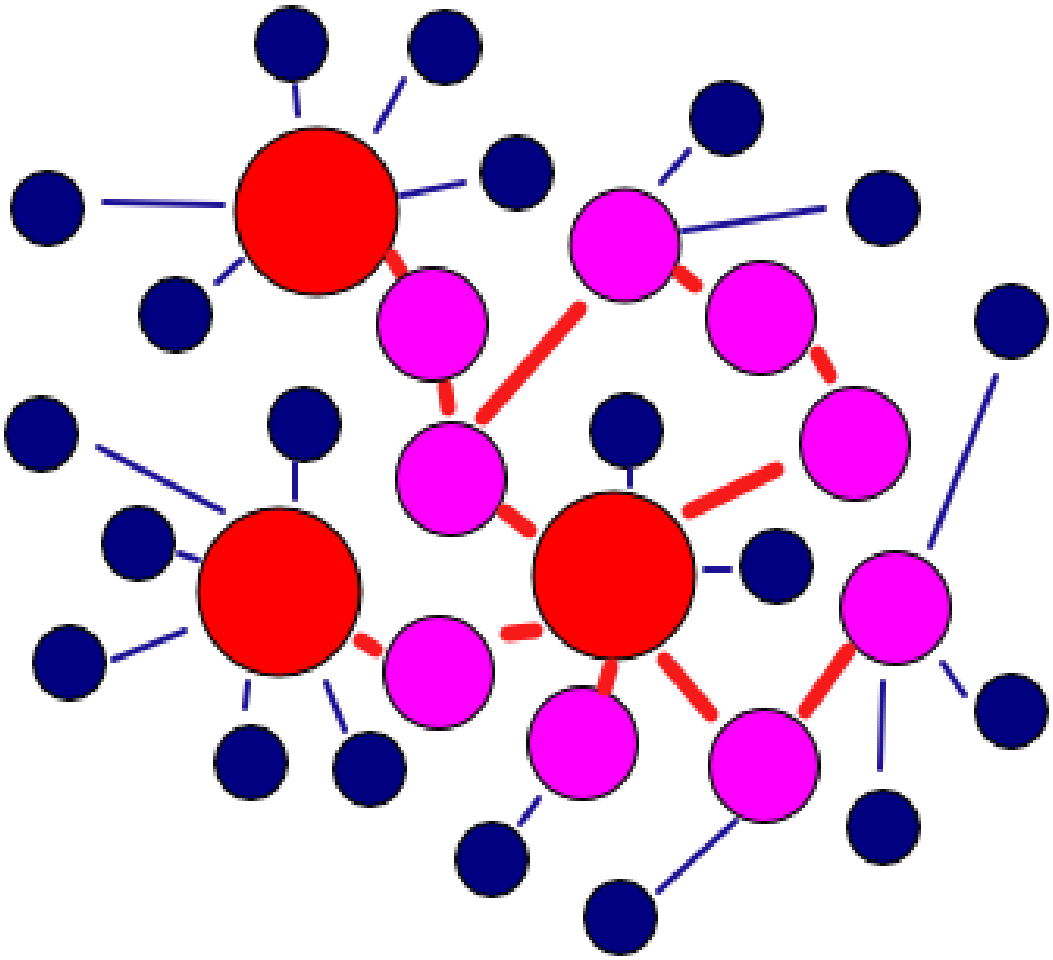
Dossier Annexe

Architecture et génération de réseaux de types Scale-Free



Bourse Yoann - Lemaire Alan

Plan du dossier

Consultez le dossier

Dossier – Le fonctionnement de Google

Le **moteur de recherche Google**, présenté dans notre dossier, doit une grande partie de son succès à la pertinence de ses résultats. Elle est garanti par l'algorithme maintenant légendaire : le **Pagerank**. L'idée maîtresse était **d'attribuer automatiquement une note à chaque page**, afin de mesurer sa popularité.

Mais sur quels critères se baser ? Les ordinateurs, incapables de comprendre le sens des phrases qu'ils lisaient, se révélaient dans l'incapacité de juger efficacement le contenu sémantique d'une page. Il fallait trouver un autre moyen.

Les ingénieurs de Google eurent alors l'idée lumineuse d'étudier la structure du web. Ils s'aperçurent bien vite, à leur grande surprise, que **la répartition des pages web étaient loin d'être aléatoire**. **Les pages les plus populaires concentraient logiquement le plus de liens pointant vers elles**, car elles étaient les plus citées par les internautes et les autres sites. Ainsi, on voyait émerger de la masse titanesque de données certaines pages vers lesquelles énormément de liens menaient.

Si Google se limita à exploiter cette particularité dans son algorithme de Pagerank, à la même époque, des **mathématiciens commencèrent à se pencher sur le phénomène**. Ils modélisèrent internet par un réseau et découvrirent les lois qui le gouvernaient.

Ils firent une extraordinaire découverte : le schéma qui gouvernait internet était non seulement **parfaitement défini et régulier**, mais il **s'appliquait également à de nombreux autres réseaux**. On retrouvait dans la nature et spécialement dans les interactions sociales d'innombrables réseaux de ce type. D'où la place grandissante de l'étude de ces réseaux dans les mathématiques récentes.

Les applications sont aussi nombreuses que concrètes : on peut trouver, par exemple, la stratégie la plus efficace pour vacciner la population humaine contre une maladie contagieuse. Les mêmes aspects sécuritaires se retrouvent sur internet ou les réseaux informatiques en général.

Nous avons étudié les réseaux de type Scale-Free et conçu des algorithmes qui en généraient aléatoirement, afin de servir de support à diverses simulations d'opérations.

PARTIE I

Architecture des réseaux de type « Scale-Free »

Un réseau est défini comme un ensemble de points appelés **noeuds** (ou *pôles,* en anglais *nodes*) connectés entre-eux par des **liens**, ces derniers pouvant être orientés ou non, c'est à dire avoir un départ et une fin.

On appelle **degré d'un noeud** le nombre de liens de celui-ci avec d'autres noeuds. Dans le cas d'un réseau orienté, le degré regroupe les liens entrants et sortants, mais on définit le degré entrant et le degré sortant.

Cette donnée permet de définir un réseau, par le biais de la **distribution de degré**, qui désigne la proportion de noeuds du réseau ayant le degré k.
Il apparaît immédiatement que si l'on considère un réseau de N noeuds dont Nk sont de degré k (et ont donc k liens), la distribution de degré $\mathrm{P}(k)$ du réseau vérifiera :

$$\mathrm{P}(k) = \frac{Nk}{N}$$

Cette équation simple permet de prendre conscience de cette définition.

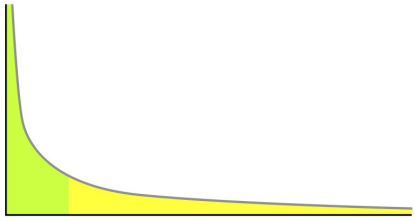
D'autre part, il faut rappeler qu'une **loi de puissance** désigne une relation de la forme

$$f(x) = a \cdot x^{\alpha}$$

Vilfredo Pareto
1848-1923
économiste et sociologue italien

Le plus célèbre exemple d'une telle loi appliquée aux probabilités fut la **distribution de Pareto (loi du 80-20)**, théorie mathématique développée par Joseph Juran et basée sur les observations d'un économiste italien qui avait repéré que 20% de la population italienne possédait 80% des richesses nationales.

Ses applications sont nombreuses, aussi bien dans la gestion de la qualité que la réassurance.



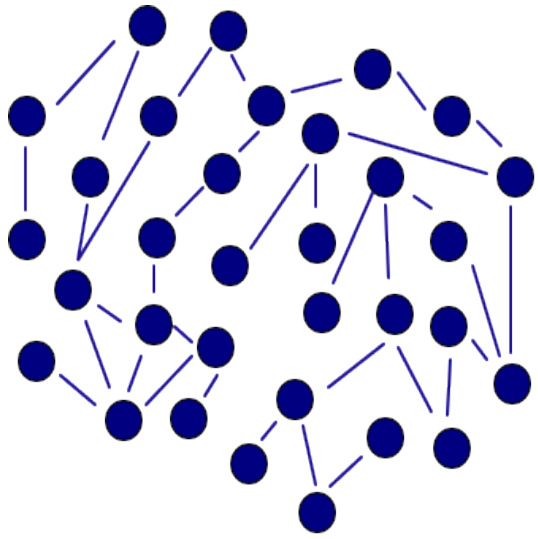
Ces outils furent utilisés dans l'étude d'internet, communément modélisé par un réseau dont les **noeuds sont les pages internet**, et les **liens sont les liens hypertextes**, phrases cliquables qui dirigent l'internaute d'un site à l'autre.

Les premières études de la topologie du *World Wide Web* furent effectuées en 1998 par Hawoong Jeong et Reka Albert de l'université de Notre-Dame. **Ils pensaient qu'internet était un réseau totalement aléatoire, ce qui semblait alors logique compte tenu de la diversité des intérêts personnels des internautes.**

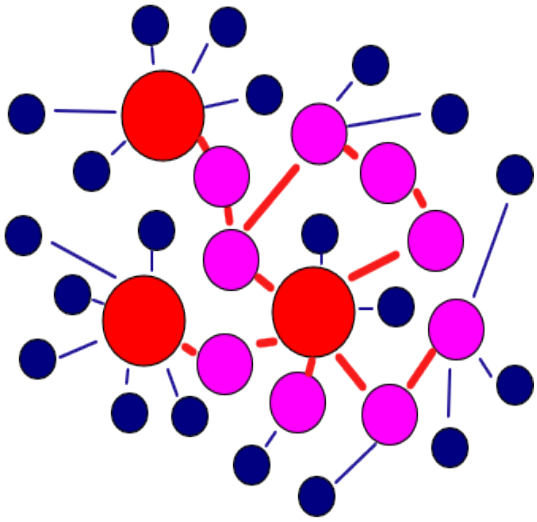
Le Web, en tant que réseau aléatoire, devrait présenter les caractéristiques majeures des distributions de degrés aléatoires : les noeuds auraient en moyenne le même nombre de connexions, et rares seraient les noeuds au degré éloigné de cette moyenne.

Même si leurs outils ne pouvaient mesurer qu'une infime fraction de la toile, ils firent une découverte surprenante : la distribution de degré se rapprochait de la distribution de Pareto : 80% des pages étaient dotées de moins de 4 liens, alors que 0,01% des noeuds en présentaient plus de 1000.

Ces observations amènent donc à considérer la toile du web en tant que **réseau social dont la distribution de degré suit une loi de puissance**. C'est un tel réseau que l'on appelle **« Scale-Free »** (littéralement : *sans échelle*). Ce terme est justifié par la présence dans le réseau de noeuds de très hauts degrés, qui semblent avoir un nombre presque illimité de connexions. Ce sont les **hubs** (*moyeu*).



Réseau aléatoire

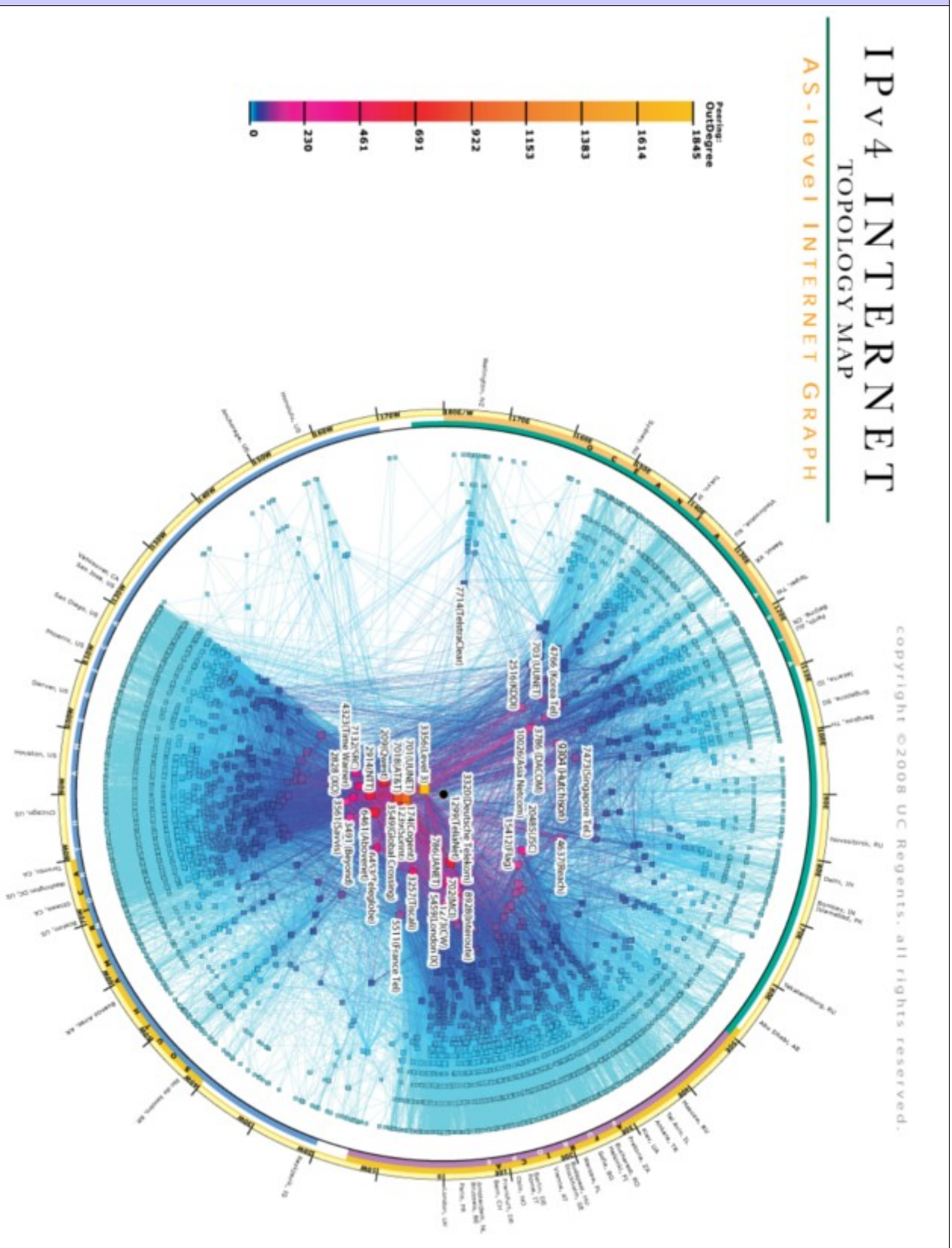


Réseau type "Scale-Free"

D'autres observations ont par la suite confirmé l'application de ce modèle à la toile. De très nombreux projets se sont en effet lancés pour cartographier cet immense réseau virtuel. Diverses cartes plus ou moins exhaustives de la toile sont montrées dans les pages suivantes.

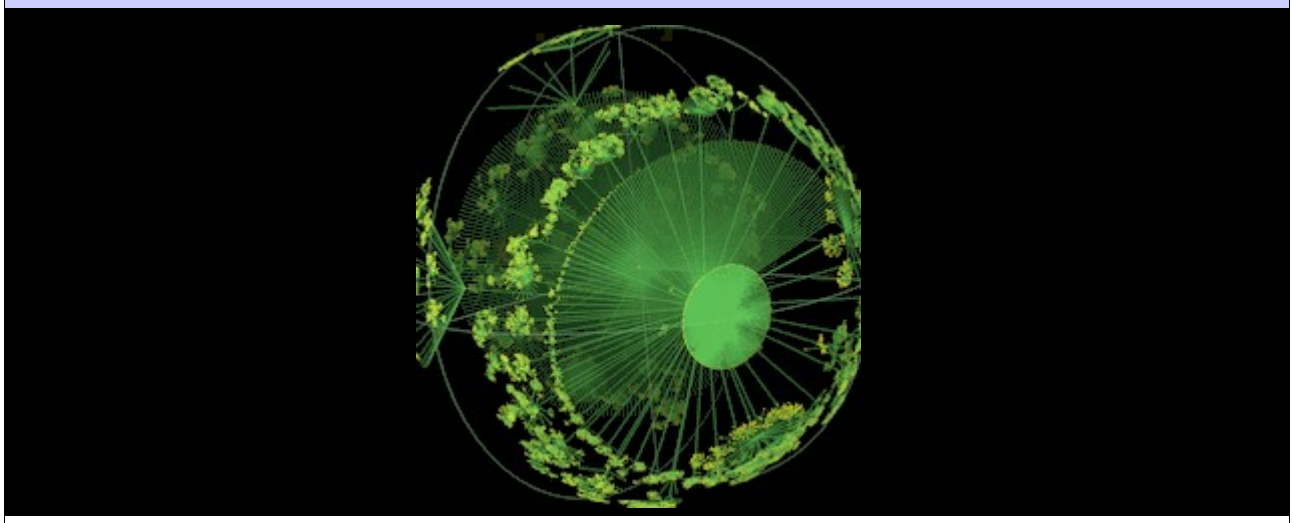
Carte d'internet de CAIDA

(cooperative association for internet data analysis)



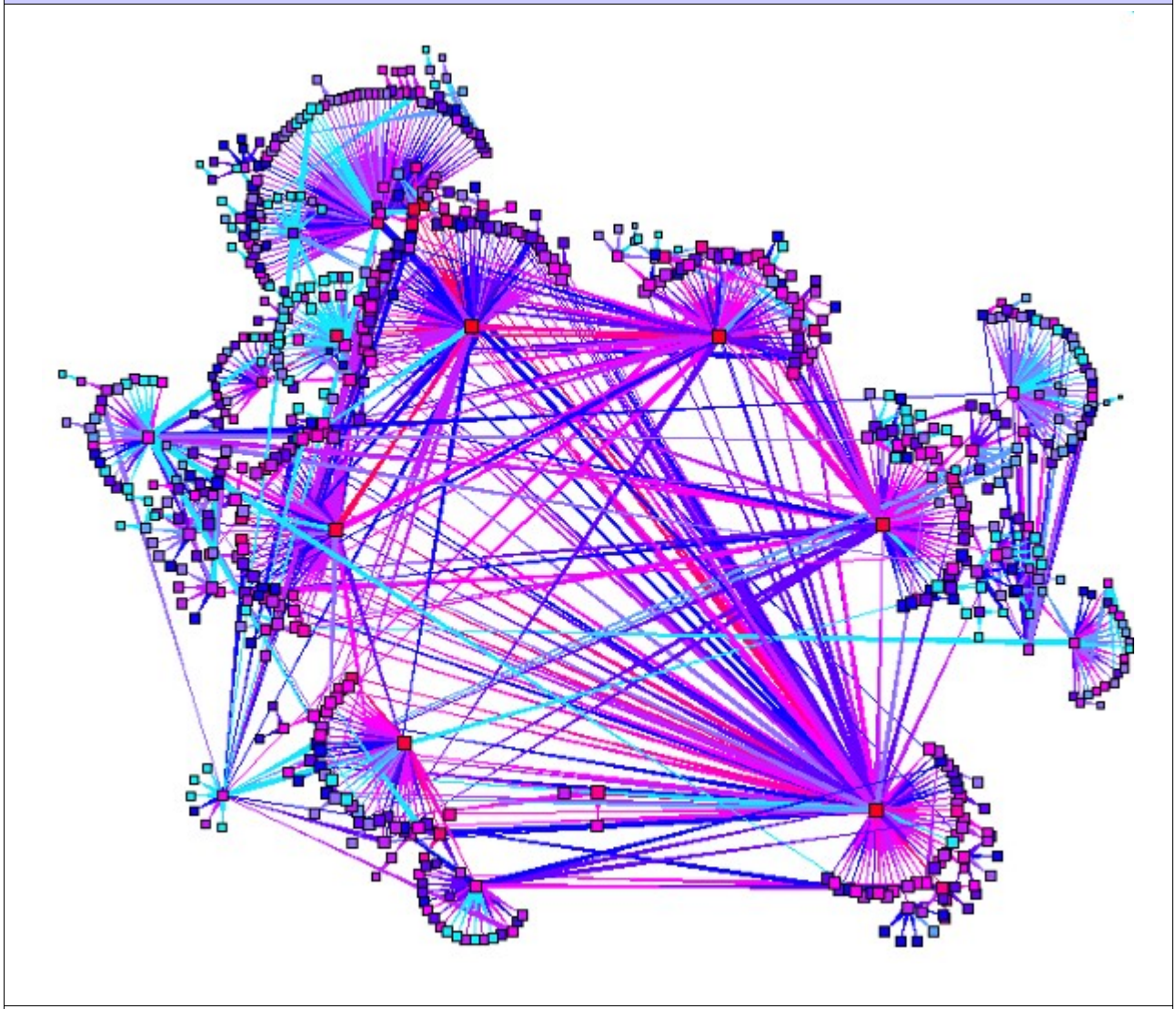
Source : CAIDA, copyright UC Regents (2008)

World Wide Web en représentation tridimentionelle



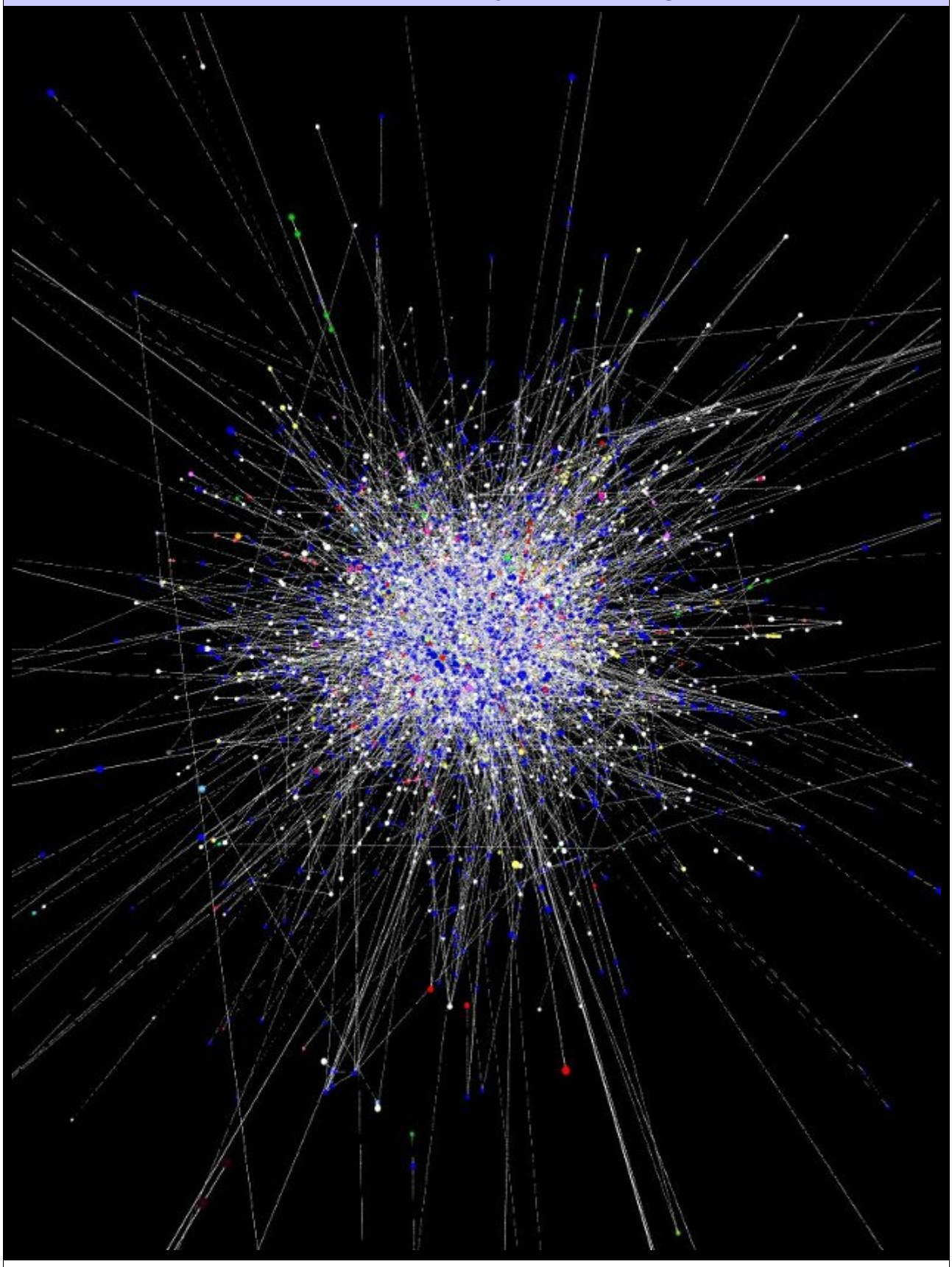
Source : CAIDA, données de Skitter visualisées par le logiciel Walrus

World Wide Web en représentation hierarchisée par Plankton



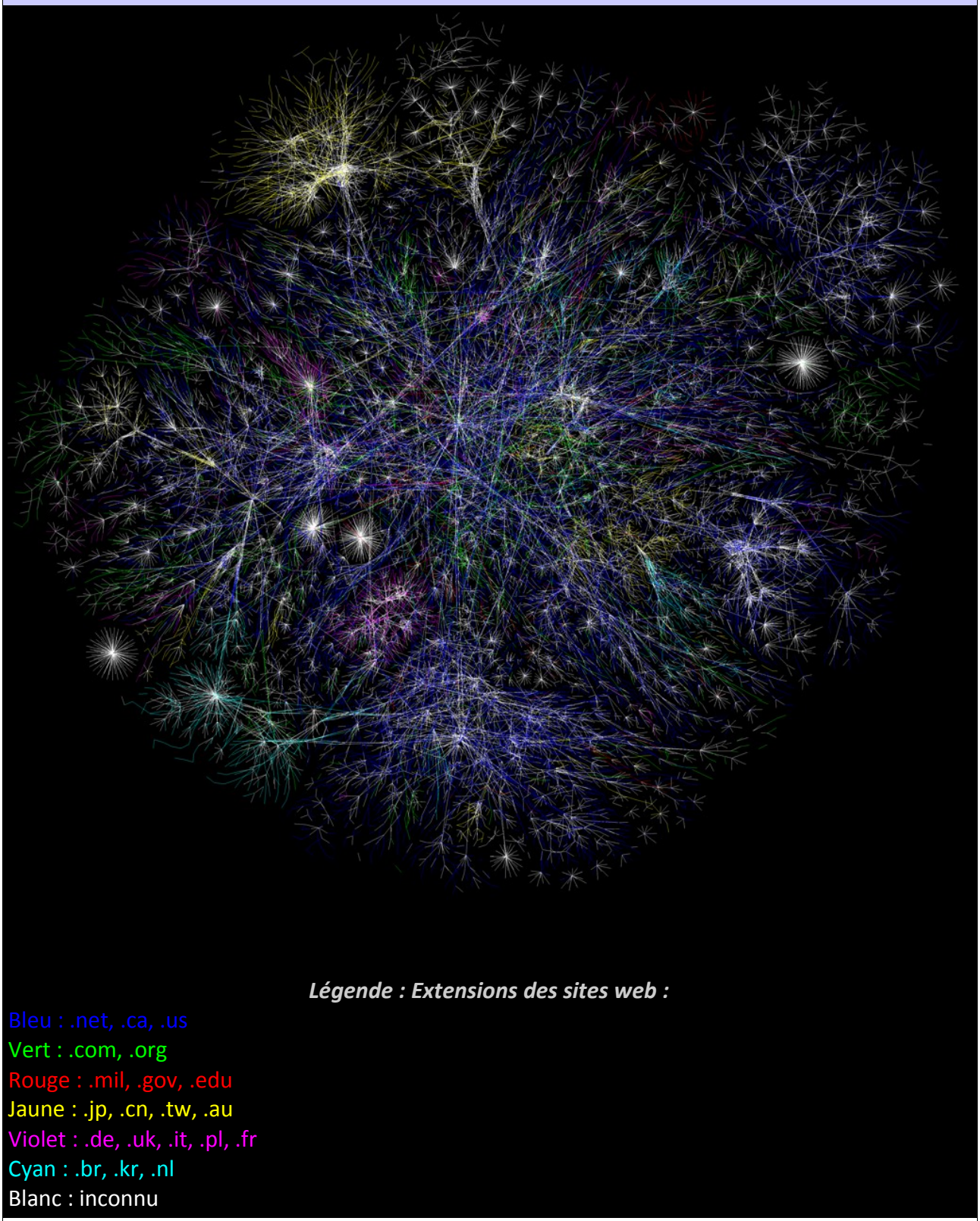
Source : CAIDA, données de Skitter visualisées par le logiciel Plankton

Carte du World Wide Web d'après le projet IP Mapping de Stephen Coast



Source : Stephen Coast (2001)

Carte du World Wide Web d'après le projet Opte



Source : Projet Opte (2005)

On y repère bien la présence de hubs qui concentrent tous les liens, et la présence de nombreux noeuds périphériques à faible degré.

Un des principes fondateurs de la science est la **modélisation par les mathématiques** des observations empiriques. Elle fait parfois des avancées spectaculaires, en découvrant des mystérieuses lois qui, comme de véritables **fondements de notre univers**, se retrouvent partout dans la nature. On peut citer les lois exponentielles découvertes au XVIIè siècle, ou la suite de Fibonacci. Les **réseaux de type "Scale-Free"** pourraient bien représenter une telle révolution, étant donné leur polyvalence.

Durant la dernière décénie, divers chercheurs ont constaté avec surprise que de nombreux réseaux observés répondaient au modèle des **réseaux Scale-Free, définis par une loi de puissance :**
La fraction de noeuds ayant *k* connexions y est :

$$P(k) = a \cdot k^{\alpha}$$

Généralement, dans la réalité α est une valeur comprise entre -3 et -2. Les plus célèbres sont récapitulés dans le tableau ci-dessous.

Réseau	Noeuds	Connexions	Source
Structure virtuelle d'internet	Pages web	Liens hypertextes	*Voir ci-dessus*
Structure physique d'internet	Routeurs	Lignes de communication	Michalis Faloutsos de l'*Université de California* à Riverside Petros Falotitsos de l'*Université de Toronto* Christos Faloutsos de l'*Université Carnegie Mellon*
Réseaux sociaux humains	Individus	Rapports sexuels	Collaboration entre les universités de Stockholm et Boston
Réseaux email	Individus	Courriers email (courriels)	Stefan Bornholdt de l'*Université de Kiel*
Publications scientifiques	Chercheurs, scientifiques	Citations	Sidney Redner de l'*Université de Boston*
Industries	Firmes	Partenariats	Walter W. Powell de l'*Université de Stanford*, Douglas R. White de l'*Université de Californie* à Irvine, Kenneth W. Koput de l'*Université d'Arizona*, Jason-Owen Smith de l'*Université du Michigan*
Hollywood	Acteurs	Participations au même film	Le jeu de Kevin Bacon
Métabolisme cellulaire	Molécules	Réactions biochimiques	Zoltan Oltvai, de la *Northwestern University*
			Source : Article du Scientific American, Mai 2003, par Albert-László Barabási et Eric Bonabeau

Les réseaux de type "Scale-Free" sont **très nombreux**, et sont de plus en plus observés. Leur **diversité** fait de l'étude de ce modèle une question clé pour beaucoup de domaines : il s'agit de découvrir les lois auxquelles tous ces réseaux, pourtant tellement variés, répondent tous.

PARTIE II

Génération aléatoire de réseaux Scale-Free

1) Aspect mathématique

Albert-László Barabási
1967-
scientifique hongrois

Le hongrois Albert-László Barabási fut l'un des premiers à travailler sur la théorie des réseaux "Scale-Free", dont on lui attribue la paternité du concept.

Une des caractéristiques de ces réseaux qu'il repéra fut le **mécanisme de liaison d'un nouveau noeud à un réseau déjà existant**, connu sous le nom d'**attachement préférentiel** (preferential attachment). Le principe en est simple : lorsqu'un nouveau noeud doit entrer dans un tel réseau, il aura une plus grande tendance à se lier aux noeuds de haut degrés (hubs) qu'à ceux de faibles degrés. On le remarque aisément sur les réseaux réels.

Il mit donc au point avec une de ses étudiantes, Réka Albert, un algorithme de création de réseaux "Scale-Free" basé sur le principe de l'attachement préférentiel. Cet algorithme est connu sous le nom de **modèle de Barabási–Albert**, ou **modèle BA**. Il est à noter que certains de leurs papiers sont cosignés par H. Jeong, spécialiste de l'informatique.

Réka Albert

Ce modèle considère un réseau initial de m_0 noeuds (m_0 >2) non connectés entre eux initialement, et indique comment l'étendre de façon à ce qu'il soit de type "Scale-Free".

Il suffit que chaque noeud ajouté se lie à un noeud existant de degré k avec la probabilité :

$$p = \frac{k}{\sum_j k_j}$$

$\sum_j k_j$ désigne ici le nombre total de liens dans le réseau. On notera par la suite : $\sum_j k_j = k_t$

Si on écrit cette formule pour un noeud *i* présent dans le réseau, la probabilité qu'un nouveau noeud tisse un lien avec lui sera :

$$p(k_i) = \frac{k_i}{\sum_j k_j}$$

Pour vérifier qu'un tel réseau est de type “Scale-Free”, il nous faut considérer le degré de *i* comme une fonction continue du nombre total de connexions dans le réseau.
Cet algorithme ajoute à chaque “tour” un noeud de degré fixé, *d*, inférieur ou égal au nombre de noeuds du réseau initial m_0.
Comme le réseau est sans aucun lien au début de l'application de l'algorithme, chaque tour introduira un nouveau noeud avec *d* liens. Au bout de *t* tours, on aura donc *d*t* liens dans le réseau. Or, ceux-ci ont deux extrémités, toutes deux comptées dans la somme $\sum_j k_j = k_t$.

Ainsi, il apparait que lorsqu'on parcours tous les noeuds, la somme de leurs degrés vaut :

$$\sum_j k_j = k_t = 2 \cdot d \cdot t$$

Comme on considère les variations de degrés continues, on peut écrire la variation du degré d'une node *i* sous la forme différentielle. Elle est justifiée par le fait que, en un tour, un nouveau noeud tisse *d* liens au total dont un éventuelle au noeud *i* avec la probabilité $p(k_i)$.

$$\frac{\delta k_i}{\delta t} = d \cdot p(k_i) = d \cdot \frac{k_i}{\sum_j k_j} = d \cdot \frac{k_i}{2 \cdot d \cdot t} = \frac{k_i}{2 \cdot t}$$

$$\frac{\delta k_i}{k_i} = \frac{\delta t}{2 \cdot t}$$

C'est une équation différentielle linéaire que l'on peut aisément résoudre en considérant les dérivées partielles comme des dérivées simples (c'est à dire que k_i ne dépend que de *t*, ce qui apparait comme évident)..
Ce qui aboutit à la solution de la forme :

$$\ln(k_i) = \frac{1}{2} \cdot \ln(t) + cste$$

On détermine la constante en considérant que ce noeud a été ajouté à un temps t_i avec d noeuds, comme les autres, ce qui donne la solution finale :

$$k_i(t_i) = d \qquad k_i(t) = d\sqrt{\frac{t}{t_i}}$$

Ce résultat peut être exploité pour mettre en évidence le caractère “Scale-Free” du réseau ainsi obtenu. En effet, la condition qu'un noeud ait un degré k inférieur strictement à k_i équivaut à :

$$k < k_i(t) \Leftrightarrow k < d\sqrt{\frac{t}{t_i}}$$

$$\Leftrightarrow \frac{k^2}{d^2} < \frac{t}{t_i} \Leftrightarrow t_i > \frac{d^2}{k^2} \cdot t$$

Considérant que les noeuds sont ajoutés dans le réseau à intervalles de temps égaux, on a donc $m_0 + t$ intervalles de temps passées, ce qui donne une densité de probabilité pour t_i égale à :

$$p_i(t_i) = \frac{1}{m_0 + t}$$

Les équivalences ci-dessus permettent d'autre part de mettre en évidence deux probabilités complémentaires, ce qui nous permet donc d'écrire :

$$p(k < k_i(t)) = p(t_i > \frac{d^2}{k^2} \cdot t)$$

$$= 1 - p(t_i \leqslant \frac{d^2}{k^2} \cdot t) = 1 - \frac{d^2}{k^2} \cdot t \cdot p_i(t_i)$$

$$= 1 - \frac{d^2}{k^2} \cdot t \cdot \frac{1}{m_0 + t}$$

On peut finalement utiliser la propriété suivante pour mettre en évidence :

$$P(k) = \frac{\delta p(k < k_i(t))}{\delta k} = \frac{2 \cdot d^2 \cdot t}{m_0 + t} \cdot \frac{1}{k^3}$$

Ceci nous permet de conclure qu'un réseau créé par cet algorithme est bien de type “Scale-Free” dont la distribution de degré répond à la loi :

$$P(k) = a \cdot k^{-3}$$

Dans le cas d'un réseau ou les liens ne sont pas réciproques, on dit qu'il est de type “Scale-Free” si la distribution des **degrés entrants** d'une part (liens dirigés vers le noeud), et **sortants** d'autre part (liens partant du noeud) sont toutes deux de type “Scale-Free”, donc gouvernées par une loi de puissance. Cet algorithme peut donc aisément être adapté pour créer des réseaux “Scale-Free” à liens non réciproques, comme nous le verrons par la suite.

Partie II

Génération aléatoire de réseaux Scale-Free

2) Programmation

Nous avons utilisé l'algorithme mis au point par Barabási et Albert pour créer un réseau “Scale-Free” de distribution aléatoire. Pour cela, nous avons conçu un programme CAML qui considère un réseau sous la forme d'un tableau dont chaque case représente un noeud. Chaque noeud est lui-même en réalité une liste d'entiers, ces derniers étant les indices des noeuds auquel il est connecté dans le tableau réseau. Cela se traduit par la définition de types suivante :

Définitions de types

```
type indice == int;;
type noeud == indice list;;
type reseau == noeud vect;;
```

```
Type indice defined.
Type noeud defined.
Type reseau defined.
```

Exemple de réseau simple

```
exemple : reseau = [|[3; 2; 1]; [3; 2; 0]; [0; 1]; [1; 0]|]
```

Pour plus de facilité, l'algorithme commencera à partir d'un réseau de deux noeuds connectés entre eux. Cette connexion étant négligeable comparée à la somme de toutes les autres, elle ne devrait pas affecter le caractère “Scale-Free” du réseau final. D'autre part, cela induit que nous ajouterons à chaque tour des noeuds de degré 1. Nous observerons également les résultats pour des noeuds ajoutés de degré 2.

Le principal problème à gérer était bien sûr de savoir à quel noeud le nouvel arrivant devait s'attacher. Il devait en choisir un parmi tous les noeuds présents en tenant compte des probabilités, directement liées au degré de chaque noeud.

A chaque tour, le programme ajoutera un noeud avec une connexion. Il créera donc deux cases, une dans la liste du noeud ajouté et une dans la liste du noeud présent auquel il s'ajoute. Le dénominateur de la probabilité caractéristique du modèle BA devra donc augmenter de deux, du double du nombre total de connexions dans le réseau.

La méthode que nous proposons se base sur une petite astuce : il s'agit de stocker dans une référence ce nombre (en l'augmentant donc de deux à chaque tour), puis de tirer un nombre au hasard entre 0 et cette limite.
Le nouveau noeud s'ajoutera tout simplement au noeud qui contient cette connexion, indicé à partir du début du tableau.

Il faut pour cela créer un sous programme qui compte la $n^{ième}$ connexion de tout le tableau réseau, c'est à dire la $n^{ième}$ case de la liste que représenterait toutes les listes des noeuds mises bout à bout. Il y a ainsi d'autant plus de chance d'arriver dans la liste d'un noeud que la taille de cette dernière est grande, ce qui correspond bien au modèle.

La programmation par récurrence oblige de passer en paramètre le numéro du noeud où le comptage est arrivé. Il faut également donner au programme le réseau dans lequel opérer et le nombre aléatoire obtenu à compter.

Compteur du noeud correspondant au nombre tiré aléatoirement

```
let rec compte (reseau:reseau) (noeud:indice) (nombre:int) =
match nombre with
   |0 -> noeud
     (*On arrive dans la première case de la liste du noeud 'noeud'*)
   |n when (n < list_length (reseau.(noeud))) -> noeud
     (*On arrive dans une des cases de la liste du noeud 'noeud'*)
  |n -> compte reseau (noeud+1) (nombre - list_length (reseau.
(noeud)));;
     (*Sinon, on décrémente le nombre et on passe au noeud suivant*)
```

```
compte : reseau -> indice -> int -> indice = <fun>
```

Une deuxième fonction annexe sera requise : il s'agira d'insérer l'indice du nouveau noeud dans la liste du noeud préexistant auquel il se connecte. Pour ceci, une simple fonction d'édition de liste suffit. Nous avons décidé, pour plus de facilité, que les listes d'indices que représentent les noeuds seraient triées par ordre décroissant.

Insertion d'un indice dans un noeud existant

```
let rec insere (indice:indice) (noeud:noeud) =
match noeud with
   |[] -> [indice]
   |t::q when t>indice -> t:: (insere indice q)
     (*si la tête est plus grande que l'indice à insérer, on continue
       de parcourir la liste jusqu'à trouver sa place*)
   |liste -> indice :: liste;;
     (*dans ce cas, la tête sera plus petite, et on ajoute l'indice
       avant*)
```

```
insere : indice -> noeud -> noeud = <fun>
```

Plutôt que de s'encombrer d'une variable globale, peu fiable en CAML, nous avons préféré construire le programme à l'aide de deux sous fonctions. La première, qui forme un réseau de *n* noeuds, appelle la seconde, qui ajoute un noeud dans un réseau préexistant, en créant bien entendu un nouveau tableau de taille supérieure. Le programme issu est présenté ci-dessous :

PROGRAMME DE GÉNÉRATION D'UN RÉSEAU ALÉATOIRE "SCALE-FREE"

```
let creer_reseau (n:int) = let denominateur = ref 2 in
former n

  where rec former (n:int) = match n with
   |n when n <=1 -> failwith "Erreur de dimension"
   |2 -> [|[1];[0]|]
   |n -> let precedent = former (n-1) in ajouter n  precedent
            (*precedent représente ici le réseau au tour précédant
              l'ajout du noeud n*)

     where rec ajouter (n:indice) (precedent:reseau) =
        let rdm = (compte precedent 0 (random__int (!denominateur)))
        and nouveau = make_vect n [] in
         (*rdm représente le noeud aléatoire auquel le nouveau se
           connectera*)
         (*nouveau accueillera le tableau du réseau après l'ajout du
           noeud*)

          nouveau.(n-1) <- [rdm];
          nouveau.(rdm) <- insere (n-1) precedent.(rdm);
          denominateur := !denominateur + 2;
         (*on effectue le changement dans les listes, les noeuds*)

          for i=0 to n-2 do
           if (i<>rdm) then
                nouveau.(i) <- precedent.(i);
          done;
        (*on recopie le reste du tableau, qui est inchangé, grâce à
          un repère i*)
          nouveau;;
        (*on retourne le tableau final*)
```

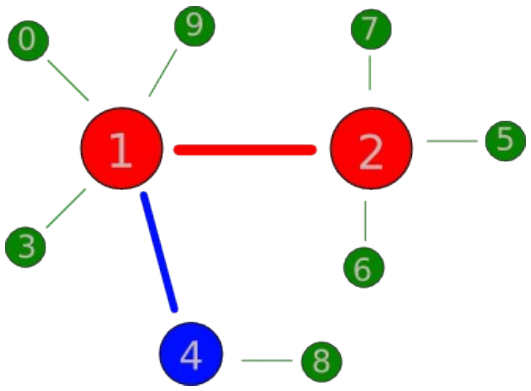
```
creer_reseau : int -> reseau = <fun>
```

EXEMPLE DE SON FONCTIONNEMENT

```
let reseau = creer_reseau 10;;
```

```
reseau : reseau =
[|[1]; [9; 4; 3; 2; 0]; [7; 6; 5; 1]; [1]; [8; 1]; [2]; [2]; [2];
 [4]; [1]|]
```

Soit le réseau :



On peut très aisément modifier le programme précédent pour qu'il insère des noeuds de degrés 2 à chaque tour :

Programme de génération d'un réseau aléatoire "Scale-Free"
Les noeuds ajoutés sont de degré 2.

```
let creer_reseau2 (n:int) = let denominateur = ref 2 in
former n

  where rec former (n:int) = match n with
   |n when n <=1 -> failwith "Erreur de dimension"
   |2 -> [|[1];[0]|]
   |n -> let precedent = former (n-1) in ajouter n  precedent

     where rec ajouter (n:indice) (precedent:reseau) =
        let n1 = (compte precedent 0 (random__int (!denominateur)))
        and n2 = ref (compte precedent 0 (random__int (!denominateur)))
        and nouveau = make_vect n [] in

          while n1= !n2 do
            n2 := (compte precedent 0 (random__int (!denominateur)));
          done;
         (*on tire n1 aléatoirement et on change n2 jusqu'à s'assurer
           d'avoir deux noeuds différents auxquels lier le nouveau
           noeud*)

          nouveau.(n-1) <- [max n1 !n2;min n1 !n2];
          nouveau.(n1) <- insere (n-1) precedent.(n1);
          nouveau.(!n2) <- insere (n-1) precedent.(!n2);
          denominateur := !denominateur + 4;
         (*on effectue le changement dans les listes, les noeuds*)

          for i=0 to n-2 do
           if (i<>n1 && i<> !n2) then
                nouveau.(i) <- precedent.(i);
          done;
        (*on recopie le reste du tableau, qui est inchangé, grâce à
          un repère i*)
          nouveau;;
```

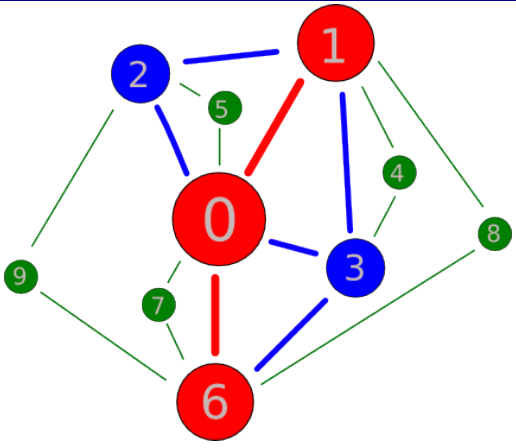
```
creer_reseau2 : int -> reseau = <fun>
```

Exemple de son fonctionnement

```
let reseau2 = creer_reseau2 10;;
```

```
reseau2 : reseau =
  [|[7; 6; 5; 3; 2; 1]; [8; 4; 3; 2; 0]; [9; 5; 1; 0]; [6; 4; 1; 0];
                                                        [3; 1];
                  [2; 0]; [9; 8; 7; 3; 0]; [6; 0]; [6; 1]; [6; 2]|]
```

Soit le réseau :



PARTIE II

Génération aléatoire de réseaux Scale-Free

3) Étude des réseaux générés

Nous nous sommes ensuite proposés de vérifier que les réseaux créés de cette façon répondaient bien à la définition d'un réseau “Scale-Free”, c'est à dire de confirmer par l'expérience ce que la théorie avait prouvé. Il a fallu pour cela créer un petit programme comptant le nombre de noeud de chaque degré. Nous les avons répertoriés dans un tableau à la dimension très largement supérieure au nécessaire.

PROGRAMME D'ANALYSE DES RÉSEAUX

```
let analyse (reseau:reseau) =
 let resultat = make_vect (vect_length reseau) 0 in
   for i=0 to (vect_length reseau - 1) do
   resultat.(list_length reseau.(i)) <-
                        resultat.(list_length reseau.(i)) + 1;
   (*on incrémente la case correspondant à la taille du noeud*)
   done;
resultat;;
```

```
analyse : reseau -> int vect = <fun>
```

Les petits réseaux de 10 noeuds étaient cités à titre exemplaire : on ne peut pas y vérifier significativement le caractère “Scale-Free” des réseaux. Nous allons le vérifier sur des réseaux plus grands de 1000 noeuds.

CRÉATION ET ANALYSE DE RÉSEAUX “SCALE-FREE”

```
let reseau_analyse = creer_reseau 1000;;
let reseau_analyse2 = creer_reseau2 1000;;

analyse reseau_analyse;;
analyse reseau_analyse2;;
```

```
reseau_analyse : reseau =
      [|[960; 946; 745; 658; 600; 578; 497; 488; 465; 460; 444; 419; 365; 356;
       305; 293; 258; 248; 219; 215; 173; 160; 98; 88; 82; 69; 65; 40; 30; 18;
                                                        9; 5; 3; 1]; ...|]

reseau_analyse2 : reseau =
      [|[947; 920; 893; 807; 786; 784; 770; 715; 708; 685; 684; 676; 667; 650;
          617; 537; 513; 485; 447; 404; 350; 327; 325; 315; 297; 291; 286; 285;
      277; 268; 232; 218; 195; 192; 176; 170; 169; 159; 151; 125; 101; 55; 54;
                          32; 31; 28; 27; 21; 19; 13; 9; 7; 5; 4; 2; 1]; ...|]

- : int vect =
  [|0; 671; 170; 68; 26; 17; 10; 9; 3; 4; 6; 2; 2; 4; 1; 0; 0; 0; 0; 1;
      0; 0; 0; 0; 0; 1; 1; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 1; 0; 0; ...|]

- : int vect =
   [|0; 0; 501; 203; 89; 70; 28; 29; 12; 14; 7; 8; 8; 3; 3; 3; 1; 1; 3;
   3; 1; 1; 0; 0; 1; 1; 0; 1; 0; 1; 0; 2; 0; 0; 0; 1; 0; 0; 0; 0; ...|]
```

Par soucis de clarté, les tableaux sont tronqués à partir de l'endroit où ils n'affichent plus de chiffres significatifs. On remarque quelques rares noeuds à très hauts degrés, les hubs.
Utilisons le logiciel de calcul formel Maple pour tracer les graphiques simultanés de la fonction prévue théoriquement et des points repérés expérimentalement.

$$P(k) = \frac{2 \cdot d^2 \cdot t}{m_0 + t} \cdot \frac{1}{k^3}$$

Nous voyons sur les graphiques suivants que la loi trouvée théoriquement est belle et bien vérifiée par cet algorithme.
Rappelons que pour le premier programme, *d=1*, alors que pour le second, *d=2*. Par soucis de clarté, nous ne tracerons que les données pour les degrés inférieurs à 15.

ANALYSE GRAPHIQUE DES RÉSEAUX GÉNÉRÉS

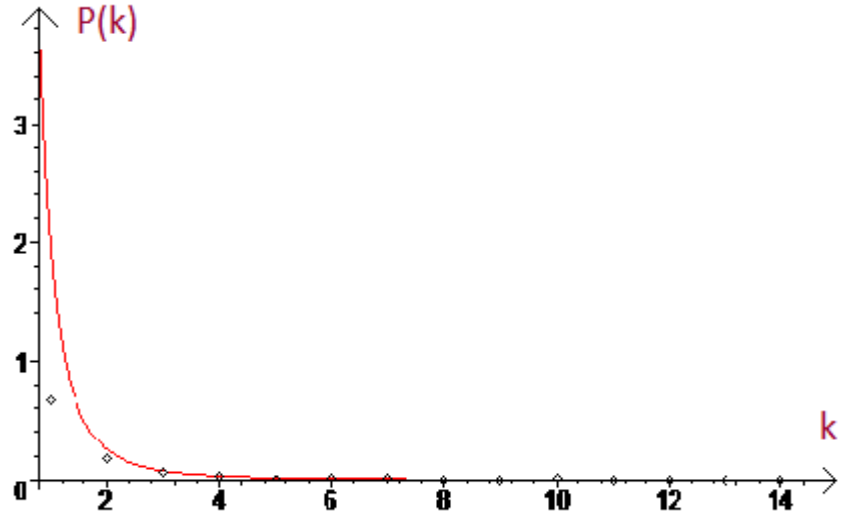
```
theorie := unapply(2*1**2*1998/(2000*k**3),k);
theorie2 := unapply(2*2**2*1998/(2000*k**3),k);
with(plots);
```

$$theorie := k \rightarrow \frac{999}{500}\frac{1}{k^3}$$

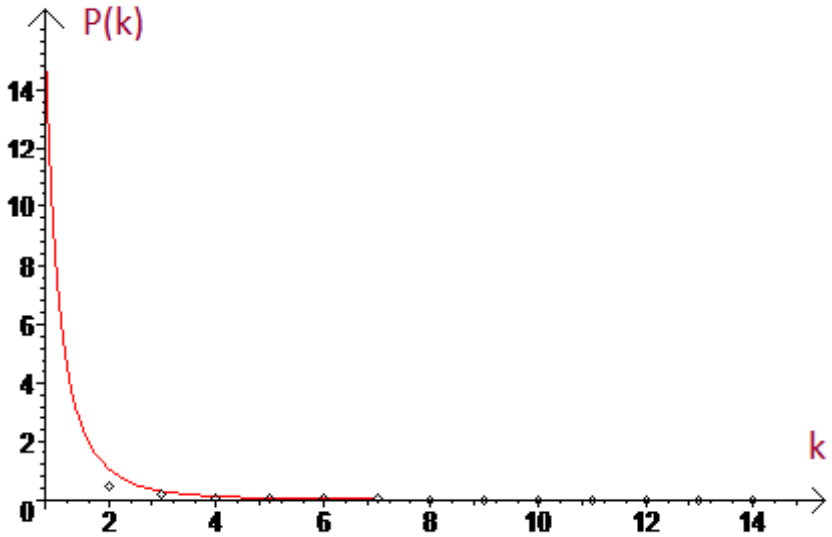
$$theorie2 := k \rightarrow \frac{999}{125}\frac{1}{k^3}$$

```
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal,
conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display, fieldplot,
fieldplot3d, gradplot, gradplot3d, graphplot3d, implicitplot, implicitplot3d, inequal, interactive,
interactiveparams, intersectplot, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d,
loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d,
polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, rootlocus, semilogplot,
setcolors, setoptions, setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d,
tubeplot]
```

```
display(plot({[[1,0.671]],[[2,0.17]],[[3,0.068]],[[4,0.026]],[[5,0.017]],
[[6,0.01]],[[7,0.009]],[[8,0.003]],[[9,0.004]],[[10,0.006]],[[11,0.002]],
[[12,0.002]],[[13,0.004]],
[[14,0.001]]},style=point,color=black),plot(theorie,0.8..15));
```



```
display(plot({[[2,0.501]],[[3,0.203]],[[4,0.089]],[[5,0.07]],
[[6,0.028]],[[7,0.029]],[[8,0.012]],[[9,0.014]],[[10,0.007]],
[[11,0.008]],[[12,0.008]],[[13,0.003]],[[14,0.003]],
[[15,0.003]]},style=point,color=black),plot(theorie2,0.8..15));
```



Les résultats sont globalement corrects, mais il montrent un léger décrochage pour des valeurs de degré faible. La crainte surgit alors que le réseau créé ne soit pas entièrement “Scale-Free”. Pour l'apaiser, cherchons une loi de puissance qui pourrait gouverner cette distribution de degré.

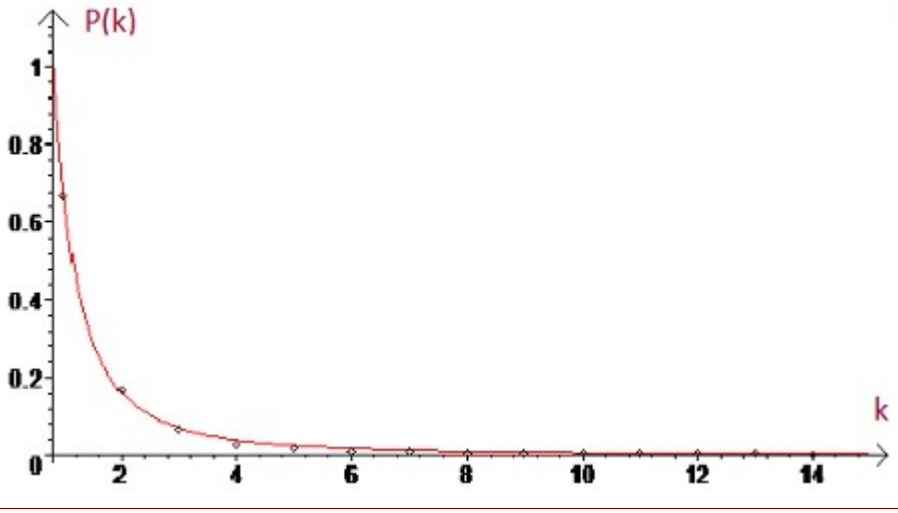
RECHERCHE D'UNE LOI DE PUISSANCE SUR LES EXEMPLES

```
inconnu:=unapply(a*k**b,k);
solve ({inconnu(1)=0.671,inconnu(10)=0.006});
```

$$inconnu := k \to a\,k^{b}$$

$$\{b = -2.048571270\,,\ a = 0.6710000000\ \}$$

```
b := -2.048571270; a := .6710000000;
display(plot({[[1,0.671]],[[2,0.17]],[[3,0.068]],[[4,0.026]],
[[5,0.017]],[[6,0.01]],[[7,0.009]],[[8,0.003]],[[9,0.004]],
[[10,0.006]],[[11,0.002]],[[12,0.002]],[[13,0.004]],
[[14,0.001]]},style=point,color=black),plot(inconnu,0.8..15));
```

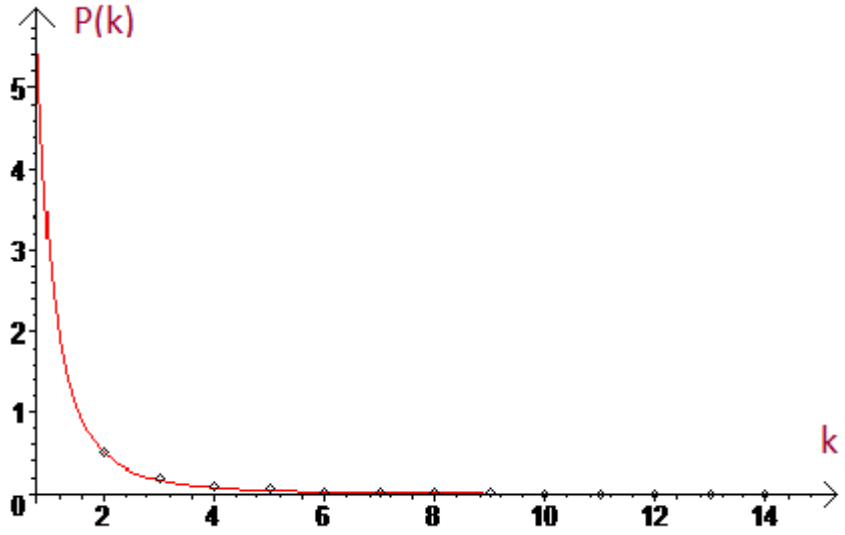


```
inconnu2:=unapply(c*k**d,k);
solve ({inconnu2(2)=0.501,inconnu2(6)=0.028});
```

$$inconnu2 := k \to c\,k^{d}$$

$$\{c = 3.091652096\ ,\ d = -2.625495473\ \}$$

```
d := -2.625495473; c := 3.091652096;
display(plot({[[2,0.501]],[[3,0.203]],[[4,0.089]],[[5,0.07]],
[[6,0.028]],[[7,0.029]],[[8,0.012]],[[9,0.014]],[[10,0.007]],
[[11,0.008]],[[12,0.008]],[[13,0.003]],[[14,0.003]],
[[15,0.003]]},style=point,color=black),plot(inconnu2,0.8..15));
```



Cette dernière opération nous prouve que, même si ces réseaux ne sont pas exactement conformes aux prévisions, ils répondent parfaitement à une loi de puissance, donc à une caractérisation “Scale-Free”, mais avec un exposant plus proche de 2.

Une autre manipulation intéressante consiste à repérer le degré maximal présent dans le réseau simulé, afin d'estimer la grosseur du hub le plus important. Le programme suivant en permet une mesure relativement simple. Par soucis de clarté, nous avons choisi d'entrer dans ce programme la taille du réseau à analyser diminuée de 1 pour en avoir l'indice du dernier noeud. Ceci évite le recours à une sous fonction peu utile et encombrante :

MESURE DU DEGRÉ MAXIMAL

```
let rec degree_max (reseau:reseau) (taille:int) =
  match taille with
   |0 -> 0
   |n -> max (degree_max reseau (taille-1))
             (list_length (reseau.(taille)));;
   (*on rétrécit à chaque tour la taille du réseau à étudier, en
   parcourant le réseau de la fin vers le début*)
```

```
degree_max : reseau -> int -> int = <fun>
```

```
degree_max reseau_analyse 999;;
```

```
- : int = 43
```

```
degree_max reseau_analyse2 999;;
```

```
- : int = 79
```

Un réseau de type “Scale-Free” présente des hubs au degré impressionant : il n'apparait donc pas surprenant que le chemin d'un noeud à l'autre, transitant par ces hubs, soit en général très court. En effet, ces réseaux sont supposés vérifier la **théorie des 6 degrés**, c'est à dire qu'un noeud est relié à n'importe quel autre par le biais d'au maximum 6 intermédiaires.
Ce constat pour le moins déroutant à fait couler beaucoup d'encre; notemment par le biais de l'acteur américain Kevin Bacon ou du mathématicien Erdős, qui se plaisaient à repérer combien de degrés les séparait des autres dans les différents réseaux présentés en exemple : Hollywood et les publications scientifiques. Cependant, il n'avait jamais été clairement montré avant cet été.
Les chercheurs Horvitz et Leskovec ont analysé le réseau social formé par le logiciel de messagerie instantanée *Windows Live Messenger* pour en conclure que les individus étaient tous connectés par une moyenne de 6,6 degrés.
Un algorithme simple permet de constater si nos réseaux répondent ou non à cette règle. Pour cela, nous nous permettrons de créer un réseau beaucoup plus vaste, de 10 000 noeuds. Nous y injecterons une information dans un noeud au hasard, et le programme simulera sa propagation à travers tous les liens pour finalement nous indiquer en combien de liens l'information a conquis tout le réseau.

Vérification de la théorie des 6 degrés

```
let information (reseau:reseau) =
  let taille = (vect_length reseau) in
  let interrupteur = ref 0
        (*l'interrupteur permettra de sortir de la boucle*)
     and compteur = ref 0 (*compteur de tours*)
     and information = make_vect taille 0 in
        (*tableau récapitulant les noeuds qu'a déjà atteint
          l'information. C'est comme cela que je modéliserai une
          information*)

      information.(random__int taille) <- 1;
        (*source de l'information, aléatoire*)
      while !interrupteur = 0 do
        interrupteur := 1;
         (*par défaut, on considère que l'information a atteint tous
           les noeuds*)

         for i=0 to taille - 1 do (*on parcours ensuite le réseau*)
           if information.(i) = 1
              then (propager reseau.(i) information;)
                 (*si un noeud possède l'information, il la transmet à
                  ses voisins via la sous-fonction ajouter*)
             else (interrupteur :=0);
                 (*dès que l'on trouve un noeud non contaminé, c'est
                   qu'il faut recommencer un tour de propagation*)
        done;
        incr compteur;
       done;
   !compteur;
     (*on retourne le compteur*)

   where rec propager (source:noeud) (information:int vect) =
      match source with
       |[] -> ()
       |t::q -> information.(t) <- 1;propager q information;;;;
(*cette petite sous-fonction étudie un noeud pour 'allumer' les noeuds
auxquel il est lié dans le tableau information*)
```

```
information : reseau -> int = <fun>
```

```
let propagationa = creer_reseau 10000;;
information propagationa;;
```

```
propagationa : reseau = [|...|]
- : int = 4
```

```
let propagationb = creer_reseau 10000;;
information propagationb;;
```

```
propagationb : reseau = [|...|]
- : int = 9
```

```
let propagation2a = creer_reseau 10000;;
information propagation2a;;
```

```
propagation2a : reseau = [|...|]
- : int = 6
```

```
let propagation2b = creer_reseau 10000;;
information propagation2b;;
```

```
propagation2b : reseau = [|...|]
- : int = 5
```

Les réseaux créés ne vérifient la théorie des 6 degrés que partiellement. Le second modèle semble être beaucoup plus fidèle à la réalité, peut-être parce que la loi de puissance des réseaux sociaux réelles est plus proche de l'exposant *-2.6* que du *-2* trouvé pour le premier modèle. Cependant, nous pouvons tout de même constater que, dans les deux cas, un nombre restreint d'intermédiaires suffit à la contamination du réseau par une information. Ceci est dû à la présence des hubs qui facilitent le transit dans un réseau “Scale-Free”.

C'est là tout l'intérêt de l'étude de ces réseaux : si la propagation d'une information est extrèmement facile, celle d'un virus devrait l'être également. Des experts se penchent donc sur ces théories pour trouver les meilleures façons d'immuniser un réseau contre une attaque. Il semble évident que la clé d'une bonne protection est la protection des hubs, mais il est parfois difficile de les repérer dans la réalité, spécialement dans les réseaux humains.

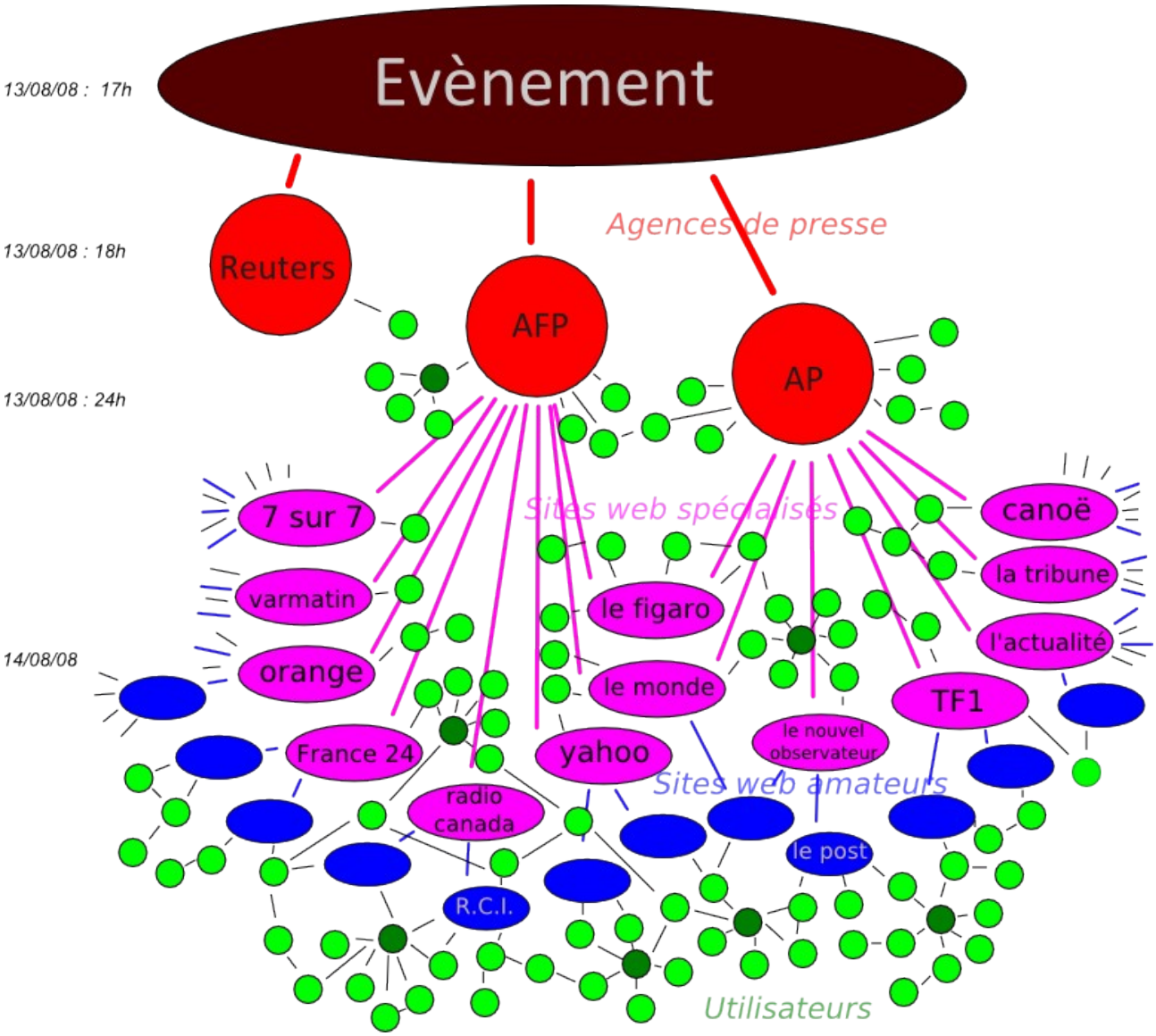
Les applications de telles stratégies pourraient être nombreuses, aussi bien pour protéger les ordinateurs de la propagation d'un virus informatique que protéger les populations humaines de diverses épidémies.

Nous avons essayé de repérer cette structure “Scale-Free” du web en étudiant le **trajet d'une information sur la toile**. Bien entendu, cette entreprise n'est que très partielle, étant donné qu'une information est très difficile à suivre sur internet : les sites sont extrèmement nombreux, et nous ne pouvons en présenter sur le schéma récapitulatif qu'un maigre échantillon.

Pour ce faire, nous avons choisi totalement aléatoirement une dépêche d'actualité : un article sur l'assassinat du président du parti démocrate de l'Arkansas. De nombreux sites célèbres ne font que reprendre intégralement l'article d'une agence de presse, mais certains sites personnels ou de journalisme amateur proposent une version modifiée de l'article, ce qui nous permet de faire apparaitre la hiérarchisation visible sur le schéma de la page suivante.

L'information est transcrite par des agences de presse puis les articles sont largement diffusés via des sites populaires spécialisés dans l'information. Certains sites personnels et blogs, ici en bleu, reprennent l'information, la commentent, la détaillent. Bien que divers, nous les qualifierons d'amateurs par contraste avec les sites spécialisés et reconnus. Tout au long de ce trajet, les

utilisateurs peuvent consulter les dépêches et informer leurs proches de l'information. Elle continue de se transmettre par les liens sociaux, représentant les amitiés, les contacts de mail ou de messagerie instantanée... On remarque dans la population des personnes plus influentes, et des regroupements entre visiteurs de mêmes sites qui se manifestent et interragissent notamment dans les commentaires des informations. On voit que l'information se répend rapidement, via peu d'intermédiaires. Il est à noter que la répartition des utilisateurs est probablement disproportionnée par soucis de lisibilité du schéma : beaucoup plus de personnes consultent les sites spécialisés que les sites amateurs.



Partie III

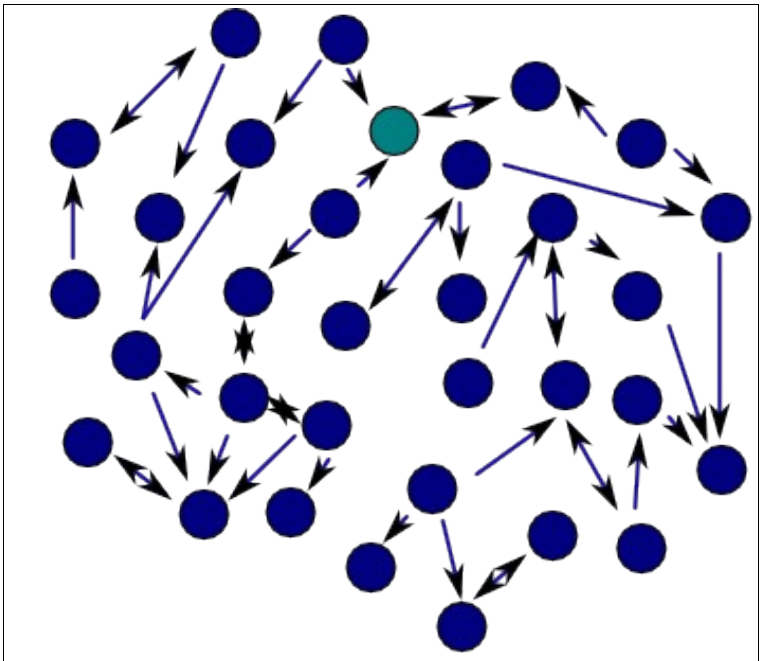
Réseaux non-réciproques

L'algorithme créé apparait donc comme convaincant : il forme un réseau aléatoire de type “Scale-Free” qui répond à une loi de puissance et présente d'importants hubs, particulièrement dans le cas de noeuds de degré 2 ajoutés.

Mais si la structure du World Wide Web apparait comme “Scale-Free”, il faut reconnaitre que le modèle n'est pas si simple : les liens sur les pages web sont rarement réciproques (on cite ses sources, mais l'inverse est extrèmement rare). Il faut alors se pencher sur un nouveau type de réseau, les réseaux **non réciproques**.

Dans de tels réseaux, les liens sont comme des flèches, à sens unique ou non. Le degré d'un noeud serait l'ensemble des flèches à la fois entrant et partant du noeud. On peut alors définir deux notions importantes dans les réseaux non réciproques : le **degré entrant** d'un noeud, qui est le nombre de connexion arrivant au noeud, et son **degré sortant**, qui qualifie le nombre de connexions dont il est à l'origine.

La définition du caractère “Scale-Free” apparait alors comme analogue à celle d'un réseau réciproque : pour qu'un réseau non réciproque soit de type “Scale-Free”, il faut que **sa distribution de degré entrant** et **sa distribution de degré sortant** répondent à une loi de puissance.



Réseau aléatoire non réciproque

A titre d'exemple, le noeud cyan a pour degré entrant 3 et pour degré sortant 1, ce qui donne un degré total de 4.

L'algorithme créé donnant des tableaux dont la répartition est de type “Scale-Free”, il apparait que la génération d'un réseau “Scale-Free” non réciproque peut s'obtenir à partir de légères modifications de l'algorithme. Nous ajouterons des noeuds de degré entrant et de degré sortant tout deux égaux à 2, car ce chiffre semble donner des résultats plus convainquants.

Cependant, ce travail recquiert une nouvelle définition de données représentatives de la direction des liens, donc un nouveau fichier CAML. Nous choisirons d'utiliser un nouveau type de noeud qui serait une alliance de deux listes, une représentant les liens entrants dans le noeud et l'autre les

liens sortants.

Définitions de nouveaux types

```
type indice == int;;
type noeud = {mutable e:indice list;mutable s:indice list};;
type reseau == noeud vect;;
```

```
Type indice defined.
Type noeud defined.
Type reseau defined.
```

Exemple de réseau non réciproque

```
exemple : reseau2 =
[|{e = [3; 2; 1]; s = [2; 1]}; {e = [2; 0]; s = [3; 2; 0]};
 {e = [3; 1; 0]; s = [3; 1; 0]}; {e = [2; 1]; s = [2; 0]}|]
```

Au vu de cette nouvelle définition, il nous faut réécrire les fonctions préliminaires :

Compteur du noeud correspondant au nombre tiré aléatoirement

```
let rec compte (reseau:reseau) (noeud:indice) (nombre:int)
(direction:char) =
 let longueur = list_length (if direction=`e`
                               then (reseau.(noeud)).e
                               else (reseau.(noeud)).s) in
                   (*c'est ici que s'opère le changement principal :
                     on parcours soit les listes de liens entrant,
                     soit les listes de liens sortant, selon le
                     caractère `e` ou `s` passé en paramètre*)
 match nombre with
  |0 -> noeud
  |n when (n < longueur) -> noeud
  |n -> compte reseau (noeud+1) (nombre - longueur) direction;;
```

```
compute : reseau -> indice -> int -> char -> indice = <fun>
```

La fonction insere sera simplement maniée habilement pour éviter la réécriture.

Insertion d'un indice dans un noeud existant

```
let rec insere (indice:indice) (list:indice list) =
match list with
   |[] -> [indice]
   |t::q when t>indice -> t:: (insere indice q)
   |liste -> indice :: liste;;
```

```
insere : indice -> indice list -> indice list = <fun>
```

Grace à ces outils, le programme peut être modifié de la façon suivante.

PROGRAMME DE GÉNÉRATION D'UN RÉSEAU NON RÉCIPROQUE "SCALE-FREE"

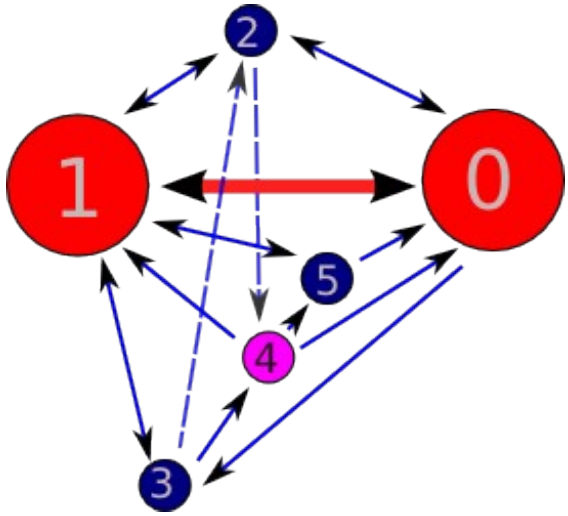
```
let creer_reseau (n:int) =
  let denominateur = ref 2 and denominateurs = ref 2 in
  (*deux références de dénominateurs seront nécessaires, une
    pour les liens entrants, et une pour les liens sortants*)
former n

  where rec former (n:int) = match n with
   |n when n <=1 -> failwith "Erreur de dimension"
   |2 -> [|{e=[1];s=[1]};{e=[0];s=[0]}|]
   (*pour des raisons pratiques, on commence avec un lien réciproque*)
   |n -> let precedent = former (n-1) in ajouter n  precedent

     where rec ajouter (n:indice) (precedent:reseau) =
  let e1 = (compte precedent 0 (random__int (!denominateur)) `e`)
  and e2 = ref (compte precedent 0 (random__int (!denominateur)) `e`)
  and  s1 = (compte precedent 0 (random__int (!denominateurs)) `s`)
  and s2 = ref (compte precedent 0 (random__int (!denominateurs)) `s`)
  (*on doit ici définir quatre noeuds choisis aléatoirement*)
  and nouveau = make_vect n {e=[];s=[]} in

       while e1= !e2 do
 e2 := (compte precedent 0 (random__int (!denominateur)) `e`);
       done;
       while s1= !s2 do
 s2 := (compte precedent 0 (random__int (!denominateurs)) `s`);
       done;
        (*si les liens peuvent être réciproques, il faut que les liens
          de même type soient dirigés vers des noeuds différents*)

nouveau.(n-1) <- {e=[max e1 !e2;min e1!e2];s=[max s1 !s2;min s1 !s2]};
(*le nouveau noeud*)
          for i=0 to n-2 do
                nouveau.(i) <- precedent.(i);
          done;
(*on recopie d'abord le tableau pour le modifier ensuite*)
           nouveau.(e1).s <- insere (n-1) (nouveau.(e1).s);
           nouveau.(!e2).s <- insere (n-1) (nouveau.(!e2).s);
           nouveau.(s1).e <- insere (n-1) (nouveau.(s1).e);
           nouveau.(!s2).e <- insere (n-1) (nouveau.(!s2).e);
           denominateure := !denominateure + 4;
           denominateurs := !denominateurs + 4;
     nouveau;;
```

creer_reseau : int -> reseau = <fun>

EXEMPLE DE SON FONCTIONNEMENT

```
let reseau = creer_reseau 6;;
```

```
reseau : reseau =
    [|{e = [5; 4; 2; 1]; s = [3; 2; 1]};
  {e = [5; 4; 3; 2; 0]; s = [5; 3; 2; 0]};
        {e = [3; 1; 0]; s = [4; 1; 0]};
           {e = [1; 0]; s = [4; 2; 1]};
           {e = [3; 2]; s = [5; 1; 0]};
             {e = [4; 1]; s = [1; 0]}|]
```



On obtient un réseau de type “Scale-Free” non réciproque. La structure semble chargée en connexions car il y a peu de noeuds. Nous pouvons, pour lui, procéder aux mêmes vérifications.

PROGRAMME D'ANALYSE DES RÉSEAUX NON RÉCIPROQUES

```
let analyse (reseau:reseau) =
 let entrant = make_vect (vect_length reseau) 0
 and sortant = make_vect (vect_length reseau) 0 in
   for i=0 to (vect_length reseau - 1) do
   entrant.(list_length (reseau.(i)).e) <-
                        entrant.(list_length (reseau.(i)).e) + 1;
   sortant.(list_length (reseau.(i)).s) <-
                        sortant.(list_length (reseau.(i)).s) + 1;
   (*on doit simplement retourner la paire de tableaux récapitulant
     les liens entrant et sortant, comptés au fur et à mesure*)
   done;
(entrant,sortant);;
```

```
analyse : reseau -> int vect * int vect = <fun>
```

CRÉATION ET ANALYSE DE RÉSEAUX “SCALE-FREE” NON RÉCIPROQUES

```
let reseau_analyse = creer_reseau 1000;;

analyse reseau_analyse;;
```

```
reseau_analyse : reseau =
                                                        [|{e =
        [945; 911; 872; 693; 524; 516; 488; 470; 414; 382; 380; 367; 320; 285;
       249; 240; 235; 228; 189; 181; 145; 111; 94; 80; 66; 57; 55; 29; 27; 15;
                                                        4; 3; 2; 1];
                                                        s =
        [917; 916; 898; 882; 773; 718; 702; 700; 668; 641; 612; 591; 572; 535;
       489; 473; 461; 313; 309; 259; 251; 238; 169; 142; 132; 106; 98; 93; 64;
                                                58; 48; 42; 34; 26; 17; 6; 2; 1]};
                                                        ...|]


                                                - : int vect * int vect =
    [|0; 0; 463; 218; 105; 73; 30; 24; 26; 10; 9; 3; 5; 6; 4; 2; 1; 1; 0; 0; 1;
     2; 1; 4; 4; 1; 0; 0; 0; 0; 0; 0; 1; 0; 2; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;
                                        0; 0; 1; 0; 1; 0; 0; 1; 0; 1; ...|],
    [|0; 0; 461; 210; 118; 55; 44; 33; 20; 11; 8; 7; 1; 2; 3; 2; 2; 4; 3; 2; 0;
     2; 0; 0; 2; 3; 0; 0; 0; 0; 0; 0; 1; 0; 0; 1; 0; 0; 1; 0; 0; 0; 0; 0; 0; 1;
                                0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 1; 1;  ...|]
```

RECHERCHE D'UNE LOI DE PUISSANCE SUR LES EXEMPLES NON-RÉCIPROQUES

```
entrant:=unapply(ea*k**eb,k);
sortant:=unapply(sa*k**sb,k);
solve ({entrant(2)=0.463,entrant(10)=0.009});
solve ({sortant(2)=0.461,sortant(10)=0.008});
```

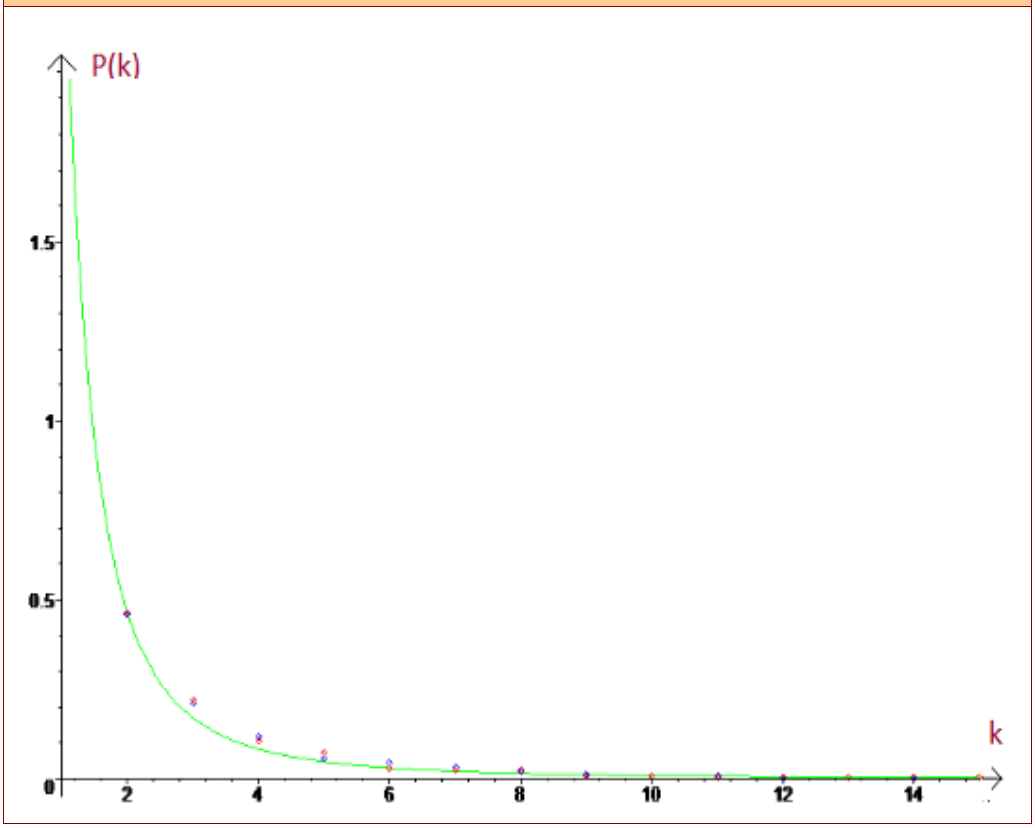
$$entrant := k \rightarrow ea\ k^{eb}$$

$$sortant := k \rightarrow sa\ k^{sb}$$

$$\{ea = 2.527053043\ ,\ eb = -2.448371849\ \}$$

$$\{sb = -2.518864797\ ,\ sa = 2.642133663\ \}$$

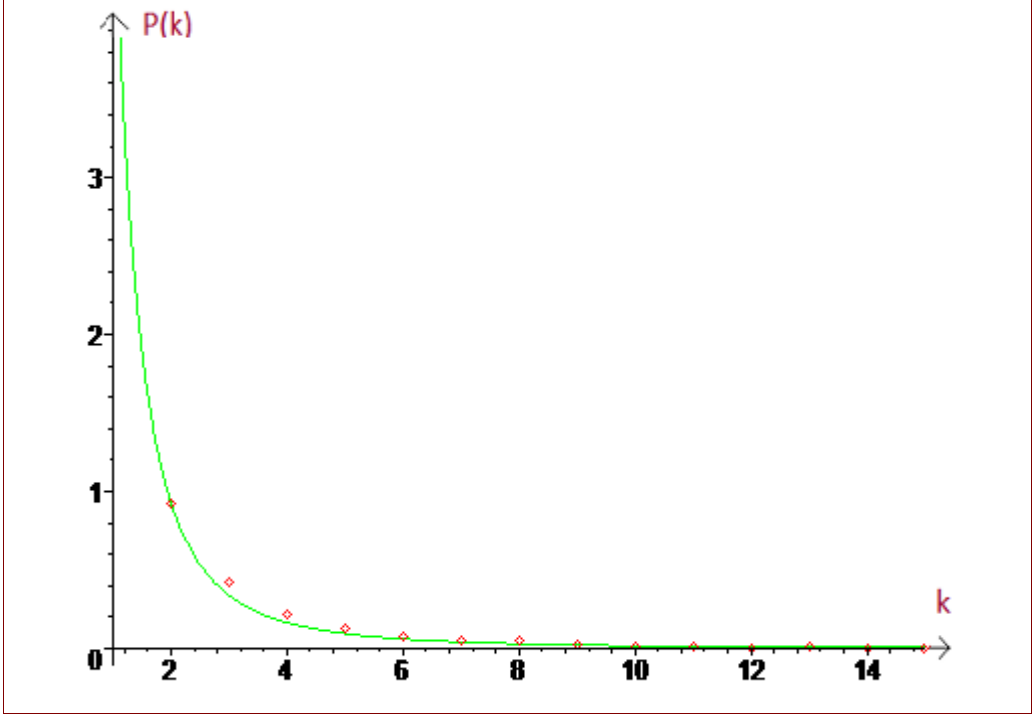
```
Dans la mesure ou les exposants sont relativement proches, nous
n'utiliserons qu'une courbe moyenne pour plus de clarté.
with(plots);
puissance:=unapply(a*k**b,k);
b := -(2.518864797+2.448371849)/2; a := (2.5270530434+2.642133663)/2;
display(
     plot({[[2,0.463]],[[3,0.218]],[[4,0.105]],[[5,0.073]],[[6,0.03]],
[[7,0.024]],[[8,0.026]],[[9,0.010]],[[10,0.009]],[[11,0.003]],
[[12,0.005]],[[13,0.006]],[[14,0.004]],
[[15,0.002]]},style=point,color=red),
    plot({[[2,0.461]],[[3,0.210]],[[4,0.118]],[[5,0.055]],[[6,0.044]],
[[7,0.033]],[[8,0.020]],[[9,0.011]],[[10,0.008]],[[11,0.007]],
[[12,0.001]],[[13,0.002]],[[14,0.001]],
[[15,0.003]]},style=point,color=blue),
    plot(puissance,1..15,color=green));
```



Les degrés entrants sont en rouge et sortants en bleu. La modélisation est en vert clair. La similitude des termes *sa* et *ea*, *sb* et *eb*, permet de confirmer que les degrés entrant et sortant suivent bien tous deux la même loi de puissance. On peut d'ailleurs remarquer que les données étaient relativement semblables. Le réseau créé est donc de type “Scale-Free”. On peut aller plus loin en montrant que, par conséquent, le degré total (somme des deux degrés) répond aussi à une loi de puissance qui serait le double de la première :

RECHERCHE D'UNE LOI DE PUISSANCE SUR LE DEGRÉ TOTAL

```
    plot({[[2,0.924]],[[3,0.428]],[[4,0.223]],[[5,0.128]],[[6,0.074]],
[[7,0.057]],[[8,0.046]],[[9,0.021]],[[10,0.017]],[[11,0.010]],
[[12,0.006]],[[13,0.008]],[[14,0.005]],
[[15,0.005]]},style=point,color=red),
    plot(2*puissance(k),k=1..15,color=green));
```



On remarque donc que la loi de puissance s'applique aussi à la distribution de degré total. Mesurons les degrés maximums en entrée et sortie.

MESURE DU DEGRÉ MAXIMAL DANS UN RÉSEAU NON RÉCIPROQUE

```
let rec degree_max (reseau:reseau) (taille:int) =
  match taille with
    |0 -> (0,0)
    |n -> let precedent = (degree_max reseau (taille-1)) in
     ((max (fst precedent) (list_length (reseau.(taille)).e)),
      (max (snd precedent) (list_length (reseau.(taille)).s)));;
   (*le programme renvoie la paire du degré maximal entrant suivi du
     degré maximal sortant. On procède par récurrence en épluchant le
     réseau de la fin au début.*)
```

```
degree_max : reseau -> int -> int * int = <fun>
```

```
degree_max reseau_analyse 999;;
```

```
- : int * int = 55, 59
```

On remarque donc la présence de hubs aussi bien pour les connexions entrantes que sortantes. Ils ne sont pas forcément les mêmes, ce qui correspond à la structure réelle du World Wide Web : en effet, une page célèbre comme un article de Wikipedia se verra beaucoup citée, mais ne proposera que de maigres liens sortants vers ses quelques sources. A l'inverse, les sites de fans, ou les forums, proposent de nombreuses pages qui ne sont pas toujours réputées : ils n'ont que peu

de liens entrant mais peuvent présenter beaucoup de connexions sortantes, en affichant des listes de sites favoris ou partenaires.

Il ne nous reste plus qu'à étudier la propagation d'une information dans un tel réseau. Nous procèderons, comme précédemment, par l'injection d'une information dans un noeud aléatoire. Il s'agira alors de considérer les connexions sortantes des noeuds touchés par l'information pour remarquer sa propagation. Le programme est très semblable au précédent, la nouvelle structure de données choisie étant adéquate.

VÉRIFICATION DE LA THÉORIE DES 6 DEGRÉS POUR UN RÉSEAU NON-RÉCIPROQUE

```
let information (reseau:reseau) =
  let taille = (vect_length reseau) in
  let interrupteur = ref 0
     and compteur = ref 0
     and information = make_vect taille 0 in

      information.(random__int taille) <- 1;

      while !interrupteur = 0 do
        interrupteur := 1;

         for i=0 to taille - 1 do
           if information.(i) = 1
              then (propager reseau.(i).s information;)
             else (interrupteur :=0);
         done;
         incr compteur;
       done;
   !compteur;

     where rec propager (source:indice list) (information:int vect) =
       match source with
        |[] -> ()
        |t::q -> information.(t) <- 1;propager q information;;;;
```

```
information : reseau -> int = <fun>
```

```
let propagationa = creer_reseau 10000;;
information propagationa;;
```

```
propagationa : reseau = [|...|]
- : int = 3
```

```
let propagationb = creer_reseau 10000;;
information propagationb;;
```

```
propagationb : reseau = [|...|]
- : int = 3
```

```
let propagationc = creer_reseau 10000;;
information propagationc;;
```

```
propagationc : reseau = [|...|]
- : int = 3
```

Il suffit d'un nombre très restreints d'intermédiaires pour faire partager l'information à l'intégralité du réseau. Les résultats expérimentaux donnent une moyenne de 3 degrés sur plusieurs executions (un très grand nombre d'executions du programme a donné un résultat constant). Remarquons que les réseaux sont de taille faible par rapport aux réseaux réels : 10 000 noeuds. Une simulation sur un nombre encore plus important permettrait de vérifier la théorie des 6

degrés sur un réseau aléatoire plus semblable au réseau réel. En effet, la population mondiale d'internet est estimée à 1 450 000 000 pour le moment (*Nielsen//Netratings*), et la population totale mondiale à 6 675 000 000 (*US Census Bureau*). Pour une telle simulation, des capacités de calcul beaucoup plus puissantes sont nécessaires.

Conclusion

Des réseaux polyvalents

L'étude des réseaux présents dans la nature ont prouvé l'omniprésence du modèle “Scale-Free”. Ils semblent régir la formation des sociétés, mais sont aussi présents dans de nombreux domaines. Notre moteur de recherche pourrait être ainsi adapté, par exemple, pour trouver une publication populaire et fiable parmi les nombreuses publications scientifiques, ou une personne influente dans les réseaux sociaux.

Les **applications sont diverses**, et c'est l'intérêt de l'étude de ces réseaux. Nos programmes de simulation permettent de vérifier la théorie des 6 degrés, et pourraient sans problème être utilisés à d'autres fins, dans l'optique de **vérifier les théories établies par l'étude de ces réseaux**. Par exemple, des experts se penchent sur la question de la stratégie la plus efficace pour **immuniser un réseau contre une attaque** (réseau humain contre une épidémie, réseau virtuel contre un virus). Il est i**mpossible de repérer les hubs dans la réalité**, il faut donc ruser. Notre programme permet de **comparer l'efficacité de différentes méthodes,** avec beaucoup plus de facilité pour effectuer les test que sur une population humaine (spécialement au niveau des épidémies).

Bibliographie

Topologies et cartes d'internet et du web :

CAIDA (copyright UC Regents)
http://www.caida.org/research/topology/as_core_network/pics/ascore-simple.2008_big.png

Walrus
http://www.caida.org/tools/visualization/walrus/

Plankton
http://www.caida.org/tools/visualization/plankton/

Projet Opte
http://bitcast-a.bitgravity.com/blyon/opte/maps/static/1105841711.LGL.2D.1024x1024.png

Stephen Coast
http://www.fractalus.com/steve/stuff/ipmap/

Autres documents iconographiques :

Photographies, dessins et notes biographiques
http://fr.wikipedia.org

photo de Albert-László Barabási :
http://www.bnl.gov/bnlweb/pubaf/pr/PR_display.asp?prID=08-51

photo Réka Albert :
http://www.phys.psu.edu/~ralbert/

Propagation de l'information sur internet :
La mort du président du parti démocrate de l'Arkansas

Reuters
http://www.reuters.com/article/newsOne/idUSN1337847020080813

AP
http://hosted.ap.org/dynamic/stories/O/OBIT_GWATNEY?SITE=NCKIN&SECTION=HOME&TEMPLATE=DEFAULT

AFP
http://www.afp.com/francais/home/

7 sur 7
http://www.7sur7.be/7s7/fr/1501/Canal-Infos/article/detail/380345/2008/08/13/Le-president-du-parti-democrate-de-l-Arkansas-blesse-par-balles.dhtml

Varmatin
http://www.varmatin.com/ra/monde/139921/etats-unis-le-president-du-parti-democrate-de-l-arkansas-blesse-par-balles?utm_source=wikio&utm_medium=digg&xtor=AL-201&

Orange
http://actu.orange.fr/Inc/archives/article.php?file=journal%2Fmon%2Fnewsmlmmd.95efbbbb3c7661a36bc07a36b33179ad.3d1.xml&id=92673

France 24
http://www.france24.com/fr/20080813-bill-gwatney-mort-chef-democrates-arkansas-fusillade-bill-clinton

Radio Canada
http://www.radio-canada.ca/nouvelles/International/2008/08/13/008-Gwatney-blesse.shtml

RCI : Radio Canada International
http://www.rcinet.ca/rci/fr/actualite.shtml

Yahoo
http://fr.news.search.yahoo.com/search/news?fr=news_sb_hd&source=yahoo&c=yahoo_news&p=Gwatney&ei=windows-1252

Le Monde
http://www.lemonde.fr/elections-americaines/article/2008/08/14/le-president-du-parti-democrate-de-l-arkansas-meurt-dans-une-fusillade_1083455_829254.html

Le Figaro
http://www.lefigaro.fr/international/2008/08/14/01003-20080814ARTFIG00246-le-president-du-parti-democrate-de-l-arkansas-abattu-.php

Le Nouvel Observateur
http://tempsreel.nouvelobs.com/depeches/international/ameriques/20080814.FAP0220/le_president_du_parti_democrate_de_larkansas_abattu_dan.html

Le Post
http://www.lepost.fr/article/2008/08/14/1243832_etats-unis-le-president-du-parti-democrate-de-l-arkansas-assassine-dans-ses-bureaux.html

TF1
http://tf1.lci.fr/infos/monde/ameriques/0,,3940277,00-fusillade-au-siege-du-parti-democrate-de-l-arkansas-.html

L'actualité
http://www.lactualite.com/nouvelles/monde/article.jsp?content=M081385AU

La Tribune
http://www.latribune.fr/info/Le-president-du-Parti-democrate-de-l-Arkansas-abattu-dans-ses-bureaux-777-~-AP-USA-DEMOCRATE-ARKANSAS-FUSILLADE-$Db=News/News.nsf-$Channel=Monde

Canoë
http://www.canoe.com/infos/international/archives/2008/08/20080813-181249.html

Principaux documents scientifiques

Modèle d'attachement préférentiel
http://en.wikipedia.org/wiki/Barabasi-Albert_model

Théorie des réseaux Scale-Free (principaux documents)

Mean-eld theory for scale-free random networks
Albert-Laszlo Barabasi, Reka Albert, Hawoong Jeong
Department of Physics, University of Notre-Dame, Notre-Dame, IN 46556, USA
Received 1 July 1999

Scale-Free information system networks
Wee Horng Ang (B. SC. (Hons) Computer Engineering, University of Illinois at Urbana-Champaign 2004)

Théorie des 6 degrés :
http://www.infos-du-net.com/actualite/14179-messenger-microsoft-degre.html
http://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93Bacon_number
http://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon
Loi de pareto et loi de puissance :
http://fr.wikipedia.org/wiki/Distribution_de_Pareto
http://fr.wikipedia.org/wiki/Loi_de_Pareto
http://fr.wikipedia.org/wiki/Vilfredo_Pareto
http://en.wikipedia.org/wiki/Power_law
Théorie basique des réseaux :
http://en.wikipedia.org/wiki/Scale_free_network
http://en.wikipedia.org/wiki/Degree_distribution

Chiffres :

estimation de la population internet totale :
www.worldinternetstats.com, de Nielsen//Netratings via http://www.slideshare.net/researchreinvented/the-world-of-internet-internet-usage-statistics/
Population mondiale :
US census bureau

Logiciels utilisés :

Rédaction : Open Office
Editeur CAML : Wincaml
Calcul Formel : Maple 10
Retouche et créations graphiques : Paint, Inkscape, Gimp