Stabilité des routes dans les réseaux inter-domaines.

Johanne Cohen* — **Sylvie Delaët** **

* *Laboratoire PRiSM (CNRS - Université de Versailles, Saint Quentin)*
45, avenue des États-Unis – 78035 Versailles Cedex
Johanne.Cohen@prism.uvsq.fr

** *Laboratoire LRI (Université Paris Sud 11, Orsay)*
Bâtiment 490, université Paris Sud 91405 Orsay Cedex
Sylvie.Delaet@lri.fr

RÉSUMÉ. *Le problème de la construction d'arbre de plus courts chemins est étudié en présence de noeuds ayant des métriques propres. Ce problème est dérivé du problème de routage inter-domaine où chaque noeud (ou système autonome -AS) à sa propre politique de routage. Ici, nous reprenons la modélisation de Griffin et al sous forme de jeu en l'étendant au fait qu'il puisse exister des coalitions. Ensuite nous caractérisons une condition nécessaire sur l'existence d'un tel arbre. Finalement, nous présentons un algorithme auto-stabilisant sur une instance possédant un arbre de plus courts chemins dans ce contexte.*

ABSTRACT. *The shortest path tree problem are studied in the context that each node has this own metric. This problem is connected to inter-domain routing where each node (AS) has policy-based metric. Here, we extend the work of Griffin et al. In fact, we also considere that there exist some coalitions. Next, we characterize the structure of the instance having such tree. Finally, we present a self-stabilizing algorithm and we show that it converges to a solution of an instance if it instance contains such shortest path tree.*

MOTS-CLÉS : *Plus court chemin, routage inter-domaine, auto-stabilisation, théorie des jeux.*

KEYWORDS: *Shortest path, inter-domain routing, self-stabilization, game theory*

1. Introduction

Le problème de la construction d'arbre de plus courts chemins enracinés en un sommet u dans un graphe $G = (V, E)$ a été étudié de façon intensive dans le cadre centralisé (l'algorithme a une vue globale du graphe). Deux principaux algorithmes ont trouvés : celui de Bellman et Ford (Cormen *et al.*, 2001) et celui de Dijkstra (Dijkstra, 1971). Par la suite, des variantes distribuées (chaque noeud a une vue locale du graphe : son voisinage et son propre état) de ces algorithmes ont été réalisées, puis adaptées aux réseaux sous forme de protocole de routage.

Les protocoles de routage construisent les chemins permettant d'achemeniner les données entre deux endroits du réseaux. Par exemple, le protocole *Routing Information Protocol* (RIP) correspond à une adaptation de l'algorithme de Bellman et Ford, et le protocole *Open Shortest Path First* ($OSPF$) à celui de Dijkstra (voir (Huitema, 2001)). En effet les protocoles de routage dynamiques des réseaux locaux (une seule entité gérante) tel que RIP et texttt$OSPF$ sont essentiellement basés sur des algorithmes distribués pour la résolution du problème du plus court chemin. Dans le cadre du réseau Internet, ce n'est plus le cas. En effet, Internet est composé de systèmes autonomes (*Autonomous System*) qui s'échangent des messages afin de pouvoir assurer le routage. Chaque AS correspond à un ensemble de réseaux et de routeurs sous une administration unique et posséde sa propre politique de routage (correspondant aux préférences de cette adminitration d'utiliser une partie du réseau)

Actuellement, le protocole de routage *Border gateway protocol* BGP est utilisé pour le routage dans l'Internet. En effet, il permet à chacun de ses AS d'utiliser ses propres politiques de routage pour déterminer ces propres routes. Par conséquent, il peut diverger dans le sens où les différentes politiques de routage sont en conflit et cela implique une certaine instabilité des routes. Dans (Griffin *et al.*, 2002), la stabilité des routes est étudiée. Il y est prouvé qu'étant donné une fonction de préférence par sommet, le problème de stabilité des routes est NP-complet. De plus, des propriétés sur les instances ayant des routes stables y sont extraites et, à partir de cela, un algorithme autostabilisant (simple) est donné. Par la suite, dans (Griffin *et al.*, 2000), un algorithme détectant des cycles liés à des conflits de politiques grâce à l'historique des chemins. Plusieurs améliorations ont été trouvées en utilisant des mécanismes de jetons (Ahronovitz *et al.*, 2006), en utilisant des algorithmes aléatoires (Ibrahim *et al.*, 2004).

Dans (Dasgupta *et al.*, 2006), le problème a été étendu au fait que les sommets pourraient avoir les mêmes politiques de routage. Il a été prouvé que ce problème était NP-complet (Cohen *et al.*, 2008) même si uniquement deux types de politiques de routage étaient présents et dans (Dasgupta *et al.*, 2006), un algorithme faiblement auto-stabilisant a été donné. Le travail présenté dans cet article est une extention de ces travaux. Il étudie la construction (auto-stabilisante) distribuée d'un arbre de plus courts chemins dans le contexte où des sommets de graphes peuvent former une coalition.

Le document s'organise de la façon suivante. La section suivante est une présentation formelle de notre problème. Dans la section 3, nous présenterons des condi-

tions suffisantes sur des instances possèdant des arbres stables. Ensuite, dans les sections 4 et 5, nous présentons un algorithme autostabilisant construisant un arbre stable.

2. Problème du plus court chemin concurrent

Dans cette section, nous présentons notre modèle et les notations que nous allons utiliser par la suite.

Nous considérons le problème du plus court chemin dans un graphe $G = (V, E)$ avec V correspondant à l'ensemble des sommets et E à celui des arêtes. Dans l'algorithmique classique, des algorithmes centralisés et des algorithmes distribués voire auto-stabilisants (voir (Tel, 2000)) existent pour résoudre ce problème. Quel que soit le modèle retenu (centralisé ou distribué), les sommets collaborent pour construire une structure minimisant la même fonction objective. Or ces algorithmes ne sont pas adaptables si chaque nœud suit son propre objectif. C'est le cadre de cette étude.

En effet, nous supposerons ici que tous les sommets de V ont d'abord un but commun : construire un arbre couvrant enraciné en un sommet distingué noté r. Chaque arête portant des coûts différents associés aux différents sommets, chaque sommet a par ailleurs pour objectif de minimiser le coût de son chemin dans l'arbre à destination de r. Ceci peut entraîner des conflits entre les ensembles de sommets définis par une métrique commune.

Plus précisement, l'ensemble des sommets V est découpé en p parties disjointes $V_1, V_1, V_2, \ldots, V_p$ tel que $V = \bigcup_{i=1}^{p} V_i$. Dans un souci de simplification, nous considérons que chaque élément de cette partition correspond à une couleur. Les arêtes sont pondérées par un poids de chaque couleur. Formellement, les arêtes du graphe G possèdent un poids, et donc le graphe G est pondéré par la fonction $w : E \rightarrow \mathbb{N}^{*p}$. Pour chaque entier $i \in [1 \ldots p]$, la fonction $w_i : E \rightarrow \mathbb{N}^*$ est définie comme

$$\forall e \in E, w_i(e) = x_i \text{ si et seulement si } w(e) = (x_1, \ldots, x_i, \ldots, x_p)$$

Le coût du chemin $C_{u \rightarrow v}$ d'origine u et de destination v pour la couleur i est

$$w_i(C_{u \rightarrow v}) = \sum_{e \in C_{u \rightarrow v}} w_i(e)$$

Un arbre couvrant T de G possède un unique chemin $T_{v \rightarrow u}$ entre n'importe quelle paire de sommets (v, u). Le but de cet article est de construire un arbre couvrant stable c'est-à-dire non améliorable en coût pour chaque sommet. En effet, l'objectif de chaque sommet v est de minimiser le coût pour sa couleur i de son chemin vers la racine r . Nous notons $cost(v) = w_i(T_{v \rightarrow r})$ le coût à minimiser du sommet v de couleur i (*i.e.* appartenant à V_i).

La construction de l'arbre couvrant est faite par la sélection d'un de ses arcs sortant par chaque sommet non racine du graphe. Par conséquent, un sommet peut potentiel-

lement améliorer son coût en changeant l'arc sélectionné. Une telle modification ne doit pas nuire pas à la propriété de connexité de l'arbre couvrant (voir figure 1).

Nous formulons maintenant les définitions de sommet et d'arbre stables. Un sommet v est stable pour un arbre T s'il ne peut pas choisir un autre voisin $z \in \Gamma_G(v)$ qui lui permettrait de diminuer son coût $cost(v)$ dans l'arbre $T^{'}$ engendré.

Définition 1 (Sommet stable) *Soit $G = (V, E)$ un graphe pondéré par une fonction $w : E \rightarrow \mathbb{N}^{*p}$ et r un sommet distingué de V. Soit $\mathcal{P}$ une partition de V composée de p parties $V = \cup_{i=1}^{p} V_i$. Soit T un arbre couvrant de G. Le sommet $v \in V_i$ est* stable dans T pour la métrique w *si et seulement si v satisfait la condition suivante :*

$$\forall z \in \Gamma_G(v), cost(v) = w_i(T_{v \rightarrow r}) \leq w_i((v, z)) + w_i(T_{z \rightarrow r})$$

Considérons un graphe de trois sommets $\{r, v_1, v_2\}$ avec $V_1 = \{v_1, r\}$ et $V_2 = \{v_2\}$. Les coûts sont représentés grâce aux couleurs sur à la figure 1. Le premier dessin représente l'instance du graphe sans construction d'arbre. Les deux autres dessins représentent des choix différents d'arêtes par les sommets menant aux arbres T_1 et T_2. La couleur du noeud étant reportée sur l'arête sélectionnée. Le sommet v_2 est stable dans l'arbre T_1. En effet, son coût dans T_1 est 2 tandis que s'il choisit l'arête (v_2, r), le chemin allant v_2 lui coutera 3. De même, le sommet v_1 est stable dans l'arbre T_1 car s'il choisit une autre arête, il n'existe plus de chemin le reliant à r.

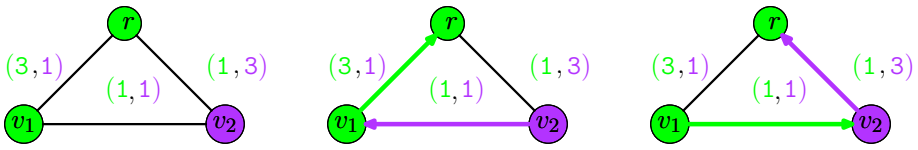

Figure 1. *Instance ayant deux arbres stables T_1 et T_2*

On dira qu'un arbre T est stable si tous les nœuds sont stables.

Reprenons l'exemple précédent (représenté par la figure 1). Comme nous l'avons dit précédemment, les sommets V_1 et V_2 sont stables dans T_1 et donc par définition de la stabilité d'un arbre, T_1 est stable. De plus, en utilisant les même arguments que précédemment, T_2 est stable.

Par contre, il existe des cas où il n'existe pas d'arbre stable. Par exemple, en adaptant l'exemple donné dans (Griffin *et al.*, 2002), considérons un graphe de quatre sommets $\{r, v_0, v_1, v_2\}$ avec $V_0 = \{v_0, r\}$, $V_1 = \{v_1\}$ et $V_1 = \{v_2\}$ représenté par la figure 2. Pour chacun des sommets v_j avec $j \in \{0, 1, 2\}$ le plus court chemin est de coût 2, il passe par le voisin $v_{(j+1) mod 3}$. Le chemin de poids immédiatement supérieur est le chemin direct à la racine qui coûte 3. Tout autre chemin est de poids supérieur à 4. Un arbre couvrant du graphe contient nécessairement au moins l'une des arêtes

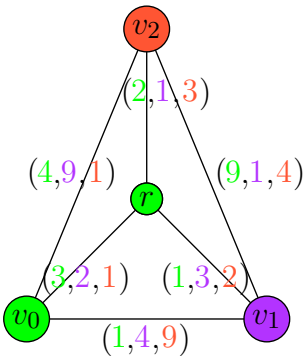

Figure 2. *Instance n'admettant aucun arbre stable.*

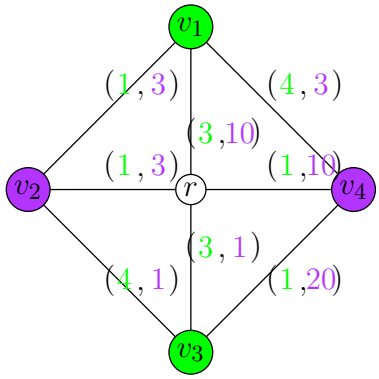

Figure 3. *Instance n'admettant aucun arbre stable.*

(v_j,r). Sous l'hypothèse que le nœud v_j soit sélectionné l'arête (v_j,r) (chemin de coût 3), le sommet $v_{(j-1)mod\ 3}$ n'est stable que s'il sélectionne l'arrête $(v_{(j-1)mod\ 3}, v_j)$ car c'est le chemin le plus court. De plus, le sommet $v_{(j+1)mod\ 3}$ ne peut sélectionner son plus court chemin et se voit contraint de choisir le chemin de poids 3 directement vers la racine. En conséquence le nœud v_j n'est pas stable car en choisissant l'arête $(v_j, v_{(j+1)mod\ 3})$, il peut maintenant atteindre un chemin de coût 2 inférieur à 3. Nous venons de démontrer qu'aucun arbre couvrant de ce graphe ne peut être stable.

De plus, dans (Cohen *et al.*, 2008), même si il y a deux couleurs ($p = 2$), il existe des instances qui ne possèdent pas d'arbre stable : voir par exemple l'instance représentée par la figure 2 que l'on étudira plus longuement dans la section 3.

Analogie avec la théorie des jeux

Dans notre contexte, chaque sommet a son l'objectif correspondant à celui de sa couleur. Cet optique est proche de celle de la théorie des jeux permettant de modéliser la concurrence entre deux joueurs. C'est pour cette raison que nous introduisons ici certains concepts de base de la théorie des jeux.

Un *jeu* (non-répété) est composé d'un ensemble de joueurs J ayant chacun un ensemble de stratégies pures noté S_i pour le joueur $i \in J$. Un *profil pure* de stratégies $s = (s_1, \ldots, s_n)$ est un vecteur qui contient toutes les stratégies pures des joueurs où chaque joueur i, $i \in I$ choisit la stratégie s_i. Pour chaque profil s, chaque joueur $i \in J$ a une fonction de préférence u_i dépendant des stratégies jouées de tous les autres joueurs. Un *équilibre de Nash* est un profil $s = (s_1, \ldots, s_n)$ tel que

$$\forall i \quad \forall s'_i \in S_i \quad u_i(s_1, \cdots, s_i, \cdots, s_n) \geq u_i(s_1, \cdots, s'_i, \cdots, s_n)$$

La table 1 représente la correspondance entre les différentes notions introduites dans la section 2 et la théorie des jeux.

Théorie des jeux	Problème de plus court chemin
Joueurs	Sommets du graphe
Coalition	Ensembles des sommets d'une même couleur
Ensemble de stratégies	Ensembles des arcs sortants
Fonction de préférence	Opposé de la fonction du coût
Equilibre de Nash	Arbre stable

Tableau 1. *Correspondance entre la théorie des jeux et notre problème*

Résultats de la littérature

Dans (Cohen *et al.*, 2008), il a été prouvé que

Théorème 1 *Décider si il existe un arbre stable est NP-complet même si* $p = 2$.

Dans (Dasgupta *et al.*, 2006), il a été prouvé que

Théorème 2 *Pour* $p = 2$*, s'il existe un arbre stable dans* G *avec une métrique* w*, un algorithme faiblement auto-stabilisant le construit.*

A notre connaissance, aucun résultat sur des propriétés d'existence d'arbre stable sur les instances n'a été trouvé lorsque le nombre de couleurs (coalisions) est faible. Par contre, dans (Griffin *et al.*, 2002), lors de la première étude d'instabilité des routes, le problème est presque semblable au notre en supposant que personne n'est dans notre coalision. Leur exemple le plus petit ayant aucun arbre stable est composé de trois sommets et donc en analogie de notre problème à trois couleurs. C'est pour cette raison nous étudions particulièrement le cas où le nombre de coalitions est égale à 2.

3. Condition d'existence d'un arbre stable

Dans (Cohen *et al.*, 2008), il a été prouvé que déterminer si il existait un arbre stable est NP-complet. C'est donc pour cette raison que dans cette section, nous allons concentrer sur des conditions suffisantes de l'existence d'arbre stable.

Tout d'abord, nous allons présenter des résultats préliminaires sur des instances simples du problème.

3.1. *Résultats préliminaires*

Lemme 1 *Soit* $G = (V, E)$ *un graphe et* u *un sommet de* V*. Soit* $\mathcal{P}$ *une partition de* V *ayant deux éléments* V_1 *et* V_2*. Soit* w *une fonction de* $E \rightarrow \mathbb{N}^2$.

Soit v un sommet appartenant à V_i avec $i \in \{1,2\}$ tel qu'il existe un plus court chemin entre v et r selon la métrique w_i composé uniquement de sommets de V_i (à l'exception de la destination r), alors tous les arbres stables dans G contient un plus court chemin entre v et r dans la métrique w_i

Preuve : Soit $pcc = \{v_\ell, \ldots, v_1, u\}$ un plus court chemin entre v et u dans la métrique w_i composé uniquement de sommets de V_i ($\forall j, v_j \in V_i$) avec $v_\ell = v$. Nous allons prouver ce lemme par récurrence sur le paramètre ℓ.

Si $\ell = 1$ alors $w_i(pcc) = w_i(v_1, r)$. Soit T un arbre stable tel que $(v_1, r) \notin T$. Comme v_1 est stable dans T, on a

$$\forall z \in \Gamma_G(v_1), \sum_{e \in T_{v_1 \to r}} w_i(e) \leq w_i((v_1, z)) + \sum_{e \in T_{z \to r}} w_i(e)$$

Comme $r \in \Gamma_G(v_1)$, on a $\sum_{e \in T_{v_1 \to r}} w_i(e) \leq w_i(u, v_1) \leq w_i(pcc)$. Donc T relie r à v_1 par un plus court chemin via la métrique w_i.

Supposons que le lemme soit vrai pour tout $i < \ell$.

Soit T un arbre stable. Par hypothèse de récurrence, T relie r à $v_{\ell-1}$ par un plus court chemin via la métrique w_i. Comme v_ℓ est stable dans T, on a

$$\forall z \in \Gamma_G(v_\ell), \sum_{e \in T_{v_\ell \to r}} w_i(e) \leq w_i((v_\ell, z)) + \sum_{e \in T_{z \to r}} w_i(e)$$

Comme $v_{\ell-1} \in \Gamma_G(v_\ell)$, on a $\sum_{e \in T_{v_\ell \to r}} w_i(e) \leq w_i((v_{\ell-1}, z)) + \sum_{e \in T_{z \to r}} w_i(e) \leq w_i(pcc)$. Donc T relie r à v_ℓ par un plus court chemin via la métrique w_i.

Théorème 3 *Soit G un graphe et soit w une fonction de pondération. Soit une partition de V de deux éléments V_1 et V_2 telle que V_1 contient un unique sommet x. Il existe toujours un arbre couvrant stable enraciné en r dans G.*

Preuve : Si $x = r$, alors on se trouve dans le problème classique de la construction de l'arbre couvrant de plus court chemin utilisant la métrique w_2. Maintenant, nous supposerons que $x \neq r$. Ce théorème est prouvé par induction sur le degré d de x.

Cas où $d = 1$: Tout arbre de plus court chemin utilisant la métrique w_2 est un arbre stable car x a une unique arête incidente.

Supposons que le théorème soit vrai pour les graphes dont le degré de x est inférieur strict à d.

Cas d'induction : Soit $z_1, \ldots, z_d$ les voisins de x. Soit pcc_i un plus court chemin entre x et u passant par z_i selon la métrique w_2. Par simplicité, les voisins sont triés de façon croissante en fonction de $w_2(pcc_i)$.

Considérons le graphe construit G_1 de la façon suivante

– $V(G_1) = V(G)$ et $E(G_1) = E(G)\backslash\{(x, z_1)\}$
– $w'(e) = w(e)$ pour $\forall e \in E(G_1)$

Par hypothèse d'induction, il existe un arbre stable T^1 dans G_1 selon la métrique w'. Supposons que T^1 n'est pas stable dans G selon la métrique w. Par définition de w' et G_1, seul le sommet x ne peut pas être stable : donc

$$w_1((x, z_1)) + \sum_{e \in T^1_{z_1 \to r}} w_1(e) \leq \sum_{e \in T^1_{x \to r}} w_1(e) \quad [1]$$

Considérons T qui est une copie de T^1 en remplaçant x par z_1. Maintenant x est stable dans T. Nous allons démontrer que T est stable dans G selon la métrique w. Soit $Y = V \cap \{y : x \in T^1_{y \to r}\}$

Tout d'abord, tous les sommets $V\backslash Y$ sont stables. Maintenant concentrons nous sur les sommets $y \in V_2$ de Y. Soit t le voisin de y tel qu'il appartient au chemin $T^1_{y \to r}$. Nous avons

– y est stable dans G_1 pour la métrique w' (car $x \neq y$), on a

$$\forall s \in \Gamma(w)\backslash\{t\}, \sum_{e \in T^1_{y \to r}} w_2(e) \leq \sum_{e \in T^1_{s \to r}} w_2(e) + w_2((s, y)) \quad [2]$$

– Par définition, on a

$$\sum_{e \in T_{y \to r}} w_2(e) = \sum_{e \in T^1_{y \to x}} w_2(e) + w_2((x, z_1)) + \sum_{e \in T^1_{z_1 \to r}} w_2(e) \quad [3]$$

Le chemin $T^1_{z_1 \to r}$ reliant r à z_1 appartenant à T^1 est un plus court chemin pour la métrique w_2 car tous les sommets de ce chemin sont stables et ils appartiennent au même ensemble V_2 (voir le lemme 1).

$$\sum_{e \in T^1_{z_1 \to r}} w_2(e) + w_2((x, z_1)) = w_2(pcc_1) \quad [4]$$

En combinant les équations 3 et 4, on obtient

$$\sum_{e \in T_{w \to r}} w_2(e) = \sum_{e \in T^1_{w \to x}} w_2(e) + w_2(pcc_1) \quad [5]$$

Par définition $\forall z_i \in \Gamma_G(x)$, on a $w_2(pcc_i) \geq w_2(pcc_1)$ En combinant la remarque précédente et l' équation 5, on a

$$\sum_{e \in T_{y \to r}} w_2(e) = \sum_{e \in T^1_{y \to x}} w_2(e) + w_2(pcc_1) \leq \sum_{e \in T^1_{y \to x}} w_2(e) + w_2(pcc_i) \quad [6]$$

$$\sum_{e \in T_{y \to r}} w_2(e) \leq \sum_{e \in T^1_{y \to x}} w_2(e) + \sum_{e \in T^1_{x \to r}} w_2(e) \quad [7]$$

$$\sum_{e \in T_{y \to r}} w_2(e) \leq \sum_{e \in T^1_{y \to r}} w_2(e) \quad [8]$$

En combinant les équations 2 et 8, on peut déduire que y est stable dans l'arbre T pour la métrique w. Donc T est stable dans G pour la métrique w. Ceci conclut la preuve.

Maintenant, nous allons définir un type de graphes dans lequel il n'existe pas toujours des arbres stables.

3.2. *Graphes de type* dispute wheel *("roue de la discorde")*

Définition 2 *Un graphe G est un graphe de type* roue *ayant k rayons si et seulement si il existe un sous-ensemble $U(G) = \{u_0, u_1 \ldots, u_{k-1}\}$ de sommets de V tel que pour tout sommet u_j de $U(G)$, il existe*

– un unique chemin Q_j reliant u_j à r ne traversant aucun autre sommet de U.

– un unique chemin R_j reliant u_j à u_{j+1} ne traversant aucun autre sommet de $U(G)$.

Les indices doivent être interprétés comme modulo k. La figure 4 correspond à une représentation graphique de graphe de type roue.

Définition 3 *Une instance du problème du plus court chemin concurrent est dit une* dispute wheel *de taille k si elle se définit de la façon suivante*

– G est un graphe de type roue *ayant k rayons*

– les sommets sont partitionnés en deux sous-ensembles V_1 et V_2

– la métrique w satisfait les propriétés suivantes telles que pour tout $j, 0 \leq j \leq k-1$,

1) u_j et u_{j+1} ne sont pas de même couleur.

2) si $j > 0$, en supposant i la couleur u_j, alors
$w_i(R_j) + w_i(R_{j+1}) + w_i(Q_{j+2}) > w_i(Q_j) > w_i(R_j) + w_i(Q_{j+1})$

3) si $j = 0$*, alors*

$$w_2(R_0) + w_2(Q_1) > w_2(Q_0) > w_2(R_0) + w_2(R_1 \cdot Q_2)$$

en supposant que u_0 *est de couleur* 2*.*

On peut remarquer que l'instance réprésentée dans la figure 2 est une instance de type *dispute wheel* de taille 4. Il suffit de constater que les sommets v_j, $j \in \{1, 2, 3\}$ jouent le rôle de u_j avec $Q_j = \{v_j, r\}$ et $R_j = \{v_j, v_{j+1}\}$. Le sommet v_4 a le rôle de u_0 car $w_2(R_0) + w_2(Q_1) > w_2(Q_0) > w_2(R_0) + w_i(R_1 \cdot Q_2)$ en posant $R_0 = \{v_4, v_1\}$, $Q_0 = \{v_4, r\}$. Pour chaque sommet v_j, $j \in \{1, 2, 3\}$ de couleur ℓ la propriété $w_\ell(R_j) + w_\ell(R_{j+1}) + w_\ell(Q_{j+2}) > w_\ell(Q_j) > w_\ell(R_j) + w_i(Q_{\ell+1})$ est satisfaite.

De plus, comme nous l'avons précisé dans la section 2, aucun arbre couvrant n'est stable. Pour le prouver, il suffit de faire un raisonnement par l'absurde. S'il existe un arbre couvrant T, T contient au moins une arête incidente en r. Supposons qu'il contient l'arête (v_2, r). Cela implique qu'il contient aussi l'arête (v_1, v_2) puisque le plus court chemin entre v_1 et r dans la métrique w_1 est $\{(v_1, v_2), (v_2, r)\}$. En utilisant le même argument T doit aussi contenir l'arête (v_4, v_1). Afin que v_3 soit stable, T doit contenir l'arête (v_3, r) qui entraine l'instabilité du sommet v_2. Donc T ne contient pas l'arête (v_2, r). On peut faire le même raisonnement pour les arêtes (v_i, r), avec $i \in \{1, 3, 4\}$.

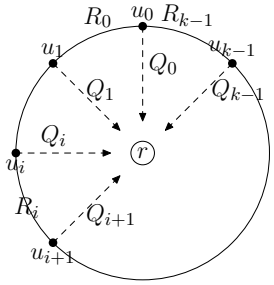

Figure 4. *un graphe de type* roue

Maintenant, à partir des instances de type *dispute wheel*, nous allons définir une condition suffisante pour déterminer si un graphe G contient un arbre stable.

Théorème 4 *Soit* G *un graphe ne contenant aucune* dispute wheel *et soit* w *une fonction de pondération. Soit une partition de deux éléments* V_1 *et* V_2 *telle que* V_1 *contient deux sommets notés* $x_1,\ x_2$*. Il existe toujours un arbre couvrant stable vers* r *dans* G*.*

Preuve : Sans perte de généralité, nous supposerons que $r \in V_2$. Considérons le sommet x_1 ayant d voisins dans G que l'on notera $z_1^1, \ldots, z_d^1$. On raisonnera par récurrence sur d pour prouver qu'il existe un arbre stable dans G pour la métrique w.

Cas où $d = 1$: il suffit de calculer l'arbre stable T dans le graphe G privé de x_1 avec la métrique w (il en existe un par le théorème 3). Ensuite il suffit d'ajouter l'arête incidente à x_1 dans l'arbre T. Il est facile de voir que l'arbre ainsi calculé est stable dans G avec la métrique w.

Supposons que le théorème soit vrai pour les graphes dont le degré de x_1 est inférieur strictement à d.

Cas d'induction : Nous allons prouver par contradiction que la propriété est vraie pour le cas d'induction en supposant qu'il n'existe pas d'arbre stable dans G selon la métrique w. Nous allons construire une suite d'arbres jusqu'à obtenir un arbre stable dans G selon la métrique w (c'est-à-dire jusqu'à obtenir une contradiction). Pour cela nous introduisons les notations suivantes :

– $br(z,T) = min_{x\in\Gamma_G(z)}(w_i(x,z) + w_i(T_{x\to r}))$.
– $\mathcal{BR}(z,T) = argmin_{x\in\Gamma_G(z)}(w_i(x,z) + w_i(T_{x\to r}))$.

avec z un sommet de V de couleur i et T un arbre couvrant de G.

Soient $z_1^1, \ldots, z_d^1$ les voisins de x_1. Considérons le graphe construit G_1 de la façon suivante :

– $V(G_1) = V(G)$ et $E(G_1) = E(G)\backslash\{(x_1, z_1^1)\}$
– $w'(e) = w(e)$ pour $\forall e \in E(G_1)$

Comme G ne contient aucun graphe de type *roue*, G_1 n'en possède pas non plus. Par hypothèse d'induction, il existe un arbre stable T^1 dans G_1 selon la métrique w'. Comme $T^{(1)}$ est un arbre couvrant dans G, $T^{(1)}$ n'est pas donc stable dans G selon la métrique w (par hypothèse de contradiction). Seul le sommet x_1 ne peut pas être stable. De plus, l'arbre $T^{(1)}$ a les propriétés suivantes :

(1) seul le sommet x_1 est non-stable dans $T^{(1)}$ selon la métrique w.

(2) $\forall z \in \Gamma_G(x_1)$, $w_2(T^{(1)}_{x_1\to r}) \leq w_2(T^{(1)}_{z\to u}) + w_2(z, x_1)$ (stabilité de z)

Ces propriétés résultent de la façon que $T^{(1)}$ a été construit.

A partir de $T^{(1)}$, nous allons contruire un arbre couvrant tel que x_1 sera stable. Considérons l'arbre $T^{'}$ qui est une copie de $T^{(1)}$ à l'exception que le père de x_1 dans $T^{'}$ est z_1^1 (car $z_1^1 \in \mathcal{BR}(x_1, T^{(1)})$). Dans l'arbre $T^{'}$, le sommet x_1 est stable.

$$w_1(T^{(1)}_{z\to r}) + w_1(z, x_1) < w_1(T^{(1)}_{x_1\to r}) \qquad [9]$$

Comme x_1 et x_2 sont de même couleur, le sommet x_2 l'est aussi est stable. Par hypothèse, comme aucun arbre couvrant n'est stable, cela implique que des sommets de V_2 ne sont pas stables : seul les sommets voisins de x_1 peuvent être non-stables dans $T^{'}$. Par l'équation [9], les sommets y de V_2 non-stables dans $T^{'}$ sont tels que $x_1 \in T^{(1)}_{y\to r}$ et par conséquent $x_1 \in T^{'}_{y\to r}$.

Nous allons transformer l'arbre $T^{'}$ en un arbre que l'on notera $T^{(2)}$ en rendant les sommets de V_2 non-stables sans changer l'arc sortant de x_2. A partir de l'ensemble $\mathcal{U}$ des voisins non-stables de x_1 dans $T^{'}$, nous considérons un sommet v tel que $br(v, T^{'}) \leq br(y, T^{'})$ pour tout $y \in \mathcal{U}$. Nous transformerons l'arbre $T^{'}$ afin qu'un sommet de $\mathcal{BR}(v, T^{'})$ soit le père de v. Ce nouvel arbre a des sommets non-stables voisins de x_1 ou de v appartenant à V_2. Nous effectuons la même opération avec un autre sommet non-stable, ainsi de suite jusqu'il n'existe plus de sommet non stable dans V_2. Cet arbre existe car par cette transformation, les sommets de V_2 ne peuvent pas redevenir non-stable dans cette suite d'arbres puisque les plus courts chemins se basent sur la métrique w_2. L'algorithme ci-dessous décrit plus formellement cette transformation :

Entrée : un graphe G avec une pondération w et une partition de sommets, un arbre $T^{'}$

Sortie : un arbre couvrant

Instructions :

1) $T^{''} \leftarrow T^{'}$;

2) $\mathcal{U} \leftarrow$ ens. de sommets non stables dans $T^{''}$ privé des sommets de V_1 ;

3) tant que $\mathcal{U}$ non-vide faire

a) Trier en ordre croissant la liste $\mathcal{U}$ en fonction de l'ordre suivant $\preceq$: $x \preceq y$ si et seulement si

$min_{z \in \Gamma_G(x)}(w_2(x,z) + w_2(T^{''}_{z \to r})) \leq min_{z \in \Gamma_G(y)}(w_2(y,z) + w_2(T^{''}_{z \to r}))$

b) Soit v le premier sommet de la liste $\mathcal{U}$

c) Changer l'arc sortant de v dans $T^{''}$ tel que v deviennent stable.

d) réactualiser l'ensemble $\mathcal{U}$

4) retourner l'arbre $T^{''}$ ainsi construit ;

Par hypothèse, l'arbre $T^{(2)}$ ainsi construit n'est pas stable : il a les propriétés suivantes :

1) seul le sommet x_2 n'est pas stable.

2) $T^{'}_{x_1 \to u} = T^{(2)}_{x_1 \to u}$ (ceci est du au fait que tous les sommets de V_2 dans $T^{(1)}_{x_1 \to u}$ sont stables.)

L'instabilité de x_2 dans $T^{(2)}$ a pu être causé par un de ces deux cas :

Cas 1 : $w_2(T^{\prime}_{x_1 \to r}) < w_2(T^{(1)}_{x_1 \to r})$. Cela implique qu'il existe y_1 un sommet de V_2 tel que $T^{(2)}_{y \to r} \neq T^{'}_{y \to r}$ et tel qu'il existe $z \in BR(x_2, T^{(2)})$ avec $y_1 \in T^{(2)}_{z \to r}$

Cas 2 : $w_2(T^{\prime}_{x_1 \to r}) > w_2(T^{(1)}_{x_1 \to r})$. Cela implique qu'il existe y_1 un sommet de V_2 tel que $T^{(2)}_{y \to r} \neq T^{'}_{y \to r}$ et tel que $y_1 \in T^{'}_{x_2 \to r}$

Traitons les deux cas.

Cas 1 : Supposons que $y_1 \in V_2$ est tel que $T^{(2)}_{y\to r} \neq T^{(')}_{y\to r}$ et tel qu'il existe $z \in BR(x_2, T^{(2)})$ avec $y_1 \in T^{(2)}_{z\to r}$. Par la propriété **(2)** de l'arbre $T^{(2)}$ et par définition de y_1, on a

$$w_2(T^{(2)}_{y_1\to x_1}) + w_2(T^{(2)}_{x_1\to r}) \leq w_2(T^{(1)}_{y_1\to r}) \quad [10]$$

$$w_1(T^{(2)}_{x_2\to z}) + w_1(T^{(2)}_{z\to r}) \leq w_1(T^{(2)}_{x_2\to r}) \quad [11]$$

A partir de $T^{(2)}$, nous allons contruire un arbre $T^{''}$ tel que x_2 sera stable. Considérons l'arbre $T^{''}$ qui est une copie de $T^{(2)}$ à l'exception que le père de x_2 dans $T^{''}$ est maintenant z. Dans l'arbre $T^{''}$, le sommet x_2 est stable.

En fait, nous sommes en train de construire un sous-graphe de G correspondant à une *dispute wheel* où y_1 (resp. x_2) est u_j (resp. u_{j+1}) avec $j \neq 0$. Maintenant, il reste à trouver quel est le sommet qui joue le rôle de u_0.

Par définition $T^{''}$ n'est pas stable mais pour tout sommet de V_1, il l'est. L'arbre $T^{''}$ peut être transformé en un arbre $T^{(3)}$ afin de rendre tout sommet de V_2 stable en utilisant les mêmes arguments pour transformer l'arbre $T^{'}$ en $T^{(2)}$. C'est-à-dire pour tout sommet y de V_2 non stable dans $T^{''}$, on a

$$w_2(T^{(3)}_{y\to r}) < w_2(T^{''}_{y\to r}) \quad [12]$$

En utilisant les mêmes arguments, seul le sommet x_1 est non-stable dans $T^{(3)}$. Nous allons considérer deux cas :

Cas 1.1 : $w_2(T^{''}_{x_2\to r}) > w_2(T^{(2)}_{x_2\to r})$

Cas 1.2 : $w_2(T^{''}_{x_2\to r}) < w_2(T^{(2)}_{x_2\to r})$

Remarque : il est impossible que $w_2(T^{''}_{x_2\to r}) = w_2(T^{(2)}_{x_2\to r})$. Si tel est le cas, $T^{''}$ serait stable. Cela contredit les hypothèses. Traitons les deux cas.

Cas 1.1 : Supposons que $w_2(T^{''}_{x_2\to r}) > w_2(T^{(2)}_{x_2\to r})$. Par définition, les sommets non-stables dans $T^{''}$ sont des sommets y de V_2 tel que $x_2 \in T^{''}_{y\to r}$ et par conséquent $x_2 \in T^{(2)}_{y\to r}$. On peut déduire que pour tout sommet instable y dans $T^{(2)}$

$$w_2(T^{(2)}_{y\to x_2}) + w_2(T^{(2)}_{x_2\to y_1}) + w_2(T^{(2)}_{y_1\to r}) < w_2(T^{''}_{y\to x_2}) + w_2(T^{''}_{x_2\to r}) \quad [13]$$

$$w_2(T^{(3)}_{y_1\to r}) < w_2(T^{''}_{y\to x_2}) + w_2(T^{''}_{x_2\to r}) \quad [14]$$

Comme x_1 n'est pas stable dans $T^{(3)}$, il existe $y_2 \in (T^{''}_{x_1\to r})$ tel que y_2 est non stable dans $T^{''}$. Cela signifie qu'il existe $z \in BR(x_1, T^{(3)})$ tel que

$w_1(T^{(3)}_{x_1\to z}) + w_1(T^{(3)}_{z\to r}) < w_1(T^{(3)}_{x_1\to y_1}) + w_1(T^{(3)}_{y_1\to r})$. Par définition de la stabilité de $T^{(1)}$ dans G_1, on a $w_1(T^{(1)}_{x_1\to r}) = w_1(T^{(3)}_{x_1\to z}) + w_1(T^{(3)}_{z\to r})$

Nous avons construit une *dispute wheel* de taille 4 avec y_2 qui joue le rôle de u_0 . Ceci contredit notre hypothèse de départ. Donc, ce cas est impossible.

Cas 1.2 : Supposons que $w_2(T^{(2)}_{x_2\to r}) > w_2(T^{''}_{x_2\to r})$. Par définition, les sommets non-stables y dans $T^{''}$ vont changer d'arc sortant afin que $x_2 \in T^{(2)}_{y\to u}$. Comme x_1 n'est pas stable dans $T^{(3)}$, il n'existe pas $y_2 \in (T^{''}_{x_1\to r})$ tel que y_2 est non stable dans $T^{''}$. Si tel était le cas, x_1 serait stable dans $T^{(3)}$.

Soit $z \in BR(x_1, T^{(3)})$. On a $w_1(T^{(1)}_{x_1\to r}) = w_1(T^{(3)}_{x_1\to z}) + w_1(T^{(3)}_{z\to u})$ par définition. Soit y_2 instable dans $T^{(2)}$ tel que $y_2 \in T^{(3)}_{z\to r}$. En combinant toutes les inégalités,

$$w_1(T^{(3)}_{x_1\to z}) + w_1(T^{(3)}_{z\to y_2}) + w_1(T^{(3)}_{y_2\to x_2}) + w_1(T^{(3)}_{x_2\to r}) \leq w_1(T^{(2)}_{x_1\to r}) \quad [15]$$

$$w_2(T^{(2)}_{x_1\to r}) < w_1(T^{(2)}_{x_1\to y_2}) + w_1(T^{(2)}_{y_2\to r}) \quad [16]$$

Nous avons construit une *dispute wheel* de taille 4 avec x_1 qui joue le rôle de u_0 . Ceci contredit notre hypothèse de départ. Donc, ce cas est impossible.

Cas 2 : Il faut raisonner de la même façon que le cas 1. Ce cas est aussi impossible.

Donc, le cas d'induction est prouvé. On peut en déduire le théorème.

Théorème 5 *Un graphe ne contenant pas une* dispute wheel *implique l'existence d'un arbre stable.*

Preuve : La preuve du théorème correspond à une double récurrence. La première est sur le nombre de sommets de V_1 dont le cas de base correspond au théorème 4. Pour le cas d'induction, il suffit de reprendre la preuve du théorème 4 en considérant que la taille de la *dispute wheel* ne peut pas excéder $2|V_1|$.

4. Description de l'algorithme résolvant le problème du plus court chemin concurrent

Le but de cette section est de décrire l'algorithme construisant un arbre stable de plus courts chemins dans le cas précis d'une compétition entre différentes coalitions de nœuds. Cet algorithme est auto-stabilisant mais il ne fonctionne correctement que dans le cas de l'existence d'un arbre stable dans le réseau. Celui ci est basé sur l'utilisation de deux résultats (Delaët *et al.*, 2002) et (Griffin *et al.*, 2002) qui, combinés, forment notre algorithme.

Le principe de l'algorithme est de découvrir la topologie du réseau et les pondérations des arêtes via l'algorithme de recensement de (Delaët *et al.*, 2002) dont on

surcharge les messages. Ensuite chaque nœud ayant pu calculer un ordre préférentiel des chemins vers la destination grâce à sa connaissance parfaite de la topologie exécute l'algorithme de (Griffin *et al.*, 2002) de calcul d'arbre stable et indique au vu du résultat de ce calcul l'arc qu'il sélectionne en local pour l'arbre stable. Evidement le calcul d'arbre stable ne convergera qu'en cas d'existence d'un arbre stable d'où le travail des sections précédentes.

Le travail principal de cette partie est donc de formaliser la surcharge des messages de l'algorithme de recensement afin de récolter de manière auto-stabilisante une information complète sur la topologie du graphe et les pondérations colorées de ses arcs nécessaires à l'algorithme de (Griffin *et al.*, 2002).

Les messages de notre algorithme se composent de trois parties, une partie recensement, une partie découverte de la topologie et une partie construction d'arbre.

Par hypothèse, lorsqu'un nœud reçoit un message, celui çi est associé au numéro du canal entrant sur le quel circulait ce message et au coût pour chaque coalition du passage de ce message sur ce canal.

La présente section se compose de trois sous sections, dans la première nous rappelons l'algorithme de recensement auto-stabilisant de (Delaët *et al.*, 2002). Dans la deuxième, nous présentons comment la surcharge des messages permet d'acquérir, de manière auto-stabilisante, une connaissance complète de la topologie du réseau et des pondérations sur chaque arc. La troisième est un rappel de l'algorithme de construction d'arbre stable à partir d'une liste de routes classées par priorité.

4.1. *Rappel de l'algorithme de recensement*

Cette sous section est un rappel de l'algorithme de recensement de (Delaët *et al.*, 2002). Sa lecture peut être omise par tout lecteur connaissant préalablement le résultat.

Le recensement consiste en une découverte des nœuds présents dans un réseau et de leur position relative. Ceci répond pour chaque nœud à la question : "qui peut m'envoyer des messages, à quelle distance et via quel canal ?"

Le principe de l'algorithme de faire circuler des messages dans le réseau qui récoltent des informations sur celui-ci. Les nœuds font systématiquement confiance à la dernière information reçue depuis un canal de communication. Ils la combinent avec l'information en leur possession sur le réseau pour ainsi recenser de manière auto-stabilisante l'ensemble des identifiants des nœuds présents dans le réseau, leur distance au nœud courant et les nœuds intermédiaires entre le nœud cité et le nœud courant. L'algorithme étant auto-stabilisant, il assure qu'après le temps de stabilisation toutes les mémoires des nœuds contiennent une information correcte et que ne circulent que des messages canoniques qui ne modifieront pas ces mémoires.

La figure 5 présente un exemple de configuration correcte des mémoires pour l'algorithme de recensement. Chaque nœud y est représenté par une tête coiffée d'un cha-

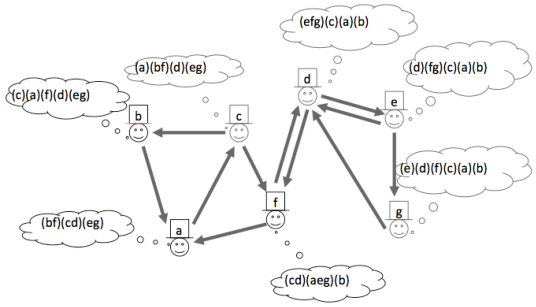

Figure 5. *configuration avec uniquement des mémoires correctes pour l'algorithme de recensement.*

peau portant son identifiant, le contenu des mémoires y est représenté par les bulles de pensée. Plus précisément, pour le réseau de la figure 5 la mémoire correcte du nœud a est $(\text{b/A f/B})(\text{c/AB d/B})(\text{e/B g/B})$. En effet, a possède deux canaux entrant qu'il peut par exemple numéroter A et B via la canal numéroté A, b est à distance 1, c est à distance 2 et il n'y a pas de nœud à distance 3 ; via le canal B, f est à distance 1, c et d sont à distance 2 et e et g sont à distance 3. Autrement dit, ceci signifie que le nœud a peut recevoir des messages de deux nœuds voisins à distance 1 les nœuds b et f , qu'il possède deux nœuds à distance 2 les nœud c et d et deux nœud à distance 3 les nœud e et g. Par ailleurs la la forme canonique d'un message circulant entre f et a est $(f)(cd)(aeg)(b)$. Le propos ici n'est pas de prouver de nouveau cet algorithme mais de rappeler par l'exemple son fonctionnement et de pointer le fait qu'après stabilisation il continue à transmettre continuellement des messages et à réécrire éternellement le même contenu dans les mémoires.

Remarquons que cet algorithme ne permet pas directement d'avoir une image complète de la topologie du réseau dans chacune des mémoires des nœuds du graphe mais seulement une information de recensement sur les identifiants de tous les nœuds du graphe. En effet, considérons le même graphe auquel on ajoute un arc de g vers e, la mémoire canonique des nœuds a, b, c, d, f et g serait la même malgré une topologie différente du réseau. Toutefois, par définition des mémoires correctes, chaque nœud a une connaissance totale de ses voisins entrants (topologie à distance 1).

4.2. *surcharge des messages*

Dans cette sous section nous présentons notre adaptation de l'algorithme de recensement de (Delaët *et al.*, 2002) qui servira de base à l'algorithme de de construction d'arbre stable de (Griffin *et al.*, 2002).

Il est nécessaire de connaître en chaque nœud une image complète de la topologie du réseau ainsi que du poids pour chacune des couleurs (coalitions) affectées à chacun

des arcs du graphe. Pour cela nous allons surcharger les messages transmis par l'algorithme présenté dans la section précédente de manière à récolter les poids affectés à chacun des arcs à distance 1 pour chacune des coalitions. Ainsi un message n'est plus seulement, comme dans l'algorithme de recensement, une simple liste de nœuds, mais, il possède aussi une seconde partie qui indique, pour chaque nœud, sa topologie à distance 1 et les pondérations des arcs selon les couleurs (coalitions). Sur réception d'un message depuis un canal numéroté A l'algorithme de recensement s'applique normalement sur la partie recensement du message. De plus pour la partie topologie du message, l'information sur le nouveau nœud à distance 1 via le canal numéroté A est mise à jour avec les nouvelles informations sur les poids par coalitions via le canal numéroté A. Ainsi dès que l'algorithme de recensement est en phase stabilisée et qu'un message a parcouru tout le graphe, il contient une image complète de la topologie du graphe.

4.3. *rappel de l'algorithme de construction d'arbre stable*

Cette sous section est un rappel de l'algorithme de construction d'arbre stable de (Griffin *et al.*, 2002). Sa lecture peut être omise par tout lecteur connaissant préalablement le résultat. L'algorithme de construction d'arbre stable de (Griffin *et al.*, 2002) est basé sur une connaissance à priori d'une liste ordonnée par préférence des chemins possibles en chaque nœud vers la destination.

Chaque nœud maintient au niveau de sa mémoire un chemin courant vers la destination. Sur réception d'un nouveau chemin proposé par un de ses voisins, le nœud actualise son chemin en mémoire s'il préfère le chemin proposé à son chemin courant suivant sa liste de préférence.

Dans notre cas, la notion d'ordre préférentielle sur les chemins sera calculée à la volée en fonction des données topologiques et des pondérations présentes au sein du nœud. L'algorithme de (Griffin *et al.*, 2002) s'applique donc directement sauf que le calcul du meilleur chemin chemin est local. En phase de stabilisation des algorithmes de recensement et du calcul de la topologie, ce calcul local du meilleur chemin n'a pas de pertinence puisque la topologie et les poids considérés ne sont pas nécessairement les bons. Par contre dès que les deux algorithmes sous-jacents sont en phase stabilisée, les poids des chemins courants et proposés sont calculés et comparés, pour que le meilleur soit conservé pour former l'arbre stable. Notons que cet algorithme ne convergence sur des instances contenant aucune *dispute wheel* et respectant des conditions de l'algorithme de (Griffin *et al.*, 2002) concernant la stabilisation de celui ci (à savoir en l'absence de dispute wheel sur les ordres de préférences des chemins - voir (Griffin *et al.*, 2002) -).

5. Formalisation de l'algorithme

5.1. *rappel de la notion d'auto-stabilisation*

Une configuration est le produit cartésien à un instant donné de toutes les variables de tous les nœuds du réseau. Chaque nœud possède ses propres variables et communique avec les autres processeurs par passage de messages.

Un système de transitions est un couple $S = (C, \rightarrow)$, où C est un ensemble de configurations et $\rightarrow$ une relation binaire sur C. Une exécution de S est une suite maximale de configurations (finie ou infinie) $E = (\gamma_0, \gamma_1, \ldots, \gamma_i, \ldots)$ telle que pour tout $i \geq 0, \gamma_i \rightarrow \gamma_{i+1}$. Une spécification est un prédicat sur l'ensemble des exécutions. Un système de transition $S = (C, \rightarrow)$ est auto-stabilisant pour la spécification P s'il existe un ensemble $L \subseteq C$ de configurations légitimes vérifiant les propriétés suivantes :

1) *Correction :* toute exécution débutant par une configuration dans L satisfait P ;

2) *Convergence :* toute exécution atteint une configuration dans L.

5.2. *Hypothèse liées aux communications dans le cadre de ce problème*

La communication se fait par échange de message point à point. Un nœud origine envoie un message sur un de ses ou sur tous ses arcs sortants. En fonctionnement normal (absence de pannes) les messages ne sont pas corrompus. Toutefois un nœud ne connaît pas à priori l'identifiant du nœud à l'origine du message qu'il reçoit, il connaît seulement une étiquette locale de ses arcs entrants. Sur réception d'un message, un nœud apprend le message (contenu de la variable transmise par l'émetteur), l'étiquette locale de l'arc par lequel est entré ce message (information locale) et la pondération de cet arc entrant pour chacune des couleurs représentant les coalitions (information propre au canal apprise en parcourant le canal).

5.3. *Description de l'algorithme*

L'algorithme étant formé de trois parties (recensement découverte de la topologie, construction d'arbre), ses variables sont partagées en trois groupes : les variables de recensement, les variables de topologie et les variables de construction d'arbre. Ces trois groupes seront aussi utilisés pour le contenu des messages transmis sur le réseau. De même les messages contiennent trois parties, une partie recensement, une partie découverte de la topologie et une partie construction d'arbre.

Pour la compréhension de l'algorithme, nous considérons dans les exemples que les identifiants des processeurs sont des mots formés de lettre de l'alphabet en minuscules, les noms des arcs sont des mots formés de l'alphabet en majuscules, les noms

des coalitions des couleurs, les poids des entiers et les chemins des suites d'identifiants.

La variable de recensement correspondent à une suite de liste d'identifiants, chacun étant associé un ensemble d'arcs entrant du nœud. La première liste représente les voisins à distance 1, la ième liste celle des voisins à distance i du nœud, les ensembles d'arcs sont l'ensemble des noms des arcs par les quels on obtient l'information sur ce nœud. La mise à jour de cette variable est faite sur réception d'un message par le nœud courant via un canal numéroté en local conformément à l'algorithme auto-stabilisant de recensement de (Delaët *et al.*, 2002).

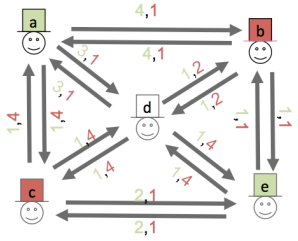

Figure 6. *cas de deux coalitions(les verts et les rouges)*

– Exemple de variable de recensement au sein du nœud d du réseau de la figure 5 avec des contenus de mémoires différents : (b/A c/C)(d/A e/A f/B), cette valeur de la variable est trivialement incorrecte puisqu'il est impossible de connaître le nœud f à distance 2 via le canal B, c'est pourtant une valeur initiale possible de la partie recensement du nœud d, puisqu'elle est correcte du point de vue de la syntaxe.

– Exemple de variable de recensement au sein du nœud a du réseau de la figure 5 avec les contenus des mémoires de la figure 5 : (b/A f/B)(c/AB d/B)(e/B g/B), cette valeur de la variable est la valeur correcte pour le recensement dans le cas du réseau de la figure 5 en phase stabilisée elle sera éternellement réécrite à l'identique.

La variable de découverte de la topologie est une suite d'identifiants d'arcs auquel sont associés pour chacun un poids par couleur. La mise à jour de cette variable est faite sur réception d'un message par le nœud courant via un canal numéroté en local. La partie recensement est d'abord mise à jour conformément à l'algorithme de (Delaët *et al.*, 2002) ensuite la partie découverte de la topologie du message est recopiée à l'exception des informations sur l'arc entrant du message qui sont elles mises à jour avec le nouvelles informations de pondération des couleurs récoltées à la réception du message. Remarquons que même si on ne connaît que l'identifiant local de l'arc entrant (par exemple A), son identifiant global (par exemple $b \rightarrow a$) est déduit d'une part par l'identifiant du nœud courant (par exemple a) et d'autre part par la première liste de l'algorithme de recensement qui associe l'identifiant local de l'arc entrant avec l'identifiant global du nœud à l'origine de l'arc (par exemple b/A).

– Exemple de variable de recensement au sein d'un nœud :
$a \rightarrow b$ / $vert$ 4 / $rouge$ 1 - $a \rightarrow c$ / $vert$ 1 / $rouge$ 4 - $a \rightarrow d$ / $vert$ 3 / $rouge$ 1

$b \rightarrow a$ / $vert$ 4 / $rouge$ 1 - $b \rightarrow d$ / $vert$ 1 / $rouge$ 2 - $b \rightarrow e$ / $vert$ 1 / $rouge$ 1
$c \rightarrow a$ / $vert$ 4 / $rouge$ 1 - $c \rightarrow d$ / $vert$ 1 / $rouge$ 4 - $c \rightarrow e$ / $vert$ 2 / $rouge$ 1
$d \rightarrow a$ / $vert$ 3 / $rouge$ 1 - $d \rightarrow b$ / $vert$ 1 / $rouge$ 2 - $d \rightarrow c$ / $vert$ 1 / $rouge$ 4 - $d \rightarrow e$ / $vert$ 1 / $rouge$ 1
$e \rightarrow b$ / $vert$ 1 / $rouge$ 1 - $e \rightarrow c$ / $vert$ 2 / $rouge$ 1 - $e \rightarrow d$ / $vert$ 1 / $rouge$ 4
Cette valeur de variable est correcte pour le réseau de la figure 6, en phase stabilisée elle sera éternellement réécrite à l'identique. Par contre elle est possible mais totalement erronée pour le réseau de la figure 5.

La variable de construction d'arbre est le chemin courant vers la destination. Sur réception d'un message et après mise à jour de la variable de recensement et de découverte de la topologie, l'ensemble des chemins possibles vers la destination sont évaluées (ceci correspond à l'entrée de classement des chemins de l'algorithme de (Griffin *et al.*, 2002)). Ensuite le nouveau chemin est selectionné. C'est le chemin de coût minimum entre aller vers le voisin (si l'arc existe) puis utiliser son chemin vers la destination (reçu dans le message) et le chemin courant (si c'est bien un chemin possible).

Après réception et modification des variables un message est transmis à tous les voisins, celui ci contient la nouvelle valeur de la variable de recensement précédée du nom de l'identifiant du nœud émetteur, la valeur de la nouvelle variable de découverte de la topologie et le nouveau chemin courant vers la destination.

6. Conclusion

Nous avons étudié le problème du plus court chemin en présence de concurrence, en étendant la modélisation de Griffin en incluant la possibilité de présence de coalision, en caractérisant des propriétés sur la structure de l'instance permettant de garantir l'existence d'un tel arbre, en donnant un algorithme auto-stabilisant. Cet algorithme ne convergence sur des instances contenant aucune *dispute wheel* et respectant des conditions données dans (Griffin *et al.*, 2002). La taille mémoire de l'algorithme présenté pourrait certainement être optimisée, toutefois cela rendrait plus confus la compréhension de ses principes de fonctionnement. En outre le facteur d'optimisation ne serait pas nécessairement grand et la litérature montre au moins un exemple de résultat dans un domaine proche où l'optimisation de la taille des mémoires et des messages n'est pas un souci majeur (voir (Chen *et al.*, 2005)).

7. Bibliographie

Ahronovitz E., König J.-C., Saad C., « A distributed method for dynamic resolution of BGP oscillations », *20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, 2006.

Chen Y., Datta A. K., Tixeuil S., « Stabilizing Inter-domain Routing in the Internet », *Journal of High Speed Networks*, vol. 14, n° 1, p. 21-37, 2005.

Cohen J., Dasgupta A., Ghosh S., Tixeuil S., « An exercise in selfish stabilization », *ACM Transactions of Adaptive Autonomous Systems (TAAS)*, 2008.

Cormen T. H., Leiserson C. E., Rivest R. L., Stein C., *Introduction to Algorithms, Second Edition*, The MIT Press, September, 2001.

Dasgupta A., Ghosh S., Tixeuil S., « Selfish Stabilization », *8th International Symposium Stabilization, Safety, and Security of Distributed Systems (SSS)*, vol. 4280 of *Lecture Notes in Computer Science*, Springer, p. 231-243, 2006.

Delaët S., Tixeuil S., « Tolerating Transient and Intermittent Failures », *Journal of Parallel and Distributed Computing*, vol. 62, n° 5, p. 961-981, May, 2002.

Dijkstra E. W., *A short introduction to the art of programming*, 1971.

Griffin T., Shepherd F. B., Wilfong G. T., « The stable paths problem and interdomain routing. », *IEEE/ACM Trans. Netw.*, vol. 10, n° 2, p. 232-243, 2002.

Griffin T., Wilfong G. T., « A Safe Path Vector Protocol », *INFOCOM*, p. 490-499, 2000.

Huitema C., *Routing in the Internet*, Prentice Hall, 2001.

Ibrahim S. Y., Matta I., « A Randomized Solution to BGP Divergence », *in Proceedings of the 2nd IASTED International Conference on Communication and Computer Networks (CCN'04*, 2004.

Tel G., *Introduction to Distributed Algorithms*, Cambridge university press, 2000.

ANNEXE POUR LE SERVICE FABRICATION

A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :
   L'objet. Volume 8 – n°2/2005

2. AUTEURS :
   Johanne Cohen [*] *— Sylvie Delaët* [**]

3. TITRE DE L'ARTICLE :
   Stabilité des routes dans les réseaux inter-domaines.

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
   Stabilité des routes

5. DATE DE CETTE VERSION :
   2 juillet 2009

6. COORDONNÉES DES AUTEURS :

– adresse postale :

[*] Laboratoire PRiSM (CNRS - Université de Versailles, Saint Quentin)
45, avenue des États-Unis – 78035 Versailles Cedex

Johanne.Cohen@prism.uvsq.fr

[**] Laboratoire LRI (Université Paris Sud 11, Orsay)
Bâtiment 490, université Paris Sud 91405 Orsay Cedex

Sylvie.Delaet@lri.fr

– téléphone : 01 39 25 30 52

– télécopie : 01 69 15 42 39

– e-mail : Johanne.Cohen@prism.uvsq.fr/Sylvie.Delaet@lri.fr

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :
   LaTeX, avec le fichier de style `article-hermes2.cls`,
   version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :
   Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
   `http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com