Routage avec indépendance des noms

Cyril Gavoille
LaBRI
Université Bordeaux 1

Projet `PairApair` ACI Masse de données
23 janvier 2004 – Aussois

PLAN

1. Introduction : problème, complexité, exemples

2. Facteur d'étirement moyen : critère mesurant l'évolution d'Internet ?
   (Krioukov, Fall, Yang – INFOCOM 2004)

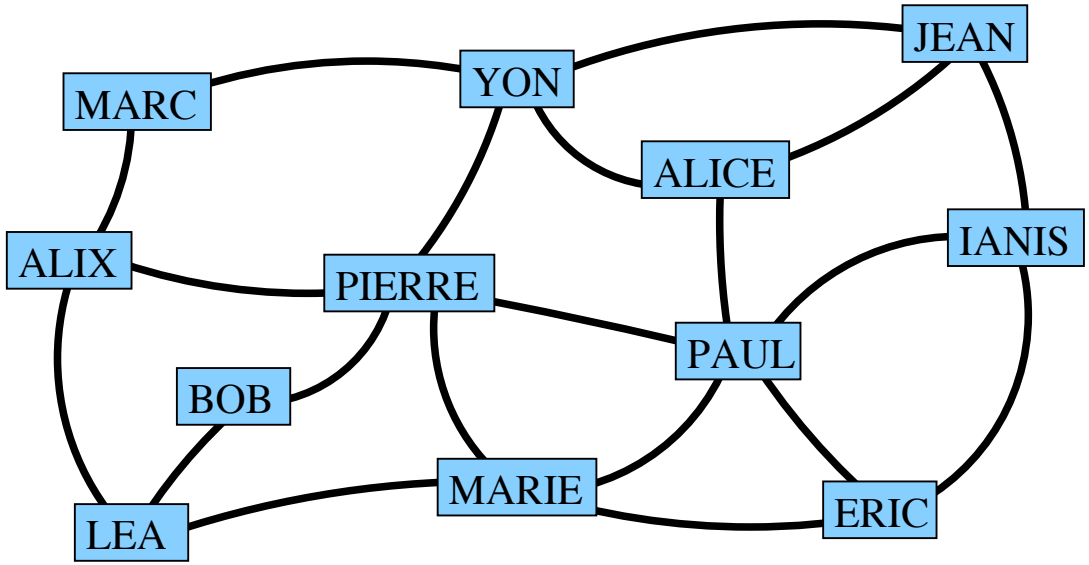
3. Indépendance des noms : schéma d'*Awerbuch, Bar-Noy, Linial et Peleg (J. of Algorithms, 1990)*

4. Une nouvelle solution
   (en commun avec Abraham, Malkhi, Nisan)

1. Introduction

Router ?

Trouver sa route dans un réseau d'entités interconnectées

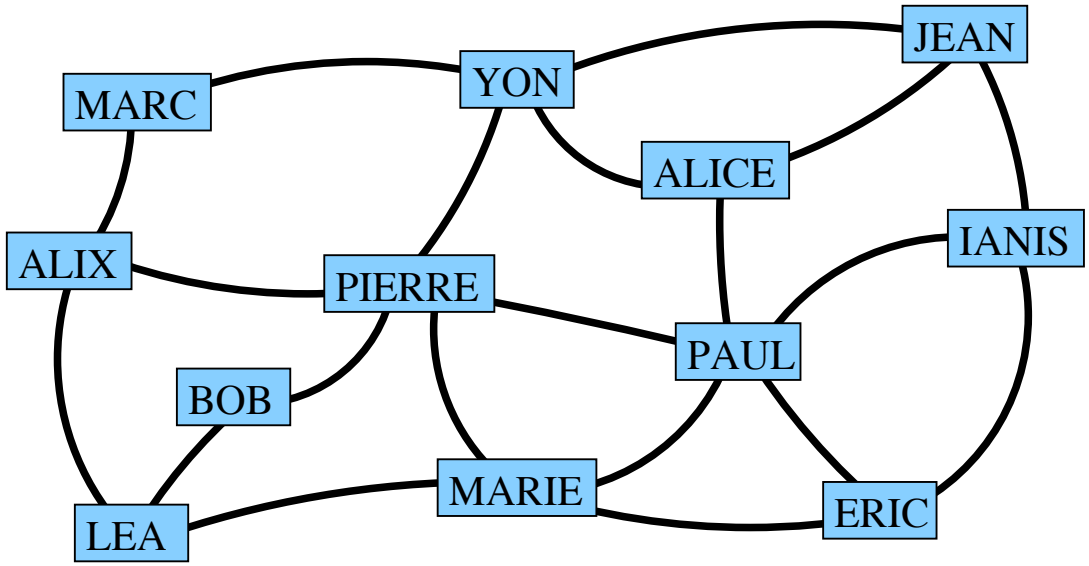


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

On s'intéresse ici au processus de bas niveau permettant de transmettre une information d'une entité à une autre via le réseau (pair à pair, entre toutes paires)

Router ?

Trouver sa route dans un réseau d'entités interconnectées

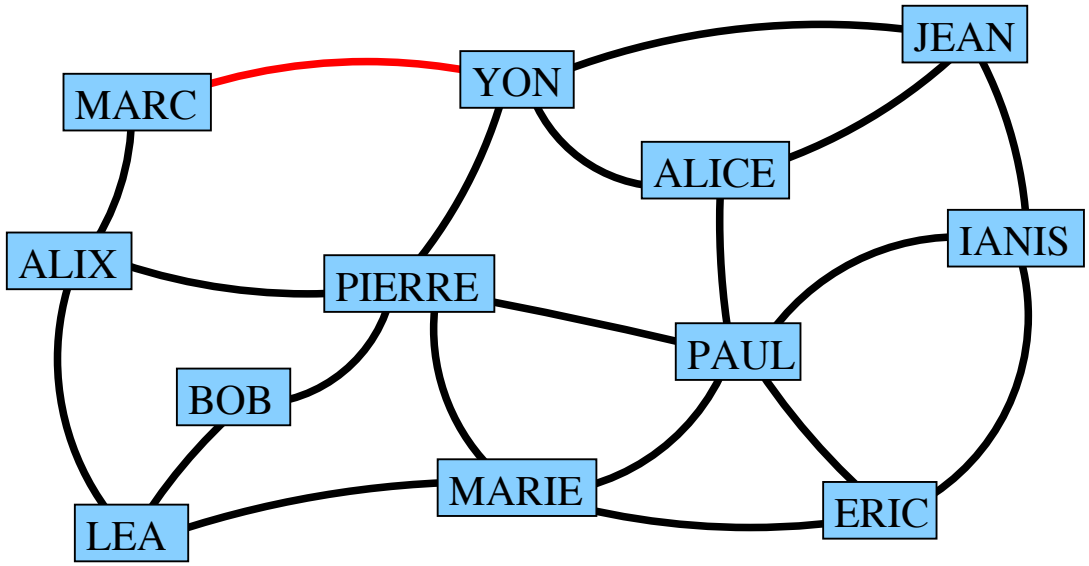


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
MARC: IANIS?
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

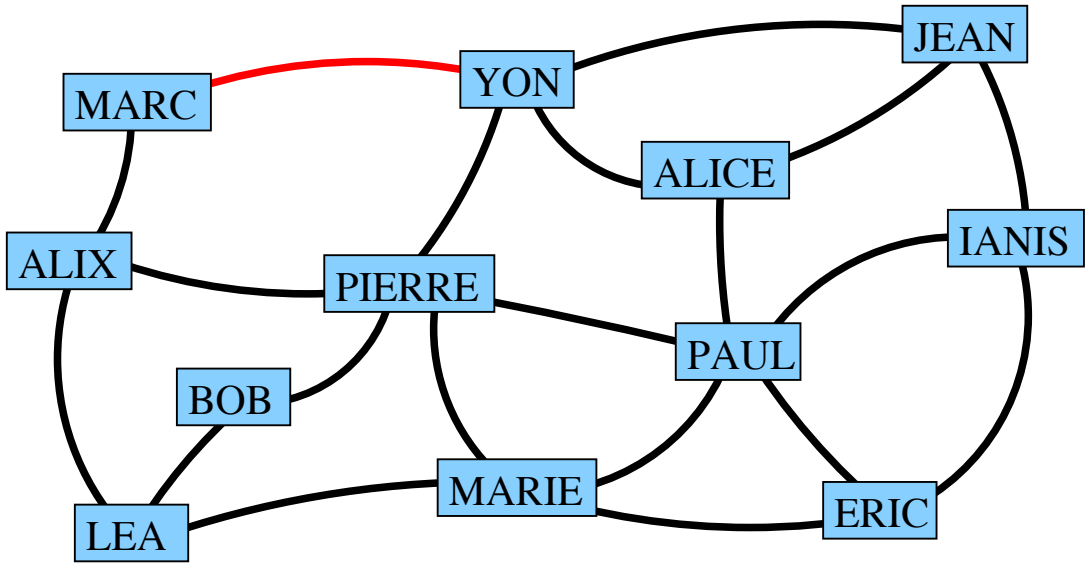


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
MARC: IANIS? > aller vers le voisin de droite
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

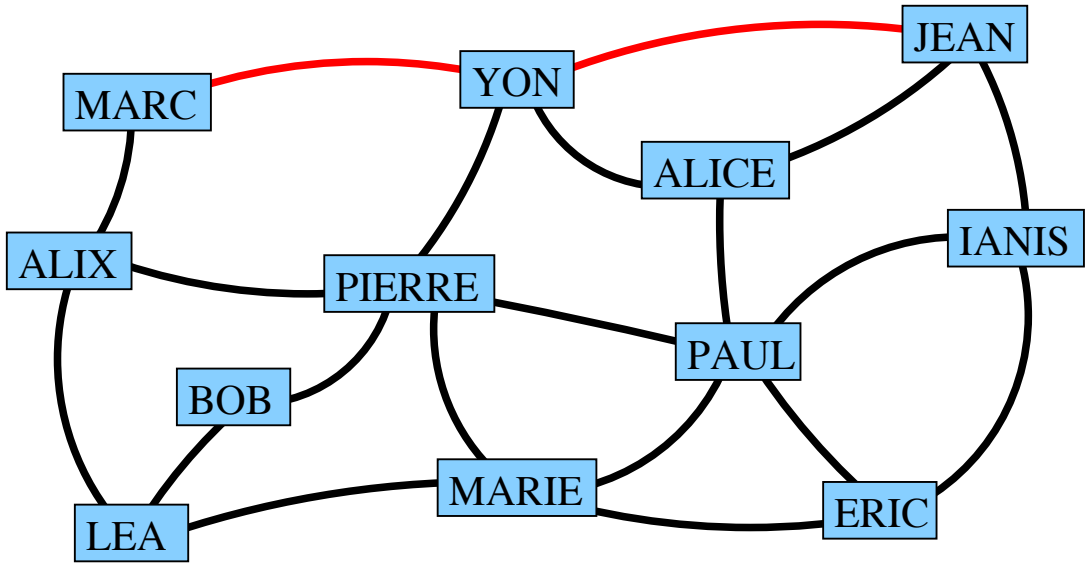


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
MARC: IANIS? > aller vers le voisin de droite
 YON: IANIS?
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

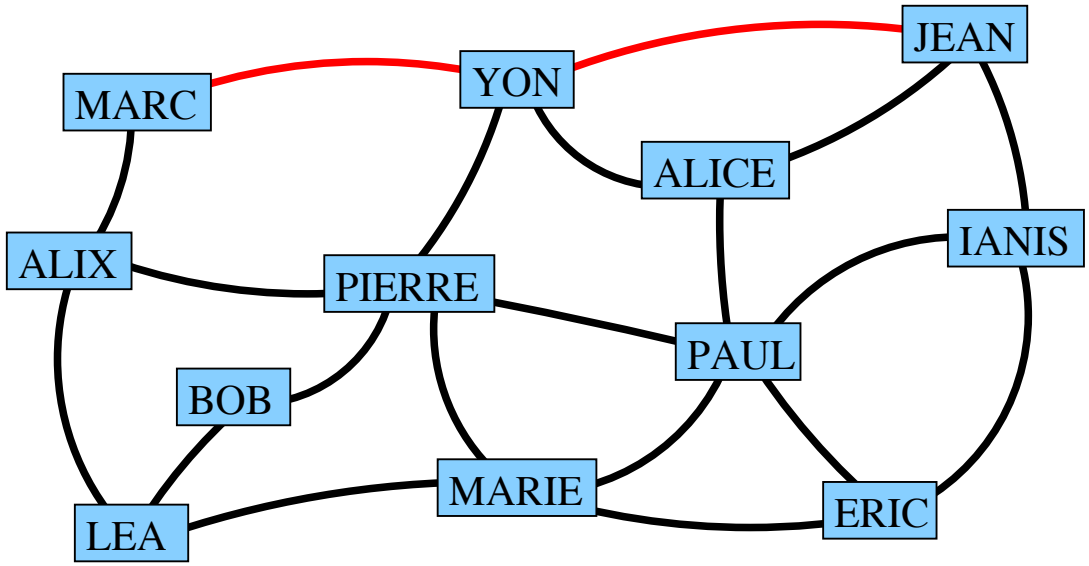


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
MARC: IANIS? > aller vers le voisin de droite
 YON: IANIS? > aller vers le voisin du haut
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

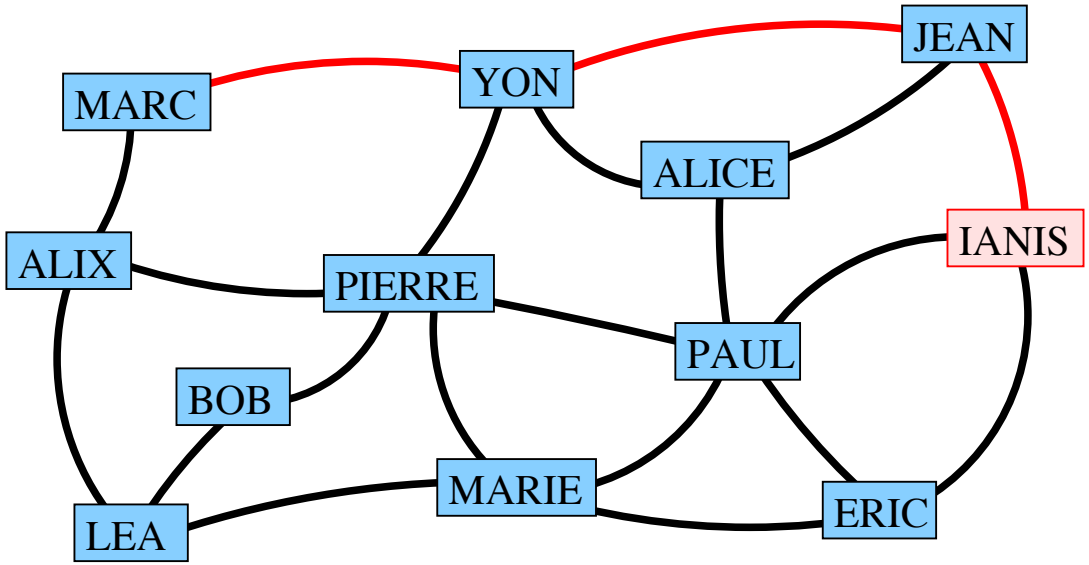


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
MARC: IANIS? > aller vers le voisin de droite
 YON: IANIS? > aller vers le voisin du haut
JEAN: IANIS?
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

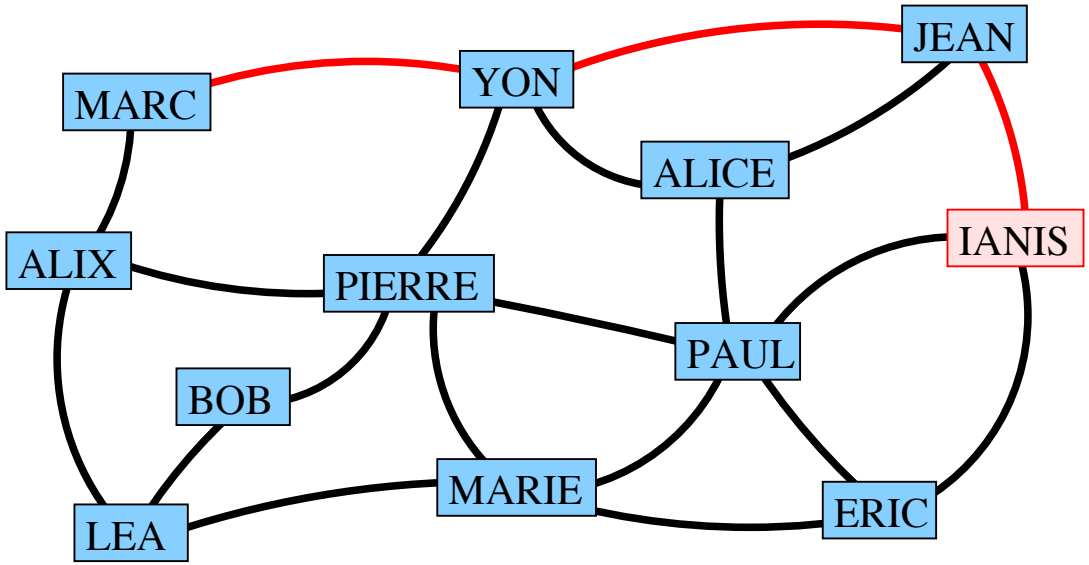


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
MARC: IANIS? > aller vers le voisin de droite
 YON: IANIS? > aller vers le voisin du haut
JEAN: IANIS? > aller vers le voisin le plus bas
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

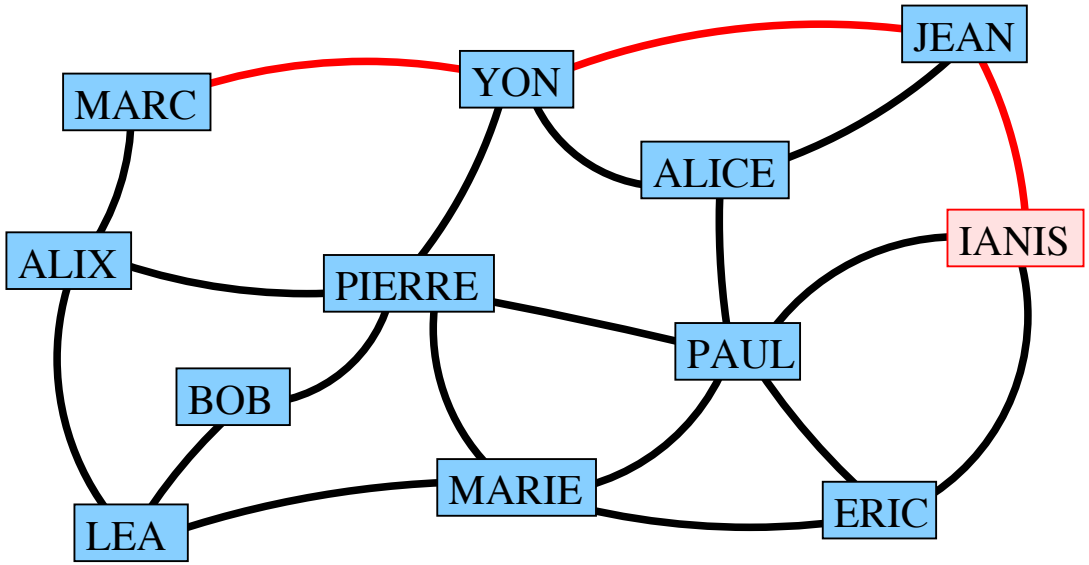


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
 MARC: IANIS? > aller vers le voisin de droite
  YON: IANIS? > aller vers le voisin du haut
 JEAN: IANIS? > aller vers le voisin le plus bas
IANIS: IANIS?
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

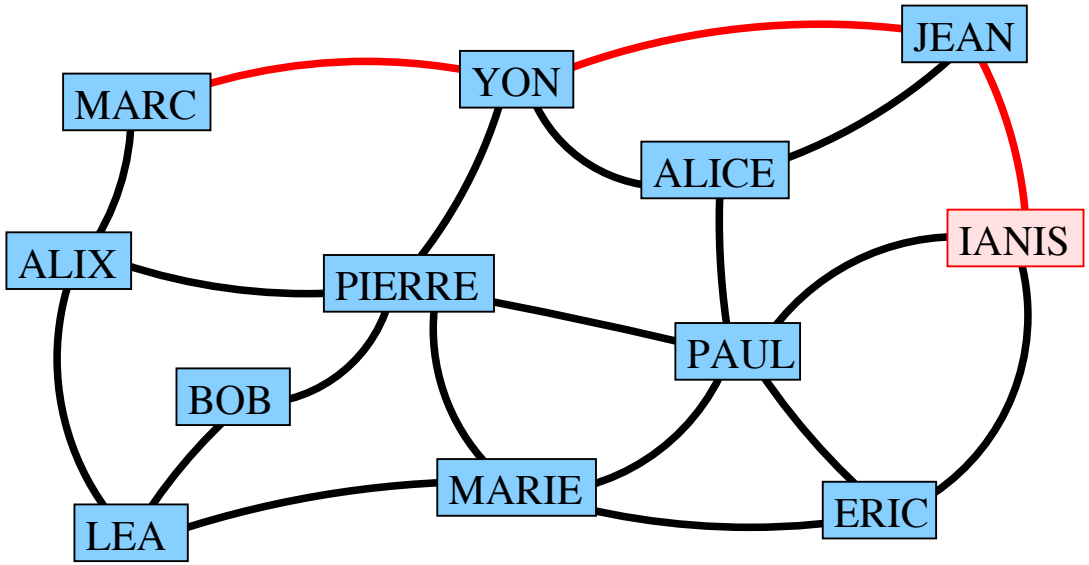


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
 MARC: IANIS? > aller vers le voisin de droite
  YON: IANIS? > aller vers le voisin du haut
 JEAN: IANIS? > aller vers le voisin le plus bas
IANIS: IANIS? > message arrivé.
```

Router ?

Trouver sa route dans un réseau d'entités interconnectées

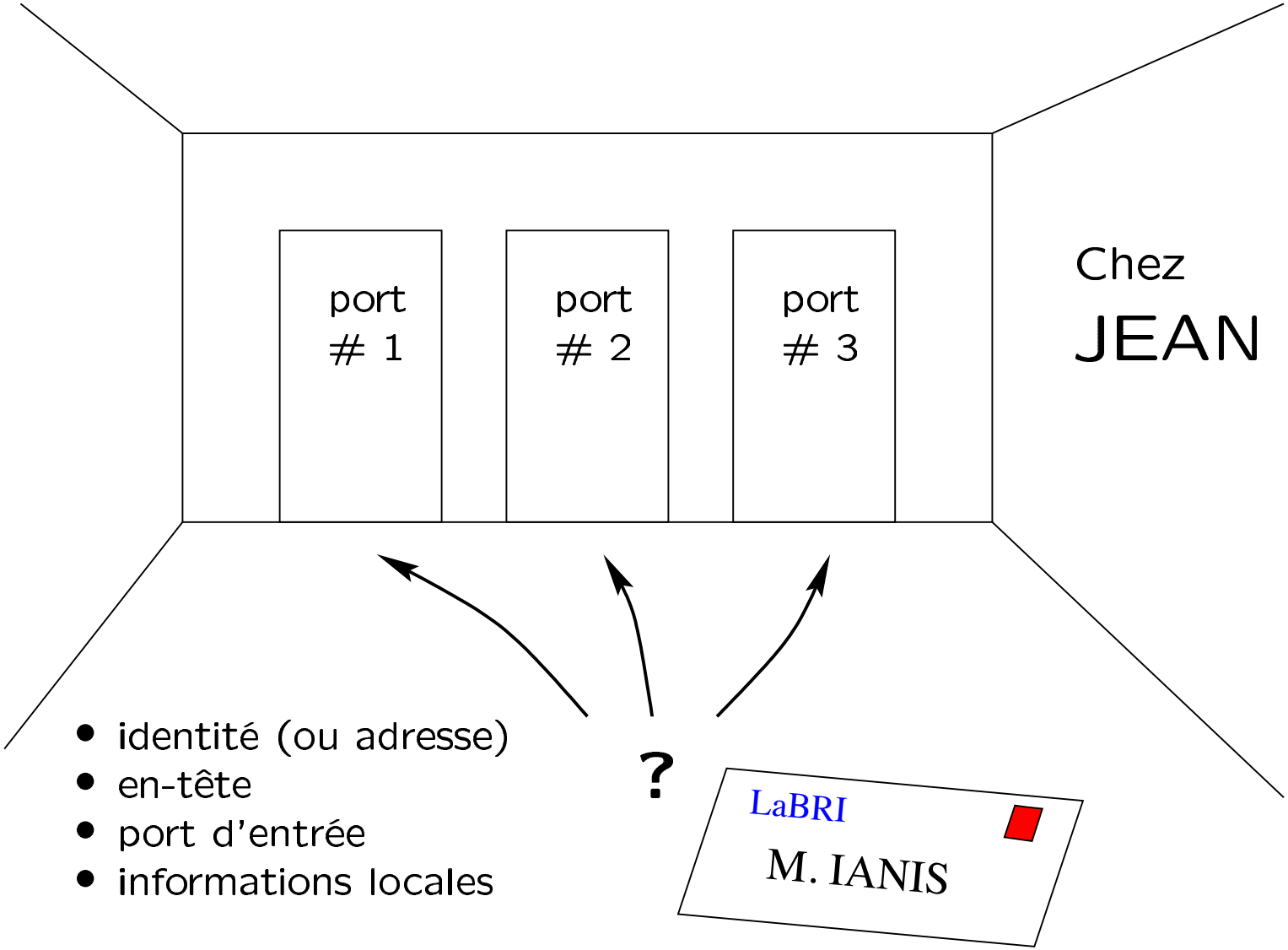


Ex: MARC veut récupérer un morceau de musique M qu'a IANIS

```
 MARC: IANIS? > aller vers le voisin de droite
  YON: IANIS? > aller vers le voisin du haut
 JEAN: IANIS? > aller vers le voisin le plus bas
IANIS: IANIS? > message arrivé.
```

Calcul distribué de la route

Décision locale



- identité (ou adresse)
- en-tête
- port d'entrée
- informations locales

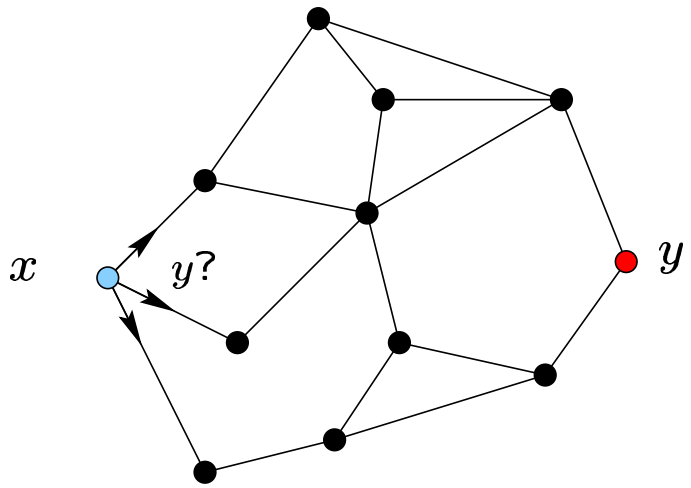
Question récurrente en théorie du Calcul Distribué :

De quoi ai-je « besoin » localement pour résoudre la tâche ?
(ici calculer la route)

besoin = connaissance
= quantité d'information
= structure de données pré-calculées

Ex: Dois-je connaître tout le réseau pour router ?

Réseau ?

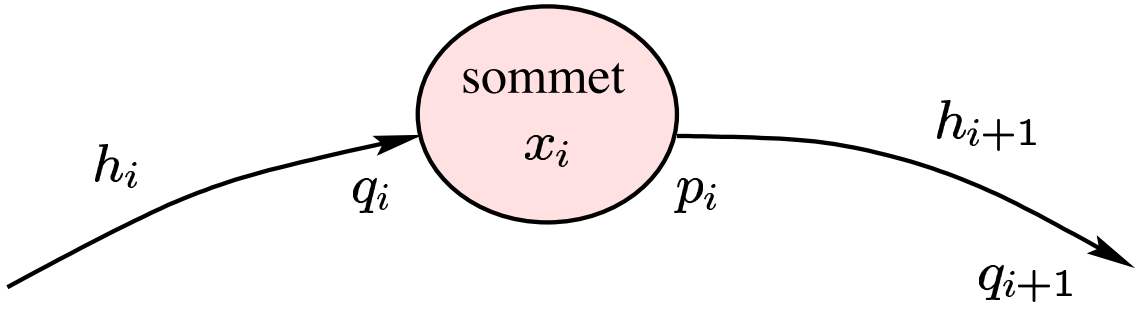


- Un graphe $G = (V, E)$ connexe, a priori non orienté, simple, sans boucle, dans un premier temps plutôt statique
- n entités, $n = |V|$
- Arêtes valuées $\omega : E \to \mathbb{R}_*^+$, fonction de coût d entre les sommets définissant une métrique (=distance),

$$d(x, z) \leqslant d(x, y) + d(y, z) \quad \text{(inégalité triangulaire)}$$

Algorithme de routage ?

$\text{ROUTE}_{x_i}(h_i, q_i) = (h_{i+1}, p_i)$



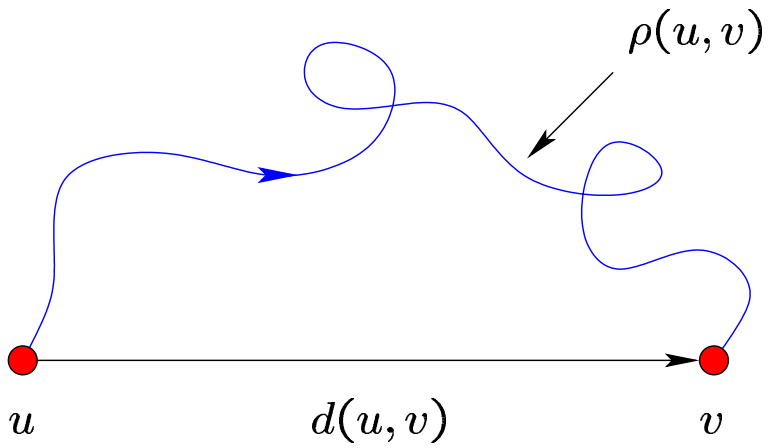
- h_i = en-têtes
- p_i, q_i = numéro (local à x_i) de port $\in \{0, 1, \ldots, \deg(x_i)\}$
- port spécial 0 = émetteur / récepteur
- p_i et q_{i+1} sans relation a priori

Problème

Étant donné G, décrire ROUTE_{x_i} pour tout $x_i \in V$

Critères de qualité d'un algo. de routage

- Longueurs (coûts) des routes : s = facteur d'étirement



$$s(u,v) = \frac{\rho(u,v)}{d(u,v)}$$

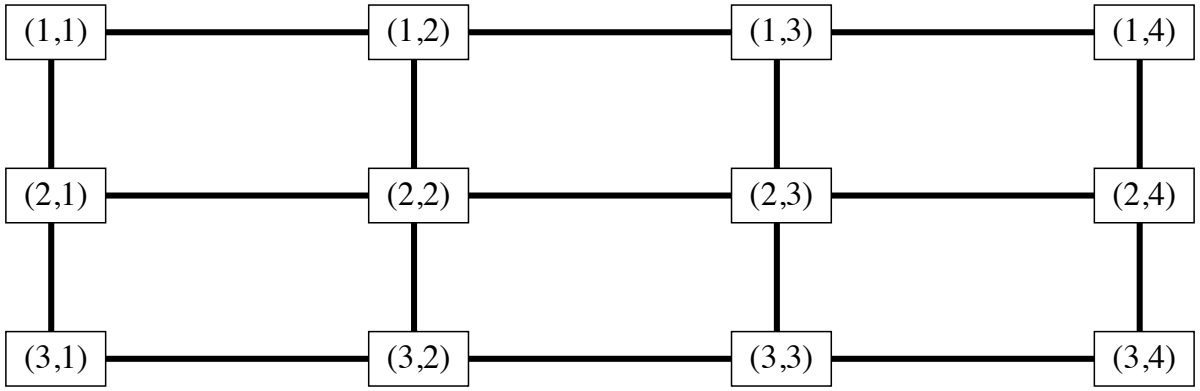
- taille des structures de données de chaque sommet (en bits), équilibre entre les sommets

Compromis : **étirement** vs. **taille des informations**

Autres critères

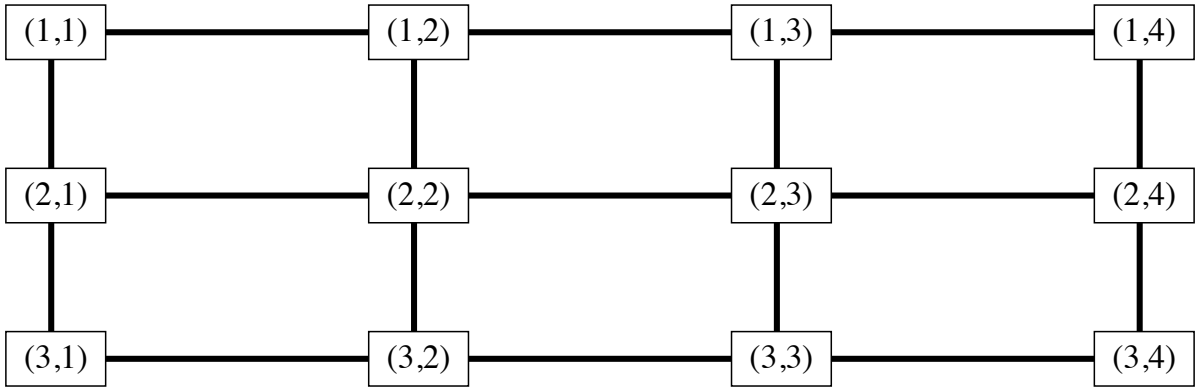
- Taille des en-têtes, en-têtes modifiables ou pas ?
- Temps de décision (complexité en temps de ROUTE_{x_i})
- Temps de calcul / mise à jour des structures de données
- ...

Cas d'école : routage XY



$$\text{ROUTE}_{(i,j)}(x,y) = \begin{cases} x > i & \text{retourner « droite »} \\ x < i & \text{retourner « gauche »} \\ x = i \text{ et } y > j & \text{retourner « bas »} \\ x = i \text{ et } y < j & \text{retourner « haut »} \\ x = i \text{ et } y = j & \text{retourner « arrivé »} \end{cases}$$

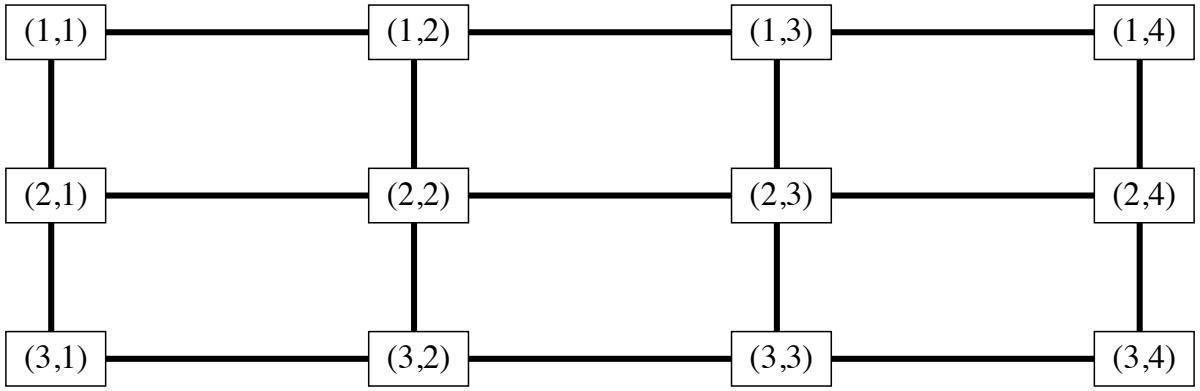
Cas d'école : routage XY



$$\text{ROUTE}_{(i,j)}(x,y) = \begin{cases} x > i & \text{retourner « droite »} \\ x < i & \text{retourner « gauche »} \\ x = i \text{ et } y > j & \text{retourner « bas »} \\ x = i \text{ et } y < j & \text{retourner « haut »} \\ x = i \text{ et } y = j & \text{retourner « arrivé »} \end{cases}$$

INFORMATION $= O(\log i + \log j)$ bits/sommet

Cas d'école : routage XY

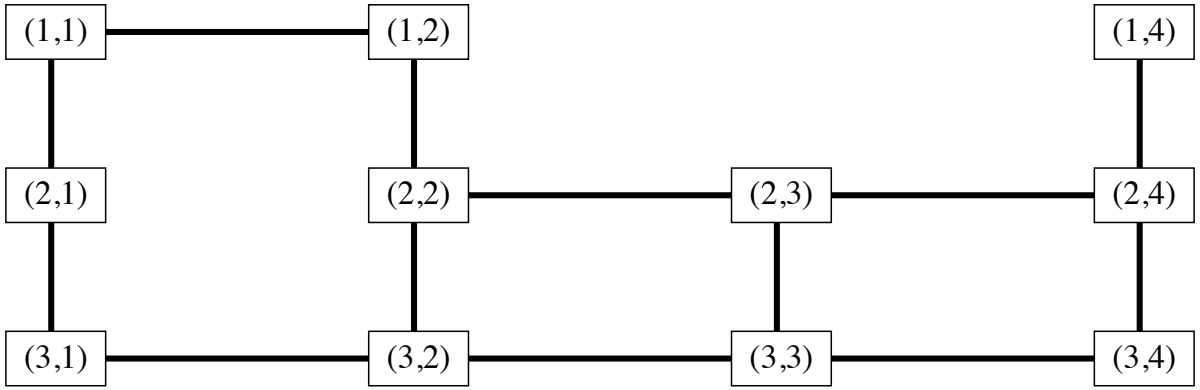


$$\text{ROUTE}_{(i,j)}(x,y) = \begin{cases} x > i & \text{retourner « droite »} \\ x < i & \text{retourner « gauche »} \\ x = i \text{ et } y > j & \text{retourner « bas »} \\ x = i \text{ et } y < j & \text{retourner « haut »} \\ x = i \text{ et } y = j & \text{retourner « arrivé »} \end{cases}$$

INFORMATION $= O(\log i + \log j)$ bits/sommet

ÉTIREMENT $= 1$

Cas d'école : routage XY

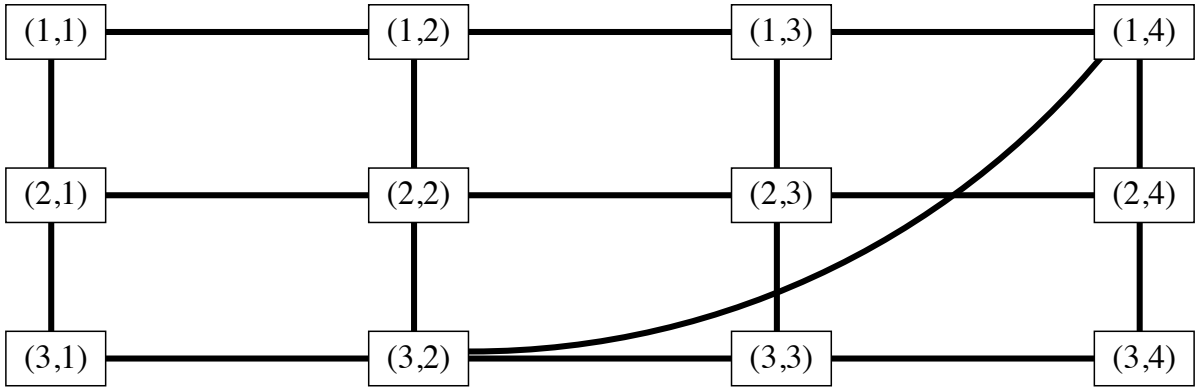


$$\text{ROUTE}_{(i,j)}(x,y) = \begin{cases} ? \\ ? \\ ? \end{cases}$$

INFORMATION = ?

ÉTIREMENT = ?

Cas d'école : routage XY

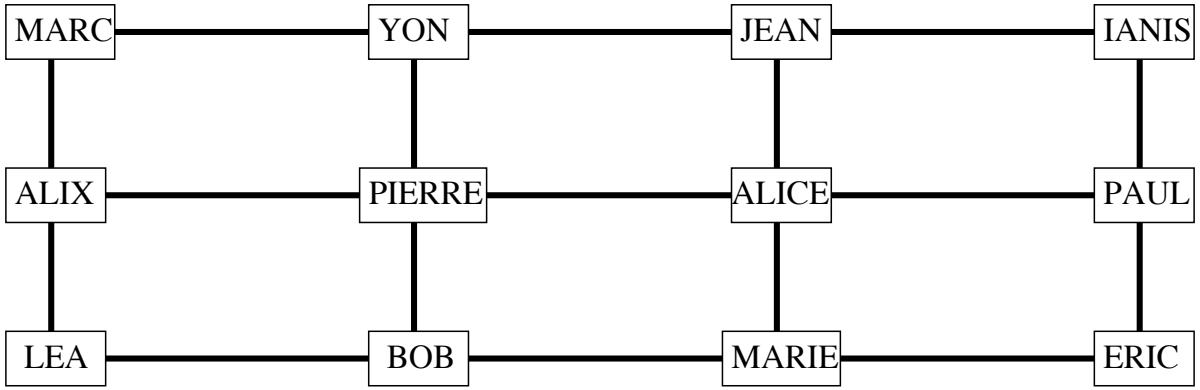


$$\text{ROUTE}_{(i,j)}(x,y) = \begin{cases} ? \\ ? \\ ? \end{cases}$$

INFORMATION = ?

ÉTIREMENT = ?

Cas d'école : routage XY



$$\text{ROUTE}_A(B) = \begin{cases} ? \\ ? \\ ? \end{cases}$$

INFORMATION = ?

ÉTIREMENT = ?

Schéma de routage UNIVERSEL

Objectif : Pour le passage à l'échelle, on souhaite calculer un algorithme de routage pour **tout graphe** (voir pour tout graphe d'une classe donnée assez grande, comme, par exemple, les graphes de diamètre logarithmique).

On parle de **schéma** (ou méthode) de routage **universel**.

Cas un peu moins d'école : Schéma de Thorup et Zwick [TZ01]

(SPAA & STOC 2001)

Cas un peu moins d'école : Schéma de Thorup et Zwick [TZ01]

(SPAA & STOC 2001)

`Caractéristiques :`

- Schéma universel
- Information : $\tilde{O}(\sqrt{n})$ bits/sommet (même si degré $\gg \sqrt{n}$)
- Étirement : $s \leqslant 3$
- En-têtes et adresses : $O(\log n)$ bits

Notation : $g(n) = \tilde{O}(f(n))$ pour $g(n) = O\left(f(n) \cdot (\log n)^{O(1)}\right)$

Cas un peu moins d'école : Schéma de Thorup et Zwick [TZ01]

(SPAA & STOC 2001)

`Caractéristiques :`

- Schéma universel
- Information : $\tilde{O}(\sqrt{n})$ bits/sommet (même si degré $\gg \sqrt{n}$)
- Étirement : $s \leqslant 3$
- En-têtes et adresses : $O(\log n)$ bits

Notation : $g(n) = \tilde{O}(f(n))$ pour $g(n) = O\left(f(n) \cdot (\log n)^{O(1)}\right)$

`Options :`

- Décision : temps $O(1)$
- Construction déterministe (après dérandomisation)
- Généralisation : $\tilde{O}(n^{1/k})$ bits/sommet et $s \leqslant 4k-5$, entier $k \geqslant 2$

À propos de l'optimalité du schéma TZ01

- si $s < 3 \Rightarrow$ information $\Omega(n)$ bits [GG97]
- si information $o(\sqrt{n})$ bits $\Rightarrow s \geqslant 5$ [TZ01, preuve ?]

Ce schéma réalise donc le meilleur compromis possible pour une information locale sous-linéaire.

Remarque :

- si $s < 1.4 \Rightarrow$ information $\Theta(n \log d)$ bits [GP96]

Question de fond :

Est-ce que ce schéma est vraiment utile ? Pourquoi optimiser le compromis information/étirement ?

2. Facteur d'étirement

&

Évolution d'Internet

Compact Routing on Internet-Like Graphs

Krioukov, Fall, Yang

INFOCOM 2004

Étude du schéma TZ01

- sur des graphes aléatoires avec loi en puissance pour la distribution des degrés :

$$\Pr(\deg(x) = k) \simeq k^{-\gamma}$$

($\gamma \simeq 2.1$ pour graphe Internet inter-domain $n = 11\,000$)

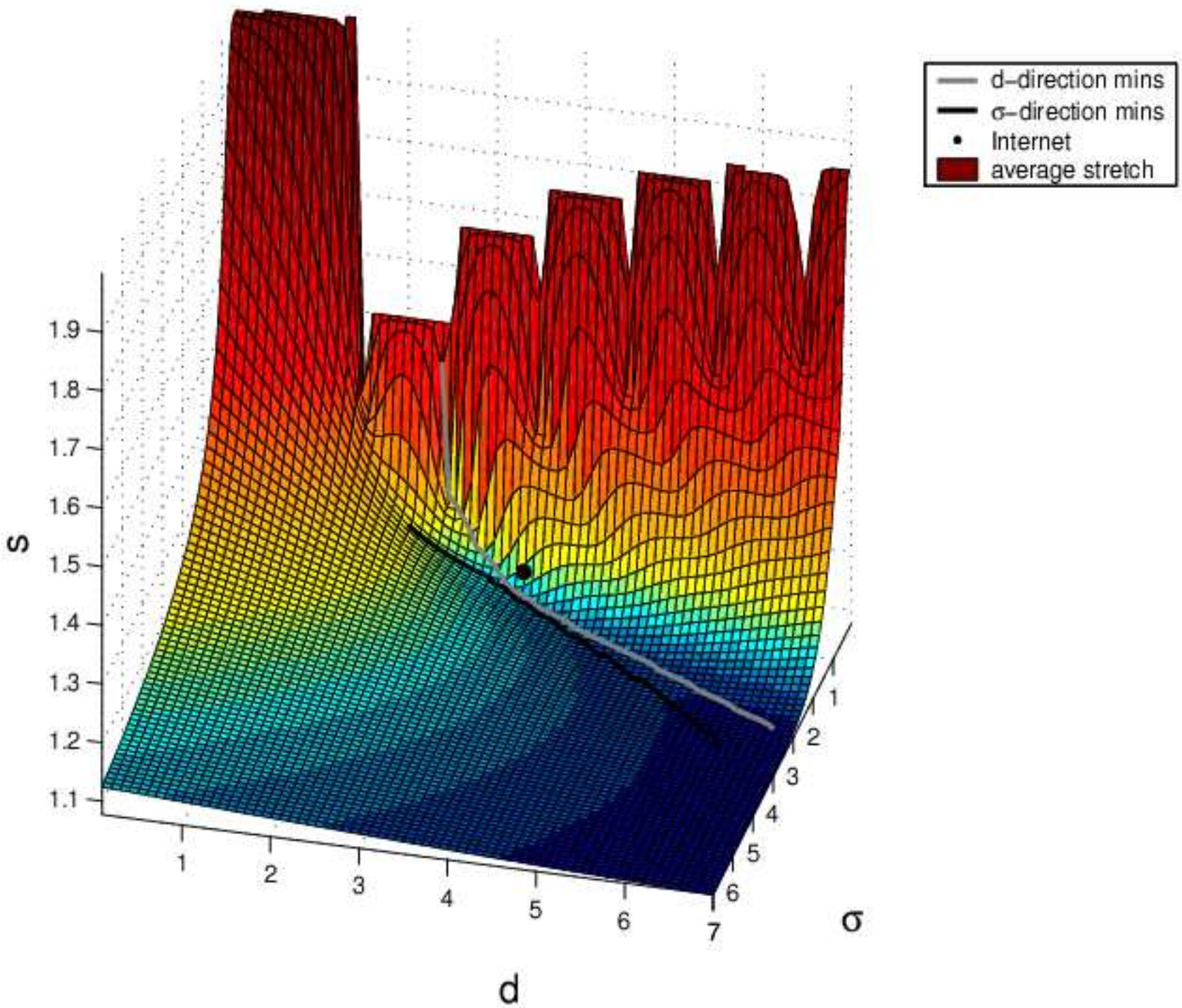
- facteur d'étirement moyen, sur toutes les paires du graphe : $\overline{s} \simeq 1.1$, jusqu'à 70% de plus courts chemins ($s = 1$)

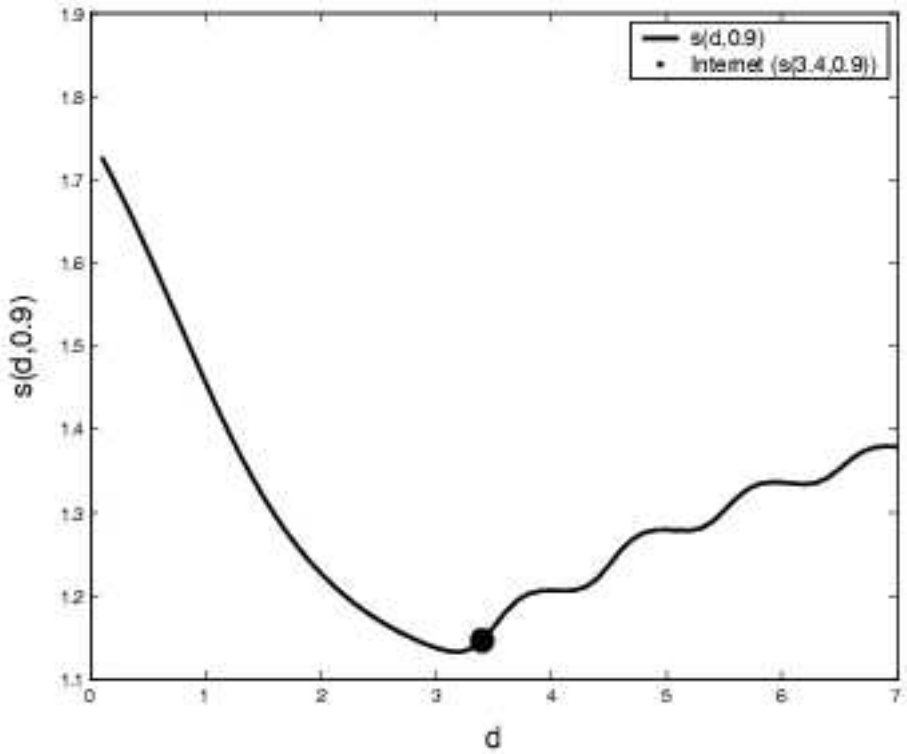
- taille des tables autour de 50 entrées pour 11 000 sommets.

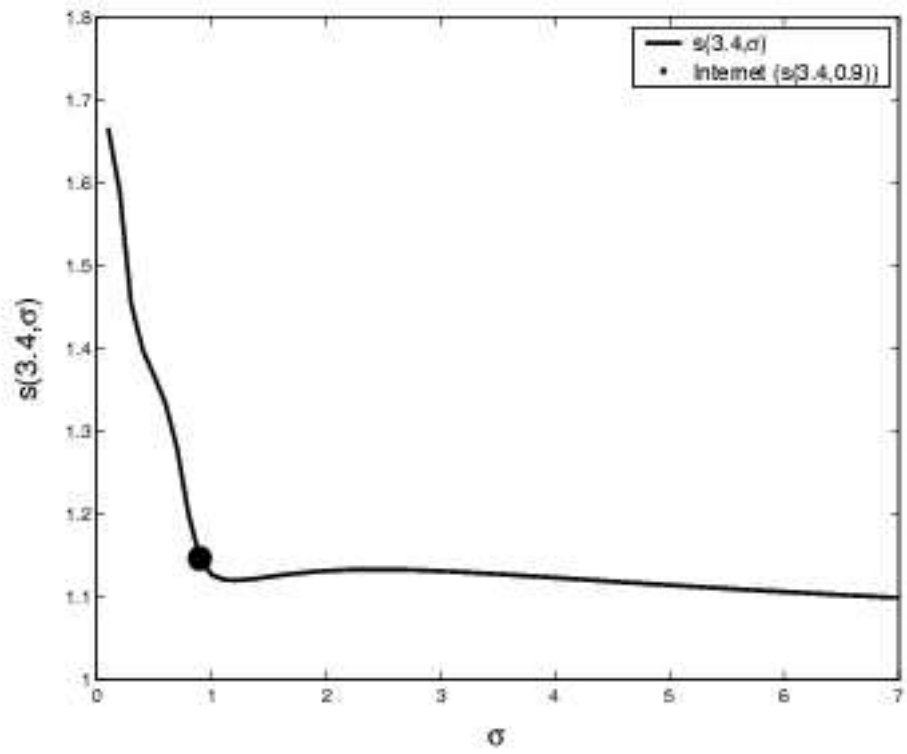
Simulations, analyses et tests sur des graphes réels. Pour les simulations, utilisation du générateur PLRG (modèle *Aiello, Chung, Lu – STOC 2000*). En faisant varier $\gamma = 2...3$, ils calculent la distribution des distances : distance moyenne $\overline{d}$ et l'écart type σ.

Pour le graphe d'Internet inter-AS ($n = 11\,000$), cela suit une Gaussienne avec $\overline{d} \simeq 3.4$ et $\sigma \simeq 0.9$.

Étude théorique de $\overline{s} = f(\overline{d}, \sigma)$ pour TZ01.


d–direction mins
σ–direction mins
Internet
average stretch
s
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
2
3
4
5
6
7
d
σ


s(d,0.9)
Internet (s(3.4,0.9))
s(d,0.9)
d

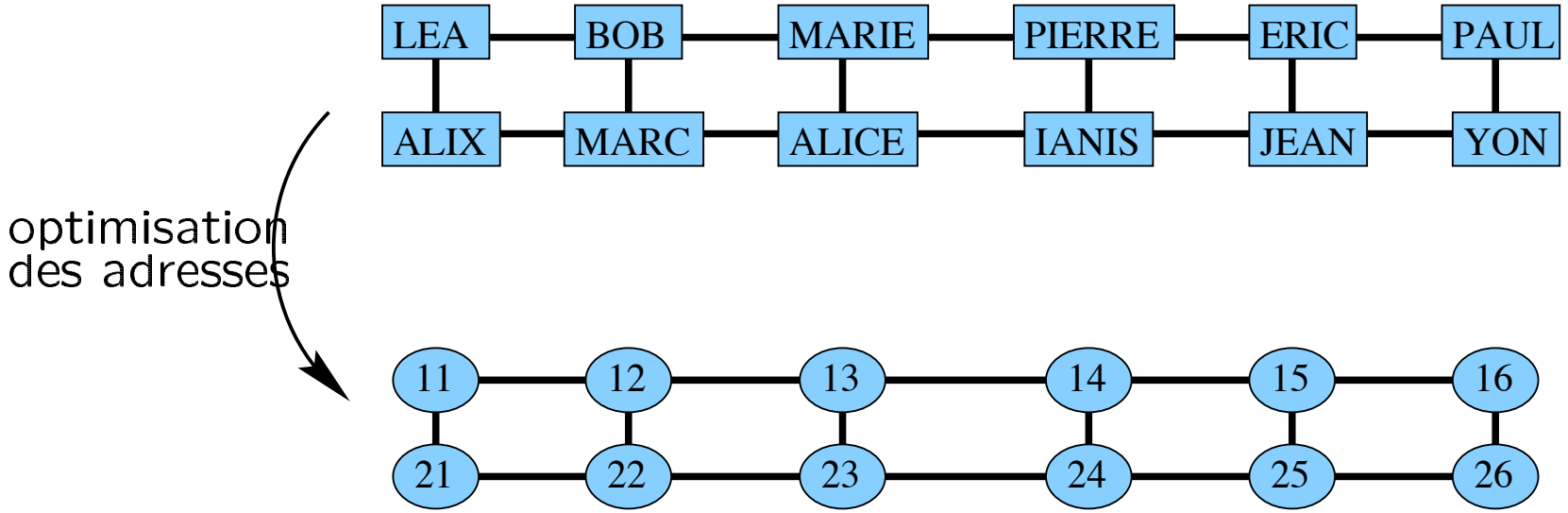

s(3.4,σ)
Internet (s(3.4,0.9))
s(3.4,σ)
σ

Leurs conclusions

« Cela donne l'impression que la topologie d'Internet a soigneusement été tissée pour avoir une distribution des distances minimisant le facteur d'étirement moyen du schéma TZ01, bien que jusqu'à présent l'évolution d'Internet n'a rien à voir avec le facteur d'étirement. Cela suggère que ce facteur peut être un indicateur du critère d'optimisation influençant l'évolution de la topologie du graphe d'Internet. »

Observation.

Le schéma TZ01 calcule et affecte de nouvelles adresses aux sommets. Ces adresses sont optimisées et dépendent de la topologies.



Un schéma possède l'indépendance des noms si le nom d'origine (avant la construction des algorithmes de routage) est conservé et peut être utiliser dans l'algorithme.

3. Schéma d'Awerbuch, Bar-Noy, Linial et Peleg [ABLP90]

Improved Routing Strategies with Succinct Tables

J. of Algorithms, 1990

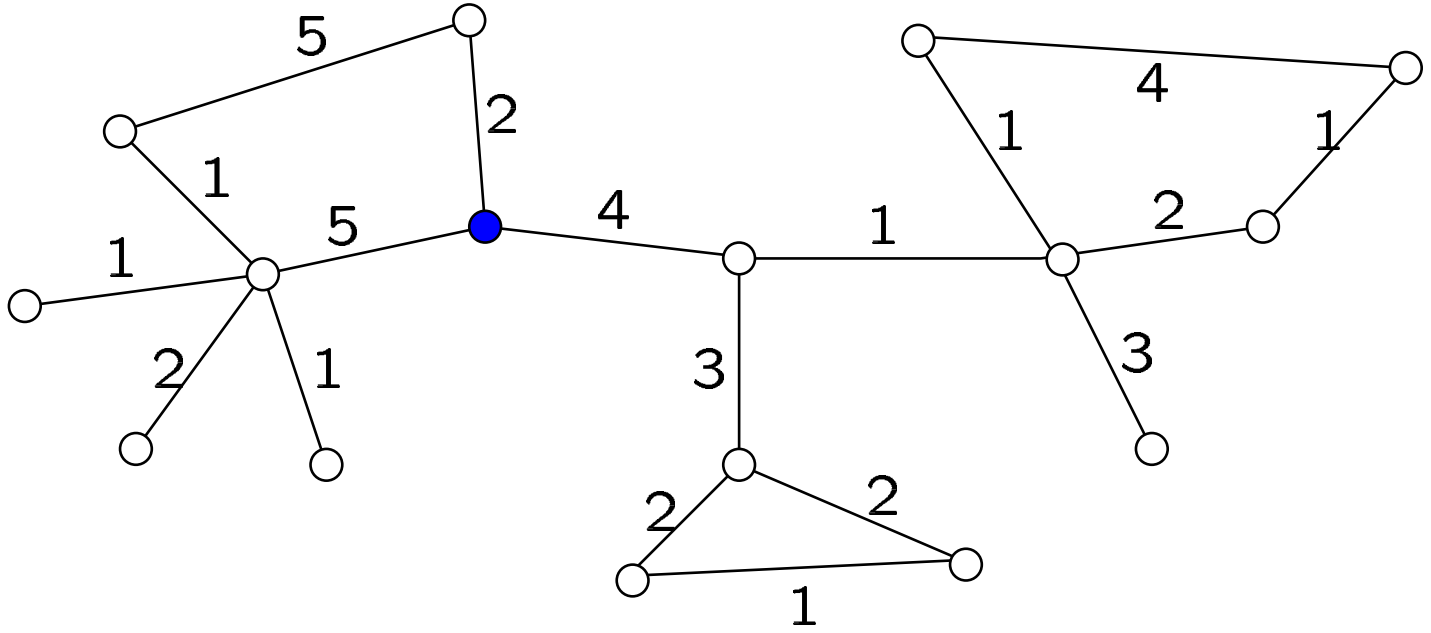
Schéma ABLP90

`Caractéristiques :`

- Schéma universel
- Indépendance des noms
- Information : $\tilde{O}(n\sqrt{n})$ bits en tout sur n sommets
- Étirement : $s \leqslant 3$
- En-têtes : $O(\log n)$ bits
- Constructible en temps polynômial
- Généralisable en $\tilde{O}(n^{1+1/k})$ bits pour $s \leqslant 2^k - 1$, entier $k \geqslant 1$

Étape 1 : voisinage défini par le volume

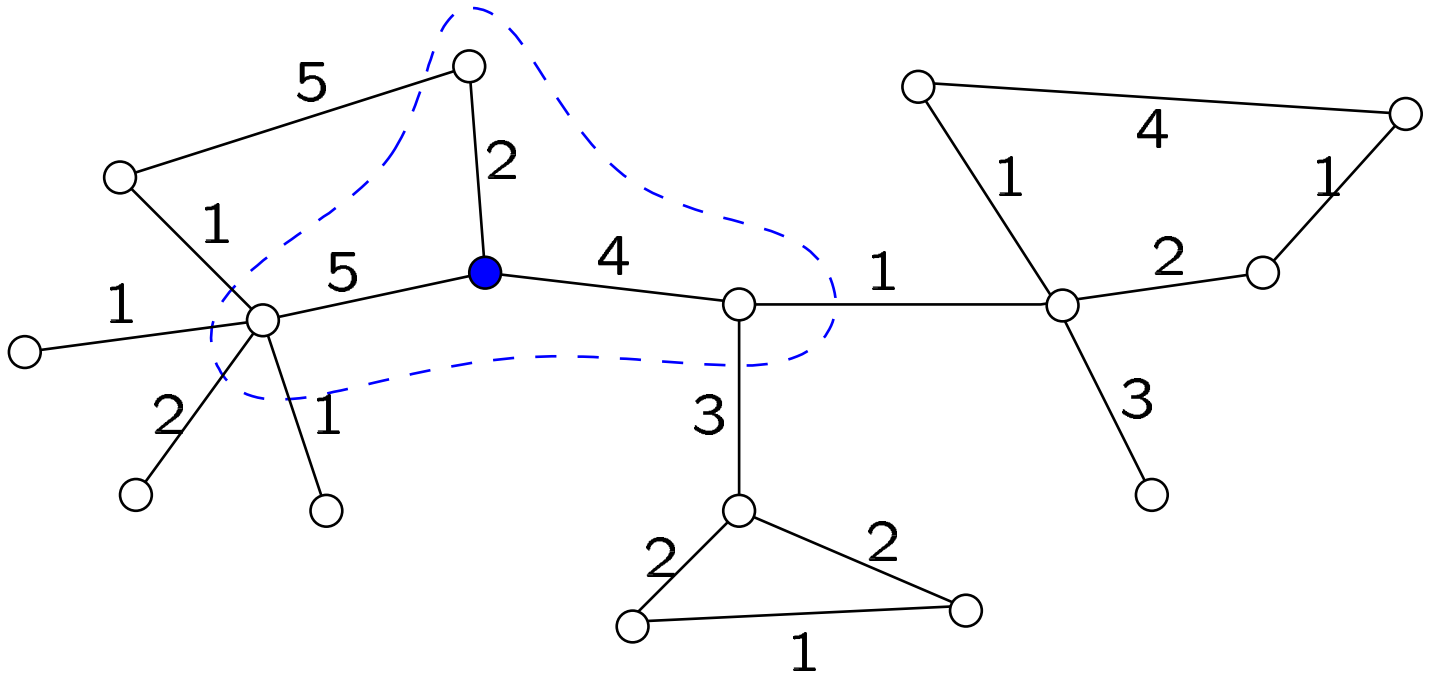
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

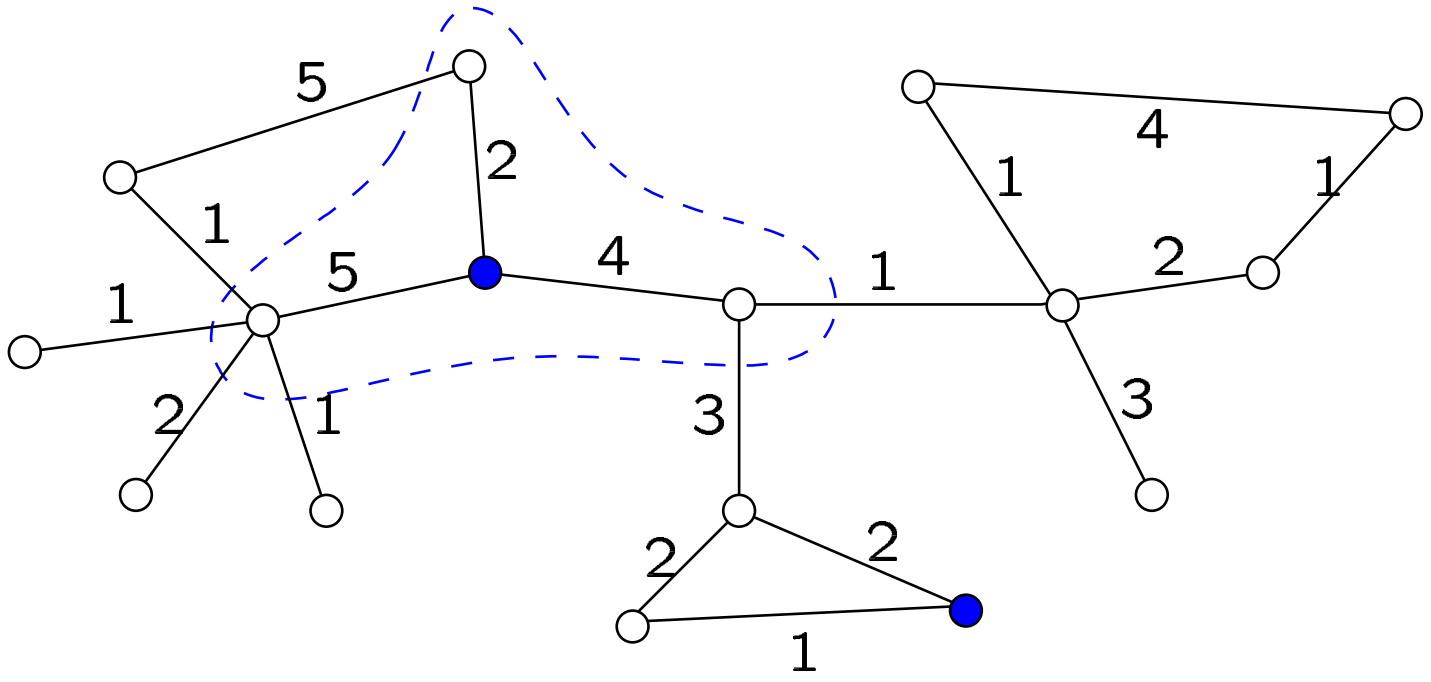
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

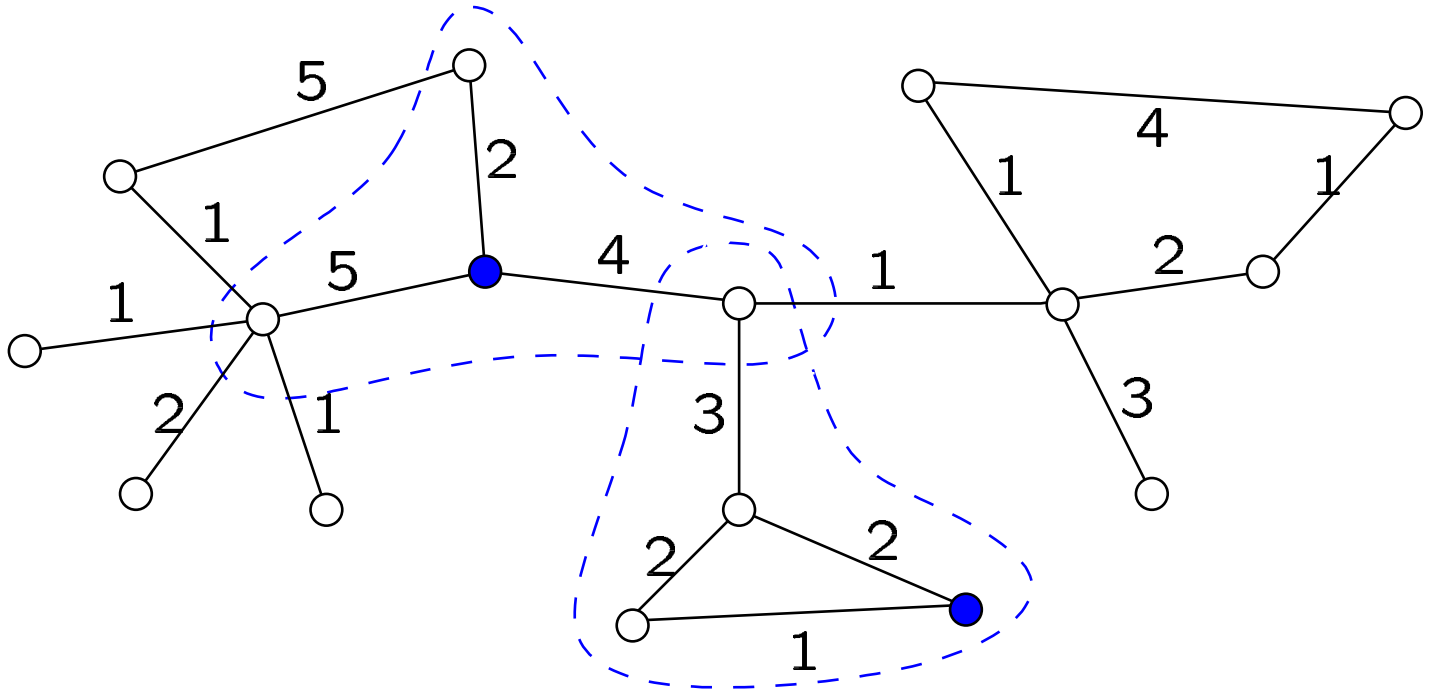
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

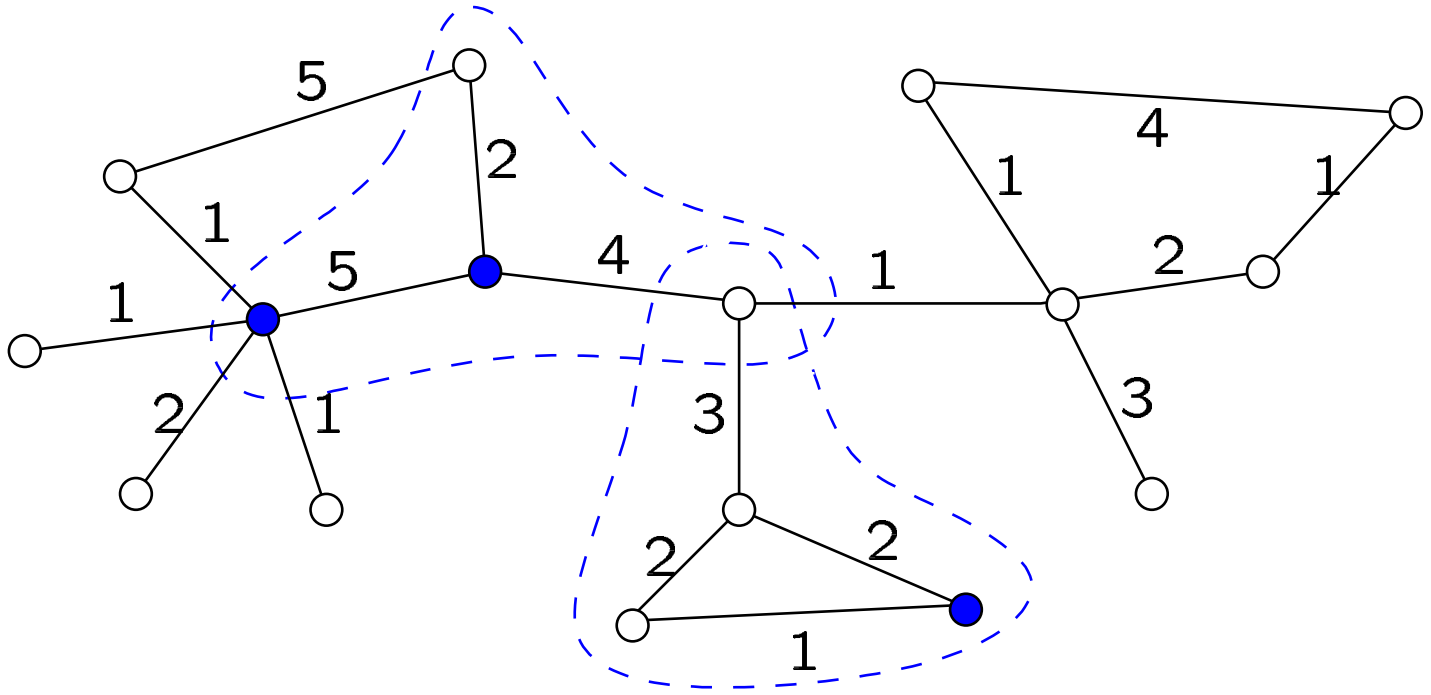
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

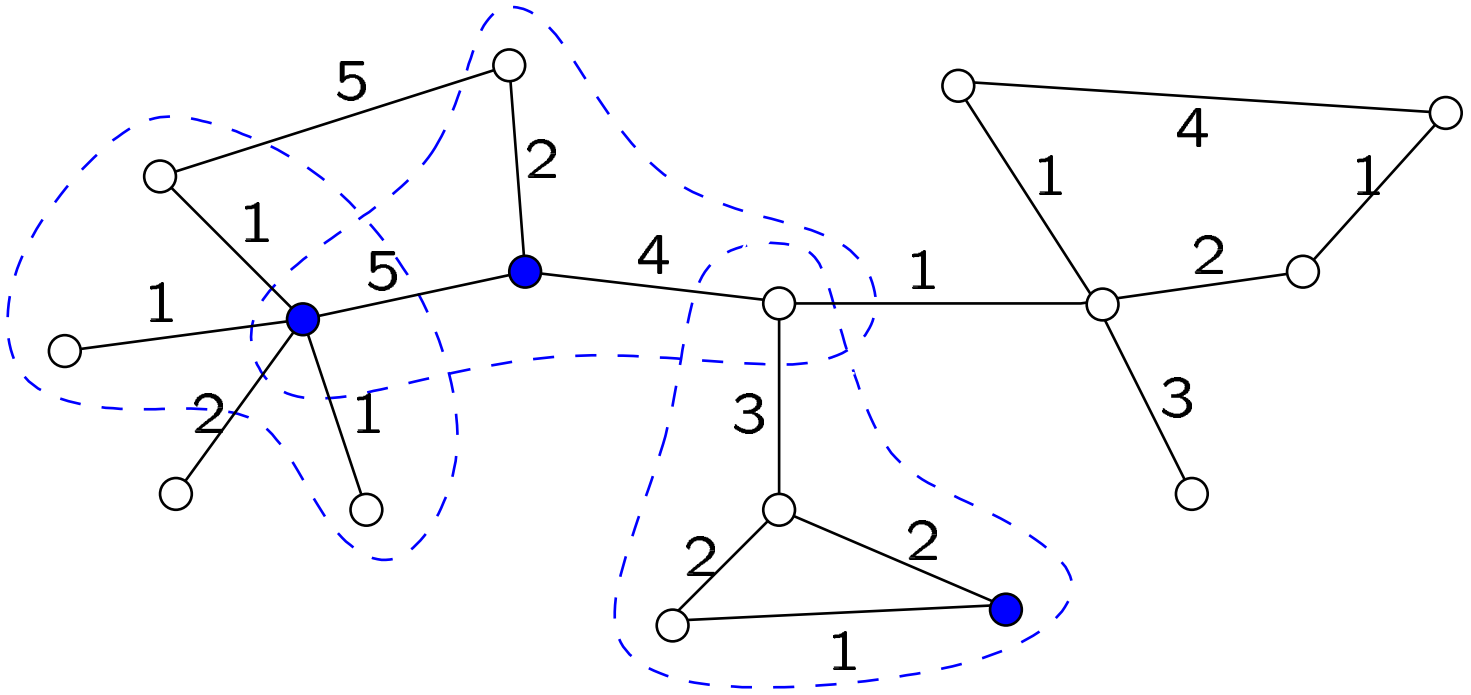
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

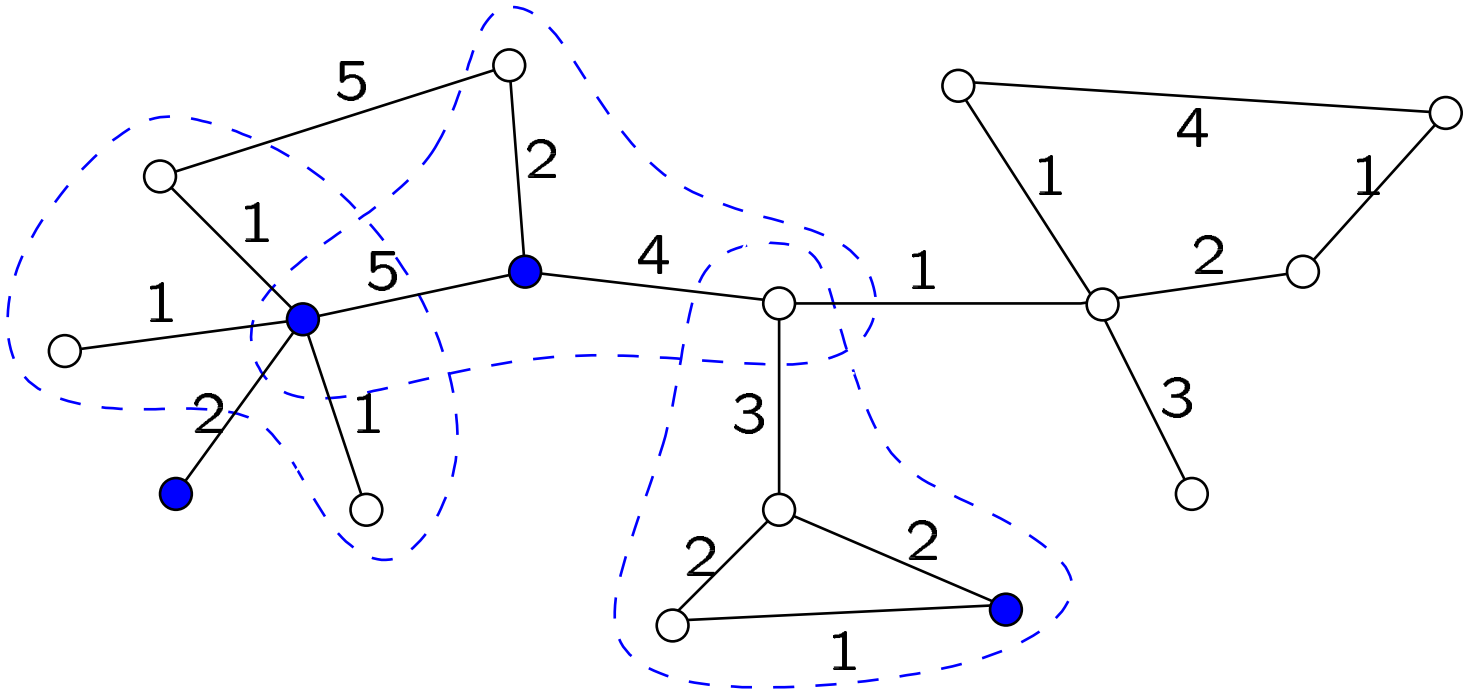
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

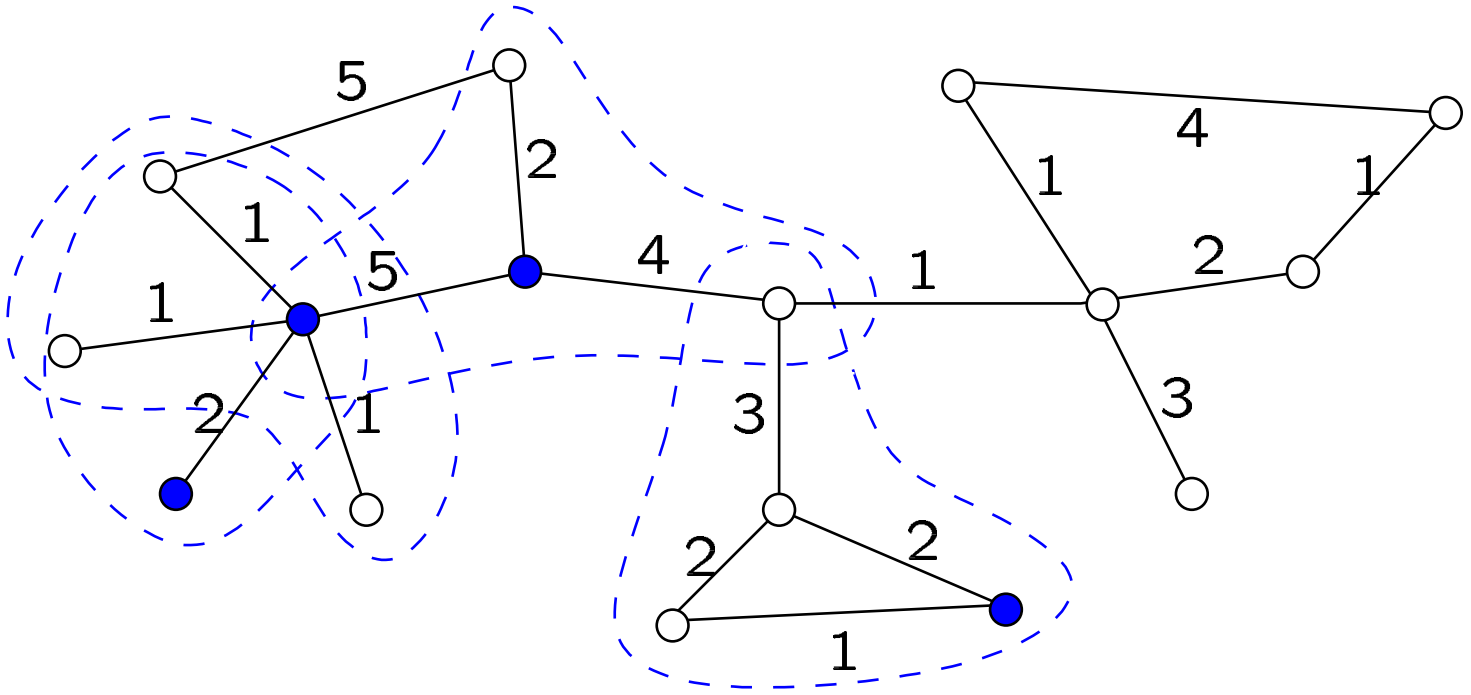
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

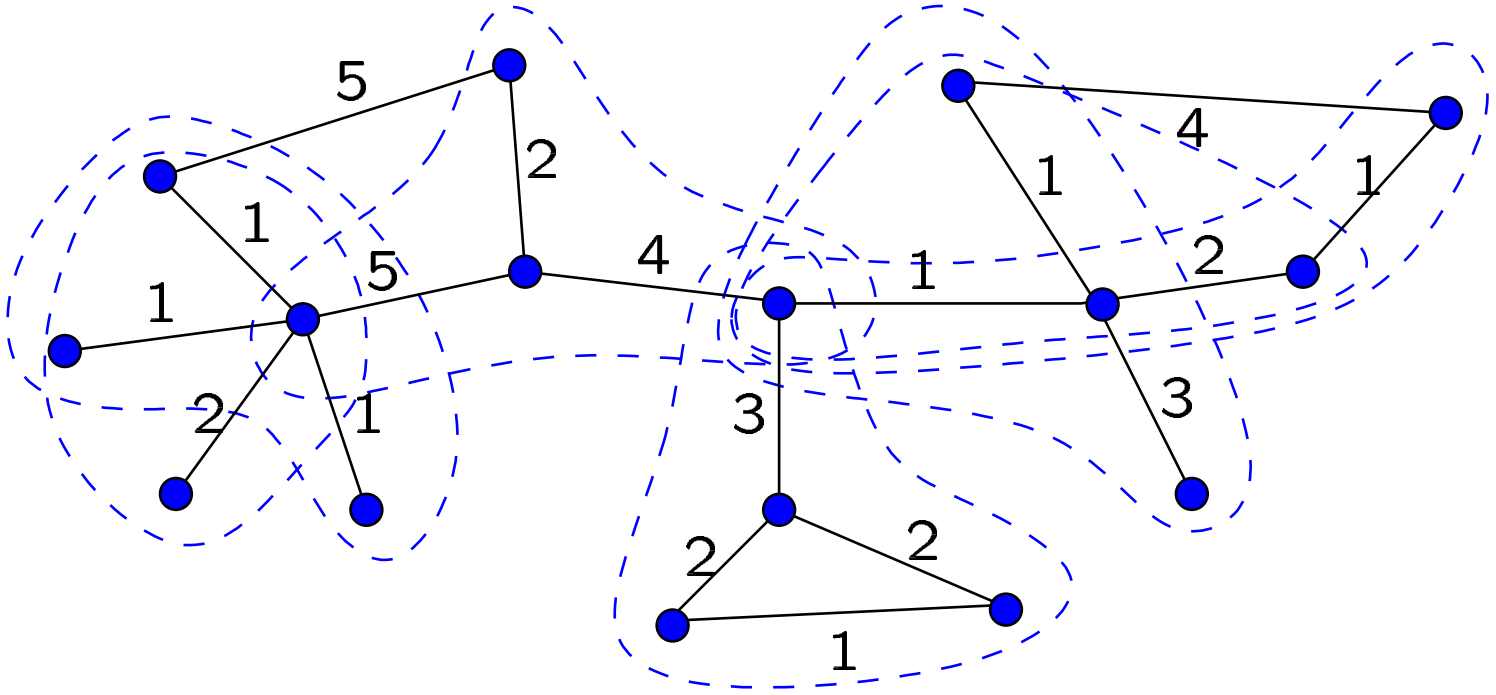
$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.



$n = 16, k = 4$

Étape 1 : voisinage défini par le volume

$x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x. On casse les égalités de la même façon pour tout les sommets (par ex. par ordre lexicographique). Ici on choisit $k \sim \sqrt{n}$.

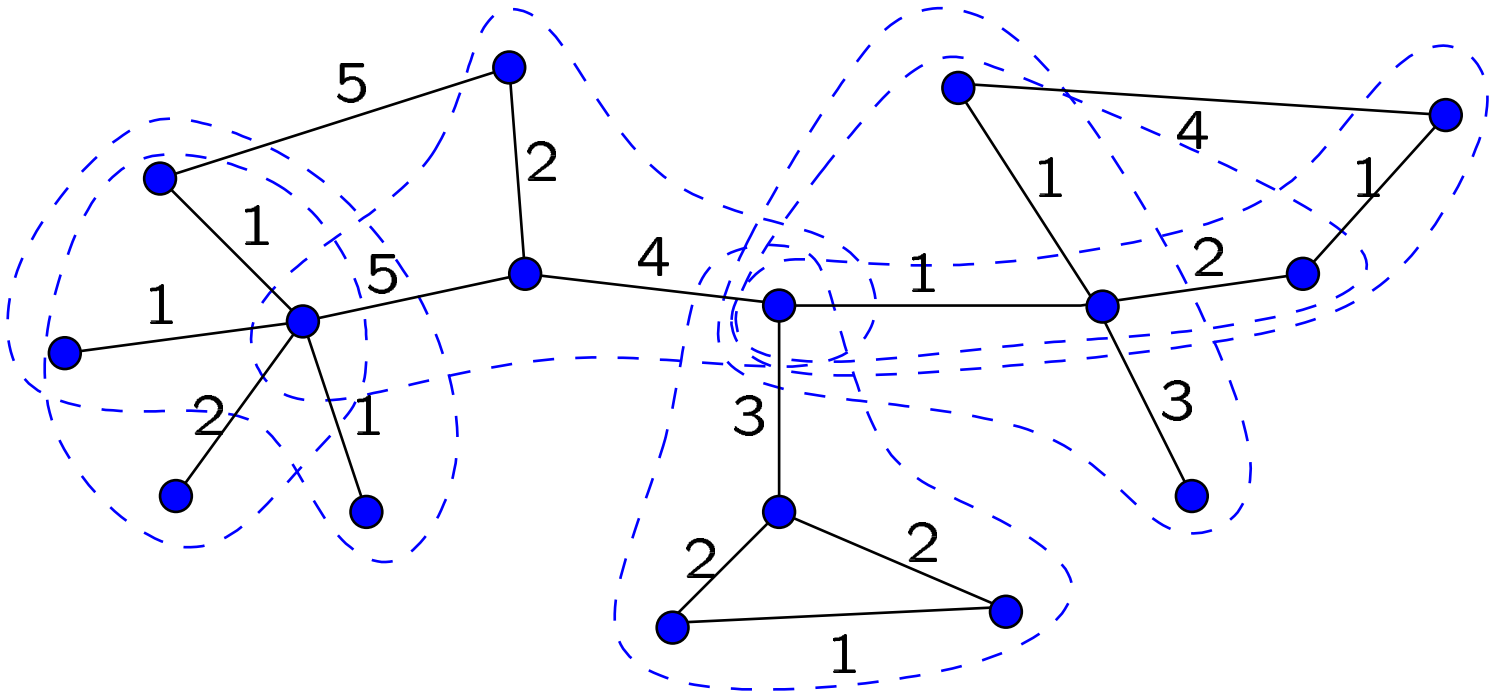


$n = 16, k = 4$

Étape 2 : ensemble couvrant

On construit un ensemble R intersectant toutes les boules $B(x)$.

Algorithme glouton $\Rightarrow$ $\boxed{|R| < n \ln(2n)/k = \tilde{O}(\sqrt{n})}$



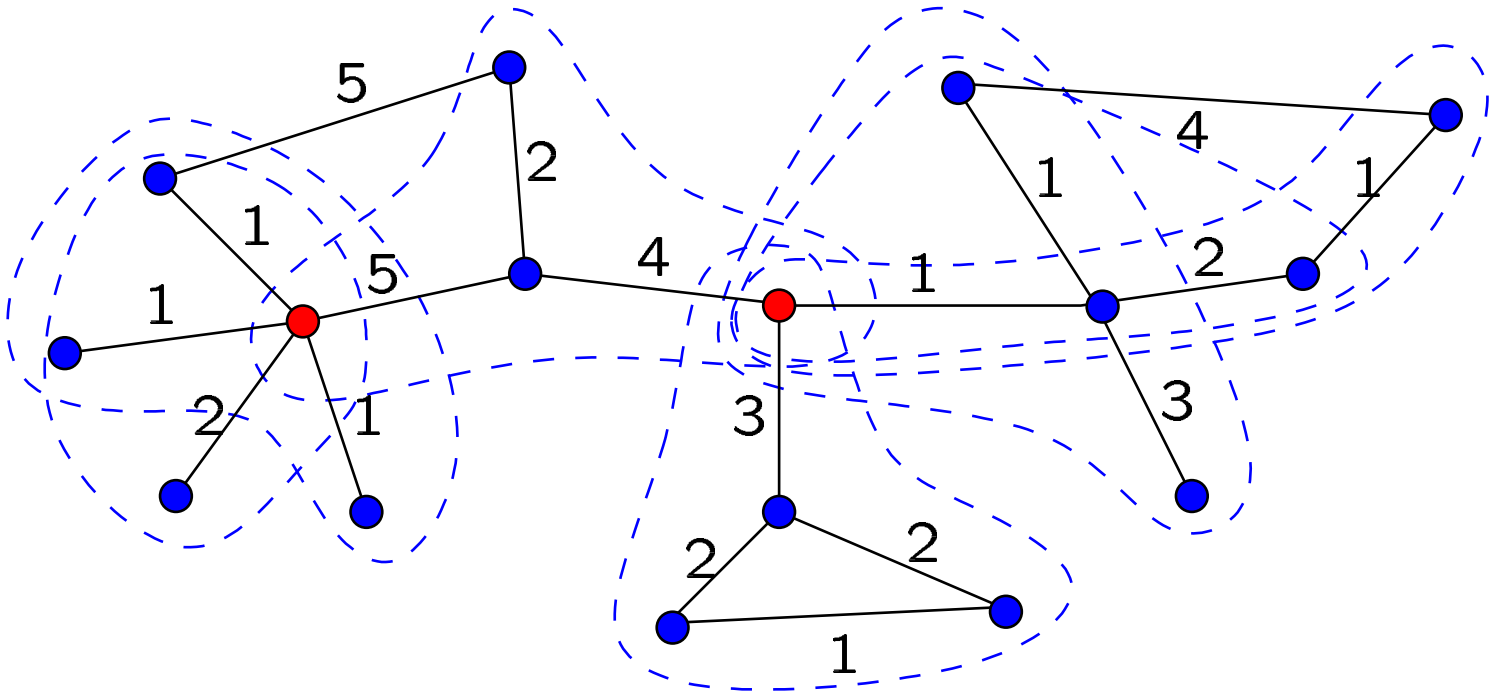
Puis,

- $x \mapsto r_x$ un sommet de $R \cap B(x)$
- $r \mapsto T_r$ arbre couvrant de coût minimum et de racine $r \in R$

Étape 2 : ensemble couvrant

On construit un ensemble R intersectant toutes les boules $B(x)$.

Algorithme glouton $\Rightarrow$ $\boxed{|R| < n \ln(2n)/k = \tilde{O}(\sqrt{n})}$



Puis,

- $x \mapsto r_x$ un sommet de $R \cap B(x)$
- $r \mapsto T_r$ arbre couvrant de coût minimum et de racine $r \in R$

Algorithme de routage de x vers y

Si $x \notin R$:
Si $y \in B(x)$ alors routage dans $B(x)$ sinon routage vers r_x.

En $r_x \in R$:
Routage dans l'arbre T_{r_x} en convertissant y en « étiquette d'arbre »,
un nombre noté $\ell_{r_x}(y) \in \{1, ..., n\}$, pour le routage efficace dans T_{r_x}.
En-tête : $\langle r_x, \ell_{r_x}(y) \rangle$

Ce routage nécessite la modification des en-têtes.

Algorithme de routage de x vers y

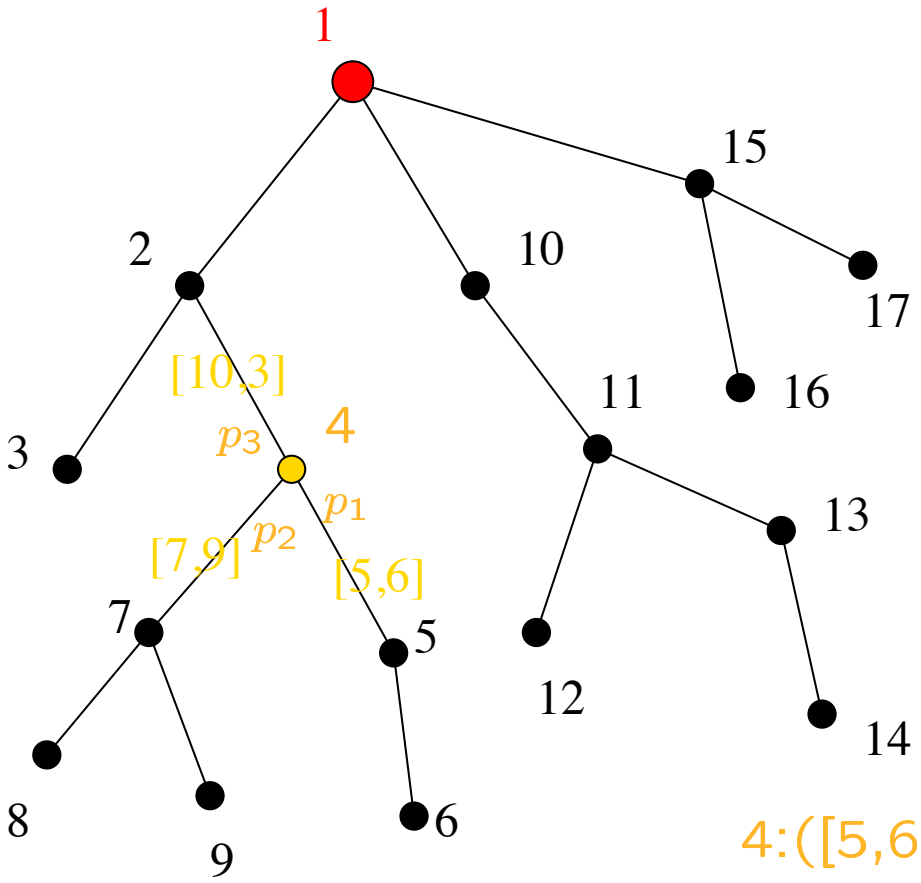
Si $x \notin R$:
Si $y \in B(x)$ alors routage dans $B(x)$ sinon routage vers r_x.

Dans ce cas x a besoin de connaître les noms des sommets de $B(x)$ ainsi que le port permettant de s'y rendre par un plus court chemin ; aussi le port pour aller vers r_x par un plus court chemin. En tout cela fait $\tilde{O}(\sqrt{n})$ bits pour x.

En $r_x \in R$:
Routage dans l'arbre T_{r_x} en convertissant y en « étiquette d'arbre », un nombre noté $\ell_{r_x}(y) \in \{1, ..., n\}$, pour le routage efficace dans T_{r_x}.
En-tête : $\langle r_x, \ell_{r_x}(y) \rangle$

Ce routage nécessite la modification des en-têtes.

Routage dans un arbre T_{r_x}



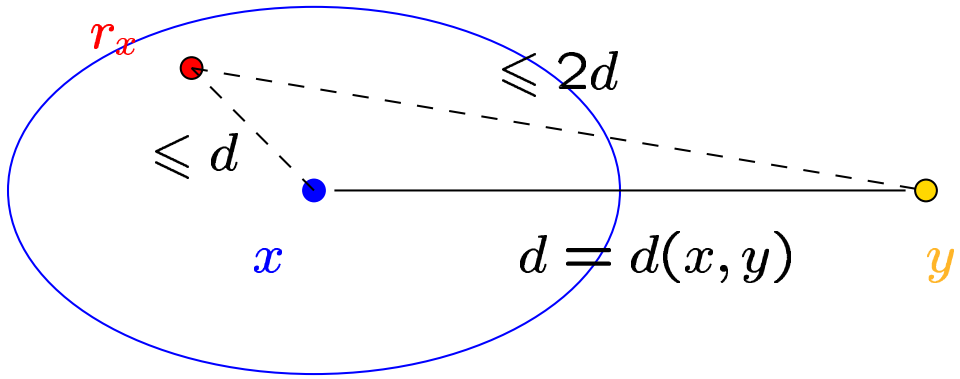
- numérotation DFS
- routage par intervalles
- $O(n \log n)$ bits en tout

4:([5,6]:p_1,[7,9]:p_2,[10,3]:p_3)

Chaque sommet r_x a besoin de stocker une table de convertion $y \mapsto \ell_{r_x}(y)$, ce qui coûte $O(n \log n)$ bits par sommet, mais seulement $|R| n \log n = \tilde{O}(n\sqrt{n})$ bits en tout. De plus chaque sommet intermédiaire participe aussi au routage dans $|R|$ arbres ($|R|$ jeux d'intervalles). Comme chaque contribue pour $n-1$ intervalles, cela rajoute en tout $|R| n \log n = \tilde{O}(n\sqrt{n})$ bits.

INFORMATION $= \tilde{O}(n\sqrt{n})$ bits en tout,
mais certain sommets ont $\Omega(n \log n)$ bits. Ce n'est pas distribué !

ÉTIREMENT : $s \leqslant 3$



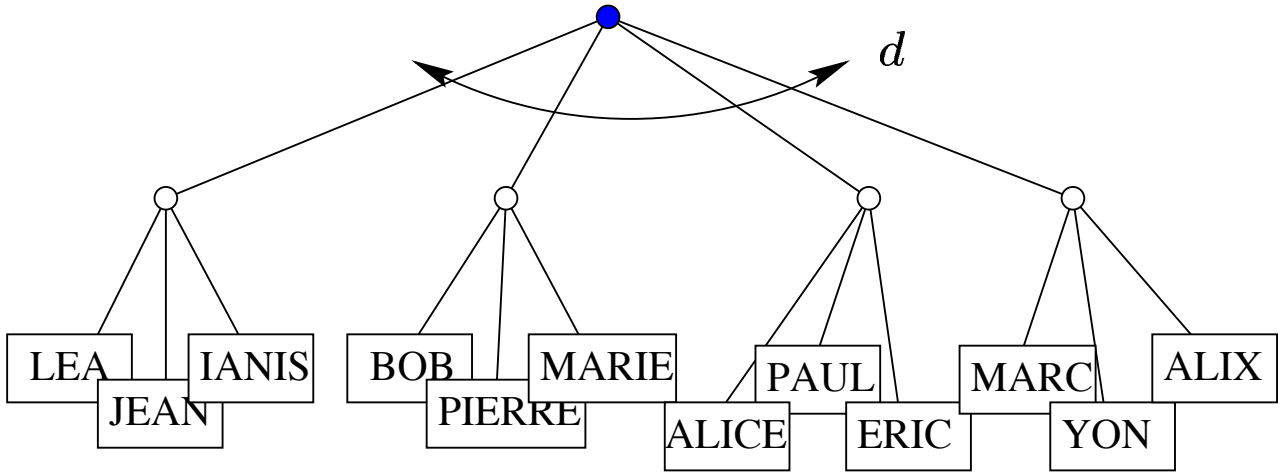
Question de fond :

Peut-on avoir $o(n)$ bits/sommet (table distribuée) et l'indépendance des noms ? Dans un arbre avec $s = 1$ par exemple ?

Question de fond :

Peut-on avoir $o(n)$ bits/sommet (table distribuée) et l'indépendance des noms ? Dans un arbre avec $s = 1$ par exemple ?

Sans l'indépendance, OUI.
Avec l'indépendance NON.



En comptant les arrangements possibles des noms, on montre que la racine doit stocker $\Omega(n \log d)$ bits. Et même, $\Omega(n \log n)$ bits si $s < 3$ dans le cas d'une étoile à n feuilles !

4. Une nouvelle solution

en collaboration avec Abraham, Malkhi et Nisan

Hebrew University of Jerusalem, 2003

Nouveau schéma

`Caractéristiques :`

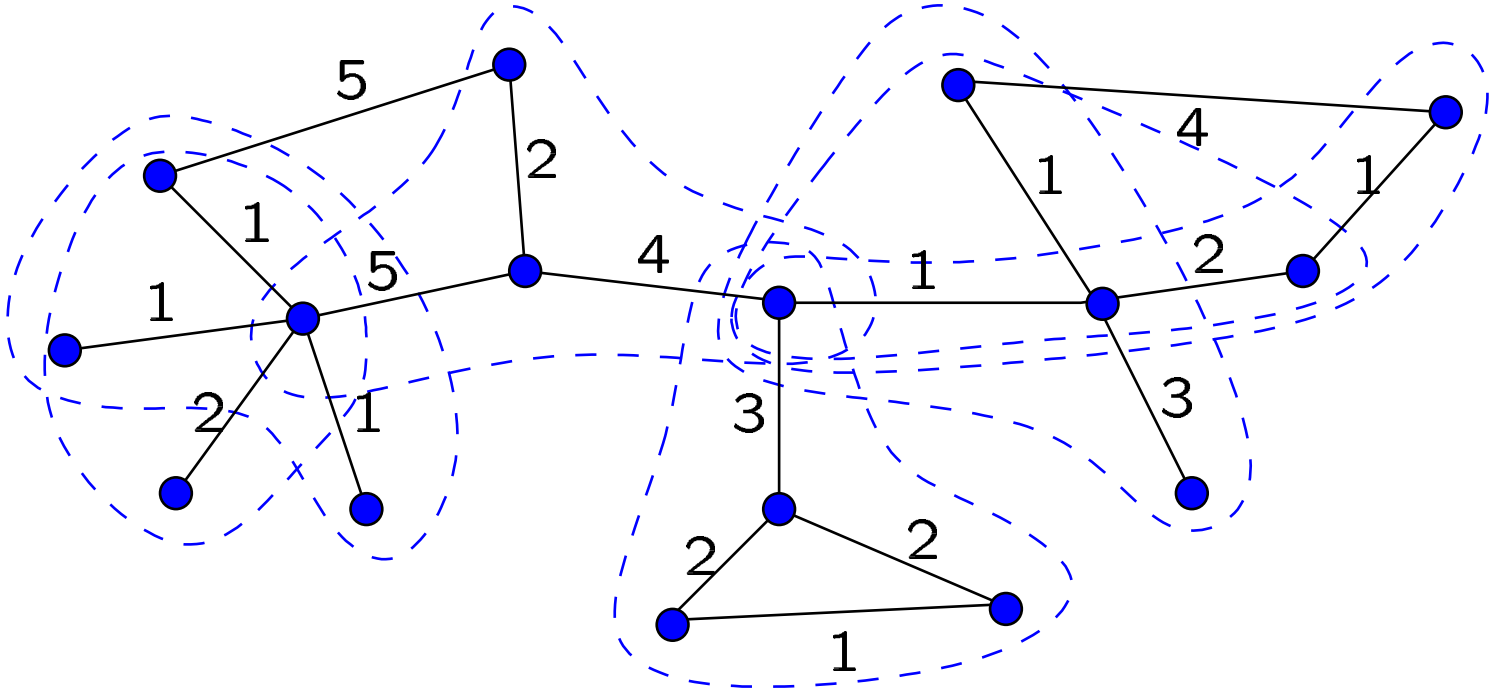
- Schéma universel
- Indépendance des noms
- Information : $\tilde{O}(\sqrt{n}\log D)$ bits/sommet
- Étirement : $s \leqslant 3$
- En-têtes : $\tilde{O}(\log D)$ bits
- Constructible en temps polynomial
- Généralisation : en cours ...

où D est le diamètre normalisé. Souvent, $D \leqslant n^{\tilde{O}(1)}$, si bien que $\log D = \tilde{O}(1)$.

$$D := \frac{\max_{x \neq y} d(x,y)}{\min_{x \neq y} d(x,y)}$$

Étape 1 : voisinage

Comme pour ABLP90, $x \mapsto B(x)$ ensemble contenant les k plus proches voisins de x, avec $k = \tilde{O}(\sqrt{n})$.

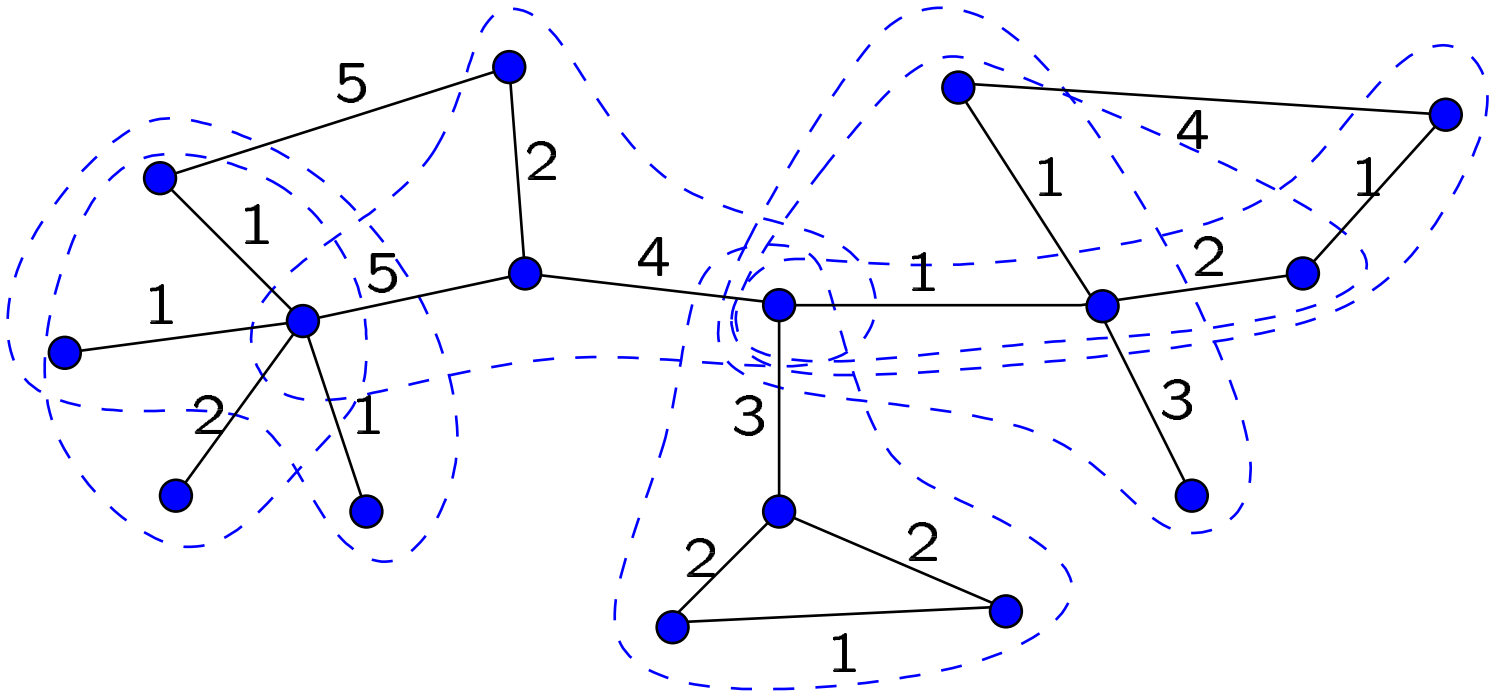


On va poser $V = \{1, ..., n\}$ fixé, numérotation non modifiable par le schéma (indépendance des noms)

Étape 2 : coloration (1/2)

Pour tout sommet $x \mapsto c(x) \in \{1, ..., \lfloor\sqrt{n}\rfloor\}$ tq

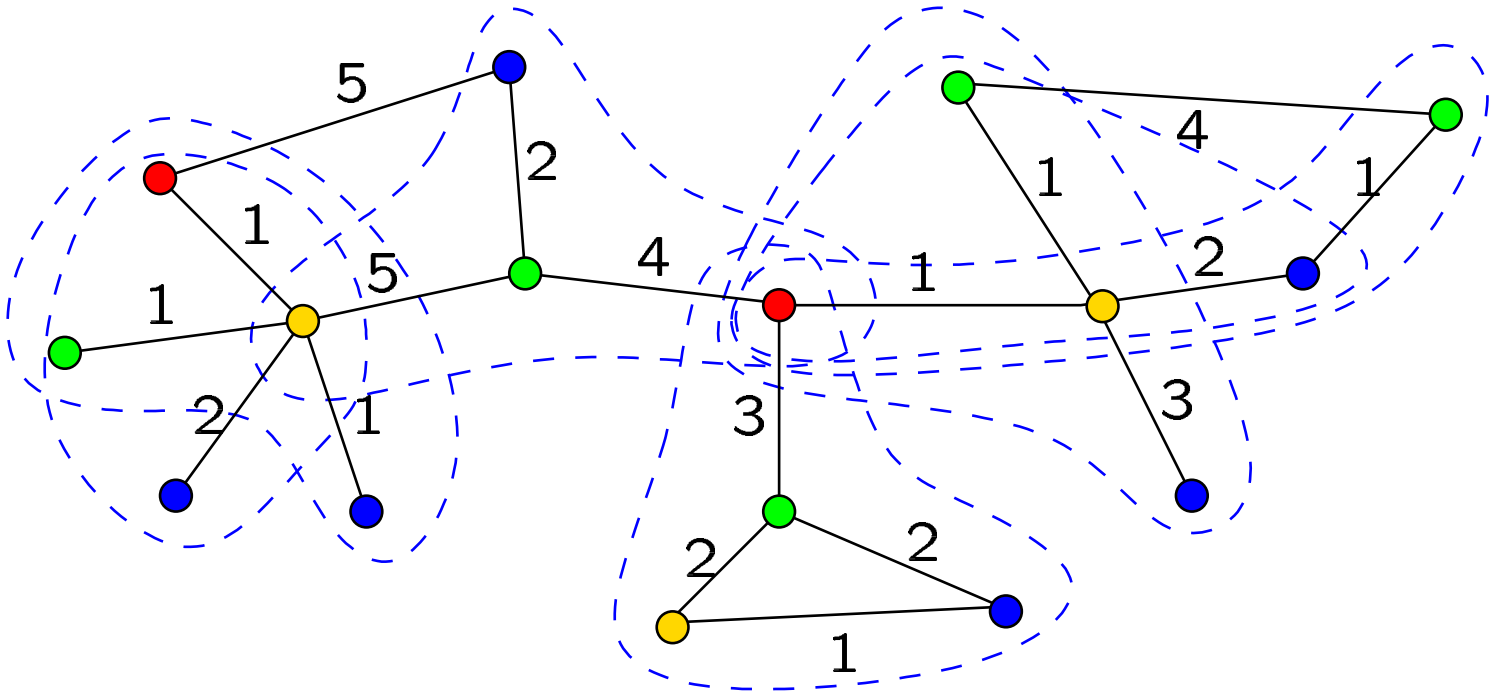
- $\forall i$, le nombre de sommets de couleur i est $\tilde{O}(\sqrt{n})$
- $\forall x$, $B(x)$ possède toutes les couleurs.



Étape 2 : coloration (1/2)

Pour tout sommet $x \mapsto c(x) \in \{1, ..., \lfloor\sqrt{n}\rfloor\}$ tq

- $\forall i$, le nombre de sommets de couleur i est $\tilde{O}(\sqrt{n})$
- $\forall x$, $B(x)$ possède toutes les couleurs.



Ici, cas optimal : on utilise 4 couleurs pour $|B(x)| = 4$

Étape 2 : coloration (2/2)

Théorème. *Une telle coloration c existe pour tout graphe. De plus elle peut être construite en temps polynomial et décrite par un programme de longueur $\tilde{O}(\sqrt{n})$ bits.*

Étape 2 : coloration (2/2)

Théorème. *Une telle coloration c existe pour tout graphe. De plus elle peut être construite en temps polynomial et décrite par un programme de longueur $\tilde{O}(\sqrt{n})$ bits.*

Preuve (l'idée).

1. Choisir aléatoirement un polynôme P de degré $\tilde{O}(\sqrt{n})$ à coefficients dans $\mathbb{Z}_p$ où $p \sim \sqrt{n}$ est premier.
   Alors,
   $$c(x) \;=\; P(x) \bmod p$$
2. Dérandomiser.

Étape 3 : sommets spéciaux

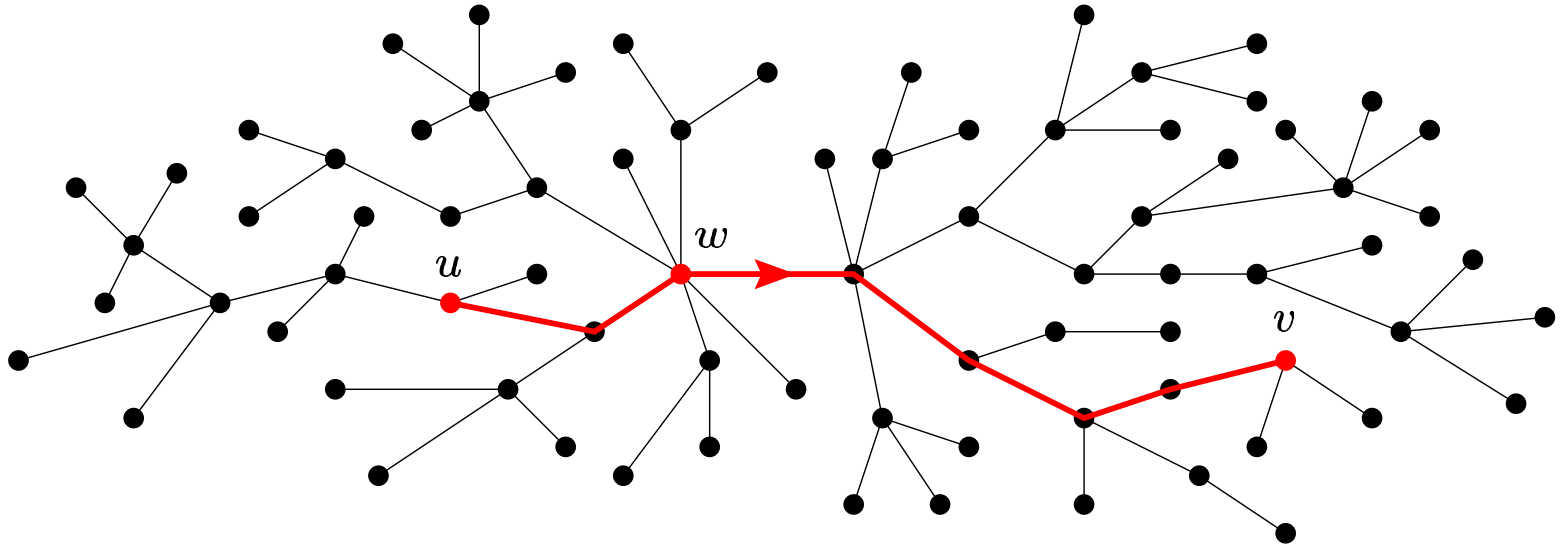
$$R := \{x \mid c(x) = 1\}$$

Puis,

- $x \mapsto r_x$ un sommet de $R \cap B(x)$
- $r \mapsto T_r$ arbre couvrant de coût minimum et de racine $r \in R$

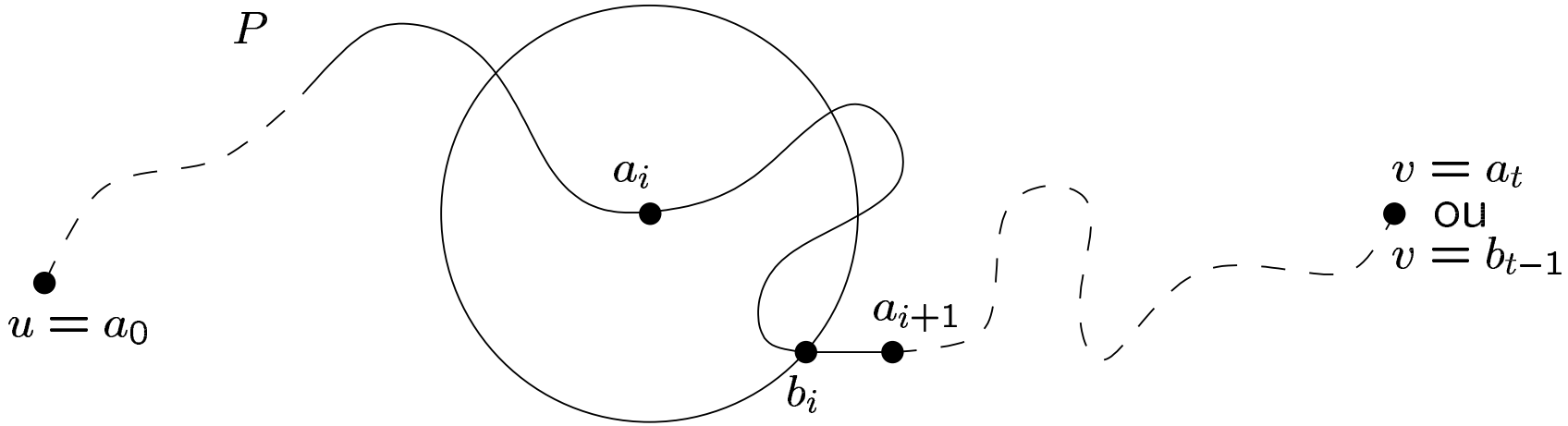
Étape 4 : routage dans un arbre T_r (v2.0)

Théorème [FG01,TZ01]. *Il existe $w \mapsto \ell(w) \in \{0,1\}^k$ tq la route de u vers v peut être déterminée par une fonction de $\ell(u)$ et de $\ell(v)$ indépendante de l'arbre. De plus $k = O(\log^2 n / \log\log n)$.*
(C'est optimal [FG02])



Ainsi, si l'étiquette $\ell_r(v)$ est contenue dans l'en-tête, un sommet w de T_r pourra router dans l'arbre en stockant seulement $\tilde{O}(1)$ bits (sa propre étiquette), et ce même si $\deg_{T_r}(w) \gg (\log n)^{O(1)}$.

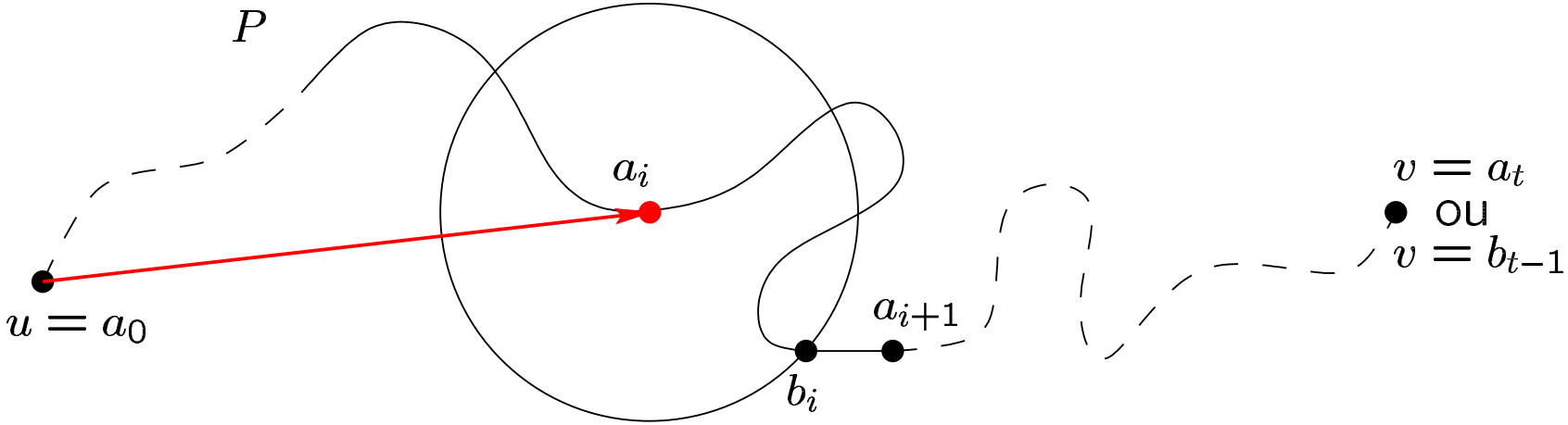
Étape 5 : chemins concis (1/2)



b_i = sommet de $B(a_i)$ le plus loin sur P, a_{i+1} le voisin de b_i
p_i = port de l'arête (b_i, a_{i+1}).

La suite $(b_0, p_1, ..., b_i, p_{i+1}, ..., b_{t-1}, p_t)$ est appelée **chemin concis** de P et t est sa **taille**.

Étape 5 : chemins concis (1/2)

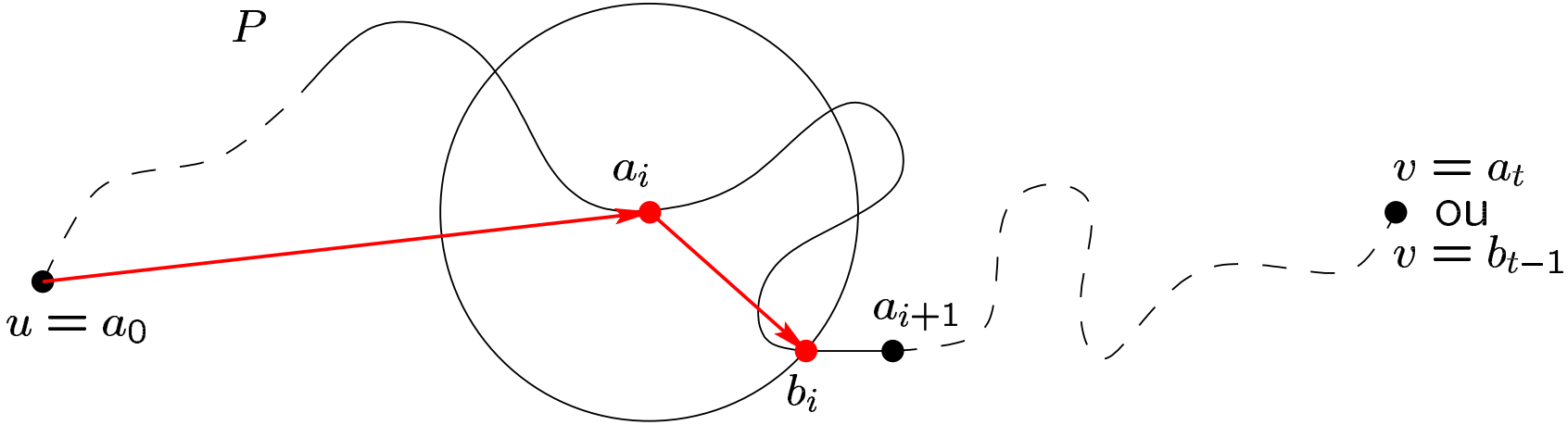


b_i = sommet de $B(a_i)$ le plus loin sur P, a_{i+1} le voisin de b_i
p_i = port de l'arête (b_i, a_{i+1}).

La suite $(b_0, p_1, ..., b_i, p_{i+1}, ..., b_{t-1}, p_t)$ est appelée **chemin concis** de P et t est sa **taille**.

Idée : disposer d'un routage « tout en-tête » qui ne charge pas les sommets intermédiaires ($\neq$ du routage dans les arbres) en s'appuyant sur le routage des boules.

Étape 5 : chemins concis (1/2)

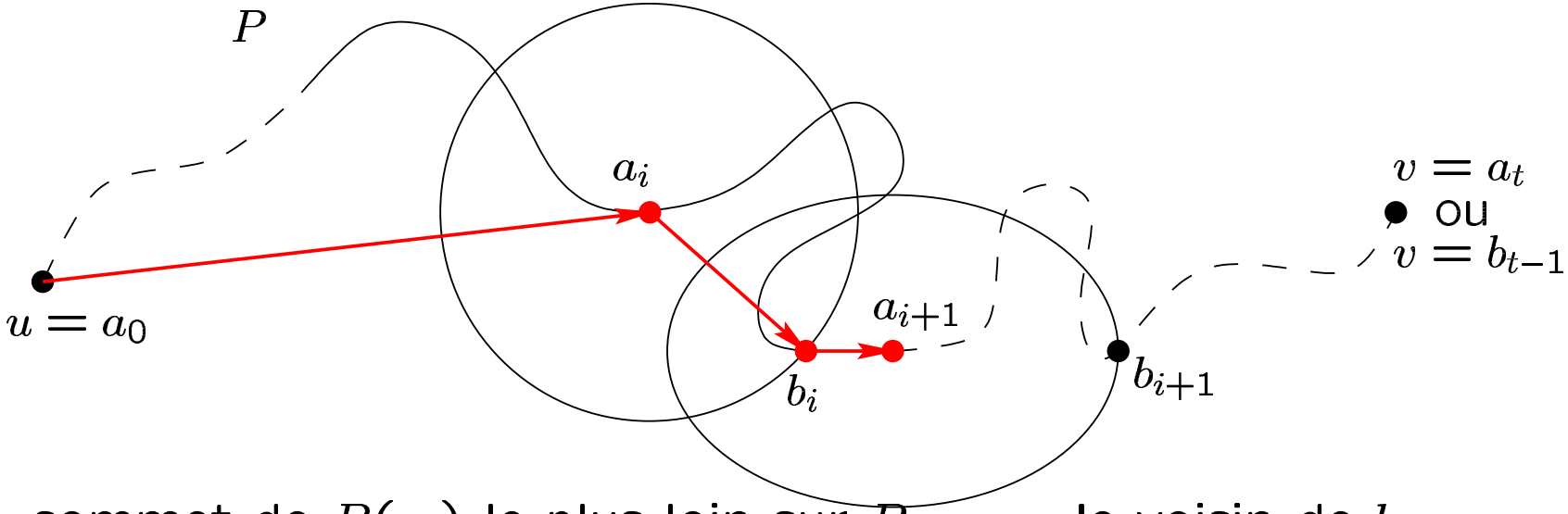


b_i = sommet de $B(a_i)$ le plus loin sur P, a_{i+1} le voisin de b_i
p_i = port de l'arête (b_i, a_{i+1}).

La suite $(b_0, p_1, ..., b_i, p_{i+1}, ..., b_{t-1}, p_t)$ est appelée **chemin concis** de P et t est sa **taille**.

Idée : disposer d'un routage « tout en-tête » qui ne charge pas les sommets intermédiaires ($\neq$ du routage dans les arbres) en s'appuyant sur le routage des boules.

Étape 5 : chemins concis (1/2)

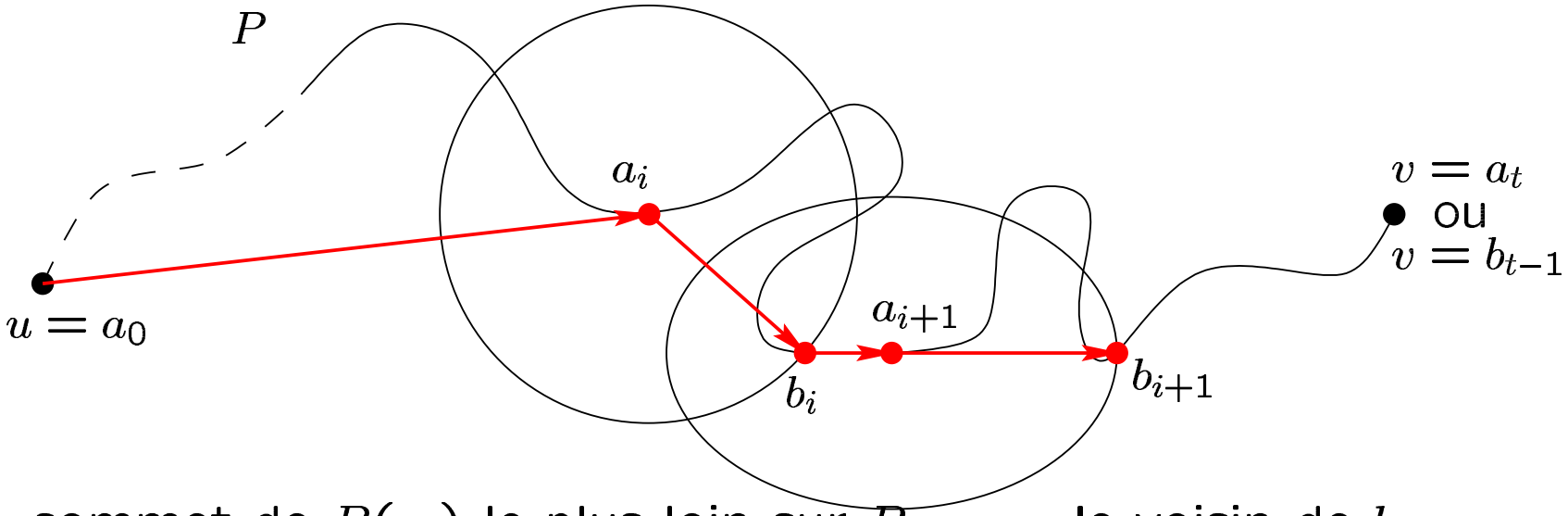


b_i = sommet de $B(a_i)$ le plus loin sur P, a_{i+1} le voisin de b_i
p_i = port de l'arête (b_i, a_{i+1}).

La suite $(b_0, p_1, ..., b_i, p_{i+1}, ..., b_{t-1}, p_t)$ est appelée **chemin concis** de P et t est sa **taille**.

Idée : disposer d'un routage « tout en-tête » qui ne charge pas les sommets intermédiaires ($\neq$ du routage dans les arbres) en s'appuyant sur le routage des boules.

Étape 5 : chemins concis (1/2)

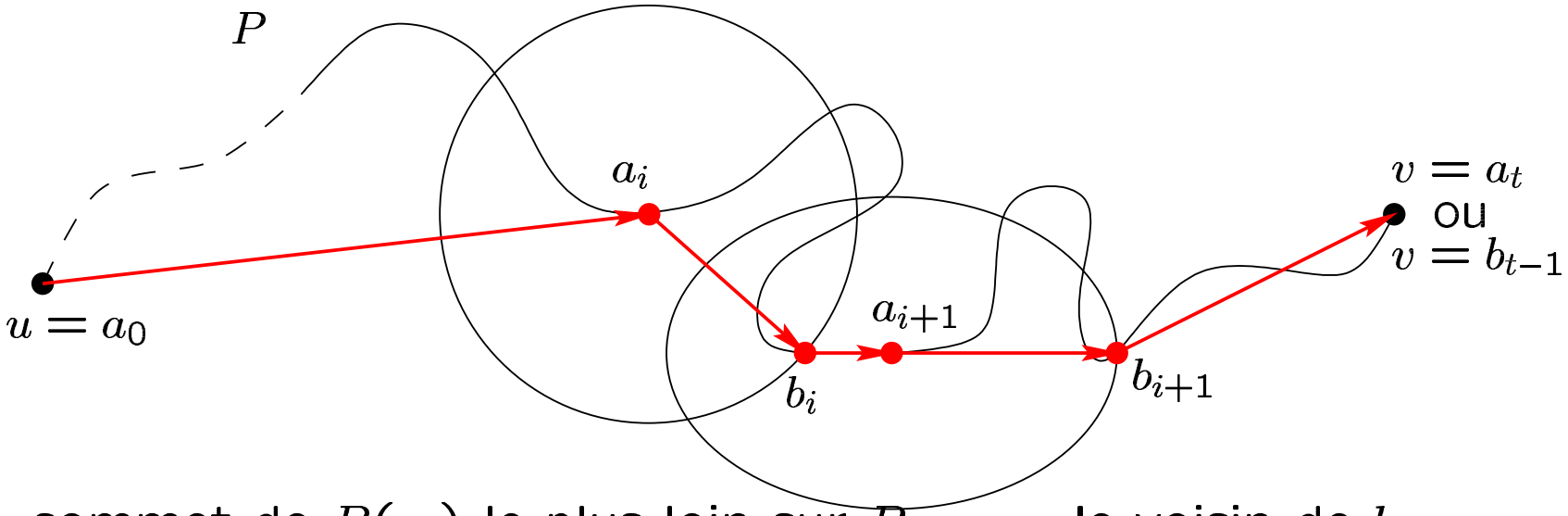


b_i = sommet de $B(a_i)$ le plus loin sur P, a_{i+1} le voisin de b_i
p_i = port de l'arête (b_i, a_{i+1}).

La suite $(b_0, p_1, ..., b_i, p_{i+1}, ..., b_{t-1}, p_t)$ est appelée **chemin concis** de P et t est sa **taille**.

Idée : disposer d'un routage « tout en-tête » qui ne charge pas les sommets intermédiaires ($\neq$ du routage dans les arbres) en s'appuyant sur le routage des boules.

Étape 5 : chemins concis (1/2)



b_i = sommet de $B(a_i)$ le plus loin sur P, a_{i+1} le voisin de b_i
p_i = port de l'arête (b_i, a_{i+1}).

La suite $(b_0, p_1, ..., b_i, p_{i+1}, ..., b_{t-1}, p_t)$ est appelée **chemin concis** de P et t est sa **taille**.

Idée : disposer d'un routage « tout en-tête » qui ne charge pas les sommets intermédiaires ($\neq$ du routage dans les arbres) en s'appuyant sur le routage des boules.

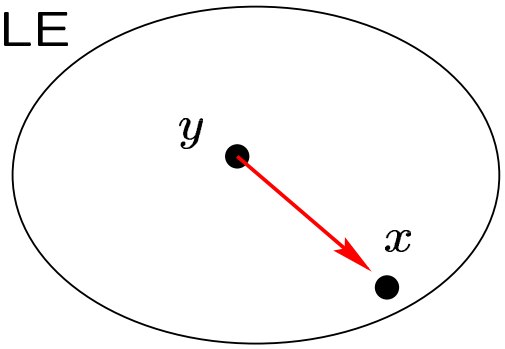
Étape 5 : chemins concis (2/2)

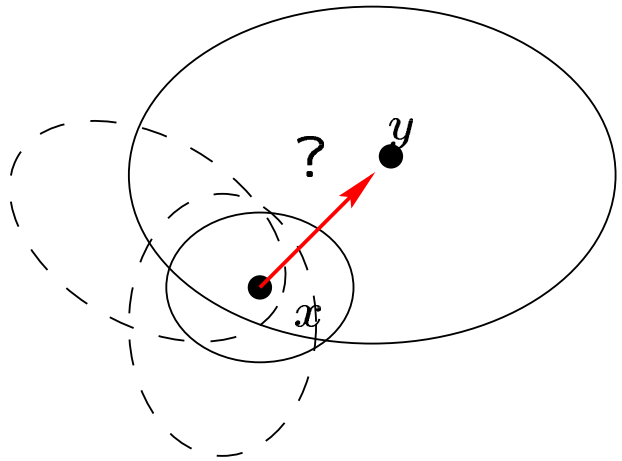
Le schéma a un étirement paramétré par $\varepsilon = 2^{-(t-1)}$, $t \geqslant 1$ entier.
On note $b(x) := \max_{w \in B(x)} d(x, w)$ le rayon de $B(x)$.

Propriété. *Si* $\boxed{x \in B(y)}$ *et* $\boxed{b(x) > \varepsilon\, b(y)}$ *alors tout plus court chemin de* x *à* y *est un chemin concis de taille au plus* $\boxed{t = 1 + \log(1/\varepsilon)}$.

Preuve. Il faut montrer que $\forall 1 < i < t$, $d(a_0, a_i) \geqslant 2^{i-1} b(a_0)$. □

FACILE





Algorithme de routage (squelette) de u vers v

1. Si $v \in B(u)$ alors router dans sa boule

2. Calculer $c = c(v)$

3. Si $c = 1$ ou $c = c(u)$ alors on se débrouille avec ses tables locales

4. Sinon router vers $w \in B(u)$ tq $c(w) = c$, puis w se débrouille

Informations stockées par u (1/2)

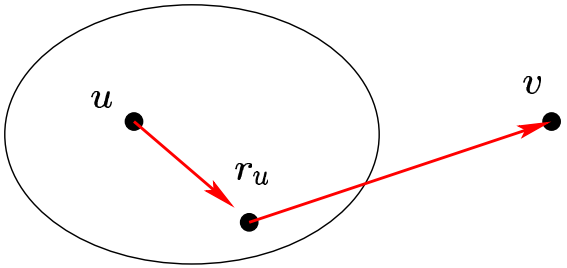
- coefficients du polynôme pour calculer la couleur
- noms des sommets de $B(u) \cup R$ ainsi que le port pour s'y rendre par un plus courts chemins
- l'étiquette $\ell_r(u)$ de routage de u dans T_r, $\forall r \in R$.

qui représentent $\tilde{O}(\sqrt{n})$ bits

... plus

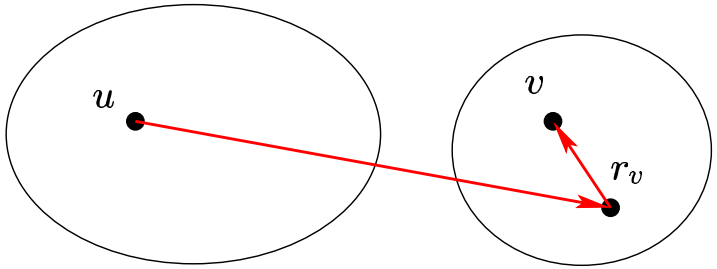
Informations stockées par u (2/2)

... pour tout sommet v de la **même couleur** que $c(u)$, on stocke les informations pour réaliser le plus courts parmi les 3 suivants :



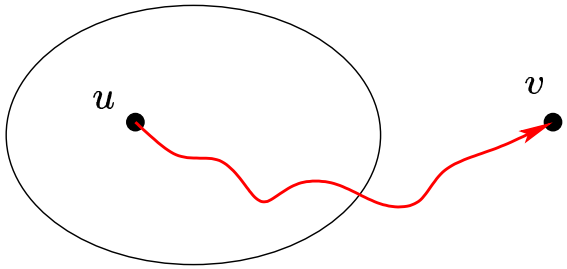
routage A

stoque $\ell_{r_u}(v)$



routage B

stoque $\ell_{r_v}(v)$



routage C

stoque un plus court chemin concis de taille $\leqslant 2t+1$

représentent $\tilde{O}(\sqrt{n}\log D)$ bits en tout pour u.

Analyse du facteur d'étirement

Cas 0. $v \in B(u) \cup R$, alors $s = 1$.

Analyse du facteur d'étirement

Cas 0. $v \in B(u) \cup R$, alors $s = 1$.

Maintenant, la route passe par $w \in B(u)$ avec $c(w) = c(v)$
(si $c(u) = c(v)$ alors $w = u$)

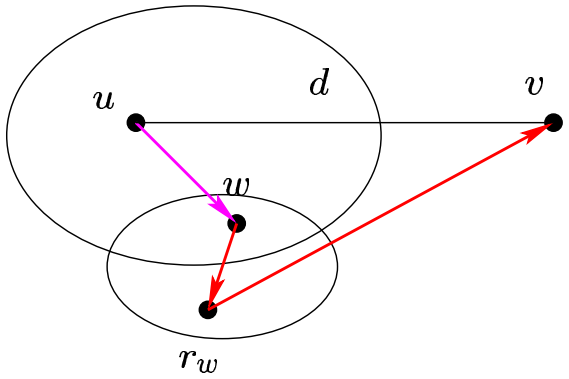
On va montrer que $\boxed{s \leqslant 3 + 4\varepsilon}$,
puis prendre $\varepsilon = 2^{-(t-1)} < 1/(4D)$, par ex. $t = \lceil \log D \rceil + 4$.

On pose $d = d(u, v)$.

Analyse du facteur d'étirement

Cas 1. $b(w) \leqslant \varepsilon b(u)$.

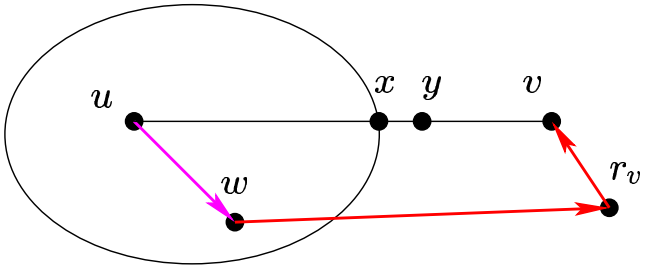
Dans ce cas, on considère le routage A en w: $u \rightsquigarrow w \rightsquigarrow r_w \rightsquigarrow v$.



$$
\begin{aligned}
d(u,w) + d(w,r_w) + d(r_w,v) &\leqslant \\
d(u,w) + d(w,r_w) + [d(r_w,w) + d(w,u) + d(u,v)] &\leqslant \\
b(u) + b(w) + b(w) + b(u) + d &\leqslant \\
(3 + 2\varepsilon)d.
\end{aligned}
$$

Analyse du facteur d'étirement

Cas 2. Sur tout plus court chemin entre u et v, il n'existe pas d'arête (x, y) avec $x \in B(u)$ et $d(y, v) < (1 - \varepsilon)b(v)$.
Dans ce cas, on considère le routage B en w: $u \rightsquigarrow w \rightsquigarrow r_v \rightsquigarrow v$.

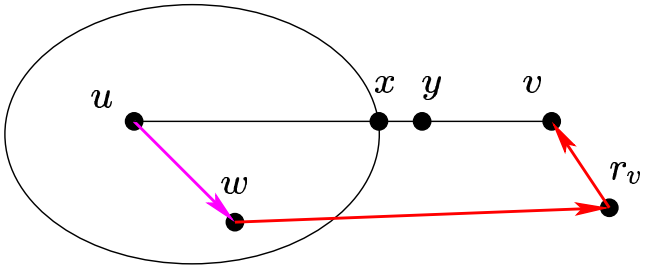


$$d(u, y) + d(y, v) = d \geqslant b(u) + (1 - \varepsilon)b(v)$$

$$\Rightarrow \boxed{b(u) + b(v)} \leqslant d + \varepsilon b(v) \leqslant \boxed{d + (\varepsilon\, 2d)} \quad \text{car } b(v) \leqslant d + b(u) \leqslant 2d$$

Analyse du facteur d'étirement

Cas 2. Sur tout plus court chemin entre u et v, il n'existe pas d'arête (x, y) avec $x \in B(u)$ et $d(y, v) < (1 - \varepsilon)b(v)$.
Dans ce cas, on considère le routage B en w: $u \rightsquigarrow w \rightsquigarrow r_v \rightsquigarrow v$.



$$d(u,y) + d(y,v) = d \;\geqslant\; b(u) + (1-\varepsilon)b(v)$$

$$\Rightarrow \boxed{b(u) + b(v)} \leqslant d + \varepsilon b(v) \leqslant \boxed{d + (\varepsilon\, 2d)} \quad \text{car } b(v) \leqslant d + b(u) \leqslant 2d$$

$$\begin{aligned} d(u,w) + d(w,r_v) + d(r_v,v) &\leqslant \\ b(u) + [b(u) + d + b(v)] + b(v) &\leqslant \\ 2(b(u) + b(v)) + d &\leqslant \\ (3 + 4\varepsilon)d. & \end{aligned}$$

Analyse du facteur d'étirement

Cas 3. Entre u et v il y a un telle arête (x, y).

Comme $d(y, v) < (1 - \varepsilon)b(v)$, on a $y \in B(v)$ mais aussi $b(y) > \varepsilon b(v)$. Donc il existe un plus court chemin concis de taille $\leqslant t$ entre y et v. Comme $x \in B(u)$ est voisin de y, on donc a un plus court chemin chemin concis de taille $\leqslant t + 1$ entre u et v.

$w \in B(u)$ et $b(w) > \varepsilon b(u)$ (cas 1). Donc il existe un chemin concis de w à u de taille t. Au total, il existe un chemin concis entre w et v de taille $\leqslant 2t + 1$. En considérant le routage C, on a alors un routage de coût au plus $u \rightsquigarrow w \rightsquigarrow u \rightsquigarrow v$, c'est-à-dire $3d$. □

... amélioration

en collaboration avec Mikkel Thorup

AT&T Labs – Research, 2004

Nouveau schéma (bis)

`Caractéristiques :`

- Schéma universel
- Indépendance des noms
- Information : $\tilde{O}(\sqrt{n})$ bits/sommet (pour tout diamètre D)
- Étirement : $s \leqslant 3$
- En-têtes : $O(\log^2 n / \log\log n)$ bits
- Constructible en temps polynomial
- Décision : temps $O(1)$ par sommet

Questions ?