

# **Auto-Stabilisation**

Alain BUI

Professeur

Département de Mathématiques et Informatique

[alain.bui@univ-reims.fr](mailto:alain.bui@univ-reims.fr)

## Motivation

- Tolérance aux pannes transitoires
- Finir par garantir la correction du comportement global du système vis à vis des spécifications de l'algorithme

## Définition

- □ les conditions de départ de l'algorithme, il finit par fonctionner correctement vis à vis des spécifications du problème qu'il est censé résoudre

## Origine du problème

- Dijkstra 74
- Lamport 83

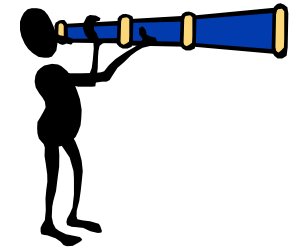
# Les modèles

- Modèle à états [Dij74]
  - modèle originelle
  - abstraction des communications : tout site perçoit l'état des variables de ses sites adjacents (voisins)
- Modèle à registres
  - Modélisation des communications
  - Chaque site détient un registre par ligne de communication sur lequel il peut lire ou écrire des informations et accéder en lecture au registre associé au site adjacent.
- Modèle à passage de messages

## Modèles à états

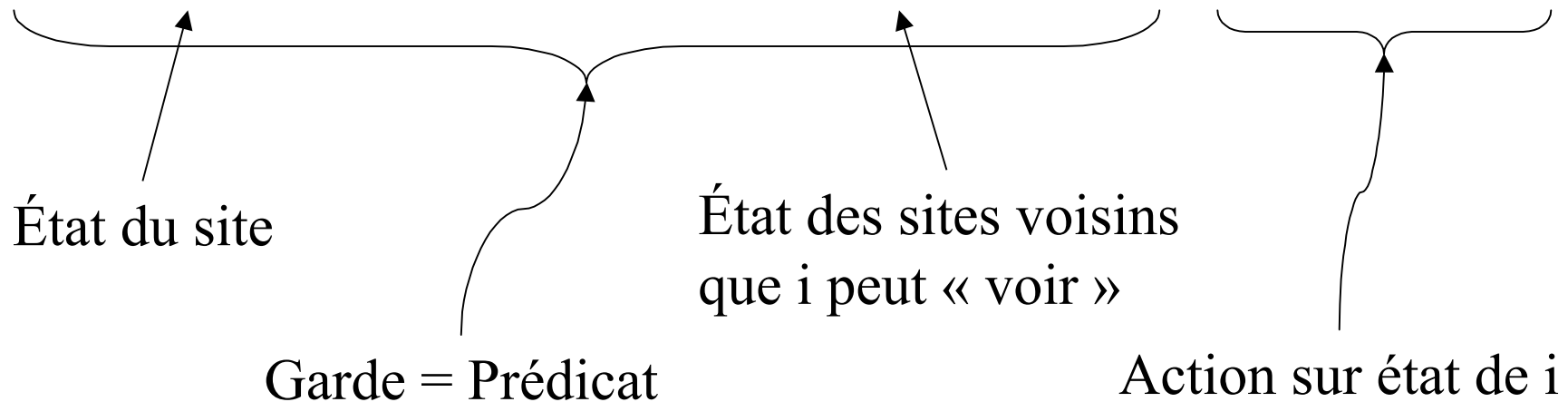
- Abstraction des communications : tout site peut voir l'état des ses voisins
- Etat d'un site = valeurs de chacune de ses variables
- Configuration = états de chacun des sites
- Algorithmes = ensemble de règles gardées
- Notion de démon
  - Ordonnancement des actions de l'algorithme qui peuvent s'effectuer
  - Notion de démon centralisé / distribué
  - Propriété dite d' «équité » du démon : faiblement / fortement
- Complexité

## Modèle à états : algorithme



- { < garde >  $\square$  < action > }
  - Exemple : site d'identité i

$(\text{état}_i = \text{blanc}) \square (\square j \square \text{Vois}_i, \text{état}_j = \text{noir}) \square \text{état}_i = \text{noir}$



- Comment l'algorithme s'exécute t-il?

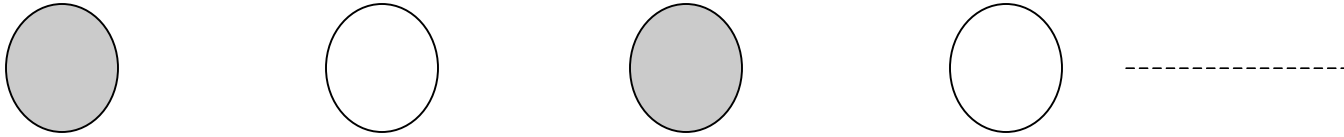
## Modèle à états : exécution de l'algorithme

- Algorithme = { règles }
- Si pour une règle, Garde est Vrai alors la règle est dite *déclenchable*.
- Un site est dit déclenchable si une des ses règles l'est.
- Dans un algorithme, plusieurs sites peuvent être déclenchables
  
- Utilisation d'un démon = ordonnanceur qui choisit un ou plusieurs sites pour qu'il(s) exécute(nt) leurs actions déclenchables

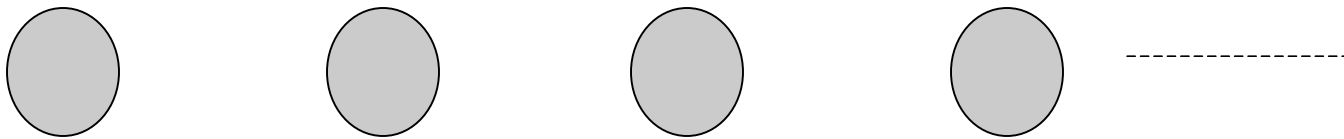
# Démon

- Démon centralisé : choix d'un et un seul site déclenchable
- Démon distribué : choix d'un ou plusieurs sites déclenchables
- Notion d'équité pour le démon
  - *Non équitable* : démon peut ne jamais choisir un site même si celui ci est infiniment déclenchable
  - *Faiblement équitable* : tout site infiniment continûment déclenchable **finira** par être choisi par le démon
  - *Fortement équitable* : tout site infiniment (pas forcément continûment) déclenchable **finira** par être choisi par le démon

∞ment déclenchable



∞ment continûment déclenchable



# Complexité

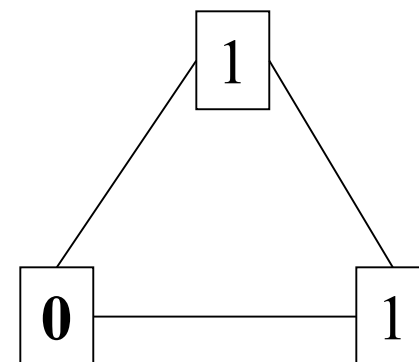
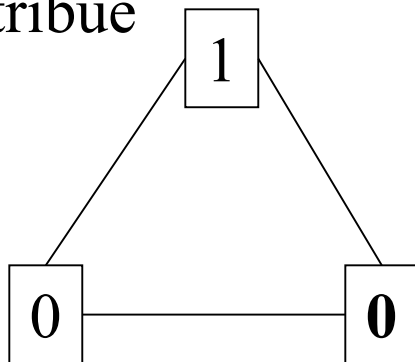
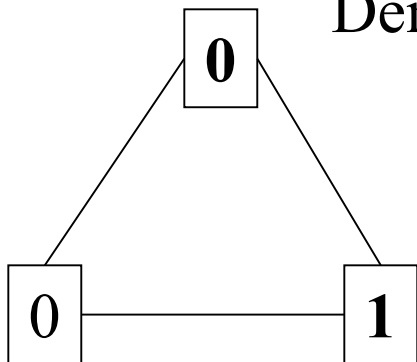
- Nombre d'états par site
- Nombre d'états global
  
- Temps: nombre de « tours »; Tour = chaque fois que des sites qui étaient déclenchables ont exécuté une action ou n'ont plus la possibilité d'exécuter une action à la suite de la modification de l'état d'un ou plusieurs sites

## Exemple: Exclusion Mutuelle

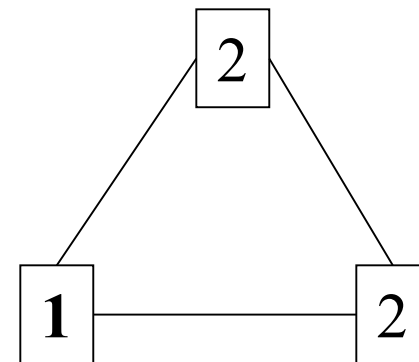
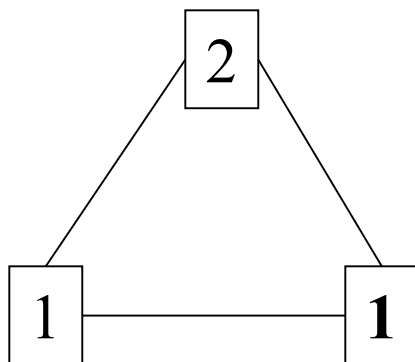
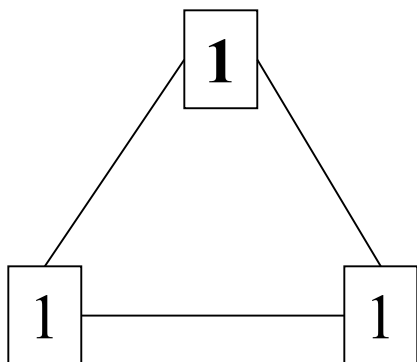
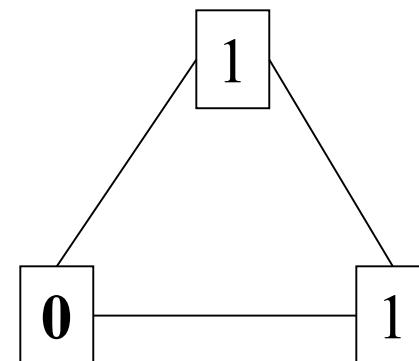
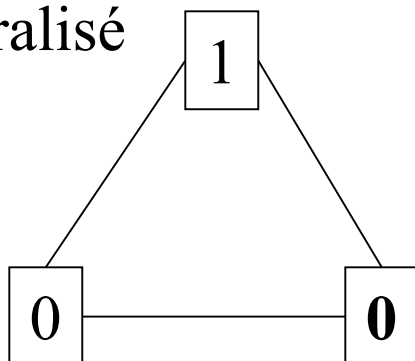
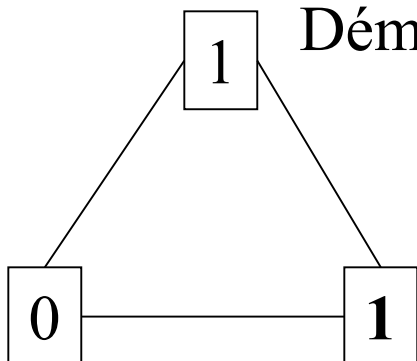
- Rappel : notion de sûreté (safety) et de vivacité (liveness)
- Topologie: anneau unidirectionnel de  $n$  sites d'identités  $1, 2, \dots, n$
- Non utilisation des identités (commodités) on ne particularise qu'un seul site par exemple site d'identité 1.
- La notion de privilège (permettant à un site d'entrer en SC) est matérialisé par le fait qu'un site est susceptible d'effectuer une action

- Variables
- $\square \ i \ 1 \leq i \leq n : M_i \text{ booléen}$ 
  - Règles
  - $\square \ i \ 2 \leq i \leq n : M_i \neq M_{i-1} \ \square \ M_i := M_{i-1}$
  - $i = 1 : M_1 = M_n \ \square \ M_1 := (M_1 + 1) \bmod K \ (K > n)$
- Auto-stabilisation : initialisation des variables n'est pas nécessaire  $\Rightarrow$  la configuration de départ peut être quelconque.

### Démon distribué



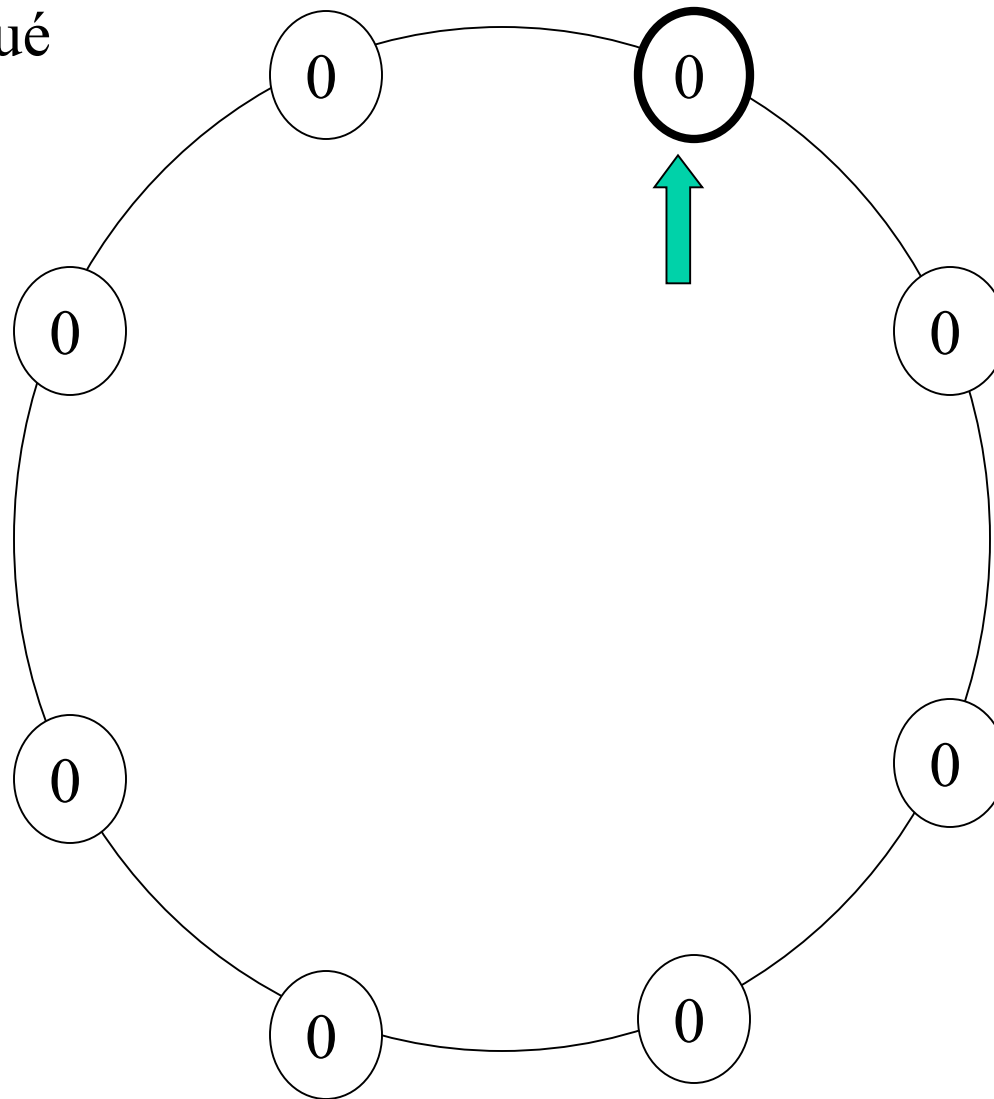
### Démon centralisé



Exécuter l'algorithme sur cet exemple avec un démon

-Centralisé

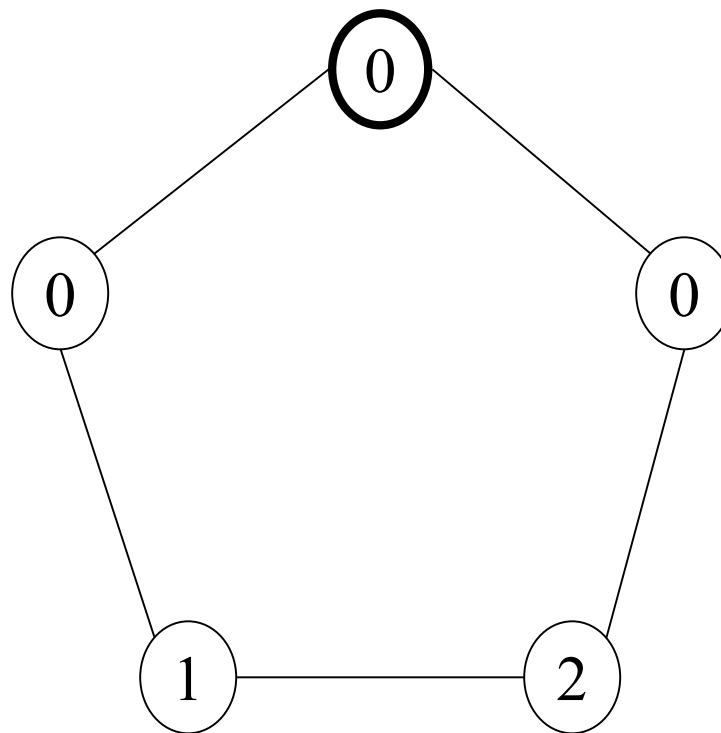
-Distribué



- Sur le même exemple: donner une configuration initiale où la propriété de sûreté n'est pas vérifiée (au moins 3 sites sont déclenchables)
- Dérouler l'algorithme avec démon
  - Centralisé
  - Distribué Equité forte
  - Distribué Equité faible
  
  - Ecrire dans le modèle à états un algorithme de diffusion avec retour sur une chaîne (ne pas considérer l'auto-stabilisation)

Et si  $K < N$  ?

- Donner une exécution à partir de l'exemple suivant qui fasse que l'algorithme ne stabilise jamais.
- On prendra  $K = N-2$



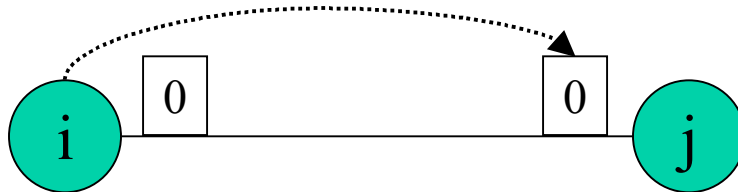
# Modèle à registres



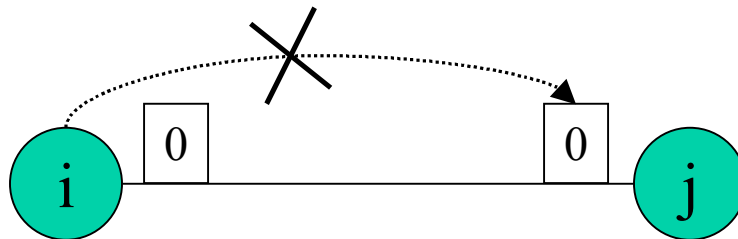
*i* peut écrire dans son registre



*i* peut lire dans son registre



*i* peut lire dans le registre de *j*



*i* ne peut pas écrire dans le registre de *j*

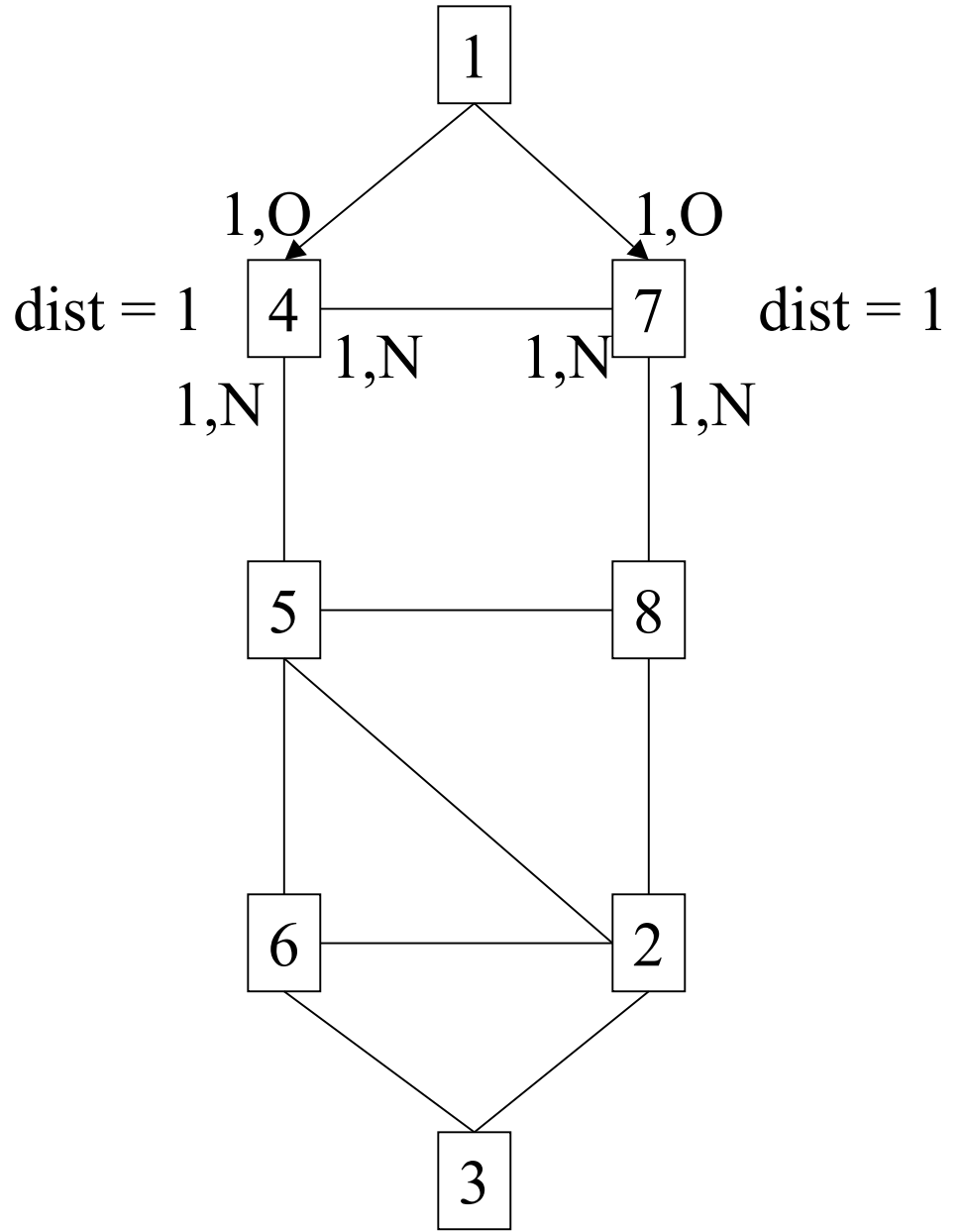
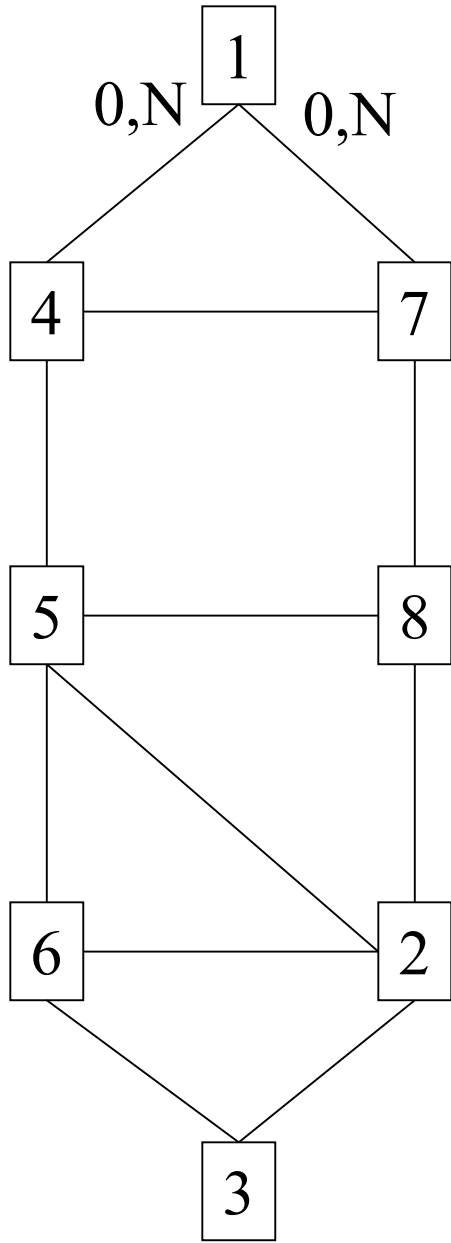
# Un algorithme de construction d'Arbre Couvrant

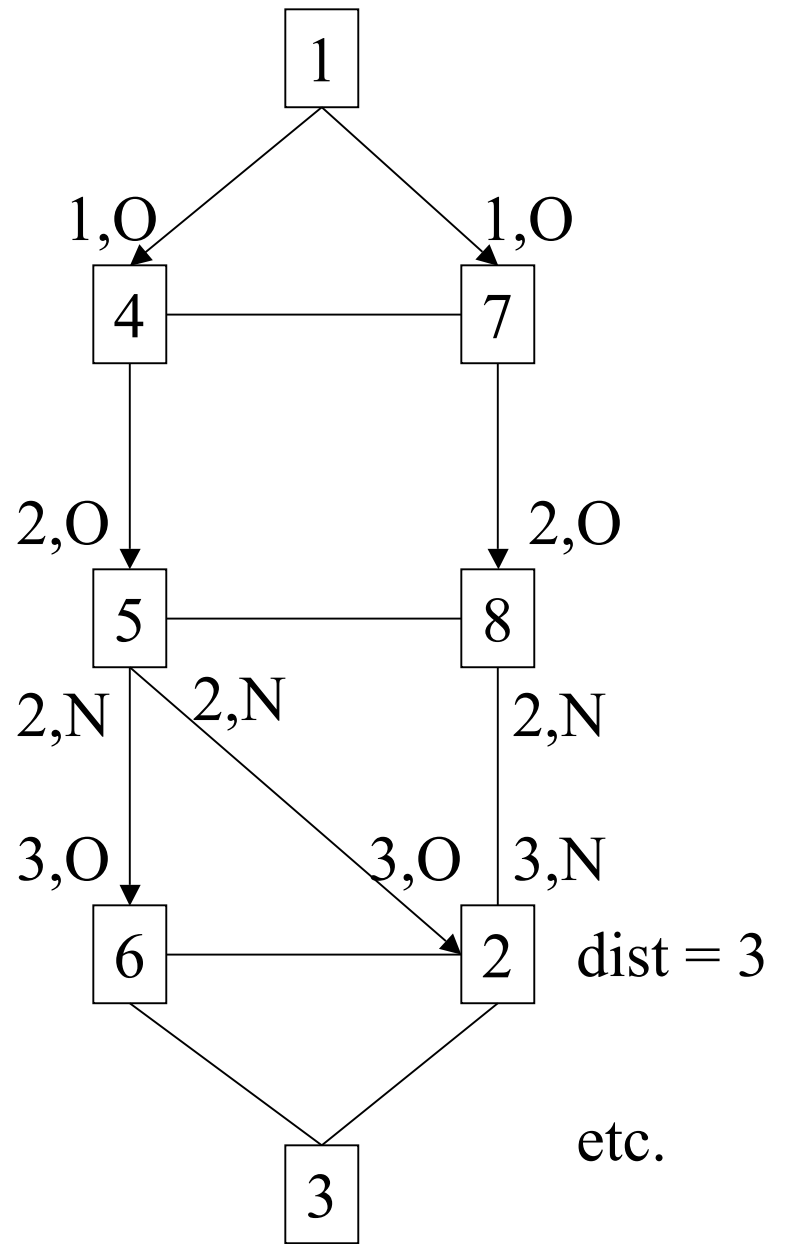
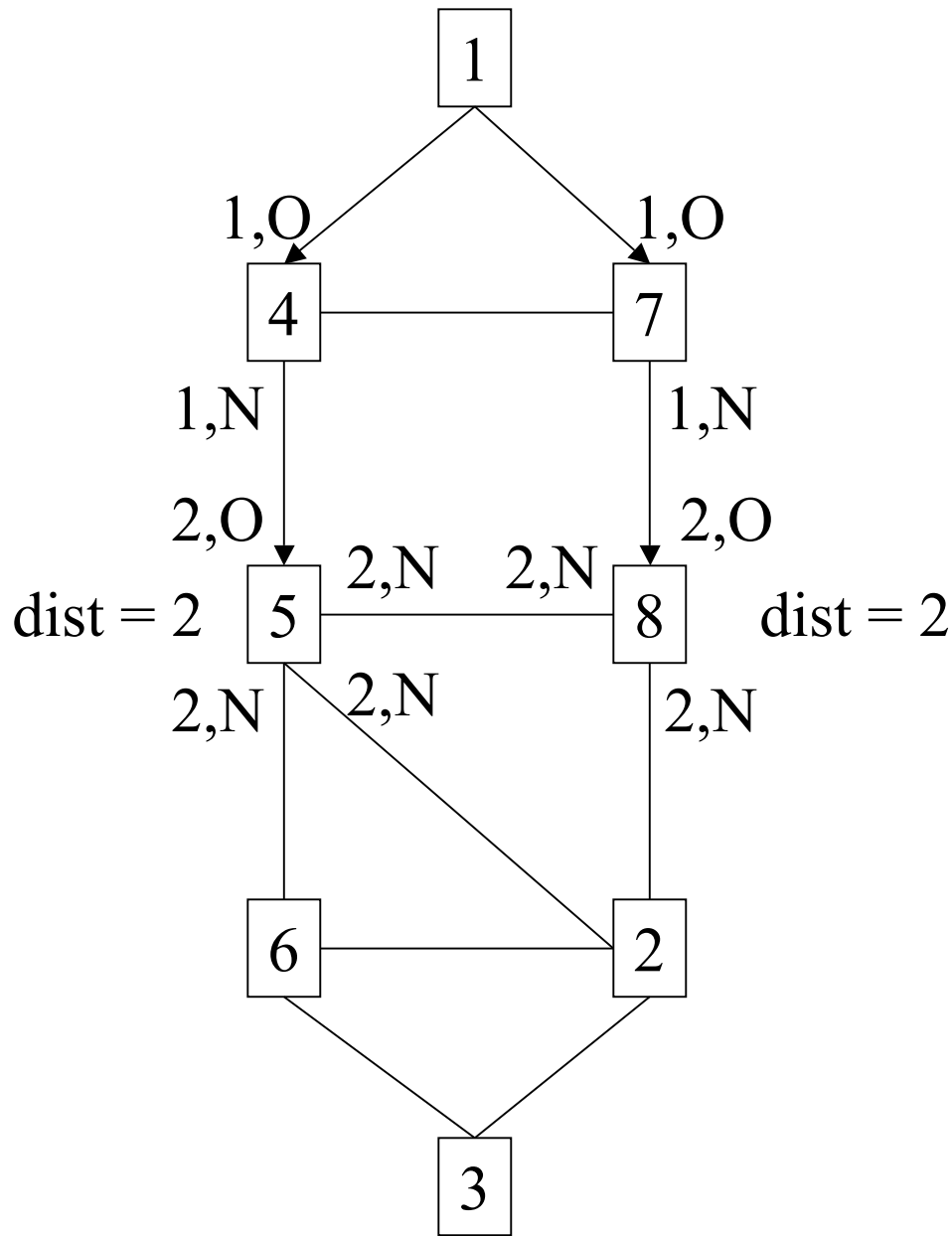
- Principe
  - Algorithme écrit dans le modèle à registre
  - Parcours BFS
  - Un site particulier appelé la racine.
  - Les autres sites ne sont pas différenciables (même algorithme local, non utilisation d'identités)
  - $r_{ij}$  : registre de  $i$  sur la ligne  $(i,j)$
  - $i$  écrit dans  $r_{ij}$  et  $i$  lit dans  $r_{ji}$

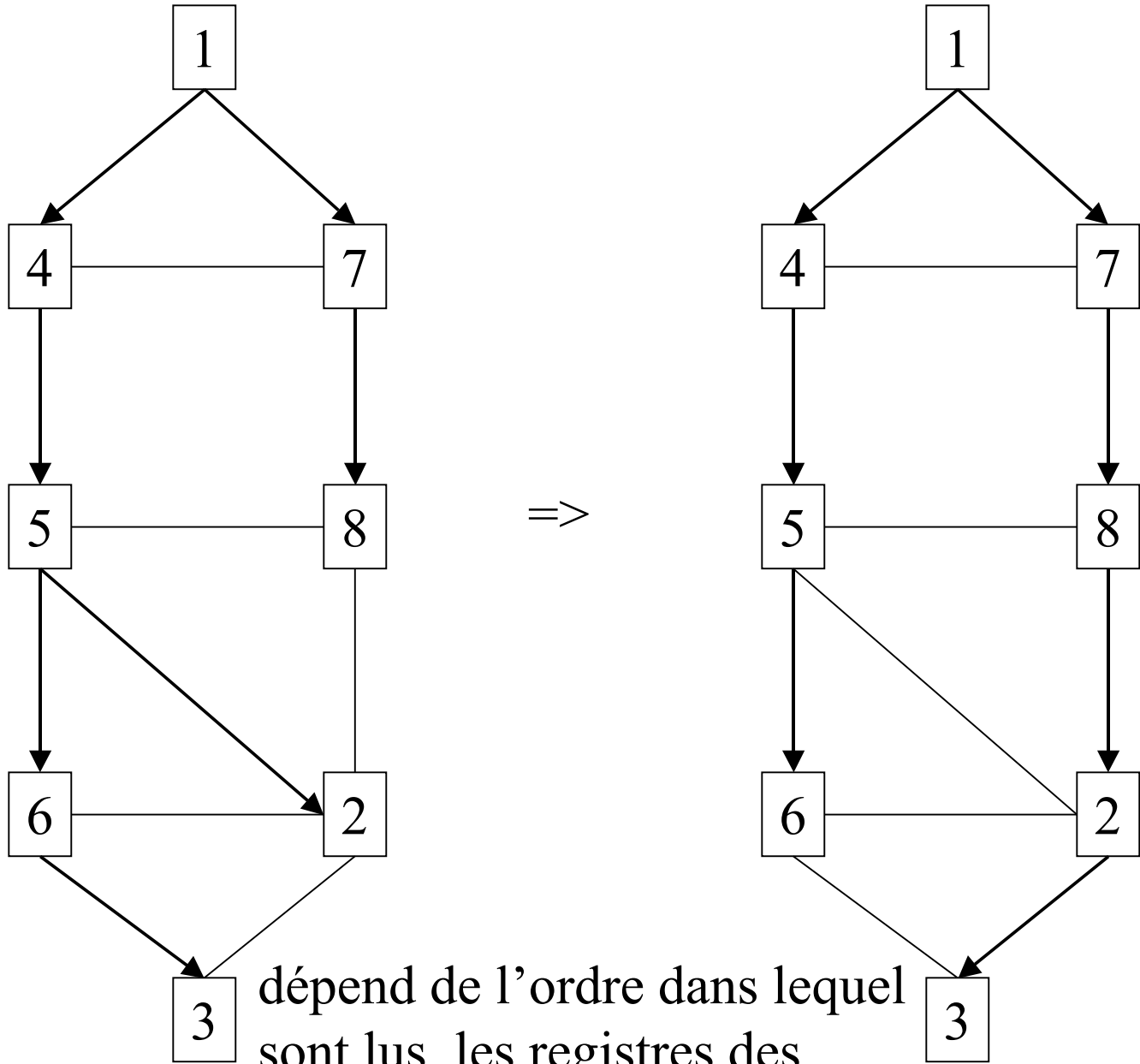
# BFS

- Chaque site calcule sa distance à la racine et transmet (via son registre) le résultat à ses voisins. Chaque site choisit en même temps un site père qui constituera la construction de l'arbre
- 2 informations stockées dans le registre  $r_{ij}$  (x,y)
  - x = distance à la racine :  $r_{ij}.dist$  (valeur entière)
  - y = site père :  $r_{ij}.pere$  ( $\in \{Oui, Non\}$ )
- Construction de proche en proche
  - Racine est à distance 0 d'elle même
  - Voisins de la racine à distance 1
  - Voisins des voisins à distance 2 etc.
  - Père de i = 1 voisin de i qui a sa distance à la racine la plus petite

- Algorithme pour la racine
  - $\forall j \in \text{Vois}_i$ , Ecrire ( $r_{ij} \leftarrow (0, \text{Non})$ ) boucle  $\infty$
- Algorithme pour tous les sites  $\neq$  racine
  - $\forall j \in \text{Vois}_i$ , Lire ( $r_{ji}$ )
  - $\text{Vu} \leftarrow \text{Faux}$
  - $\text{madist} \leftarrow \text{Min}_{j \in \text{Vois}_i} \{ r_{ji} \} + 1$
  - Tant que  $\text{nbre} \neq |\text{Vois}_i|$  faire
    - Si  $\neg (\text{Vu})$  ET  $r_{ji} \cdot \text{dist} = \text{madist} - 1$  alors boucle  $\infty$ 
      - Ecrire ( $r_{ij} \leftarrow (\text{madist}, \text{Oui})$ )
      - $\text{Vu} \leftarrow \text{Vrai}$
    - Sinon Ecrire ( $r_{ij} \leftarrow (\text{madist}, \text{Non})$ )
    - $\text{Nbre} \leftarrow \text{Nbre} + 1$
    - Finfaire



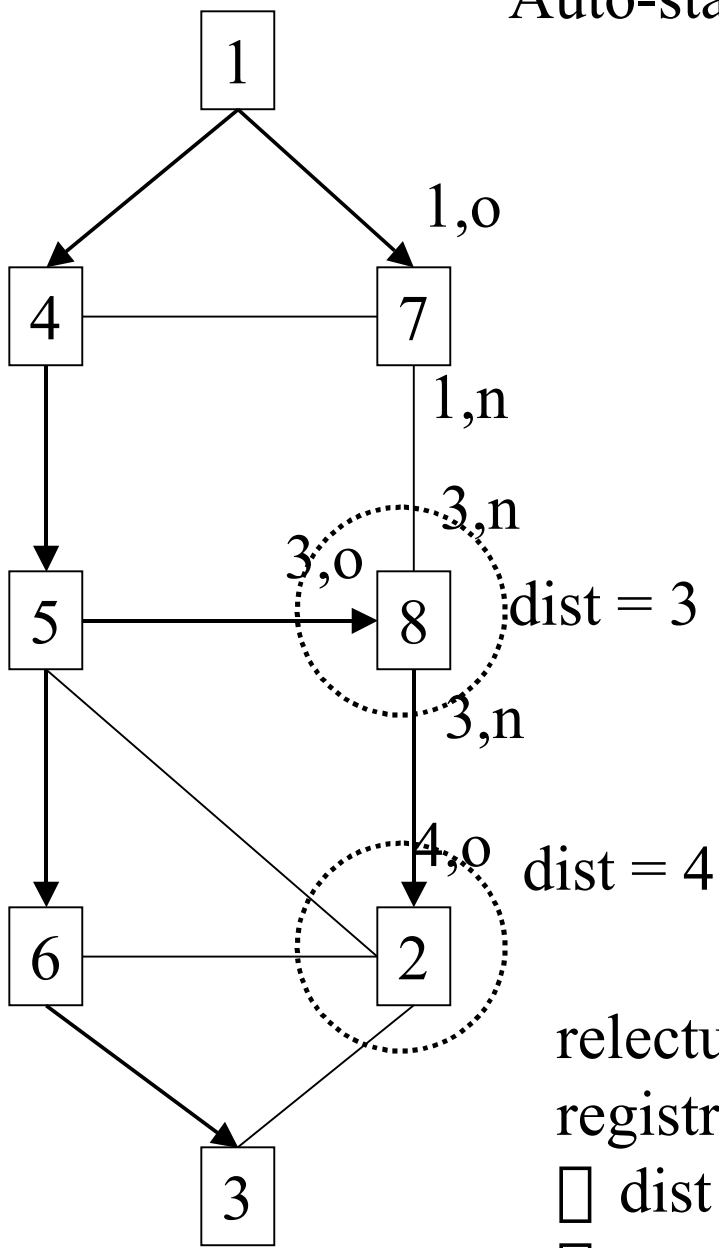




dépend de l'ordre dans lequel  
sont lus les registres des  
sites adjacents

# Auto-stabilisation

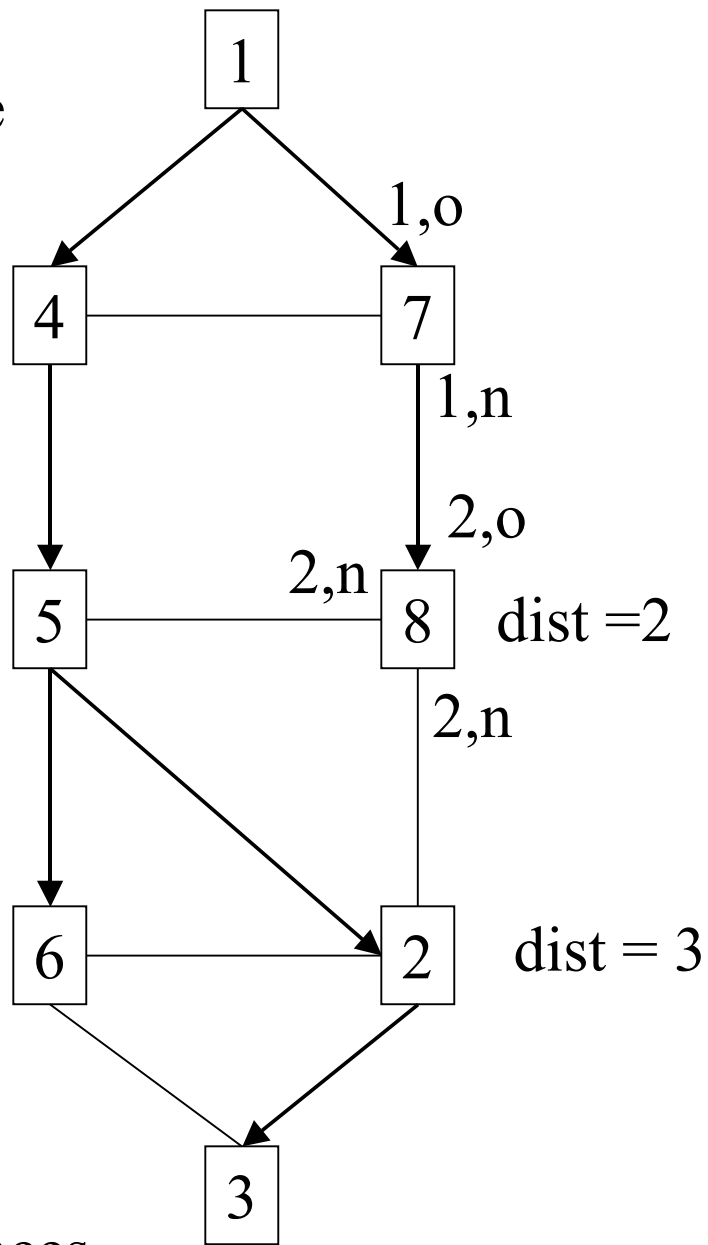
variables père  
corrompue



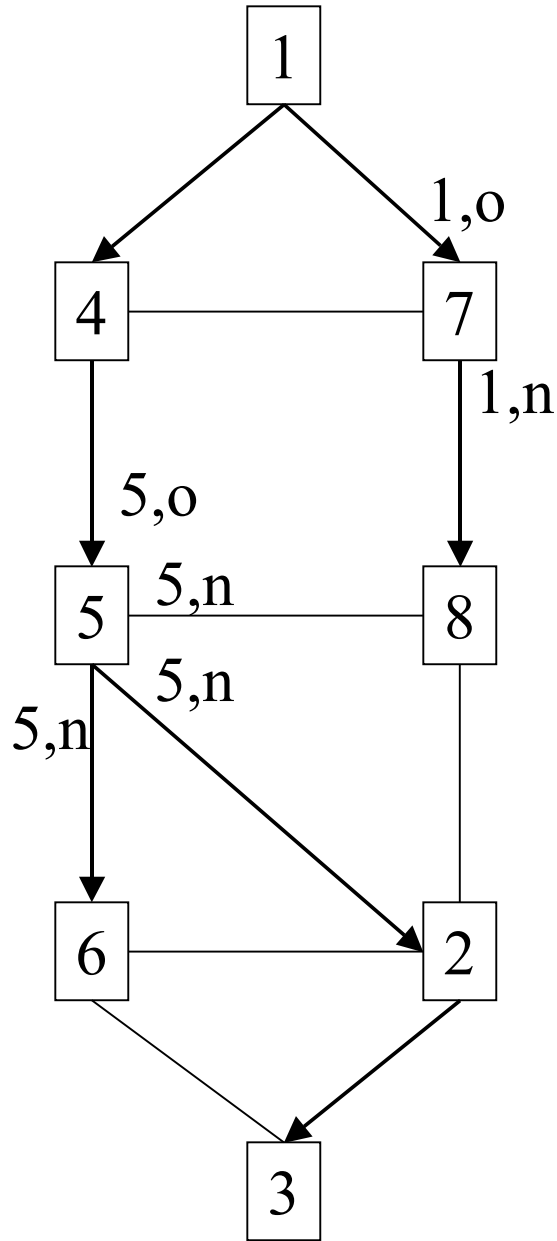
relecture des registres

□  $\text{dist} = 2$

□ recalcul des distances



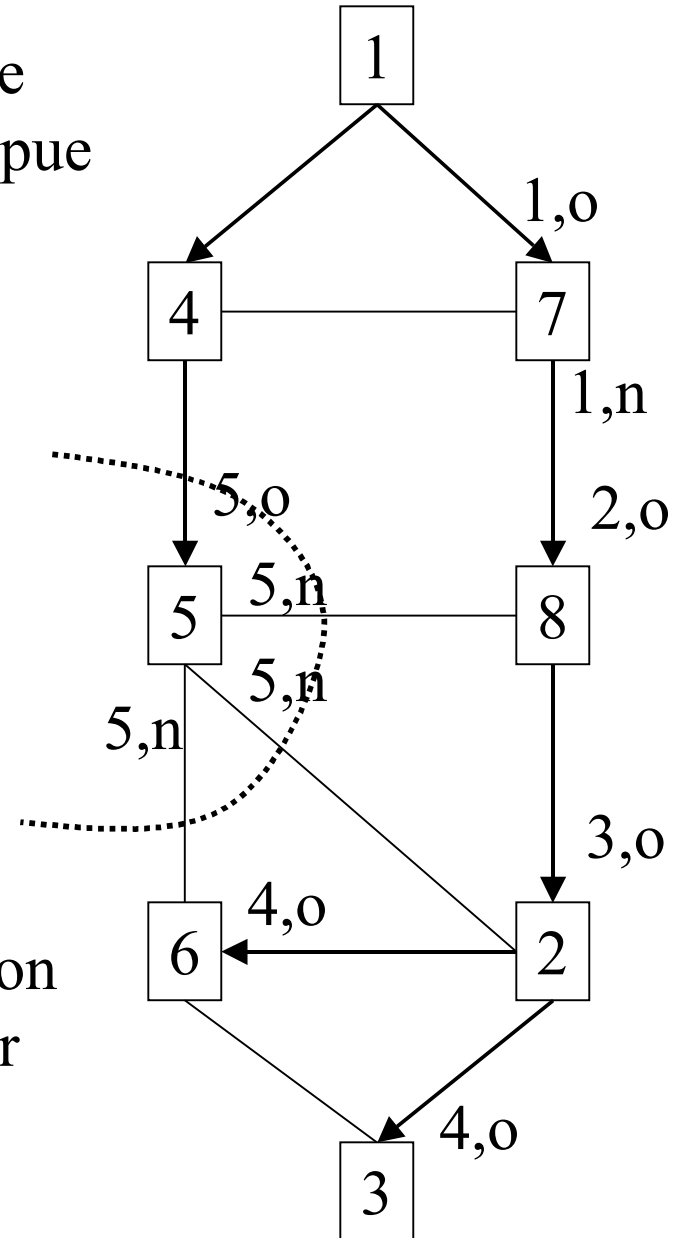
# Auto-stabilisation



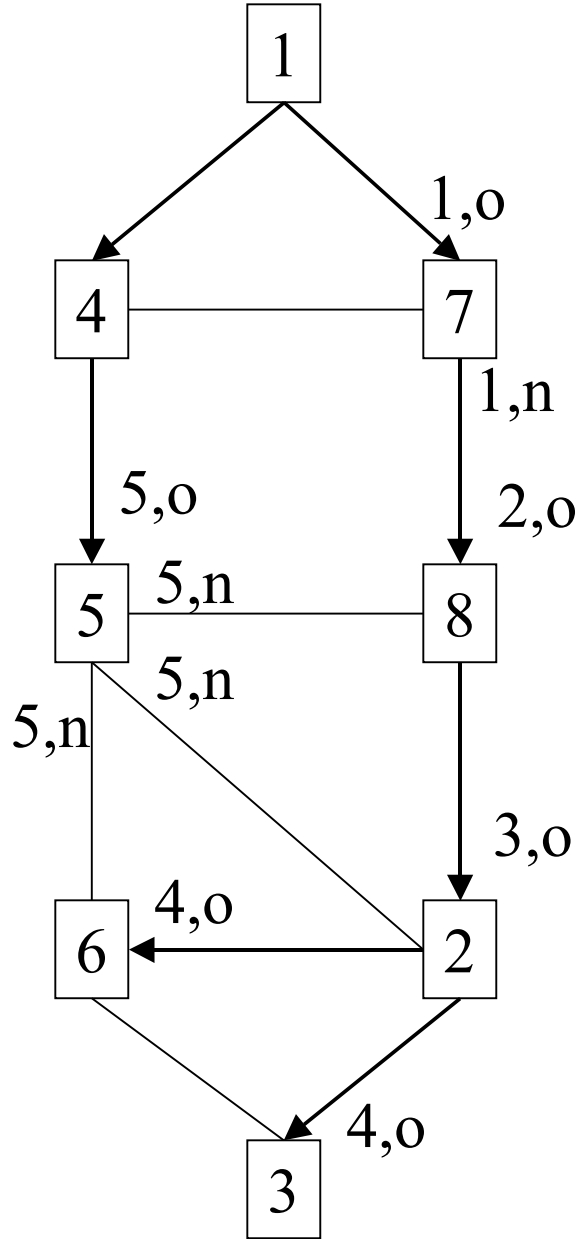
distance  
corrompue

⇒

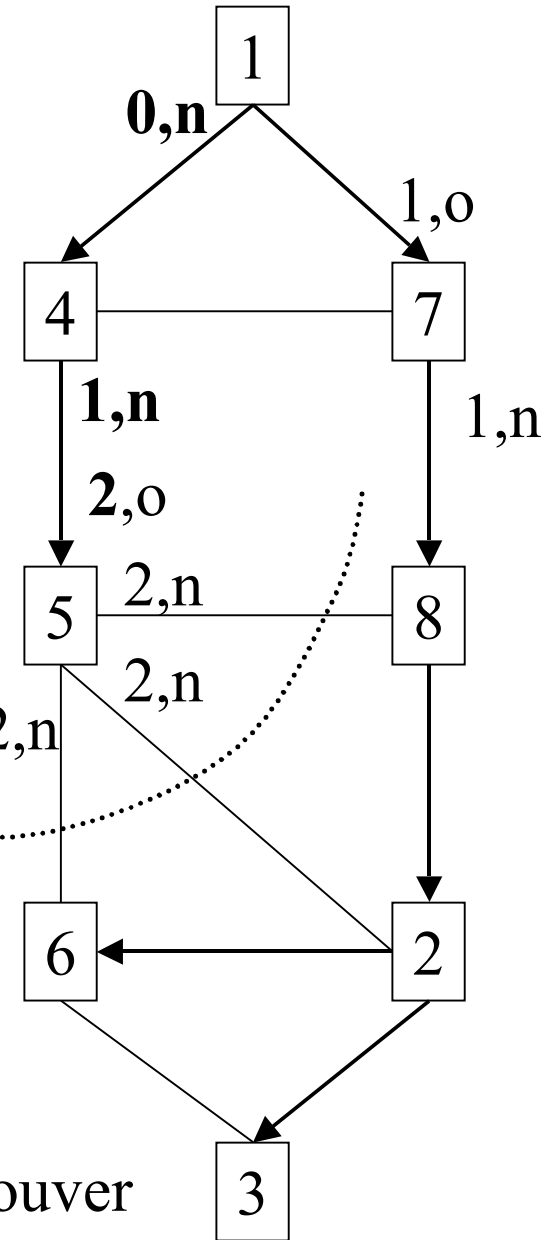
propagation  
de l'erreur



# Auto-stabilisation



distance  
corrompue



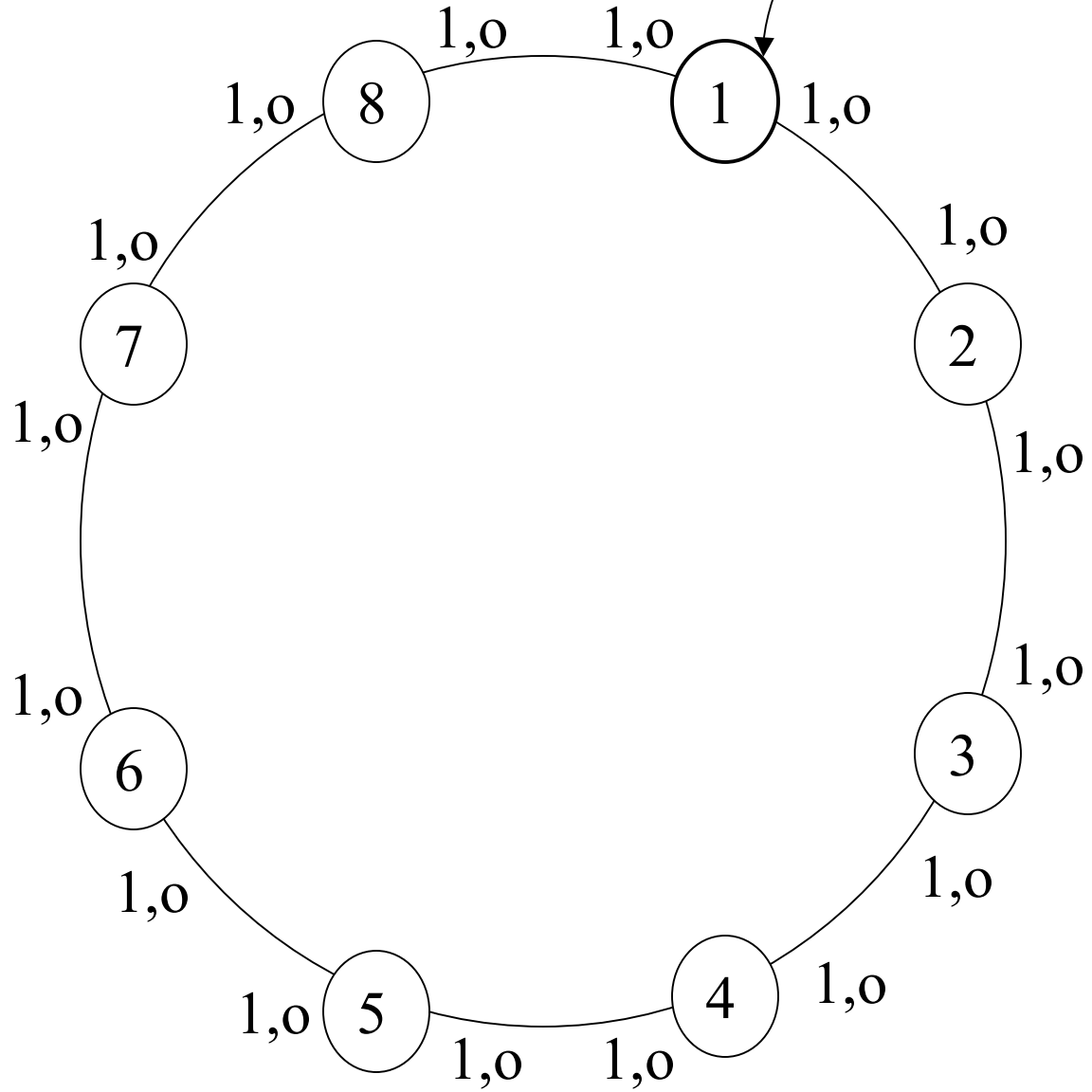
correction  
par la racine

propagation  
de la correction

On finira par retrouver  
un AC correct

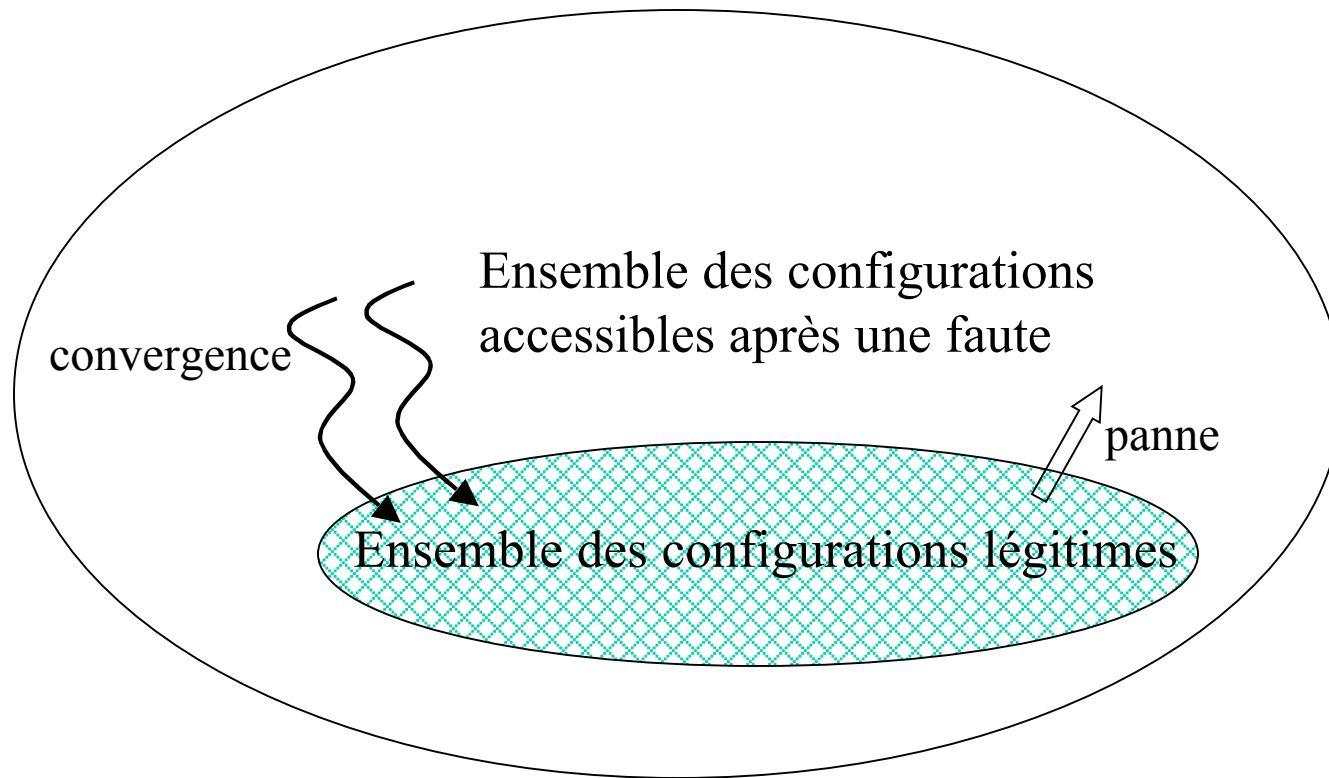
Exercice: appliquer l'algorithme

RACINE



# Propriétés des algorithmes auto-stabilisants

- Clôture et Convergence



PIF sur un arbre

# Preuve d'auto-stabilisation

# Modèles à passage de messages