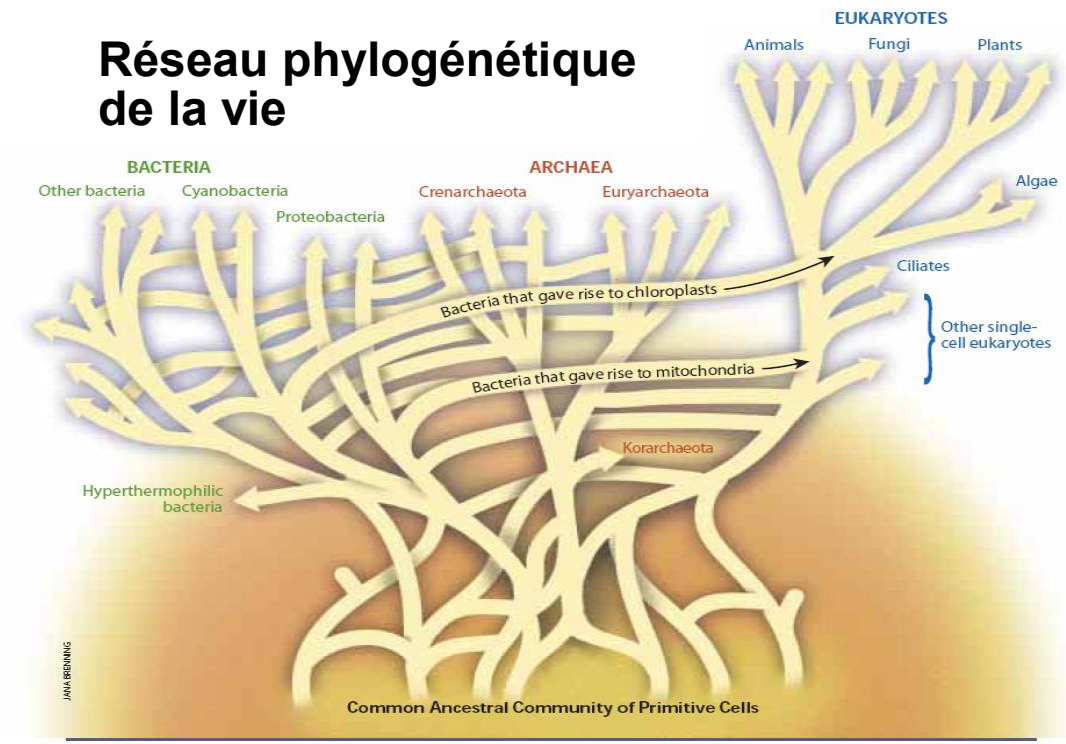


Le "réseau phylogénétique de la vie" d'après Mme Garrison dans South Park S10E12.

Les réseaux phylogénétiques

Réseau phylogénétique

Un **réseau phylogénétique** désigne un graphe utilisé pour visualiser les relations liées à l'évolution entre des espèces ou des organismes. Il doit être employé quand interviennent des événements d'hybridations, de transferts horizontaux de gènes, ou de recombinaisons génétiques.



Doolittle : Uprooting the Tree of Life, Scientific American (Fév. 2000)

Les réseaux phylogénétiques

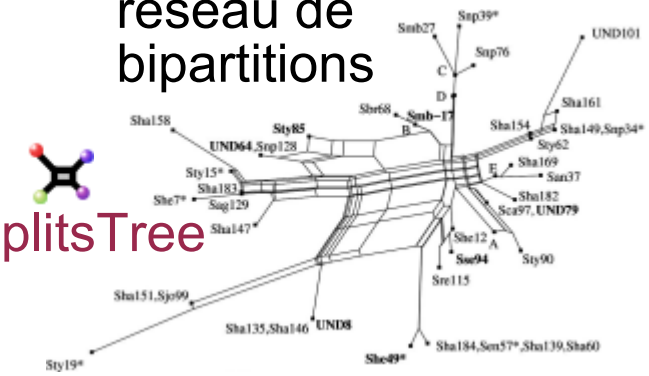
Réseau phylogénétique

Un **réseau phylogénétique** désigne un graphe utilisé pour visualiser les relations liées à l'évolution entre des espèces ou des organismes. Il doit être employé quand interviennent des événements d'hybridations, de transferts horizontaux de gènes, ou de recombinaisons génétiques.

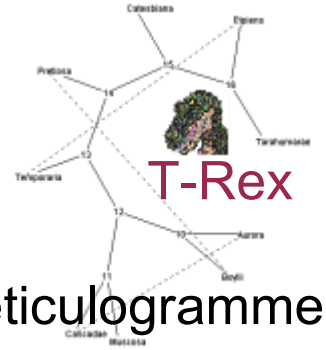
réseau de niveau 2

Level-2

réseau de bipartitions



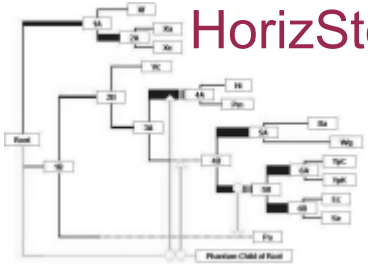
SplitsTree



T-Rex

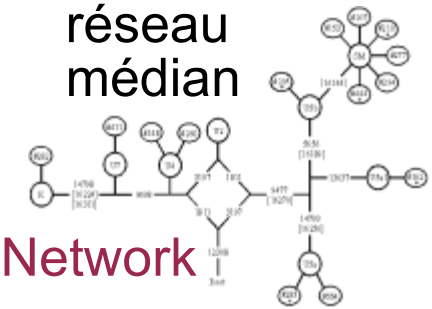
réticulogramme

diagramme de synthèse



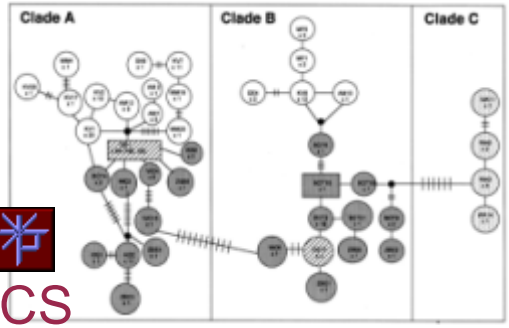
HorizStory

réseau médian



Network

réseau couvrant minimum

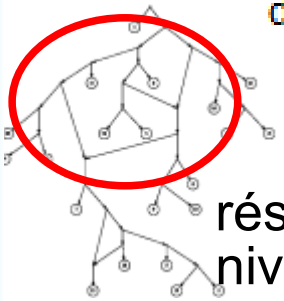


TCS

Les réseaux phylogénétiques

Réseau phylogénétique

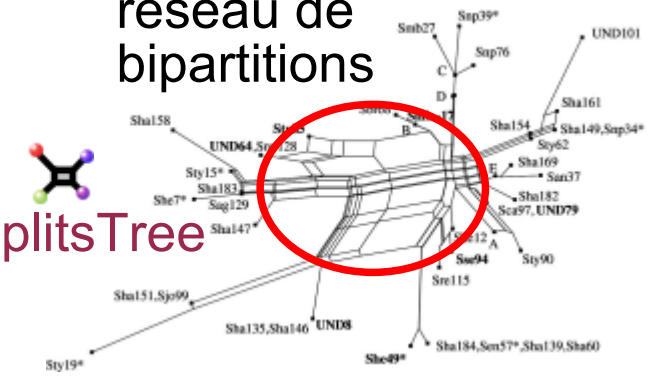
Un **réseau phylogénétique** désigne un graphe utilisé pour visualiser les relations liées à l'évolution entre des espèces ou des organismes. Il doit être employé quand interviennent des événements d'hybridations, de transferts horizontaux de gènes, ou de recombinaisons génétiques.



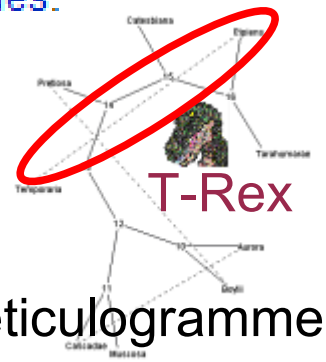
réseau de niveau 2

Level-2

réseau de bipartitions



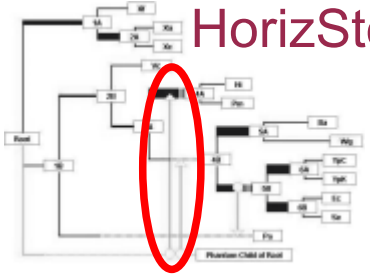
SplitsTree



T-Rex

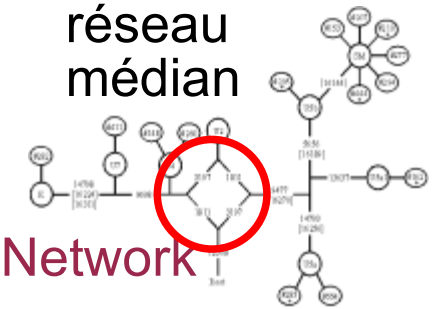
réticulogramme

diagramme de synthèse



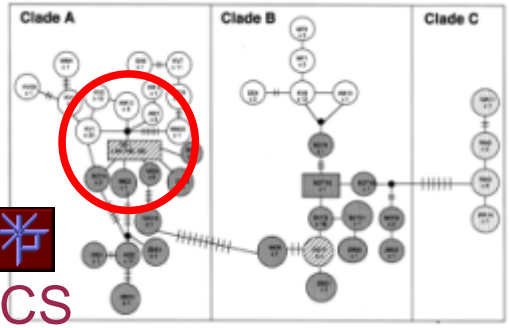
HorizStory

réseau médian



Network

réseau couvrant minimum



TCS

Réseaux abstraits ou explicites

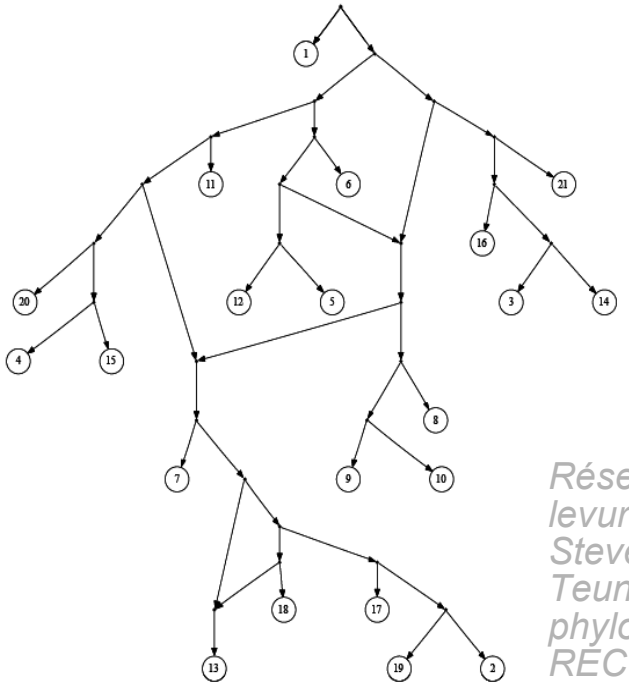
Un **réseau phylogénétique explicite** est un réseau phylogénétique dont tous les noeuds correspondent à des événements biologiques précis.

Un **réseau phylogénétique abstrait** reflète des signaux phylogénétiques sans nécessairement représenter explicitement des événements biologiques.

Réseaux abstraits ou explicites

Un **réseau phylogénétique explicite** est un réseau phylogénétique dont tous les noeuds correspondent à des événements biologiques précis.

Un **réseau phylogénétique abstrait** reflète des signaux phylogénétiques sans nécessairement représenter explicitement des événements biologiques.

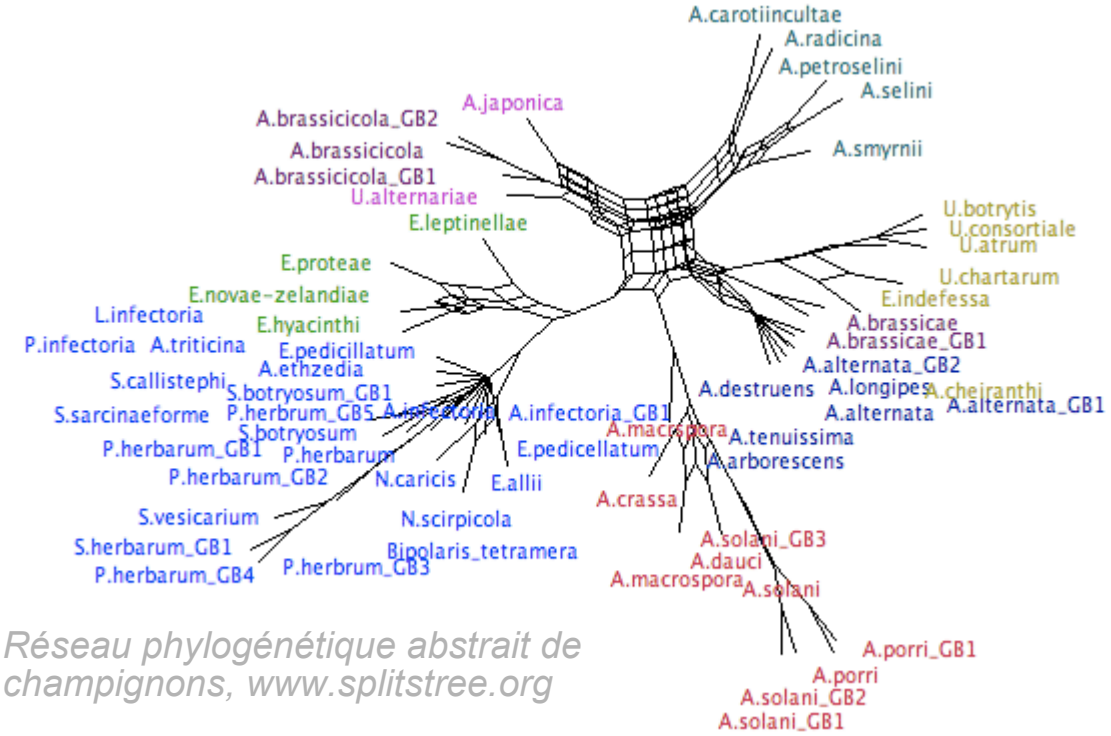


Réseau phylogénétique explicite de levures, Leo van Iersel, Judith Keijsper, Steven Kelk, Leen Stougie, Ferry Hagen, Teun Boekhout : Constructing level-2 phylogenetic networks from triplets. RECOMB'08

Réseaux abstraits ou explicites

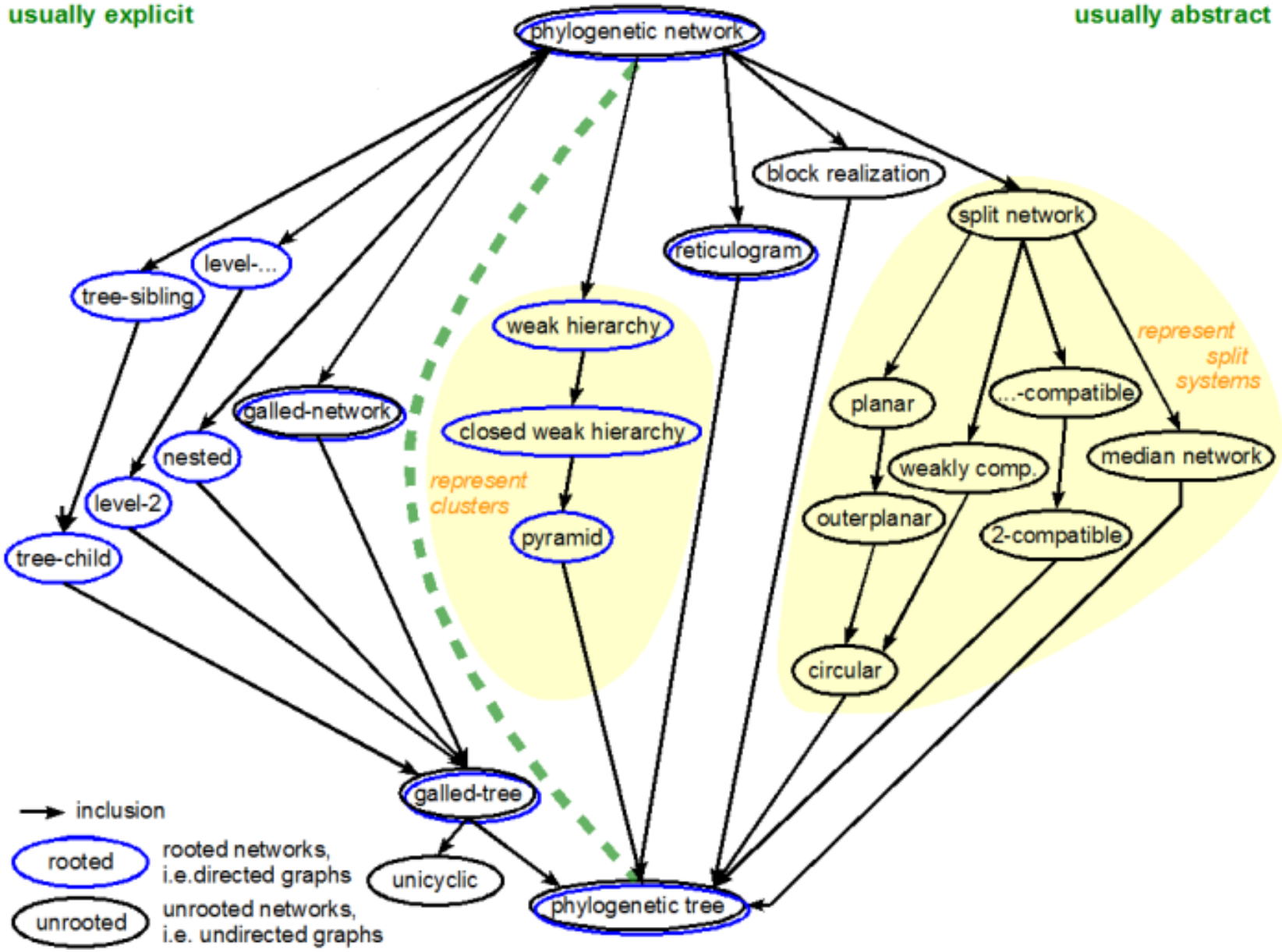
Un **réseau phylogénétique explicite** est un réseau phylogénétique dont tous les noeuds correspondent à des événements biologiques précis.

Un **réseau phylogénétique abstrait** reflète des signaux phylogénétiques sans nécessairement représenter explicitement des événements biologiques.



Réseau phylogénétique abstrait de champignons, www.splitstree.org

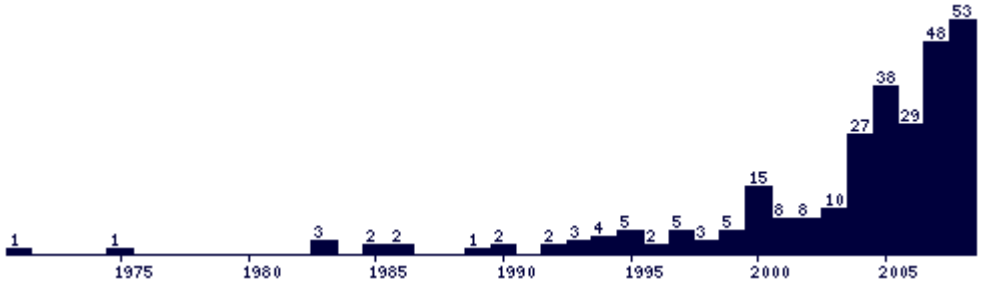
Hiérarchie de sous-classes de réseaux



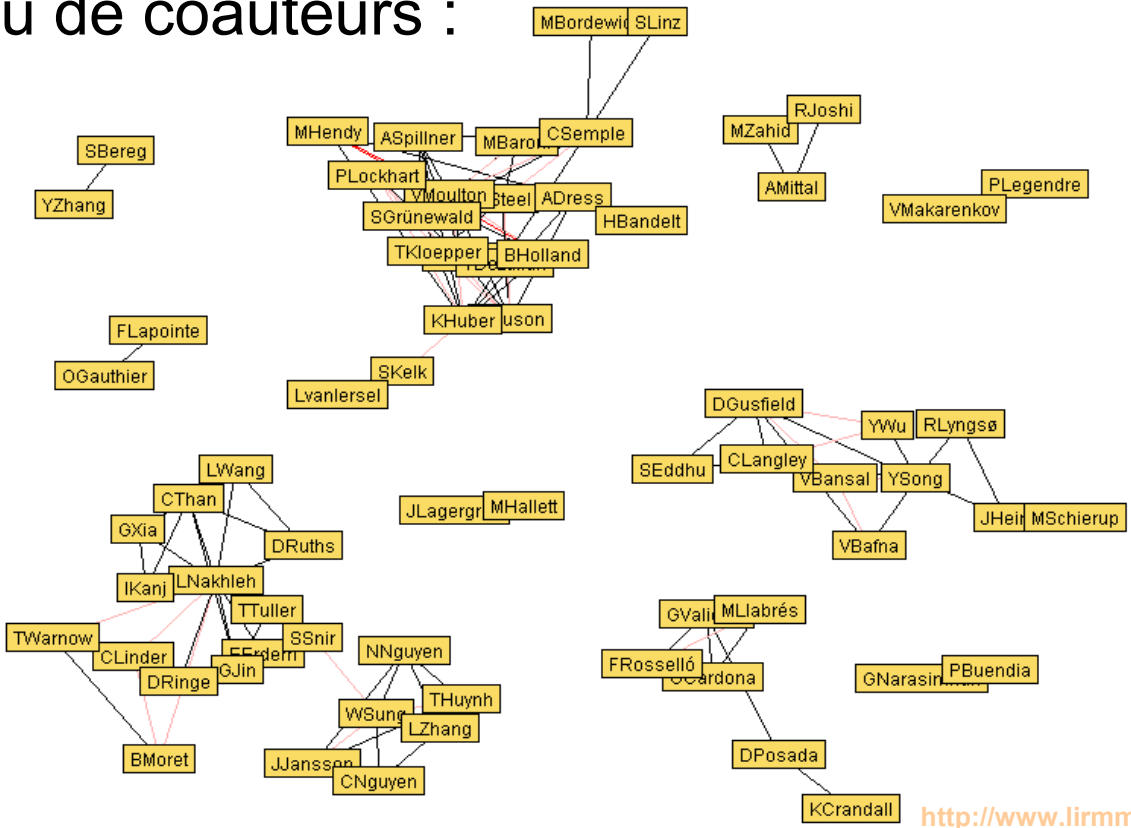
http://www.lirmm.fr/~gambette/RePhylogeneticNetworks.php

Réseaux phylogénétiques : sujet d'actualité

Publications sur les réseaux phylogénétiques :



Réseau de coauteurs :



Réseaux phylogénétiques : sujet d'actualité

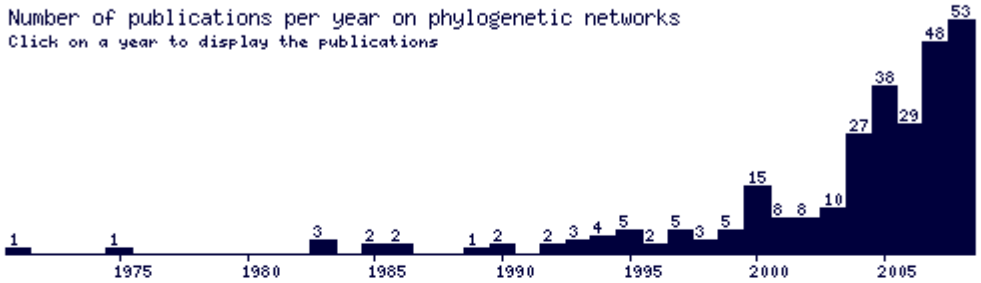
Who is Who in Phylogenetic Networks - Articles, Authors & Programs RSS

Index | Browse | Contribute! | My selection

Search: in All Go (word length ≥ 3) Login

Publications - Index (All 277 publications) Selection by: Year | Category | Keyword | Author

Selection by Year



Selection by Category

Article (Journal) (159) **InProceedings** (77) **InBook** (13)
PhdThesis (11) **Misc** (17) **Programs** (37)

Selection by Keyword

abstract-network(21) approximation(4) APX-hard(1) ARG(5) block-realization(1) bootstrap(1) bound(3) branch-and-bound(1) cactus-graph(1) characterization(3) clustering(2) coalescent(5) consensus(8) consistency(2) construction(2) distance-between-networks(19) diversity(1) enumeration(1) evaluation(22) explicit-network(23) FPT(6) from-clusters(3) from-distances(18) from-multilabeled-tree(2) from-network(5) from-quartets(5) from-rooted-trees(26) from-sequences(17) from-splits(10) from-trees(7) from-triplets(10) from-unrooted-trees(7) galled-network(2) galled-tree(26) generation(5) haplotype-network(2) haplotyping(1) heuristic(4) HMM(1) hybridization(16) labeling(3) lateral-gene-transfer(13) level-f-phylogenetic-network(7) likelihood(7) lineage-sorting(1) MASN(4) median-network(14) MedianJoining(2) minimum-number(8) minimum-spanning-network(2) mu-distance(1) NeighborNet(9) nested-network(2) netting(3) normal-network(1) NP-complete(13) optimal-realization(2) parsimony(10) perfect(5) **phylogenetic-network(151)** **phylogeny(155)** polynomial(26) Program-Arlequin(5) Program-Beagle(3) Program-Bio-PhyloNetwork(4) Program-CombineTrees(2) Program-Dendroscope(5) Program-EEEP(2) Program-GalledTree(1) Program-HapBound(1) Program-HorizStory(2) Program-HybridNumber(1) Program-LatTrans(3) Program-Level2(1) Program-Marlon(1) Program-

Réseaux phylogénétiques : sujet d'actualité

Who is Who in Phylogenetic Networks - Articles, Authors & Programs RSS

Index | Browse | Contribute! | My selection

Search: in All Go (word length ≥ 3) Login

Publications of Year << 2008 >>

<< Article (Journal) >>

1

Gabriel Cardona, Mercè Llabrés, Francesc Rosselló and Gabriel Valiente. Metrics for phylogenetic networks II: Nodal and triplets metrics. 2008. [Comment] BIBTEX

Keywords: distance between networks, phylogenetic network, phylogeny. **Note:** Submitted. [Annote]

2

Cuong Than, Derek Ruths and Luay Nakhleh. PhyloNet: A Software Package for Analyzing and Reconstructing Reticulate Evolutionary Relationships. In *BMC Bioinformatics*, Vol. 9(322), 2008. [Comment] BIBTEX

Keywords: Program PhyloNet, software. **Note:** http://dx.doi.org/10.1186/1471-2105-9-322. [Annote]

3

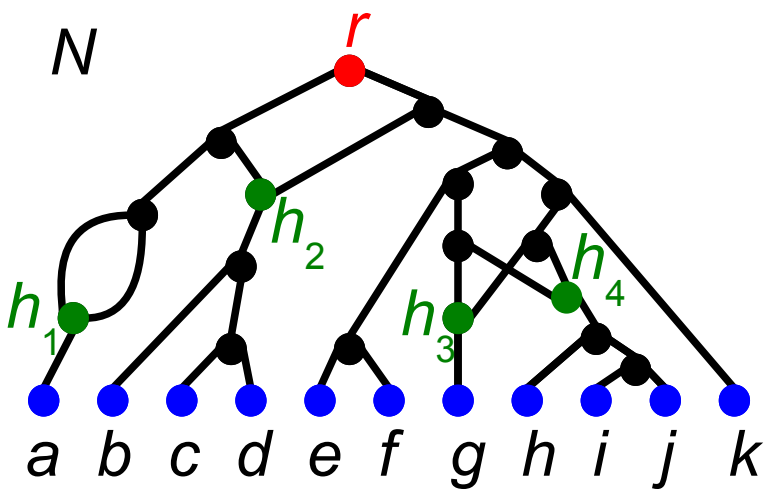
Iyad A. Kanj, Luay Nakhleh, Cuong Than and Ge Xia. Seeing the Trees and Their Branches in the Network is Hard. In *TCS*, Vol. 401:153-164, 2008. [Comment] BIBTEX

Keywords: evaluation, from network, from rooted trees, NP-complete, phylogenetic network, phylogeny.

Note: http://www.cs.rice.edu/~nakhleh/Papers/tcs08.pdf. [Annote]

Réseaux phylogénétiques de niveau *k*

Un **réseau phylogénétique de niveau *k*** sur un ensemble *X* de *n* taxons est un multigraphe orienté dans lequel :
- un sommet a degré entrant 0 et sortant 2 : la **racine**,
- tous les autres sommets ont soit :
  - degré entrant 1 et sortant 2: **sommets de spéciation**,
  - degré entrant 2 et sortant ≤ 1 : **sommets hybrides**,
  - ou degré entrant 1 et sortant 0 : **feuilles** étiquetées par *X*,
- chaque **blob** a au plus *k* sommets hybrides.

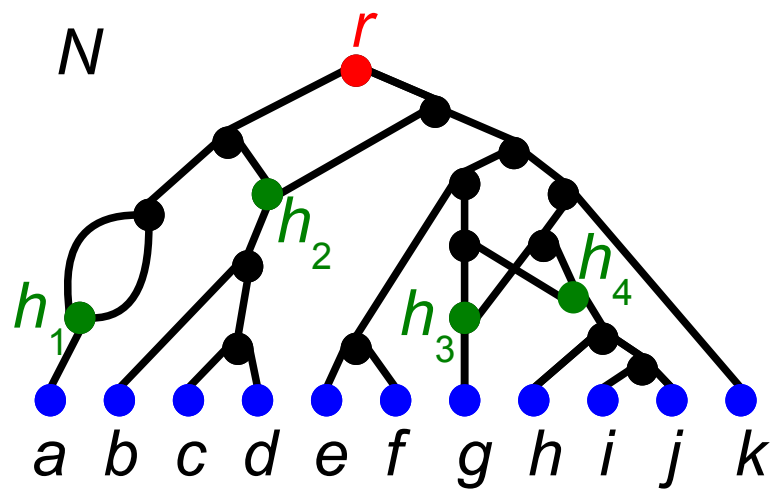


Tous les arcs sont orientés vers le bas

Avec Vincent Berry (LIRMM, MAB) et Christophe Paul (LIRMM, VAG)

Réseaux phylogénétiques de niveau *k*

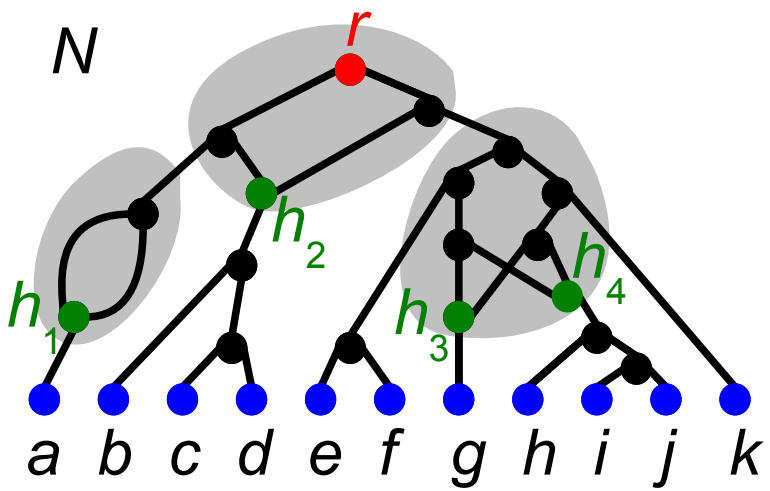
Un **réseau phylogénétique de niveau *k*** sur un ensemble *X* de *n* taxons est un multigraphe orienté dans lequel :
- un sommet a degré entrant 0 et sortant 2 : la **racine**,
- tous les autres sommets ont soit :
  - degré entrant 1 et sortant 2: **sommets de spéciation**,
  - degré entrant 2 et sortant ≤ 1 : **sommets hybrides**,
  - ou degré entrant 1 et sortant 0 : **feuilles** étiquetées par *X*,
- chaque **blob** a au plus *k* sommets hybrides.



Un **blob** est une composante **biconnexe** maximale du graphe non orienté sous-jacent, c'est à dire un sous-graphe maximal non déconnecté par la suppression d'un sommet quelconque.

Réseaux phylogénétiques de niveau *k*

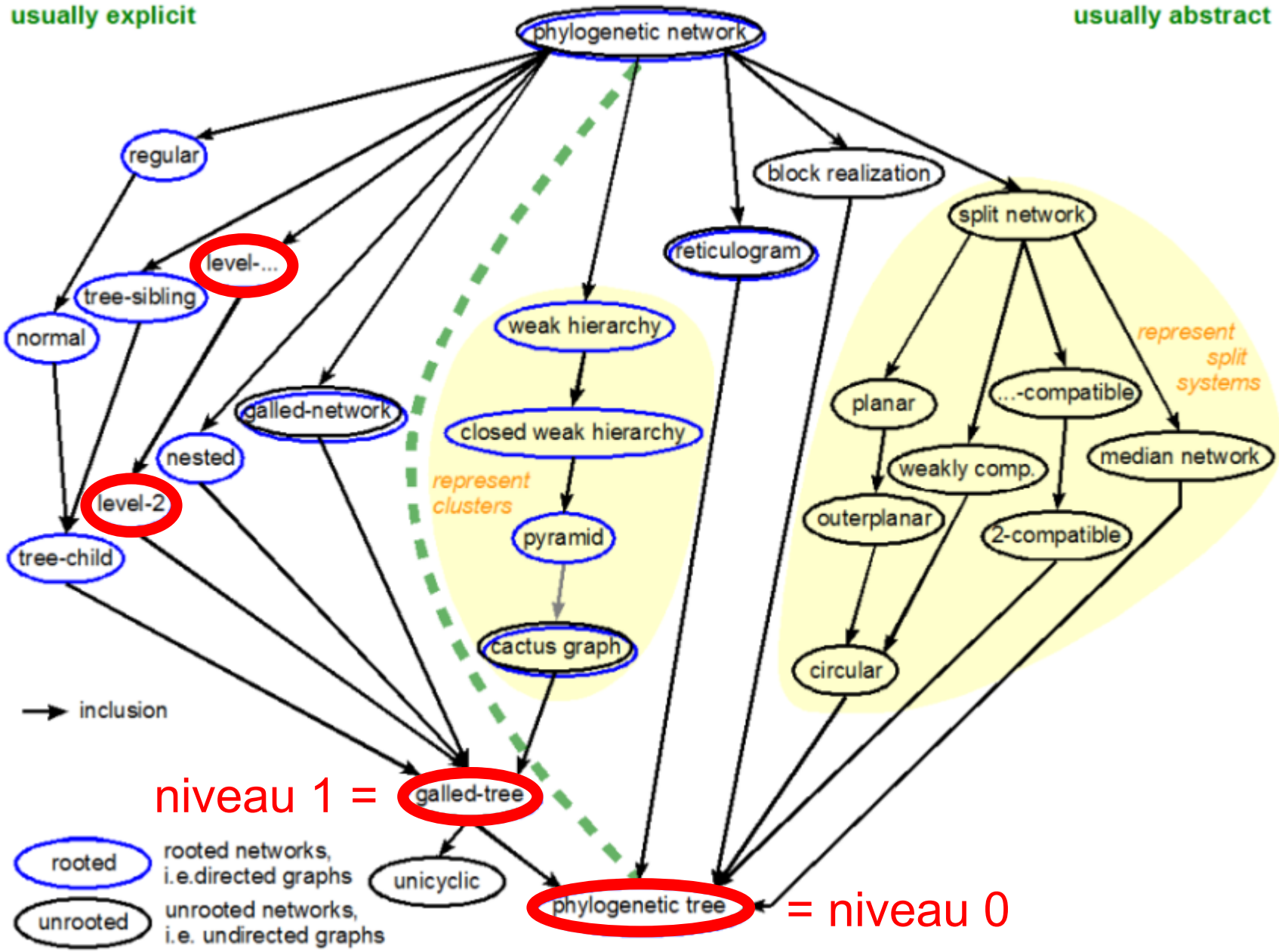
Un **réseau phylogénétique de niveau *k*** sur un ensemble *X* de *n* taxons est un multigraphe orienté dans lequel :
- un sommet a degré entrant 0 et sortant 2 : la **racine**,
- tous les autres sommets ont soit :
  - degré entrant 1 et sortant 2: **sommets de spéciation**,
  - degré entrant 2 et sortant ≤ 1 : **sommets hybrides**,
  - ou degré entrant 1 et sortant 0 : **feuilles** étiquetées par *X*,
- chaque **blob** a au plus *k* sommets hybrides.



N a niveau 2.

Un **blob** est une composante **biconnexe** maximale du graphe non orienté sous-jacent, c'est à dire un sous-graphe maximal non déconnecté par la suppression d'un sommet quelconque.

Réseaux phylogénétiques de niveau *k*



Réseaux phylogénétiques de niveau *k*

Motivation pour généraliser les “galled trees” (niveau 1) :

Table 1: Number of simulated networks falling in each class as a function of the recombination rate $\rho = 0, 1, 2, 4, 8, 16, 32$, for sample size $n = 10$.

Network class	Recombination rate 0	1	2	4	8	16	32
Regular	1,000	200	58	5	0	0	0
Tree-sibling	1,000	832	514	151	14	0	0
Tree-child	1,000	560	205	39	1	0	0
Galled-trees	1,000	440	137	21	1	0	0
Trees	1,000	139	27	1	0	0	0

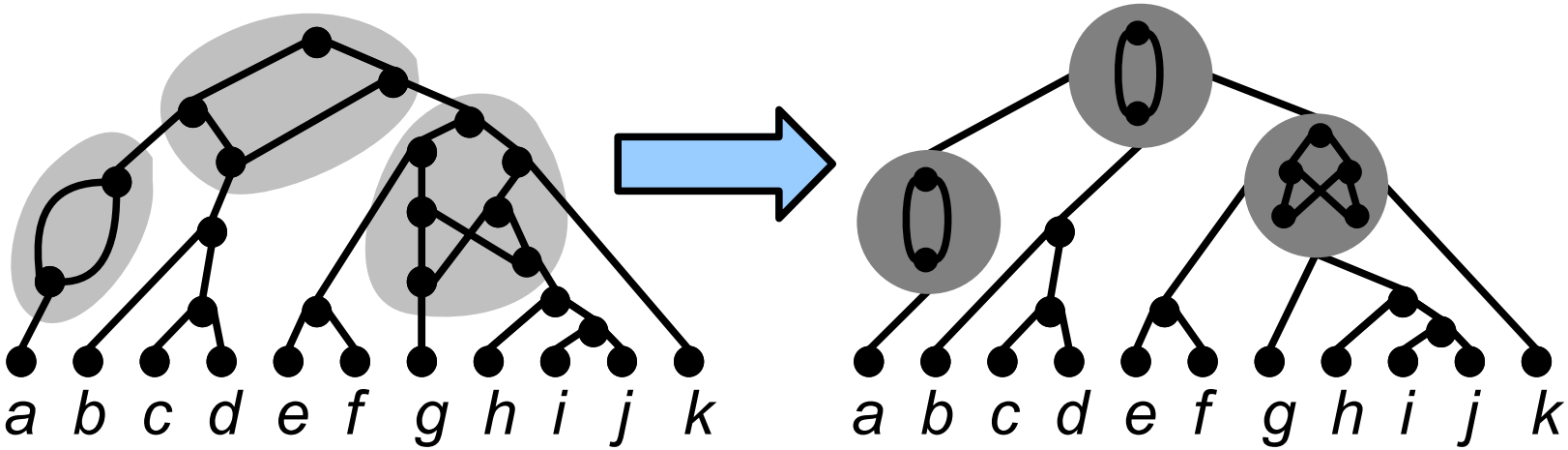
Table 2: Number of simulated networks falling in each class as a function of the recombination rate $\rho = 0, 1, 2, 4, 8, 16, 32$, for sample size $n = 50$.

Network class	Recombination rate 0	1	2	4	8	16	32
Regular	1,000	57	1	0	0	0	0
Tree-sibling	1,000	784	469	101	2	0	0
Tree-child	1,000	463	126	9	0	0	0
Galled-trees	1,000	161	5	0	0	0	0
Trees	1,000	34	0	0	0	0	0

Arenas, Valiente, Posada : Characterization of Phylogenetic Reticulate Networks based on the Coalescent with Recombination, Molecular Biology and Evolution, to appear.

Décomposition des réseaux de niveau *k*

Nous avons formalisé cette décomposition en composantes biconnexes :



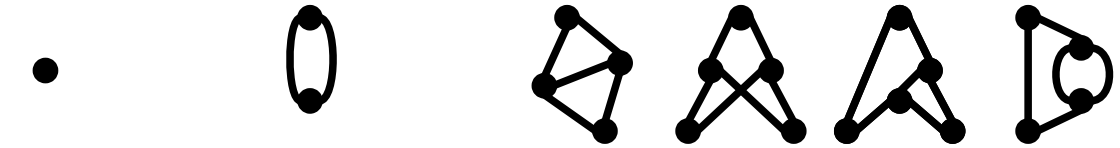
N, un réseau de niveau *k*.

N décomposé comme **arbre de générateurs**.

Générateurs introduits par van Iersel & al (Recomb 2008) pour une classe restreinte de réseaux de niveau *k*.

Générateurs de niveau *k*

Un **générateur de niveau *k*** est un réseau de niveau *k* biconnexe.



Les **côtés** d'un générateur sont :
- ses arcs
- ses sommets hybrides de degré sortant 0

$\boldsymbol{S_k}$ est l'ensemble des générateurs de niveau au plus *k*.

Décomposition des réseaux de niveau *k*

N est un réseau de niveau *k*

ssi

il existe une suite $(l_j)_{j\in[1,r]}$ de *r* emplacements
(arcs ou sommets hybrides de degré sortant 0)
et une suite $(G_j)_{j\in[0,r]}$ de générateurs de niveau au plus *k*, telles que :

- $N = \text{Attach}_k(l_r, G_r, \text{Attach}_k(\ldots \text{Attach}_k(l_2, G_2, \text{Attach}_k(l_1, G_1, G_0))\ldots))$,
- or $N = \text{Attach}_k(l_r, G_r, \text{Attach}_k(\ldots \text{Attach}_k(l_2, G_2, \text{SplitRoot}_k(G_1, G_0))\ldots))$.

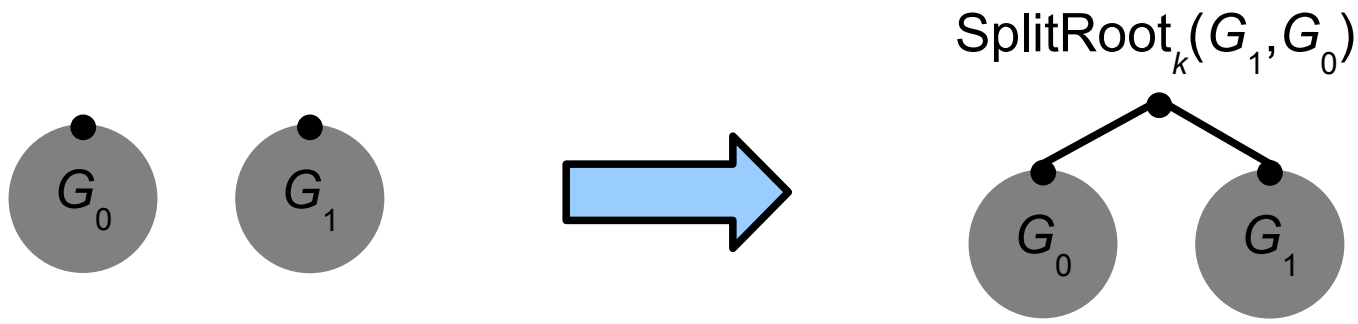
Décomposition des réseaux de niveau *k*

N est un réseau de niveau *k*

ssi

il existe une suite $(I_j)_{j\in[1,r]}$ de *r* emplacements
(arcs ou sommets hybrides de degré sortant 0)
et une suite $(G_j)_{j\in[0,r]}$ de générateurs de niveau au plus *k*, telles que :

- $N = \mathrm{Attach}_k(I_r,G_r,\mathrm{Attach}_k(\ldots \mathrm{Attach}_k(I_2,G_2,\mathrm{Attach}_k(I_1,G_1,G_0))\ldots))$,
- or $N = \mathrm{Attach}_k(I_r,G_r,\mathrm{Attach}_k(\ldots \mathrm{Attach}_k(I_2,G_2,\mathrm{SplitRoot}_k(G_1,G_0))\ldots))$.



Décomposition des réseaux de niveau *k*

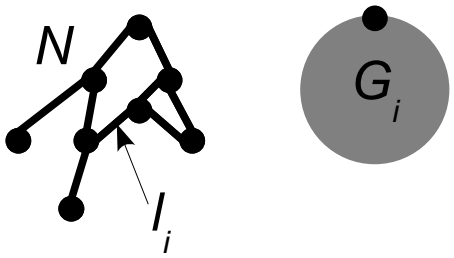
N est un réseau de niveau *k*

ssi

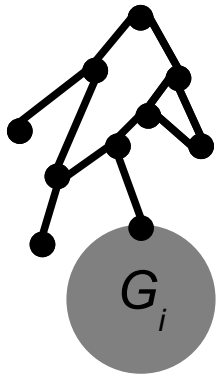
il existe une suite $(l_j)_{j\in[1,r]}$ de *r* emplacements
(arcs ou sommets hybrides de degré sortant 0)
et une suite $(G_j)_{j\in[0,r]}$ de générateurs de niveau au plus *k*, telles que :

- $N = \text{Attach}_k(l_r, G_r, \text{Attach}_k(\ldots \text{Attach}_k(l_2, G_2, \text{Attach}_k(l_1, G_1, G_0))\ldots))$,
- or $N = \text{Attach}_k(l_r, G_r, \text{Attach}_k(\ldots \text{Attach}_k(l_2, G_2, \text{SplitRoot}_k(G_1, G_0))\ldots))$.

l_i est un arc de *N*



$\text{Attach}_k(l_i, G_i, N)$



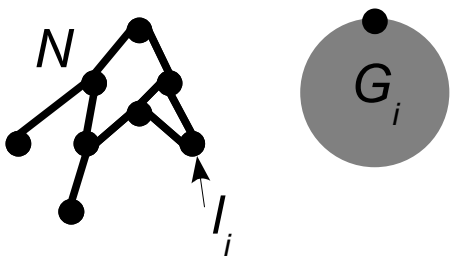
Décomposition des réseaux de niveau *k*

N est un réseau de niveau *k*

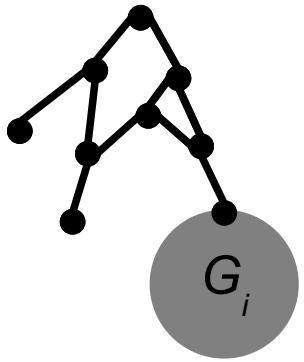
ssi

il existe une suite $(I_j)_{j\in[1,r]}$ de *r* emplacements
(arcs ou sommets hybrides de degré sortant 0)
et une suite $(G_j)_{j\in[0,r]}$ de générateurs de niveau au plus *k*, telles que :
- $N = \mathrm{Attach}_k(I_r,G_r,\mathrm{Attach}_k(\dots \mathrm{Attach}_k(I_2,G_2,\mathrm{Attach}_k(I_1,G_1,G_0))\dots))$,
- or $N = \mathrm{Attach}_k(I_r,G_r,\mathrm{Attach}_k(\dots \mathrm{Attach}_k(I_2,G_2,\mathrm{SplitRoot}_k(G_1,G_0))\dots))$.

I_i est un sommet hybride de *N*



$\mathrm{Attach}_k(I_i,G_i,N)$



Décomposition des réseaux de niveau *k*

N est un réseau de niveau *k*

ssi

il existe une suite $(I_j)_{j\in[1,r]}$ de *r* emplacements
(arcs ou sommets hybrides de degré sortant 0)
et une suite $(G_j)_{j\in[0,r]}$ de générateurs de niveau au plus *k*, telles que :

- $N = \mathrm{Attach}_k(I_r, G_r, \mathrm{Attach}_k(\ldots \mathrm{Attach}_k(I_2, G_2, \mathrm{Attach}_k(I_1, G_1, G_0))\ldots))$,
- or $N = \mathrm{Attach}_k(I_r, G_r, \mathrm{Attach}_k(\ldots \mathrm{Attach}_k(I_2, G_2, \mathrm{SplitRoot}_k(G_1, G_0))\ldots))$.

Attention, cette décomposition n'est **pas unique** !

Malgré tout, applications possibles :
- génération exhaustive
- énumération des réseaux de niveau *k*
- la décomposition en composantes biconnexes est une décomposition arborescente ... pour les graphes **non orientés**.

Décomposition des réseaux de niveau *k*

N est un réseau de niveau *k*

ssi

il existe une suite $(l_j)_{j\in[1,r]}$ de *r* emplacements
(arcs ou sommets hybrides de degré sortant 0)
et une suite $(G_j)_{j\in[0,r]}$ de générateurs de niveau au plus *k*, telles que :
- $N = \mathrm{Attach}_k(l_r, G_r, \mathrm{Attach}_k(\ldots \mathrm{Attach}_k(l_2, G_2, \mathrm{Attach}_k(l_1, G_1, G_0))\ldots))$,
- or $N = \mathrm{Attach}_k(l_r, G_r, \mathrm{Attach}_k(\ldots \mathrm{Attach}_k(l_2, G_2, \mathrm{SplitRoot}_k(G_1, G_0))\ldots))$.

Attention, cette décomposition n'est **pas unique** !

Malgré tout, applications possibles :
- génération exhaustive
- énumération des réseaux de niveau *k*
- la décomposition en composantes biconnexes est une décomposition arborescente ... pour les graphes **non orientés**.

f : {générateurs de niveau k+1} → {multigraphes biconnexes 3-réguliers sans boucle à 2k noeuds}

bien définie et **surjective** → borne inférieure pour g_k ?

Construction des générateurs

Van Iersel & al construisent les 4 générateurs de niveau 2 par une analyse de cas, généralisée par Kelk en un algorithme exponentiel pour trouver les 65 générateurs de niveau 3.

Greetings from The On-Line Encyclopedia of Integer Sequences!

1,4,65 Search Hints

Search: **1, 4, 65**

Displaying 1-2 of 2 results found. page 1

Format: long | short | internal | text Sort: relevance | references | number Highlight: on | off

A041119 Denominators of continued fraction convergents to sqrt(68). +20 2

1, 4, 65, 264, 4289, 17420, 283009, 1149456, 18674305, 75846676, 1232221121, 5004731160, 81307919681, 330236409884, 5365090477825, 21790598321184, 354014663616769, 1437849252788260, 23359602708228929 (list; graph; listen)

OFFSET 0,2

CROSSREFS Cf. A041118.
Sequence in context: A138835 A119601 A058438 this_sequence A015475 A025585 A048828
Adjacent sequences: A041116 A041117 A041118 this_sequence A041120 A041121 A041122

KEYWORD nonn,cofr,easy

AUTHOR njas

A015475 q-Fibonacci numbers for q=4. +20 1

0, 1, 4, 65, 4164, 1066049, 1091638340, 4471351706689, 73258627454030916, 480107741329872181766 5, 1258573637505038759624004676, 1319710110525284599824799048959041 (list; graph; listen)

OFFSET 0,3

FORMULA a(n) = 4^(n-1) a(n-1) + a(n-2).

CROSSREFS Sequence in context: A119601 A058438 A041119 this_sequence A025585 A048828

Construction des générateurs

Van Iersel & al construisent les 4 générateurs de niveau 2 par une analyse de cas, généralisée par Kelk en un algorithme exponentiel pour trouver les 65 générateurs de niveau 3.

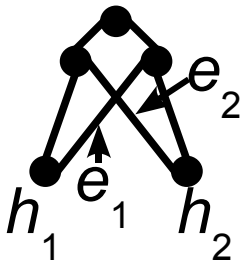
Nous donnons des **règles pour construire les générateurs de niveau *k*+1 à partir des générateurs de niveau *k*.**

Construction des générateurs

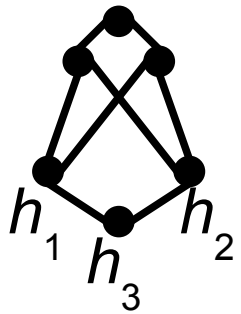
Nous donnons des **règles pour construire les générateurs de niveau *k*+1 à partir des générateurs de niveau *k*.**

N est un générateur de niveau *k*.
$R_1(N,X,Y)$ est obtenu en :

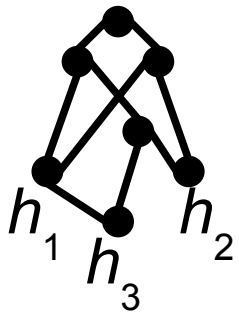
- choisissant deux côtés *X* et *Y* de *N*, tels que si *X* = *Y* alors *X* n'est pas un noeud hybride (i.e. c'est un arc),
- accrochant un nouveau noeud hybride sous *X* et *Y*.



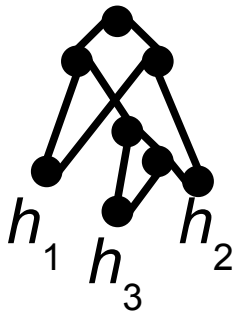
N



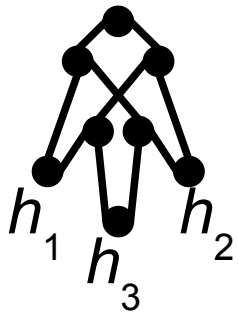
$R_1(N,h_1,h_2)$



$R_1(N,h_1,e_2)$



$R_1(N,e_2,e_2)$



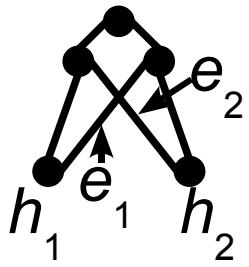
$R_1(N,e_1,e_2)$

Construction des générateurs

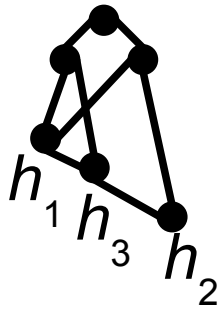
Nous donnons des **règles pour construire les générateurs de niveau *k*+1 à partir des générateurs de niveau *k*.**

N est un générateur de niveau *k*.
$R_2(N,X,Y)$ est obtenu en :

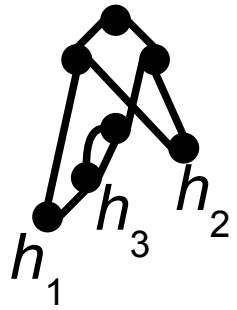
- choisissant un côté *X* de *N* et un arc *Y* de *N*, tel que *X* n'est pas sous *Y*.
- ajoutant un arc de *X* vers *Y* (ce qui crée un nouveau noeud de réticulation à l'intérieur de l'arc *Y*).



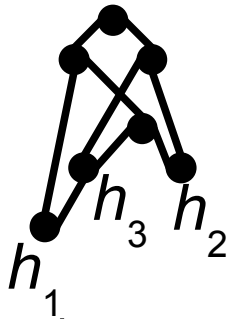
N



$R_2(N,h_1,e_2)$



$R_2(N,e_1,e_1)$



$R_2(N,e_2,e_1)$

Construction des générateurs

Nous donnons des **règles pour construire les générateurs de niveau *k*+1 à partir des générateurs de niveau *k*.**

Pour tout générateur *N* de niveau *k*, et deux côtés *X* et *Y* de *N*, si $R_1(N,X,Y)$ (resp. $R_2(N,X,Y)$) existe, alors $\boldsymbol{R_1(N,X,Y)}$ (resp. $\boldsymbol{R_2(N,X,Y)}$) est **générateur de niveau k+1**.

Pour tout générateur *N* de niveau *k*+1, il **existe un générateur N_0 de niveau *k***, et deux côtés *X* et *Y* de N_0 tels que $\boldsymbol{N = R_1(N_0,X,Y)}$ **ou** $\boldsymbol{N = R_2(N_0,X,Y)}$.

Construction des générateurs

Nous donnons des **règles pour construire les générateurs de niveau *k*+1 à partir des générateurs de niveau *k*.**

Pour tout générateur *N* de niveau *k*, et deux côtés *X* et *Y* de *N*, si $R_1(N,X,Y)$ (resp. $R_2(N,X,Y)$) existe, alors $\boldsymbol{R_1(N,X,Y)}$ (resp. $\boldsymbol{R_2(N,X,Y)}$) est **générateur de niveau k+1**.

Pour tout générateur *N* de niveau *k*+1, il **existe un générateur N_0 de niveau *k***, et deux côtés *X* et *Y* de N_0 tels que $\boldsymbol{N = R_1(N_0,X,Y)}$ **ou** $\boldsymbol{N = R_2(N_0,X,Y)}$.

Corollaire :

g_k : nombre de générateurs de niveau *k*.

$g_{k+1} \leq 50\ k^2\ g_k$

Construction des générateurs

Pour tout générateur *N* de niveau *k*, et deux côtés *X* et *Y* de *N*, si $R_1(N,X,Y)$ (resp. $R_2(N,X,Y)$) existe, alors $\boldsymbol{R_1(N,X,Y)}$ (resp. $\boldsymbol{R_2(N,X,Y)}$) est **générateur de niveau k+1**.

Pour tout générateur *N* de niveau *k*+1, il **existe un générateur N_0 de niveau *k***, et deux côtés *X* et *Y* de N_0 tels que $\boldsymbol{N = R_1(N_0,X,Y)}$ **ou** $\boldsymbol{N = R_2(N_0,X,Y)}$.

Corollaire :

g_k: nombre de générateurs de niveau *k*.

$g_{k+1} \leq 50\ k^2\ g_k$

Problème !

Certains des générateurs de niveau *k*+1 ainsi obtenus depuis les générateurs de niveau *k* sont isomorphes !

Isomorphisme des générateurs

Problème !
Certains des générateurs de niveau k+1 ainsi obtenus depuis les générateurs de niveau k sont isomorphes !

Isomorphisme de graphes de degré borné : polynomial. (Luks, FOCS 1980)

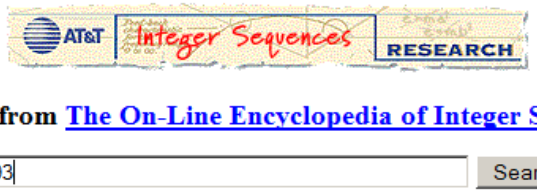
Algorithme implémentable ?
→ algorithme exponentiel qui convient pour le niveau 4 : pour trouver l'isomorphisme entre deux digraphes, les parcourir en parallèle depuis la racine et identifier leurs sommets : $O(n2^{n-h})$

Isomorphisme des générateurs

Problème !
Certains des générateurs de niveau k+1 ainsi obtenus depuis les générateurs de niveau k sont isomorphes !

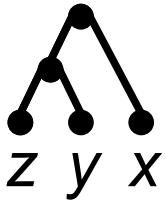
Isomorphisme de graphes de degré borné : polynomial. (Luks, FOCS 1980)

Algorithme implémentable ?
→ algorithme exponentiel qui convient pour le niveau 4 : pour trouver l'isomorphisme entre deux digraphes, les parcourir en parallèle depuis la racine et identifier leurs sommets : $O(n2^{n-h})$
→ $g_4 = 1993$
→ $g_5 > 71000$



Reconstruction depuis les triplets

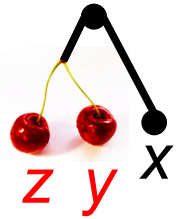
Un **triplet** *x*|*yz* est un arbre phylogénétique enraciné sur 3 taxons {*x*,*y*,*z*} tel que *x*, et le père de *y* et *z*, sont des fils de la racine.



La restriction d'un ensemble T de triplets à *X* est :
$T_{|X}$ = { *t* ϵ T | *t* est un triplet sur les taxons *x*,*y*,*z* ϵ *X* }

Reconstruction depuis les triplets

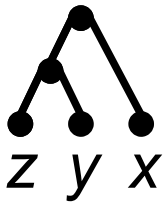
Un **triplet** $x|yz$ est un arbre phylogénétique enraciné sur 3 taxons $\{x,y,z\}$ tel que x, et le père de y et z, sont des fils de la racine.



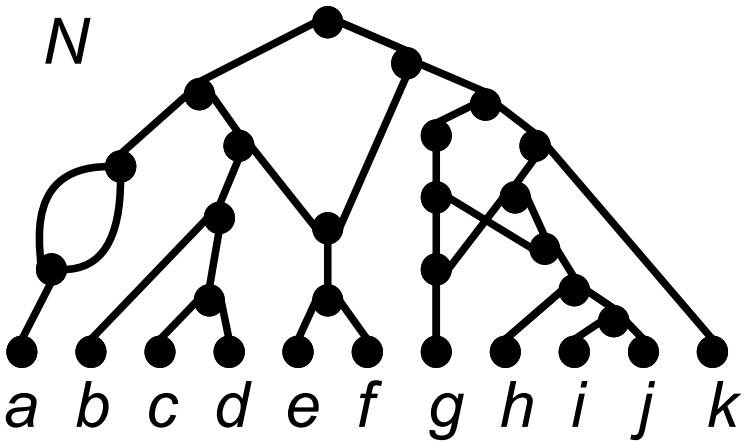
$\{y,z\}$ est appelé la **cerise** de $x|yz$.

Reconstruction depuis les triplets

Un **triplet** *x*|*yz* est un arbre phylogénétique enraciné sur 3 taxons {*x*,*y*,*z*} tel que *x*, et le père de *y* et *z*, sont des fils de la racine.

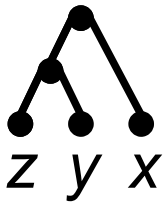


Un triplet *x*|*yz* est **compatible** avec un réseau phylogénétique *N* de niveau *k* si:
- *N* contient deux noeuds *u* et *v*
- et des chemins intérieurement disjoints deux à deux :
  - de *u* à *y*,
  - de *u* à *z*,
  - de v à *u*,
  - et de *v* à *x.*



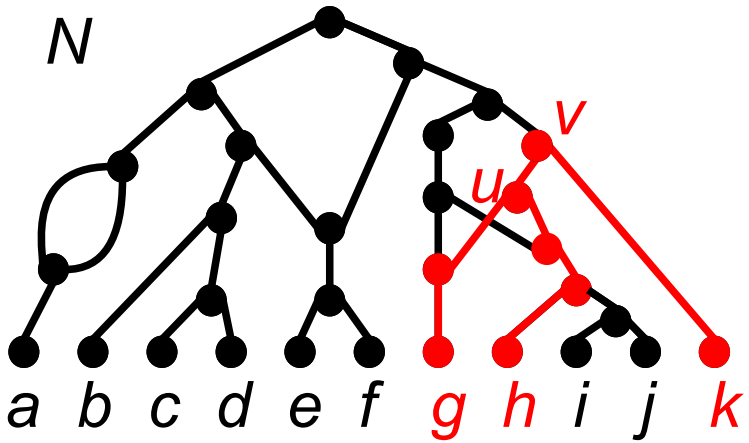
Reconstruction depuis les triplets

Un **triplet** *x*|*yz* est un arbre phylogénétique enraciné sur 3 taxons {*x*,*y*,*z*} tel que *x*, et le père de *y* et *z*, sont des fils de la racine.



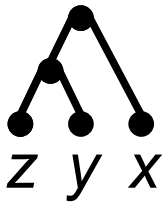
Un triplet *x*|*yz* est **compatible** avec un réseau phylogénétique *N* de niveau *k* si:
- *N* contient deux noeuds *u* et *v*
- et des chemins intérieurement disjoints deux à deux :
  - de *u* à *y*,
  - de *u* à *z*,
  - de v à *u*,
  - et de *v* à *x.*

k|*gh* **compatible** avec *N*.



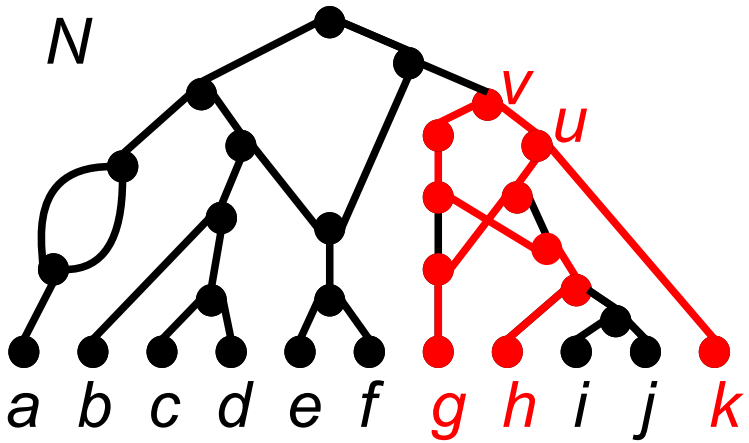
Reconstruction depuis les triplets

Un **triplet** *x*|*yz* est un arbre phylogénétique enraciné sur 3 taxons {*x*,*y*,*z*} tel que *x*, et le père de *y* et *z*, sont des fils de la racine.



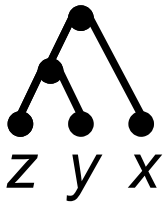
Un triplet *x*|*yz* est **compatible** avec un réseau phylogénétique *N* de niveau *k* si:
- *N* contient deux noeuds *u* et *v*
- et des chemins intérieurement disjoints deux à deux :
  - de *u* à *y*,
  - de *u* à *z*,
  - de v à *u*,
  - et de *v* à *x.*

h|*gk* **compatible** avec *N*.



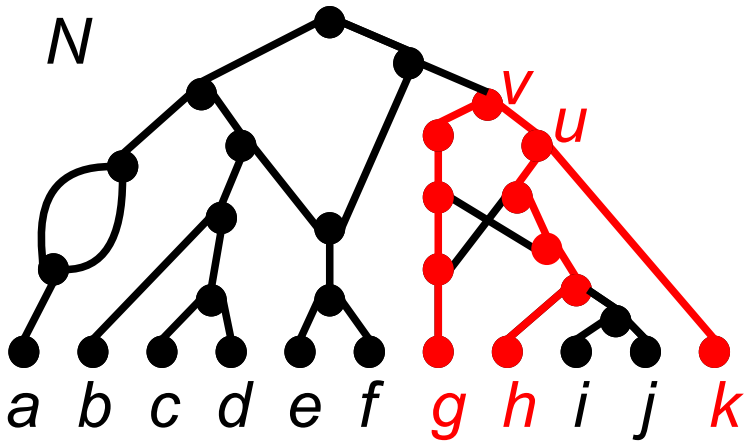
Reconstruction depuis les triplets

Un **triplet** *x*|*yz* est un arbre phylogénétique enraciné sur 3 taxons {*x*,*y*,*z*} tel que *x*, et le père de *y* et *z*, sont des fils de la racine.



Un triplet *x*|*yz* est **compatible** avec un réseau phylogénétique *N* de niveau *k* si:
- *N* contient deux noeuds *u* et *v*
- et des chemins intérieurement disjoints deux à deux :
  - de *u* à *y*,
  - de *u* à *z*,
  - de v à *u*,
  - et de *v* à *x.*

g|*hk* **compatible** avec *N.*



Reconstruction depuis les triplets

Pourquoi utiliser des triplets en entrée d'un algorithme de reconstruction d'arbre ou réseau phylogénétique ?

Sous un modèle coalescent, un arbre reconstruit sur plus de trois taxons sera probablement faux à cause des différences entre l'arbre du gène considéré et l'arbre des espèces.

(Degnan & Rosenberg, 2006)

Qui utilise des triplets en entrée d'un algorithme de reconstruction phylogénétique ?
- ???
- les gens qui utilisent des arbres en entrée ?
- les gens le feront quand ce sera implémenté dans des logiciels simples d'utilisation ?

Reconstruction depuis les triplets

L'ensemble $\mathcal{T}(N)$ de **tous les triplets compatibles** avec un réseau N de niveau k peut être construit en $O(|\mathcal{T}(N)|) = O(n^3)$

(programmation dynamique, Byrka, Gawrychowski, Huber, Kelk, 2008)

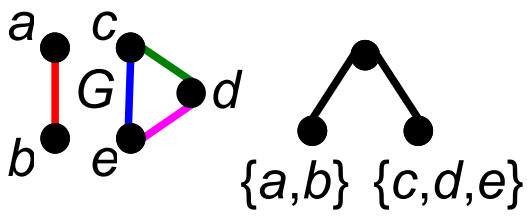
Un **arbre** T compatible avec un ensemble $\mathcal{T}$ de triplets peut être **reconstruit** en $O(|\mathcal{T}|+n^2 \log n)$.

(BUILD, algorithme top-down basé sur une structure de graphe, Aho, Sagiv, Szymanski, Ullman, 1981
implémentation efficace par Henzinger, King, Warnow, Soda 1996)

BUILD(X) :
- soit le graphe G tel que :
  - sommets = taxons,
  - arête $\{x,y\}$ si $\exists t \in \mathcal{T}_{|X}$ tel que $\{x,y\}$ est une cerise de t.
- les sommets dans différentes composantes connexes de G sont dans des sous-arbres différents.
- pour chaque composante connexe C, appliquer BUILD(C).

$X=\{a,b,c,d,e\}$

$\mathcal{T}_{|X} = \{a|cd, a|ce, c|ab, d|ab, c|de\}$



Reconstruction depuis les triplets

L'ensemble $\mathcal{T}(N)$ de **tous les triplets compatibles** avec un réseau N de niveau k peut être construit en $O(|\mathcal{T}(N)|) = O(n^3)$

(programmation dynamique, Byrka, Gawrychowski, Huber, Kelk, 2008)

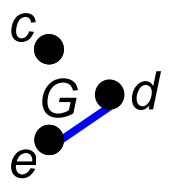
Un **arbre** T compatible avec un ensemble $\mathcal{T}$ de triplets peut être **reconstruit** en $O(|\mathcal{T}|+n^2 \log n)$.

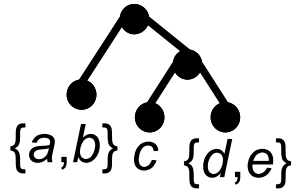
(BUILD, algorithme top-down basé sur une structure de graphe,
Aho, Sagiv, Szymanski, Ullman, 1981
implémentation efficace par Henzinger, King, Warnow, Soda 1996)

BUILD(X) :
- soit le graphe G tel que :
  - sommets = taxons,
  - arête $\{x,y\}$ si $\exists t \in \mathcal{T}_{|X}$ tel que $\{x,y\}$ est une cerise de t.
- les sommets dans différentes composantes connexes de G sont dans des sous-arbres différents.
- pour chaque composante connexe C, appliquer BUILD(C).

$X=\{c,d,e\}$

$\mathcal{T}_{|X} = \{c|de\}$





Reconstruction depuis les triplets

L'ensemble $\mathcal{T}(N)$ de **tous les triplets compatibles** avec un réseau N de niveau k peut être construit en $O(|\mathcal{T}(N)|) = O(n^3)$

(programmation dynamique, Byrka, Gawrychowski, Huber, Kelk, 2008)

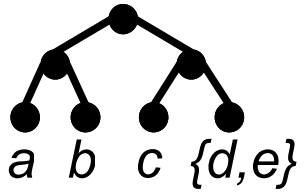
Un **arbre** T compatible avec un ensemble $\mathcal{T}$ de triplets peut être **reconstruit** en $O(|\mathcal{T}|+n^2 \log n)$.

(BUILD, algorithme top-down basé sur une structure de graphe,
Aho, Sagiv, Szymanski, Ullman, 1981
implémentation efficace par Henzinger, King, Warnow, Soda 1996)

BUILD(X) :
- soit le graphe G tel que :
  - sommets = taxons,
  - arête $\{x,y\}$ si $\exists t \in \mathcal{T}_{|X}$ tel que $\{x,y\}$ est une cerise de t.
- les sommets dans différentes composantes connexes de G sont dans des sous-arbres différents.
- pour chaque composante connexe C, appliquer BUILD(C).

$X=\{a,b\}$

$\mathcal{T}_{|X} = \{\}$



Reconstruction depuis les triplets

L'ensemble $\mathcal{T}(N)$ de **tous les triplets compatibles** avec un réseau N de niveau k peut être construit en $O(|\mathcal{T}(N)|) = O(n^3)$

(programmation dynamique, Byrka, Gawrychowski, Huber, Kelk, 2008)

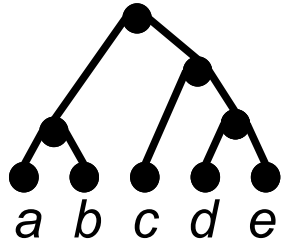
Un **arbre** T compatible avec un ensemble $\mathcal{T}$ de triplets peut être **reconstruit** en $O(|\mathcal{T}|+n^2 \log n)$.

(BUILD, algorithme top-down basé sur une structure de graphe,
Aho, Sagiv, Szymanski, Ullman, 1981
implémentation efficace par Henzinger, King, Warnow, Soda 1996)

BUILD(X) :
- soit le graphe G tel que :
  - sommets = taxons,
  - arête $\{x,y\}$ si $\exists t \in \mathcal{T}_{|X}$ tel que $\{x,y\}$ est une cerise de t.
- les sommets dans différentes composantes connexes de G sont dans des sous-arbres différents.
- pour chaque composante connexe C, appliquer BUILD(C).

$X=\{d,e\}$

$\mathcal{T}_{|X} = \{\}$



Reconstruction depuis les triplets

Le problème de **reconstruction d'un réseau de niveau 1** compatible avec un ensemble de triplets est **NP-complet**.

(Jansson, Nguyen, Sung, 2004)

Un **réseau de niveau 1** compatible avec un ensemble **dense** de triplets peut être **reconstruit** en $O(|\mathcal{T}(N)|) = O(n^3)$.
dense = sur chaque ensemble de 3 feuilles, au moins 1 triplet existe dans $\mathcal{T}$.

(Jansson, Nguyen, Sung, 2004)

Un **réseau de niveau 2** compatible avec un ensemble **dense** de triplets peut être **reconstruit** en $O(n^8)$.

(van Iersel & al, Recomb 2008)

Algorithme basé sur une décomposition de l'ensemble des triplets en **SN-ensembles**, qui correspondent à **une décomposition du réseau**.

L'approche par obstructions

Idée :
- réduire les **conflits globaux** dans l'ensemble de triplets à des **conflits locaux**,
- si aucun conflit local, **reconstruire la configuration locale** du réseau.

L'approche par obstructions

Idée :
- réduire les **conflits globaux** dans l'ensemble de triplets à des **conflits locaux**,
- si aucun conflit local, **reconstruire la configuration locale** du réseau.

Conditions nécessaires :
- assez d'information sur l'ensemble de triplets pour trouver les conflits
 → ensemble **dense**.
- assez d'information pour reconstruire la configuration locale
 → ensemble **extrêmement dense**, i.e. aucun triplet n'est manquant (données = $\mathcal{T}(N)$).

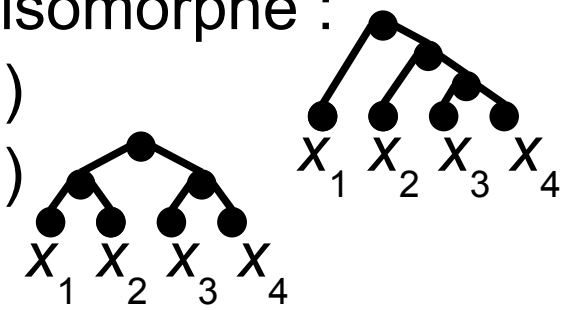
Caractérisation des arbres par les triplets

Un ensemble dense $\mathcal{T}$ de triplets est compatible avec un arbre T

ssi

aucun ensemble de 3 feuilles n'est dans deux triplets de $\mathcal{T}$, et tout ensemble de triplets sur 4 feuilles est isomorphe :

- soit à $\{x_1|x_2x_3, x_1|x_2x_4, x_1|x_3x_4, x_2|x_3x_4\}$ (cas 1)
- soit à $\{x_1|x_3x_4, x_2|x_3x_4, x_3|x_1x_2, x_4|x_1x_2\}$ (cas 2)



ssi

$\mathcal{T}$ ne contient aucun ensemble de triplets isomorphe à une des quatre obstructions suivantes :
$\{a|bc, c|ab\}$, $\{a|bc, c|bd, d|ab\}$, $\{a|bc, c|bd, d|ac\}$, $\{a|bc, a|bd, d|ac\}$.

Des caractérisations similaires ont été trouvées indépendamment par Dress (1997), Guillemot & Berry (2007).

Conséquences de la caractérisation

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

Les **algorithmes certifiants** renvoient, pour tout résultat, un **certificat** facilement vérifiable assurant que le résultat est **correct**.

Le certificat que nous fournissons est :
- l'arbre, s'il peut être reconstruit,

- une obstruction sur quatre triplets, sinon.

Conséquences de la caractérisation

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

Les **algorithmes certifiants** renvoient, pour tout résultat, un **certificat** facilement vérifiable assurant que le résultat est **correct**.

Le certificat que nous fournissons est :
- l'arbre, s'il peut être reconstruit,
  → vérifier que tous les triplets fournis en entrée sont compatible est facile
- une obstruction sur quatre triplets, sinon.
  → vérifier qu'elle est isomorphe à une des quatre obstructions est facile

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur *X* :

- Choisir un triplet *a*|*bc* dans $T_{|X}$

- Pour toute feuille *x*, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer *x* parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

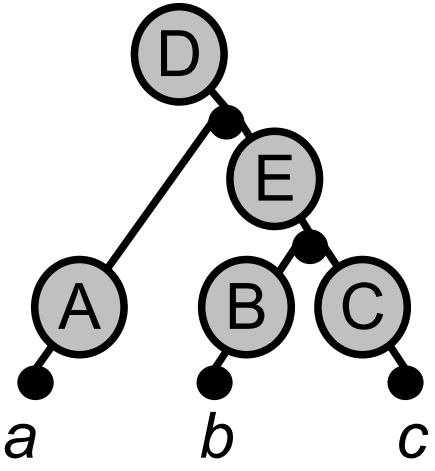
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



A:{$a|bc$,$b|ax$,$c|ax$,$x|bc$}
B:{$a|bc$,$a|bx$,$a|cx$,$c|bx$}
C:{$a|bc$,$a|bx$,$a|cx$,$b|cx$}
D:{$a|bc$,$x|ab$,$x|ac$,$x|bc$}
E:{$a|bc$,$a|bx$,$a|cx$,$x|bc$}

- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

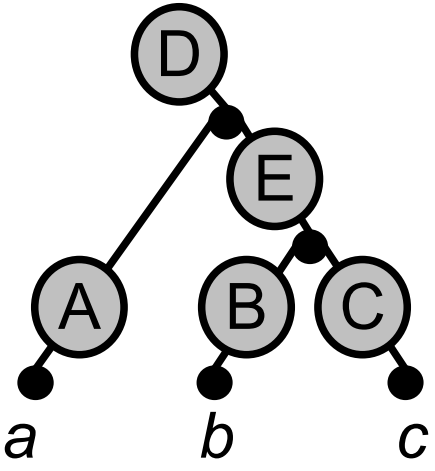
1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

Aucune zone ne convient ? Obstruction !

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



A:{$a|bc$,$b|ax$,$c|ax$,$x|bc$}
B:{$a|bc$,$a|bx$,$a|cx$,$c|bx$}
C:{$a|bc$,$a|bx$,$a|cx$,$b|cx$}
D:{$a|bc$,$x|ab$,$x|ac$,$x|bc$}
E:{$a|bc$,$a|bx$,$a|cx$,$x|bc$}

- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

Un triplet n'est pas compatible ? Obstruction !

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X : $X=\{a,b,c,d,e,f,g,h,i,j,k\}$

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus récursivement.

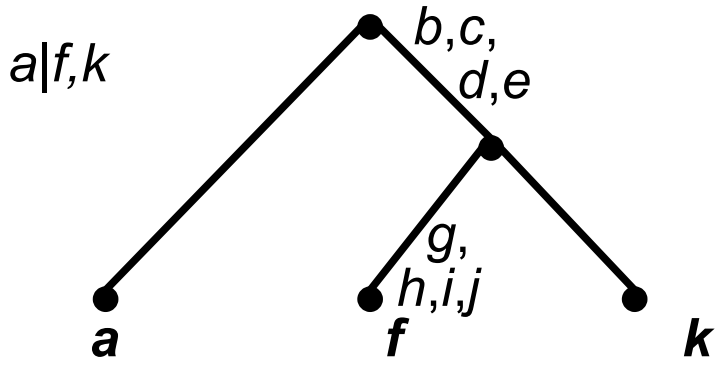
2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

X={*a*,*b*,*c*,*d*,*e*,*f*,*g*,*h*,*i*,*j*,*k*}



- Choisir un triplet *a*|*bc* dans $T_{|X}$

- Pour toute feuille *x*, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer *x* parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

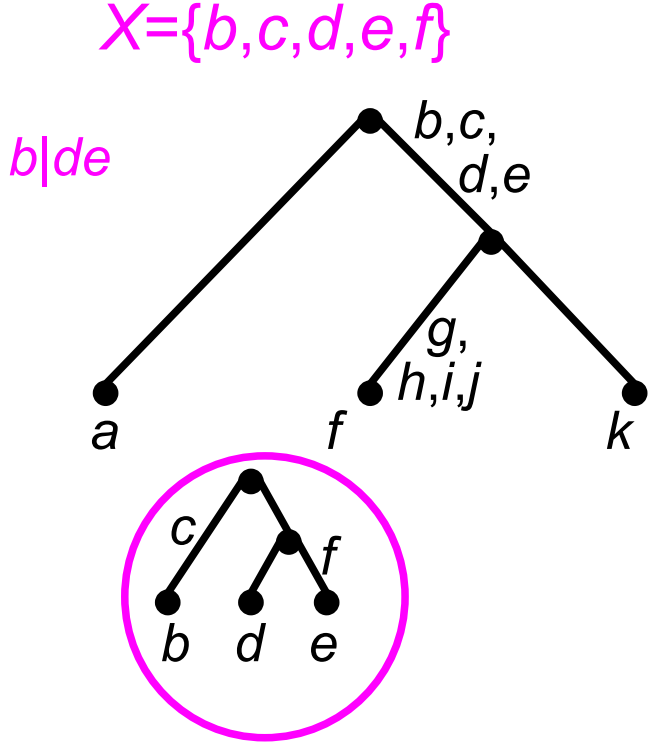
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

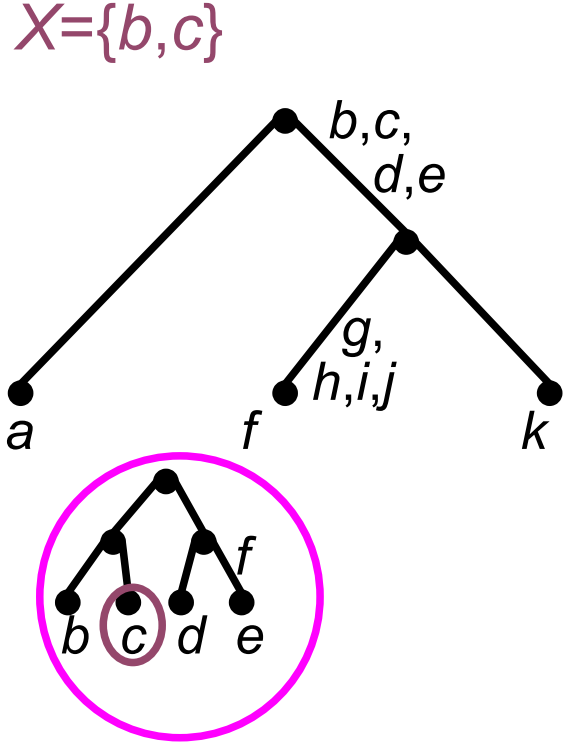
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

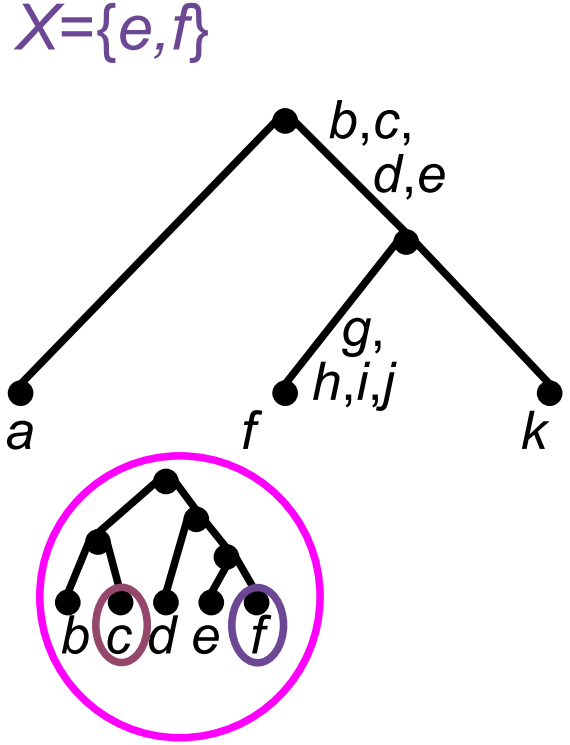
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

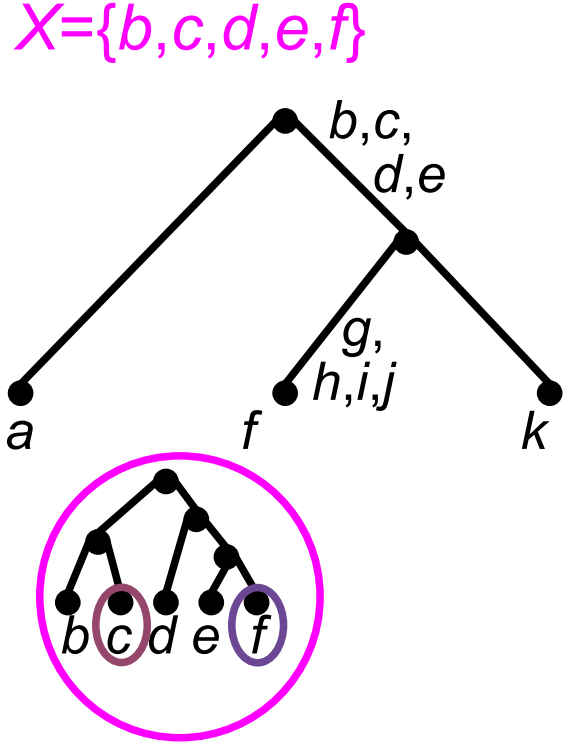
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

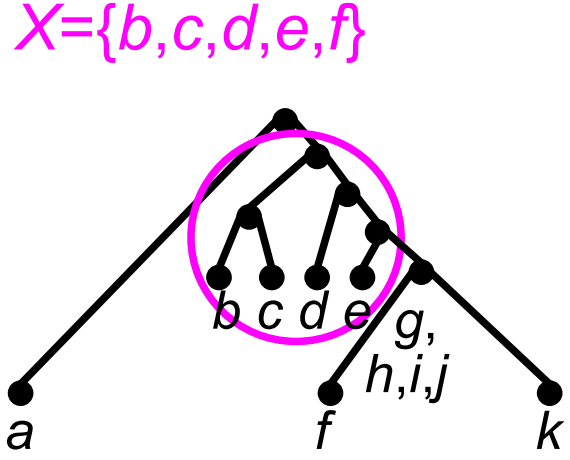
1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus récursivement.



2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

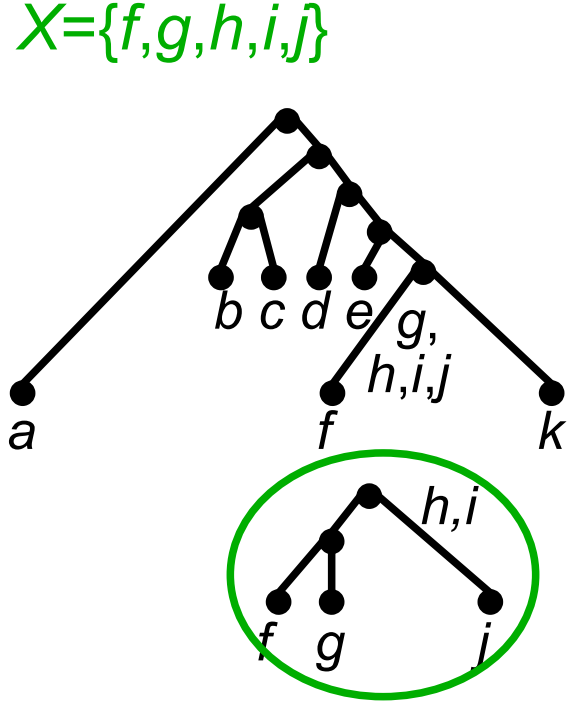
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

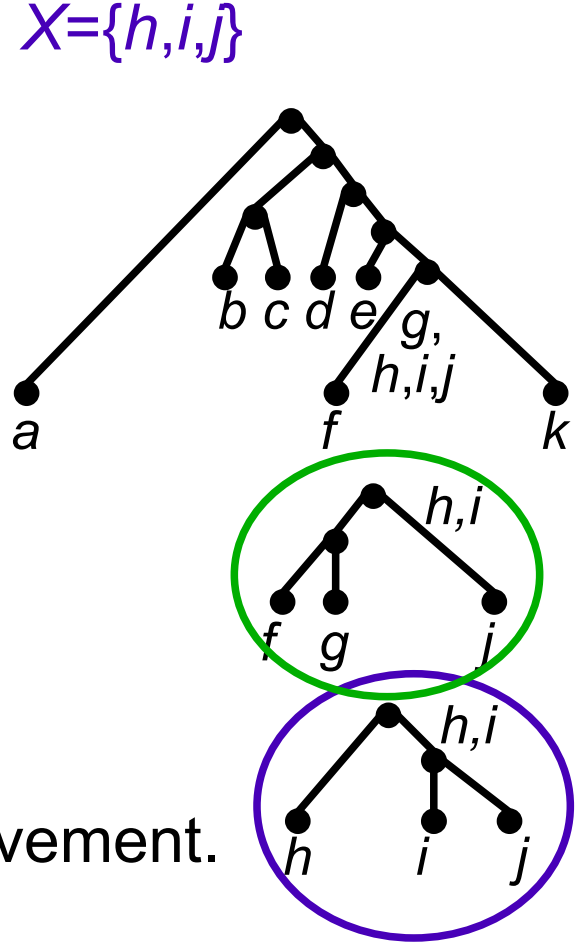
1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus récursivement.



2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

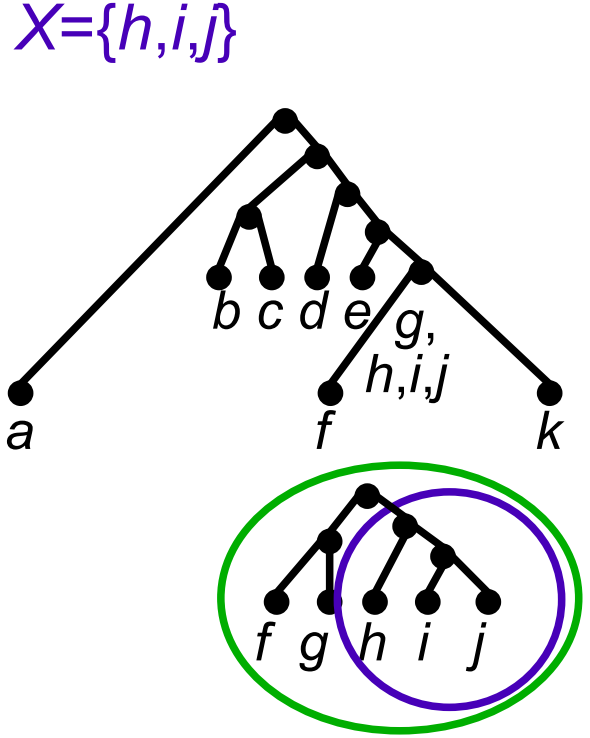
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

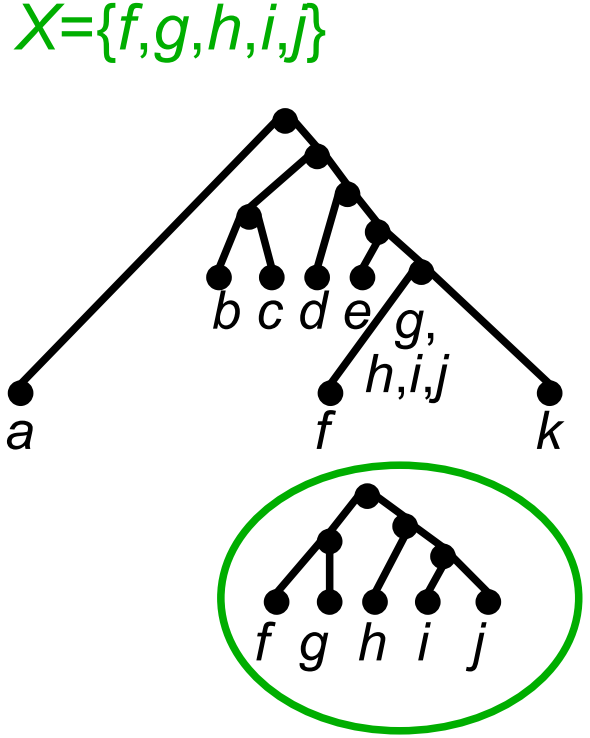
Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.



- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

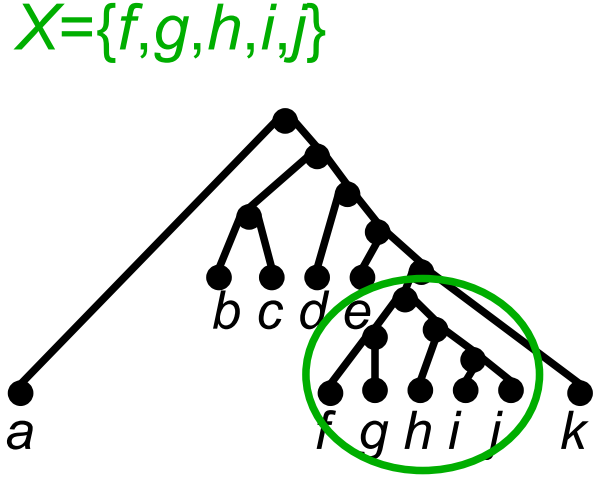
1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus récursivement.



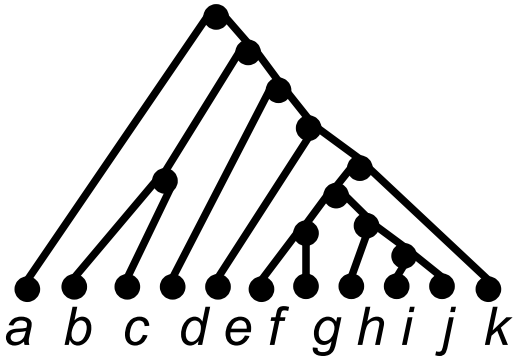
2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

1. Algorithme récursif sur X :

X={*a*,*b*,*c*,*d*,*e*,*f*,*g*,*h*,*i*,*j*,*k*}



- Choisir un triplet *a*|*bc* dans $T_{|X}$

- Pour toute feuille *x*, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer *x* parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus récursivement.

2. Vérifier que tous les triplets de T sont dans l'arbre obtenu.

L'algorithme de reconstruction d'arbre

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

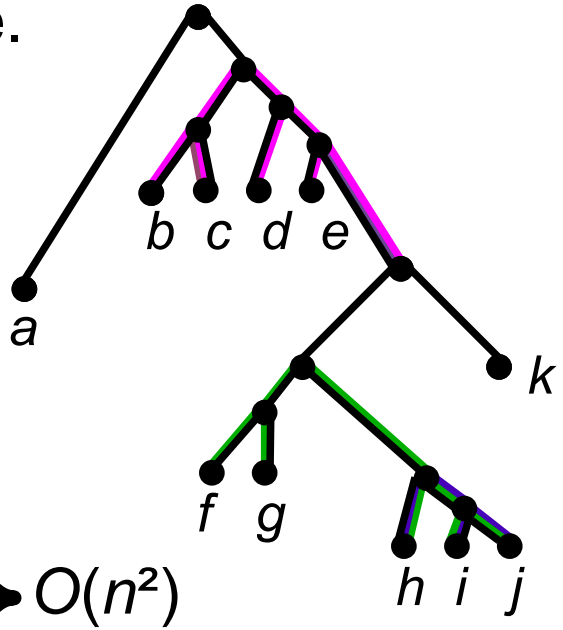
1. Algorithme récursif sur X :

- Choisir un triplet $a|bc$ dans $T_{|X}$

- Pour toute feuille x, considérer $T_{|\{a,b,c,x\}}$ pour savoir dans quelle zone placer x parmi 5 possibles.

- Pour tout ensemble de feuilles correspondant à chaque zone, appliquer l'algorithme récursif.

- Connecter les arbres obtenus.



$O(n^2)$

$O(n)$ feuilles
→ $O(n)$ arêtes
→ $O(n)$ ensembles d'arêtes non chevauchants
→ **$O(n)$ appels récursifs**

2. Vérifier que tous les triplets de T sont dans l'arbre. $O(n^3)$

Algorithme FPT pour l'édition de triplets

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble $\mathcal{T}$ de triplets quand c'est possible.

Corollaire:
Une obstruction sur 4 feuilles peut être trouvée en $O(n^3)$.

Maximum Compatible Subset of Rooted Triples:
Données : Ensemble $\mathcal{T}$ de triplets, entier $t \leq |\mathcal{T}|$.
Question : Existe-t-il un sous-ensemble de $\mathcal{T}$ de taille au moins $|\mathcal{T}|$-t compatible avec un arbre ?

Algorithme FPT pour l'édition de triplets

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble $\mathcal{T}$ de triplets quand c'est possible.

Corollaire:
Une obstruction sur 4 feuilles peut être trouvée en $O(n^3)$.

Maximum Compatible Subset of Rooted Triples:
Données : Ensemble $\mathcal{T}$ de triplets, entier $t \leq |\mathcal{T}|$.
Question : Existe-t-il un sous-ensemble de $\mathcal{T}$ de taille au moins $|\mathcal{T}|-t$ compatible avec un arbre ?

NP-complet.

(preuves par Bryant 1997, Jansson 2001, Wu 2004)

Algorithme en $O((|\mathcal{T}|+n^2)3^n)$.

(Wu, 2004)

Algorithme FPT pour l'édition de triplets

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble $\mathcal{T}$ de triplets quand c'est possible.

Corollaire:
Une obstruction sur 4 feuilles peut être trouvée en $O(n^3)$.

Maximum Compatible Subset of Rooted Triples:
Données : Ensemble $\mathcal{T}$ de triplets, entier $t \leq |\mathcal{T}|$.
Question : Existe-t-il un sous-ensemble de $\mathcal{T}$ de taille au moins $|\mathcal{T}|$-t compatible avec un arbre ?

Algorithme FPT pour des ensembles denses :
- trouver une obstruction $O(n^3)$
- éditer un de ses triplets :
  → 2 possibilités pour chaque triplet
  → total de 6 possibilités

Complexité totale : $O(6^t n^3)$

Algorithme FPT pour l'édition de triplets

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble $\mathcal{T}$ de triplets quand c'est possible.

Corollaire:
Une obstruction sur 4 feuilles peut être trouvée en $O(n^3)$.

Maximum Compatible Subset of Rooted Triples:
Données : Ensemble $\mathcal{T}$ de triplets, entier $t \leq |\mathcal{T}|$.
Question : Existe-t-il un sous-ensemble de $\mathcal{T}$ de taille au moins $|\mathcal{T}|$-t compatible avec un arbre ?

Algorithme FPT pour des ensembles denses :
- trouver une obstruction $O(n^3)$
- éditer un de ses triplets :
  → 2 possibilités pour chaque triplet
  → total de 6 possibilités

Complexité totale : $O(6^t n+n^4)$ en mettant à jour la liste des obstructions.

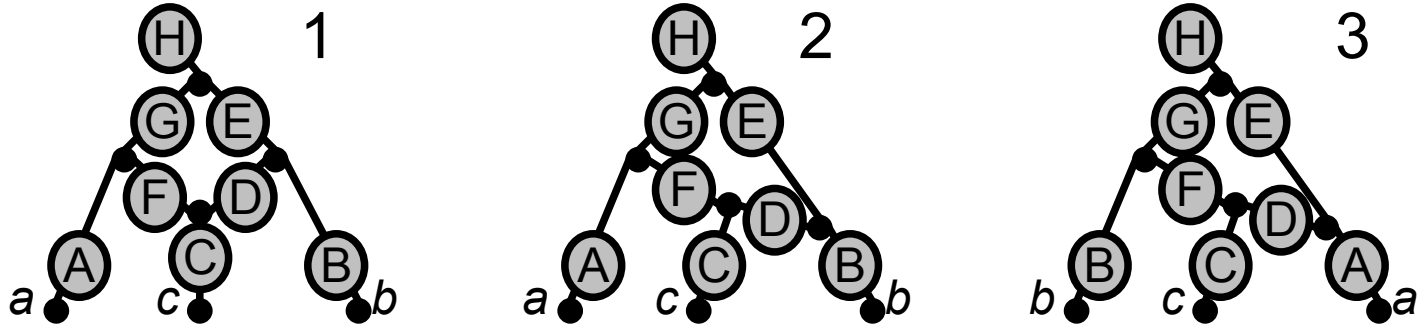
Extrême densité au niveau 1

Un algorithme **certifiant** pour reconstruire un arbre depuis un ensemble T de triplets quand c'est possible.

Corollaire :
Algorithme similaire en $O(n^3)$ pour décider s'il existe un réseau de niveau 1 dont l'ensemble de triplets est exactement T.

- un point de départ différent :
si T est l'ensemble de tous les triplets compatible avec un réseau de niveau strictement 1 alors T contient un ensemble de triplets isomorphe à {*a*|*bc*,*b*|*ac*}.
→ commencer avec {*a*|*bc*,*c*|*ab*}

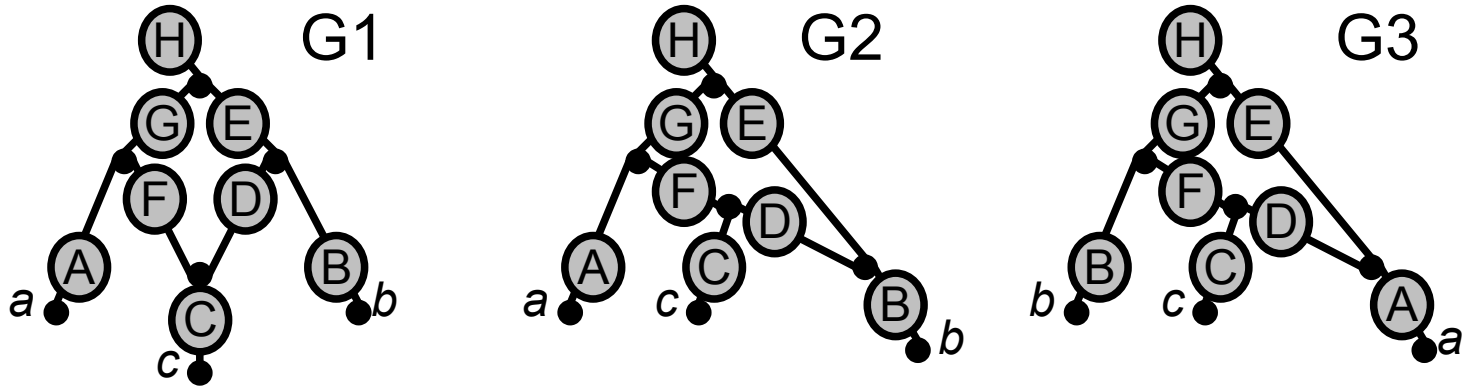
- des zones différentes :



Décision pour l'extrême densité au niveau 1

Algorithme en $O(n^3)$ pour décider s'il existe un réseau de niveau 1 dont l'ensemble de triplets est exactement T.

- des zones différentes :



Des définitions différentes pour les zones D, E, F, G selon la configuration G1, G2, G3
→ détection de certaines incompatibilités

Des définitions identiques pour les zones A, B, C, H dans les configurations G1, G2, G3
→ ambiguïté entre deux configurations
→ quel est le noeud hybride : *a*, *b* ou *c* ?

Codage des réseaux de niveau 1

Un réseau strictement de niveau 1 est **encodable** par ses triplets s'il est le seul réseau strictement de niveau 1 ayant cet ensemble de triplets.

Caractérisation des réseaux de niveau 1 encodables :

Un réseau strictement de niveau 1 est encodable par ses triplets ssi tout blob contient au moins 5 sommets

→ Caractériser les réseaux de niveau >1 encodables par leurs triplets ?
→ Caractériser les ensembles de triplets compatibles avec un **unique** réseau de niveau 1 ?
→ Caractériser les ensembles de triplets compatibles avec un réseau de niveau 1 (par des obstructions comme pour les arbres)

Collaboration avec Katharina Huber (East Anglia UK, mathématiques et bioinformatique)

Des questions ?

Merci pour votre attention !