

Durant ma thèse, j'ai principalement travaillé sur des problèmes d'algorithmique distribuée. Je m'intéresse particulièrement à déterminer ce qu'on peut calculer de manière distribuée dans différents modèles. Les résultats obtenus dans ce domaine sont présentés dans la première partie de ce document.

J'ai également travaillé avec d'autres doctorants du LaBRI sur d'autres problèmes d'informatique théorique. Avec D. Gonçalves et P. Ochem, nous nous sommes intéressés à certains problèmes de théorie des graphes présentés dans la deuxième partie de ce document. Avec P. Ochem, nous nous sommes intéressés à un problème de combinatoire des mots présenté dans la troisième partie de ce document.

Dans le cadre d'un stage de maîtrise effectué lors de l'été 2002 à l'Université de l'État du Nouveau-Mexique avec H. Leung, j'ai travaillé sur un problème de théorie des semigroupes qui est présenté dans la quatrième partie de ce document.

1 Algorithmique distribuée

Un système distribué est un environnement où plusieurs processus collaborent pour réaliser un objectif commun. Dans un réseau, généralement, les différents processus ne peuvent communiquer directement qu'avec un nombre limité d'autres processus, leurs « voisins ». L'algorithmique distribuée a pour but de décrire quelles sont les tâches qui peuvent être réalisées dans de tels systèmes. Un élément du réseau est un « noeud » et ce qui permet de distinguer un noeud d'un autre peut être un identifiant (comme une adresse IP sur Internet), mais plus généralement, c'est la position de chaque noeud dans le réseau qui permet de le distinguer. Ce qui caractérise un système distribué, c'est le fait qu'il n'existe pas a priori de système de centralisation qui peut coordonner globalement les différents processus. L'algorithmique distribuée cherche ainsi à déterminer quels sont les comportements globaux qui peuvent être obtenus dans de tels systèmes où les comportements des différents processus ont des effets locaux.

Dans la littérature, il existe de nombreux modèles d'algorithmique distribuée. Ces modèles diffèrent généralement par les procédés permettant aux processus de communiquer les uns avec les autres : les processus peuvent communiquer en échangeant des messages, en accédant à des zones de mémoire partagées, etc.

Dans tous les modèles que j'ai considérés dans ma thèse, un réseau est modélisé par un graphe simple connexe dont les sommets correspondent aux processus et dont les arêtes correspondent aux liens de communication. Les réseaux considérés sont fiables : il ne se produit aucune défaillance, ni sur les noeuds, ni sur les liens de communications. Par ailleurs, les modèles considérés sont asynchrone, au sens où il n'existe pas d'horloge globale commune à tous les processus et où les pas de calculs effectués par les processus peuvent s'exécuter à des vitesses arbitraires.

Parmi les modèles d'algorithmique distribuée que j'ai étudié dans ma thèse, on distingue trois familles de modèles. La première famille est la famille des *calculs locaux* pour lesquels un algorithme distribué est présenté sous la forme d'un système de réétiquetages locaux de graphes. Ce formalisme développé au LaBRI depuis plusieurs années autour d'Y. Métivier est un prolongement des travaux de Rosenstiehl *et al.* [RFH72] et d'Angluin [Ang80]. Dans les différents modèles étudiés, un pas de calcul nécessite une synchronisation plus ou moins « forte » entre des processus voisins. Un modèle classique pouvant être représenté dans ce formalisme est le modèle de la communication synchrone, où lorsque deux processus voisins communiquent, une synchronisation¹ a lieu entre les processus le temps de la communication (c'est le principe du téléphone).

J'ai aussi considéré un modèle plus classique où les processus communiquent en échangeant des messages en mode asynchrone. Dans ce modèle, lorsqu'un processus envoie un message à l'un de ses

¹Il faut noter que bien que chaque pas de calcul nécessite une synchronisation locale, le système est globalement asynchrone.

voisins, il est assuré que le message va arriver à destination, mais il ne sait pas au bout de combien de temps le destinataire va le recevoir (c'est le principe du courrier lorsqu'aucune lettre n'est perdue).

Enfin, j'ai étudié les systèmes à agents mobiles. Dans de tels systèmes, les différents noeuds du réseau sont passifs et ce sont des agents mobiles qui sont en charge de l'exécution de l'algorithme. Les agents peuvent se déplacer le long des liens de communication et peuvent modifier l'état des places où ils se trouvent.

1.1 Problèmes considérés

Dans ma thèse, j'ai principalement étudié des problèmes où la « symétrie » initiale du réseau doit être brisée. Le problème de l'*élection* est un problème de ce type très étudié dans la littérature. Le but d'un algorithme d'élection est de distinguer un noeud du réseau. Plus formellement, un algorithme distribué est un algorithme d'élection pour un réseau donné si toute exécution de l'algorithme sur ce réseau termine et permet d'atteindre une configuration finale où il existe exactement un processus dans l'état ÉLU, alors que tous les autres processus sont dans l'état NON-ÉLU. On suppose par ailleurs que les états ÉLU et NON-ÉLU sont des états *finaux*, i.e., une fois qu'un processus est dans cet état, son état n'est plus modifié jusqu'à la fin de l'exécution de l'algorithme.

Si les noeuds possèdent des identifiants uniques, alors il existe un algorithme simple : chaque noeud diffuse son identifiant dans le réseau, et le processus disposant du plus petit identifiant est élu. Si de tels identifiants n'existent pas, le réseau est dit *anonyme* et on sait depuis les travaux d'Angluin [Ang80] que certains réseaux trop « symétriques » n'admettent pas d'algorithme d'élection.

Un autre problème qui nécessite de briser la symétrie initiale du réseau est le problème du « nommage ». Le but d'un algorithme de nommage est d'attribuer de manière distribuée un identifiant unique à chaque noeud du réseau. Comme on l'a dit précédemment, si on dispose d'un algorithme de nommage pour un réseau donné, alors on peut utiliser les identifiants attribués à chaque sommet pour obtenir un algorithme d'élection pour ce réseau. De même, dans certains modèles d'algorithmique distribuée, il est possible d'obtenir un algorithme de nommage à partir d'un algorithme d'élection, mais ce n'est pas toujours le cas.

Ces problèmes sont très importants en algorithmique distribuée puisque de nombreux algorithmes présupposent l'existence d'un noeud distingué ou d'identifiants uniques. Un noeud distingué peut par exemple être utilisé pour centraliser ou diffuser de l'information.

Il est aussi intéressant de déterminer quelles sont les connaissances initiales nécessaires à un algorithme d'élection. Autrement dit, quelle est l'information sur le réseau dont l'algorithme doit disposer pour pouvoir toujours élire un noeud du graphe. L'algorithme peut être tout à fait spécifique à un réseau : pour un réseau différent, il faudra utiliser un autre algorithme. On peut toutefois souhaiter obtenir des algorithmes permettant d'élire dans une famille de graphes la plus large possible.

Ainsi, on peut chercher un algorithme d'élection qui nécessite la connaissance de la taille du graphe, ou d'une borne sur cette taille et non pas la connaissance de la topologie du graphe. Si de tels algorithmes existent, cela signifie qu'un même algorithme peut être utilisé pour différents réseaux qui ont la même taille, ou dont la taille est bornée par une même constante. La recherche de solutions générales de ce type permet d'obtenir des algorithmes plus « fiables » : les conditions que doivent vérifier un réseau pour que l'algorithme fonctionne correctement sont moins contraignantes. Ainsi, on préférera un algorithme d'élection qui nécessite la connaissance d'une borne sur la taille du graphe à un algorithme qui nécessite de connaître la topologie du graphe, puisque dans le premier cas, on peut ajouter ou enlever un certain nombre de sommets et continuer à utiliser le même algorithme.

1.2 Calculs locaux

Les calculs locaux sont des modèles d'algorithmique distribuée où un algorithme est décrit par un système de règles de réétiquetage local de graphes. Dans ces modèles, l'état de chaque sommet est représenté par une étiquette. L'application d'une règle du système de réétiquetages décrivant l'algorithme permet de modifier les étiquettes d'un sous-graphe connexe en fonction seulement des étiquettes des sommets de ce sous-graphe. À partir d'une configuration initiale, une exécution est

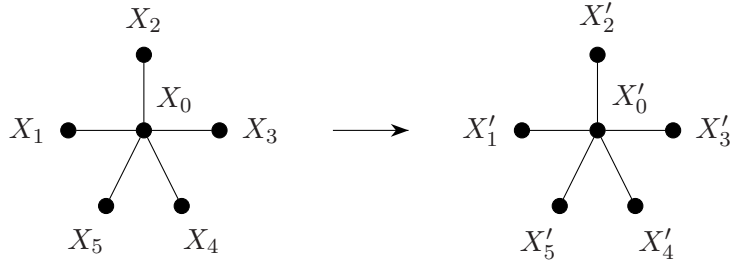


FIG. 1 – La forme d’une règle de réétiquetage dans le modèle de Mazurkiewicz.

une suite d’applications de règles. Lorsqu’aucune règle ne peut plus être appliquée, on dit que la configuration alors atteinte est *finale* et que l’exécution est terminée.

La notion d’exécution présentée ci-dessus correspond à des exécutions *séquentielles*. Cependant, il faut noter que si des règles de réétiquetages peuvent être appliquées sur des sous-graphes disjoints (qui n’ont aucun sommet en commun), alors ces règles peuvent être appliquées en parallèle. Ainsi, plusieurs pas de calculs peuvent avoir lieu simultanément dans le graphe.

1.2.1 Travaux existants

Dans [Maz97], Mazurkiewicz a considéré un modèle où un pas de calcul permet de modifier l’état d’un sommet et de tous ses voisins en fonction de sa propre étiquette et des étiquettes de ses voisins. Dans ce modèle, un algorithme peut être décrit par un ensemble de règles de réétiquetage dont le support est une étoile. Ces règles sont de la forme de la règle présentée sur la Figure 1. Ce modèle est considéré comme « puissant », puisque lorsqu’une règle est appliquée sur une étoile centrée sur un sommet v , tous les voisins de v peuvent être immédiatement informés du nouvel état de v .

Dans ce modèle, Mazurkiewicz a caractérisé les graphes admettant un algorithme d’élection ou de nommage. La bonne notion de « symétrie » permettant d’exprimer cette caractérisation est la notion de *revêtement*, qui sont des homomorphismes de graphes localement bijectifs : formellement, un graphe G est un revêtement d’un graphe H à travers un homomorphisme φ si pour tout sommet $v \in V(G)$, φ induit une bijection entre les arêtes incidentes à v dans G et celles incidentes à $\varphi(v)$ dans H . Un graphe simple G est *minimal pour les revêtements simples* s’il n’existe pas de graphe simple H non-isomorphe à G tel que G est un revêtement de H . Mazurkiewicz a montré qu’il existe un algorithme d’élection pour un graphe G donné si et seulement si G est minimal pour les revêtements simples.

Le raisonnement permettant d’obtenir le résultat d’impossibilité impliquant la condition nécessaire de ce théorème a été introduit par Angluin dans [Ang80] pour un modèle différent. Dans le modèle de Mazurkiewicz, le lemme de relèvement d’Angluin permet de montrer que si G est un revêtement de H à travers φ , alors pour tout algorithme \mathcal{A} , il existe une exécution de \mathcal{A} sur G qui est obtenue à partir d’une exécution de \mathcal{A} sur H de telle sorte que dans la configuration finale, tout sommet v de G est dans le même état que $\varphi(v)$. Cela permet de montrer que si G n’est pas minimal pour les revêtements, alors toute étiquette apparaissant sur un sommet dans la configuration finale apparaît au moins deux fois. Ainsi, si G n’est pas minimal pour les revêtements simples, il n’existe pas d’algorithme d’élection pour G .

Pour obtenir la condition suffisante, Mazurkiewicz a présenté un algorithme très élégant pour résoudre le problème du nommage. Cet algorithme attribue des numéros aux différents processus de telle sorte que dans la configuration finale, si deux sommets ont les mêmes numéros, alors les numéros apparaissant dans leurs voisinages respectifs sont les mêmes. Lorsque le graphe est minimal pour les revêtements simples, cela permet de montrer qu’un numéro unique a été attribué à chaque sommet. Cet algorithme est polynomial, au sens où chaque processus nécessite une mémoire polynomiale (en la taille du graphe) et où dans toute exécution, le nombre de pas de calcul est polynomial.

Dans [BCG⁺96], Boldi *et al.* considèrent un modèle entrelacé, où un pas de calcul permet de modifier l’état d’un seul sommet en fonction de l’état de ses voisins. Dans ce modèle entrelacé, deux sommets voisins ne peuvent pas modifier leurs états simultanément. Un algorithme peut être décrit

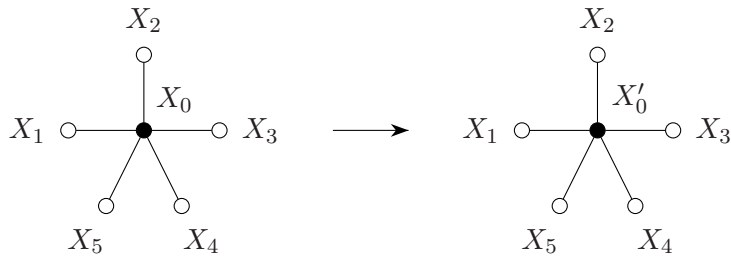


FIG. 2 – La forme d’une règle de réétiquetage dans le modèle entrelacé de Boldi *et al.* On représente en blanc les sommets dont l’étiquette n’est pas modifiée par l’application de la règle.

par un ensemble de règles de réétiquetage dont le support est une étoile, mais où seule l’étiquette du centre de l’étoile peut être modifiée. Ces règles sont de la forme de la règle présentée sur la Figure 2.

Dans ce modèle, les problèmes d’élection et de nommage ne sont pas équivalents ; Boldi *et al.* ont caractérisé les graphes admettant un algorithme d’élection. À partir de leurs résultats, on peut aussi caractériser les graphes admettant un algorithme de nommage. Dans le modèle de Boldi *et al.*, la bonne notion de « symétrie » est la notion de fibration (homomorphisme de graphes dirigés particulier).

Ici aussi, la condition nécessaire est obtenue grâce à une adaptation du lemme de relèvement d’Angluin. En revanche, l’algorithme proposé par Boldi *et al.* pour obtenir une condition suffisante est très différent de l’algorithme de Mazurkiewicz. Cet algorithme est proche de l’algorithme de Yamashita et Kameda [YK96a] initialement conçu pour le modèle où les processus communiquent par échange de messages. Il faut noter que dans l’algorithme ainsi obtenu par Boldi *et al.*, chaque processus utilise une mémoire exponentielle.

1.2.2 Contributions

Un des premiers objectifs de ma thèse était d’étudier des modèles « intermédiaires » entre le modèle de Mazurkiewicz (où chaque pas de calcul nécessite une synchronisation entre un sommet et tous ses voisins) et le modèle où les processus communiquent par échange de messages en mode asynchrone (qui ne nécessite aucune synchronisation). Le but de ce travail était de déterminer si les résultats de possibilités et d’impossibilités existants étaient spécifiques à chacun des modèles, ou s’ils étaient plus généraux.

Dans ma thèse, j’ai étudié des modèles où un pas de calcul utilise seulement une synchronisation entre deux sommets voisins dans le graphe. Dans de tels modèles, un algorithme peut être décrit par des règles de réétiquetage dont le support est une arête (et non pas une étoile, comme c’est le cas dans les modèles étudiés par Mazurkiewicz et Boldi *et al.*). Par la suite, on parle de calculs locaux sur les étoiles et de calculs locaux sur les arêtes pour distinguer les deux types de modèles.

- Lorsqu’on considère les calculs locaux sur les arêtes, on peut distinguer deux types de modèles :
- ou bien, l’application d’une règle sur une arête permet de modifier les étiquettes des deux extrémités de l’arête,
 - ou alors, l’application d’une règle sur une arête permet de modifier l’étiquette d’une seule des deux extrémités de l’arête.

On se rend rapidement compte que dans les deux cas, on obtient des modèles strictement plus puissants lorsque les arêtes peuvent être étiquetées et réétiquetées par l’application d’une règle. En effet, si les arêtes peuvent être étiquetées, alors cela permet à chaque sommet de distinguer les arêtes qui lui sont incidentes. Cet étiquetage des arêtes joue le rôle d’un étiquetage des ports habituellement disponible dans les modèles où les processus communiquent par échange de messages et permet donc à chaque sommet de distinguer ses voisins.

On obtient ainsi quatre modèles différents de calculs locaux sur les arêtes, selon que les arêtes peuvent être ou non étiquetées et selon qu’une règle de réétiquetage permet de modifier les étiquettes des deux extrémités de l’arête ou d’une seule. Dans ces différents modèles, les algorithmes sont décrits par des règles de réétiquetage de la forme de celles présentées sur la Figure 3.

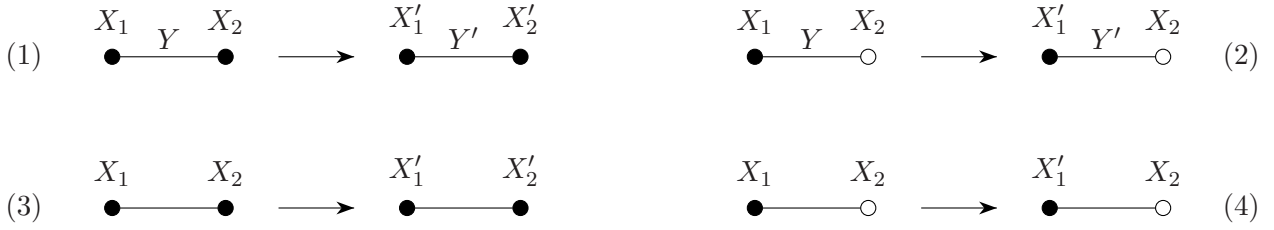


FIG. 3 – Les formes des règles de réétiquetage des différents modèles de calculs locaux sur les arêtes. Comme dans la Figure 2, on a représenté en blanc les sommets dont les étiquettes ne peuvent pas être modifiées. Dans les modèles (1) et (2), les arêtes peuvent être étiquetées et réétiquetées, mais pas dans les modèles (3) et (4).

Dans les modèles (1) et (3) de la Figure 3, un pas de calcul permet à deux sommets voisins de modifier leurs étiquettes en fonction de leurs étiquettes précédentes. La différence entre ces deux modèles est que dans le modèle (3), les arêtes du graphes peuvent être étiquetées et réétiquetées : lorsqu’une règle de réétiquetage est appliquée sur une arête, la règle peut dépendre de l’étiquette de l’arête et peut modifier cette étiquette.

Dans les modèles (2) et (4) de la Figure 3, un pas de calcul permet à un sommet de modifier son étiquette en fonction de l’étiquette d’un de ses voisins. Dans le modèle (2), comme dans le modèle (1), les arêtes du graphes peuvent être étiquetées et lorsqu’une règle de réétiquetage permet de modifier l’étiquette d’un sommet v en fonction de l’étiquette de l’un de ses voisins v' , la règle peut dépendre de l’étiquette de l’arête $\{v, v'\}$ et peut modifier cette étiquette.

Démarche générale. Pour caractériser les graphes dans lesquels on peut élire ou nommer dans les différents modèles de calculs locaux sur les arêtes, la démarche générale est la suivante.

Pour obtenir des conditions nécessaires, il faut d’abord trouver quels sont les homomorphismes qui permettent d’obtenir un lemme de relèvement « à la Angluin » pour la classe de graphes la plus large possible pour chaque modèle considéré. Pour obtenir des conditions suffisantes, il faut ensuite trouver un algorithme qui permette de nommer dans tous les graphes pour lesquels on n’a pas obtenu de modèle d’impossibilité. La recherche de ces homomorphismes et de ces algorithmes sont très liés, puisqu’il faut réussir à déterminer exactement quelles sont les informations que peut obtenir chaque sommet à propos de ses voisins dans chacun des modèles considérés.

Les algorithmes que l’on propose sont des adaptations non-triviales de l’algorithme de Mazurkiewicz : il faut dans chaque modèle utiliser des méthodes particulières afin de s’assurer que chaque sommet réussisse à collecter toute l’information à propos de ses voisins dont il a besoin.

Les caractérisations obtenues permettent de mettre en évidence les relations entre les modèles et la différence entre leurs puissances de calcul respectives.

On cherche ensuite à déterminer quelles connaissances initiales sont nécessaires pour résoudre ces problèmes. Les résultats obtenus nous permettent de mettre en évidence quels sont les résultats existants dans le modèle de Mazurkiewicz [MT00, GM02, GM03, GMM04] qu’on peut espérer pouvoir étendre dans les différents modèles considérés et quels sont les résultats spécifiques à certains modèles. Cela permet de présenter sous un angle différent les différences entre les modèles considérés.

Résultats. Avec Y. Métivier, nous avons étudié les modèles de calculs locaux sur les arêtes où les arêtes peuvent être étiquetées et réétiquetées qui correspondent aux modèles (1) et (2) de la Figure 3. Nous avons montré que ces deux modèles ont la même puissance de calcul. Par ailleurs, si on suppose qu’initialement chaque sommet connaît son degré, alors ces modèles sont équivalent au modèle étudié par Angluin [Ang80] et au modèle de communication synchrone dans un environnement asynchrone pour lesquels aucune caractérisation des graphes admettant un algorithme d’élection ou de nommage n’existait.

Dans tous ces modèles, la bonne notion de « symétrie » s'exprime à l'aide de revêtements de graphes pouvant avoir des arêtes multiples (mais pas de boucles). Il existe des algorithmes d'élection ou de nommage pour un graphe G si et seulement si G n'est revêtement d'aucun autre graphe H .

Grâce à cette caractérisation, on montre que dans ces modèles, on peut élire et nommer dans strictement moins de graphes que dans le modèle de Mazurkiewicz mais dans strictement plus que dans le modèle de Boldi *et al.*

Ces résultats ont été publiés dans [3] et correspondent au Chapitre 3 de ma thèse.

Lorsque les arêtes ne peuvent pas être étiquetées, il n'existe pas de résultat d'équivalence entre la puissance des modèles (3) et (4) de la Figure 3. Dans ces modèles, un sommet ne peut pas distinguer deux de ses voisins qui sont dans le même état.

Dans le modèle (3) de la Figure 3, la bonne notion de « symétrie » est la notion de *pseudo-revêtement*. Un graphe G est un pseudo-revêtement d'un graphe H s'il existe un homomorphisme φ de G dans H et un sous-graphe couvrant G' tel que G' est un revêtement de H à travers φ . Dans ce modèle, il existe un algorithme d'élection ou de nommage pour un graphe G si et seulement si G n'est un pseudo-revêtement d'aucun autre graphe H .

Grâce à cette caractérisation, on montre que ce modèle est strictement plus faible que le modèle de Mazurkiewicz et que les modèles correspondant aux règles (1) et (2) de la Figure 3. On montre aussi que la puissance de calcul de ce modèle est incomparable avec celle du modèle de Boldi *et al.*

Ces résultats ont été publiés dans [5] et correspondent au Chapitre 5 de ma thèse.

Avec Y. Métivier et W. Zielonka, nous avons étudié le modèle (4) de la Figure 3. Ce modèle nécessite la plus faible synchronisation possible : en un pas de calcul, un sommet observe l'état d'un de ses voisins et modifie son état. Dans ce modèle, les problèmes du nommage et de l'élection ne sont pas équivalents et les bonnes notions de « symétries » sont exprimées à l'aide de submersions. Un graphe G est une submersion d'un graphe H s'il existe un homomorphisme φ de G dans H qui est localement surjectif : pour tout sommet $v \in V(G)$, φ induit une surjection de l'ensemble des voisins de v dans G dans l'ensemble des voisins de $\varphi(v)$ dans H .

Dans ce modèle, il existe un algorithme de nommage pour un graphe G si et seulement si G n'est une submersion d'aucun autre graphe H . La caractérisation des graphes admettant un algorithme d'élection repose aussi sur la notion de submersion, mais la caractérisation obtenue et l'algorithme d'élection sont assez techniques.

Grâce à ces caractérisations, on montre que ce modèle est strictement plus faible que les modèles correspondant aux règles (1), (2) et (3) de la Figure 3 et qu'il est plus faible que les modèles de Mazurkiewicz et de Boldi *et al.*

Ces résultats ont été publiés dans [2, 4] et correspondent au Chapitre 6 de ma thèse.

1.3 Échanges de messages en mode asynchrone

J'ai aussi étudié dans ma thèse le modèle plus classique où les processus communiquent en échangeant des messages en mode asynchrone. Dans ce modèle, en un pas de calcul, chaque processeur peut ou bien envoyer un message à l'un de ses voisins, ou bien recevoir un message qu'un de ses voisins lui a envoyé, ou bien modifier son état.

Afin que chaque processus puisse distinguer les liens de communication qui lui sont incidents, on suppose que le réseau est muni d'un étiquetage des ports. Ainsi, chaque sommet attribue un numéro unique à chacune des arêtes qui lui sont incidentes.

Dans ce modèle, j'ai travaillé sur le problème de l'élection, ainsi que sur le problème classique de la détection de la terminaison et sur un problème nécessitant de briser seulement partiellement les symétries « initiales » du réseau.

1.3.1 Élection

Dans ce modèle, Yamashita et Kameda [YK96a] ont présenté la première caractérisation des graphes admettant un algorithme d'élection. La caractérisation présentée par Yamashita et Kameda

repose sur la notion de « vues », où la vue d'un sommet v d'un graphe G est un arbre infini dont les sommets correspondent aux chemins issus de v dans G .

Pour obtenir une condition nécessaire, Yamashita et Kameda montrent que pour tout algorithme, si deux sommets ont la même vue, il existe une exécution de l'algorithme lors de laquelle ces deux sommets restent toujours dans le même état. Pour obtenir une condition suffisante, ils utilisent un résultat de Norris [Nor95] qui a montré que deux sommets d'un graphe de taille n avaient la même vue si leurs vues tronquées à la hauteur n étaient les mêmes. Ainsi, Yamashita et Kameda présente un algorithme où chaque sommet construit sa vue tronquée à la hauteur n et la compare ensuite avec celle des autres sommets du graphe ; le sommet ayant la plus « petite » vue est ensuite élu. Il faut noter que lors de toute exécution de l'algorithme ainsi obtenu, les processus échangent des messages de taille exponentielle (en la taille du graphe) et utilisent une mémoire de taille exponentielle.

La caractérisation et l'algorithme de Yamashita et Kameda sont très différents de la caractérisation et de l'algorithme présentés par Mazurkiewicz dans son modèle. Toutefois, le modèle *synchrone*² de Boldi *et al.* [BCG⁺96] a la même puissance de calcul que le modèle étudié par Yamashita et Kameda dans [YK96a]. On peut ainsi présenter une caractérisation des graphes admettant un algorithme d'élection dans le modèle de Yamashita et Kameda exprimée à l'aide de revêtements dirigés (qui correspondent aux revêtements dans le cadre des graphes dirigés). Cependant, les algorithmes présentés par Boldi *et al.* pour obtenir des conditions suffisantes sont inspirés de l'algorithme de Yamashita et Kameda et leur implémentation dans des systèmes communiquant par échange de messages en mode asynchrone nécessite aussi l'utilisation de messages de taille exponentielle.

Dans [YK99], Yamashita et Kameda étudie le problème de l'élection dans trois autres modèles où les processus communiquent par échange de messages. Dans certains de ces modèles, les processus ne peuvent pas déterminer l'origine d'un message reçu : le numéro du port par lequel ce message est arrivé n'est pas disponible. Dans d'autres modèles, lorsqu'un processus envoie un message, il doit envoyer le même message à tous ses voisins. On obtient ainsi quatre modèles dont le plus puissant est le modèle étudié dans [YK96a].

Résultats. Avec Y. Métivier, nous avons présenté un algorithme « à la Mazurkiewicz » dans le modèle étudié par Yamashita et Kameda dans [YK96a]. L'algorithme que nous avons obtenu est un algorithme « totalement » polynomial au sens où le temps d'exécution, la mémoire de chaque processus, le nombre et la taille des messages échangés sont polynomiaux en la taille du graphe.

Par ailleurs, contrairement à l'algorithme de Yamashita et Kameda, l'algorithme que nous avons obtenu est totalement asynchrone : les processus n'ont pas à attendre régulièrement d'avoir reçu un message de chacun de leurs voisins pour pouvoir continuer à exécuter l'algorithme. Cette propriété permet d'assurer que pour tout graphe, il existe toujours une exécution de l'algorithme permettant de distinguer un sommet, même lorsque le graphe n'admet pas d'algorithme d'élection³. Ainsi, notre algorithme permet d'exploiter l'asymétrie qui provient de l'exécution et non du graphe.

On a aussi montré que les techniques utilisées dans ce modèle peuvent aussi être employées dans les modèles considérés par Yamashita et Kameda dans [YK99].

Ces résultats ont été publiés dans [6] et correspondent au Chapitre 7 de ma thèse.

1.3.2 Détection de la terminaison

Le problème de la détection de la terminaison est le suivant : étant donné un algorithme distribué quelconque qui réalise une tâche sur une famille de graphes, peut-on modifier cet algorithme de telle sorte que lorsque l'exécution est terminée (chaque processus a calculé sa valeur finale), alors au moins un processus puisse détecter que l'exécution est globalement terminée.

Ce problème est un problème classique en algorithmique distribuée (cf. Chap. 8 de [Tel00], [Mat87]) et de nombreux algorithmes ont été proposés lorsque certaines hypothèses sont vérifiées. Ainsi, on sait résoudre ce problème lorsqu'il existe un sommet distingué dans le réseau [Ang80], lorsque le réseau est un arbre [Ang80], ou lorsqu'une borne sur le diamètre du réseau est connue [SSP85].

²Ce modèle est légèrement différent du modèle *entrelacé* évoqué dans la partie précédente.

³C'est à dire qu'il existe aussi des exécutions de l'algorithme qui ne permettent pas d'élire un sommet.

Résultats. Avec E. Godard, Y. Métivier et G. Tel, nous avons déterminé quelles étaient les conditions nécessaires et suffisantes permettant de transformer un algorithme distribué en un algorithme distribué avec détection de la terminaison.

En fait, il s'avère que les conditions permettant d'obtenir un algorithme distribué avec détection de la terminaison ne dépendent que de la famille de graphes sur laquelle l'algorithme donné doit résoudre une tâche, et non de l'algorithme lui-même. Ainsi, nous avons caractérisé les familles de graphes pour lesquelles il était possible de réaliser des tâches de manière distribuée avec détection de la terminaison. Les résultats connus deviennent alors des corollaires de cette caractérisation. Nous avons aussi exhibé d'autres corollaires donnant de nouveaux critères simples permettant d'assurer l'existence d'un algorithme détectant la terminaison.

Ces travaux reposent sur l'algorithme présenté dans [6] et sur des généralisations de techniques employées dans [GM02, MT00].

Ces résultats ont été publiés dans [12].

1.3.3 Briser partiellement les « symétries »

Avec S. Das et N. Santoro, nous nous sommes intéressés à un problème où on ne doit pas briser totalement les « symétries » du réseau, comme c'est le cas lorsqu'on veut résoudre les problèmes de l'élection ou du nommage.

Dans ce travail, on cherche à former des « paires » dans le réseau : chaque processus doit avoir un partenaire (dont il est lui-même le partenaire) et il doit connaître les numéros de ports apparaissant sur un chemin le reliant à son partenaire. On a étudié quelles étaient les graphes dans lesquels on pouvait résoudre ce problème en fonction de la connaissance initiale disponible (connaissance de la topologie du graphe, de la taille du graphe, d'une borne sur la taille du graphe, ou pas de connaissance du tout).

On utilise les méthodes présentées dans [6] pour obtenir de l'information sur le réseau sur lequel l'algorithme est exécuté. Les résultats s'appuient aussi sur une étude des propriétés combinatoires des revêtements dirigés.

Ces résultats ont été publiés dans [10].

1.4 Agents mobiles

Le concept d'agents mobiles a été développé afin de pouvoir résoudre des problèmes dans des environnements hétérogènes et dynamiques [BR05]. Dans de tels systèmes, les différents noeuds du réseau sont passifs et ce sont des agents mobiles se déplaçant de noeud en noeud qui sont en charge de l'exécution de l'algorithme.

Dans les systèmes à agents mobiles, un problème très étudié est le problème du *rendez-vous*. Le but d'un algorithme de rendez-vous est d'arriver à une configuration où tous les agents sont réunis en un même noeud du réseau.

Dans la littérature, de nombreux résultats existent dans différents modèles [ABRS95, BFFS03b, BFFS06, BS94, DFNS05, DFNS06, DFP03, KKR06].

Résultats. Avec E. Godard, Y. Métivier et R. Ossamy, nous avons considéré un modèle assez général qui correspond au modèle étudié par Das, Barrière, Flocchini, Fraigniaud, Nayak et Santoro dans [DFNS05, DFNS06, BFFS03b, BFFS06]. Dans ce modèle, le système est asynchrone et les agents peuvent communiquer en lisant et écrivant des messages sur les sommets où ils se trouvent.

Nous avons montré qu'un tel système à agents mobiles à la même puissance de calcul qu'un système distribué où les processus communiquent par échange de messages si les graphes sous-jacents sont identiques.

On en déduit une caractérisation des systèmes à agents mobiles dans lesquels on peut résoudre le problème du rendez-vous et on généralise ainsi les résultats de Das *et al.* [DFNS05]. Avec le même raisonnement, on peut utiliser les résultats de Flocchini *et al.* [FRS03] pour généraliser les résultats de Barrière *et al.* présentés dans [BFFS03b, BFFS06].

Ces résultats ont été publiés dans [11] et correspondent au Chapitre 8 de ma thèse.

Dans [BFFS03a], Barrière *et al.* ont introduit un modèle de systèmes à agents mobiles où chaque agent dispose initialement d'une étiquette unique (le système n'est donc pas anonyme) mais il n'existe aucun ordre global sur ces étiquettes : chaque agent a une couleur et dispose d'un ordre sur l'ensemble des couleurs, mais les ordres utilisés par les agents peuvent être différents.

Ils ont étudié le problème de l'élection dans ce modèle et ont obtenu des résultats pour les systèmes où le graphe sous-jacent était un graphe de Cayley.

Un problème ouvert posé par Barrière *et al.* est de savoir s'il existe un algorithme qui pour tout système à agents mobiles permet ou bien de résoudre le problème du rendez-vous, ou bien de détecter qu'il est impossible de résoudre ce problème.

Résultats. J'ai considéré un modèle a priori plus faible que celui de Barrière *et al.* dans lequel les moyens qu'avaient les agents de communiquer entre eux étaient plus limités que dans le modèle original.

Dans ce modèle, j'ai caractérisé les systèmes à agents mobiles admettant des algorithmes d'élection et de rendez-vous et j'ai exhibé un algorithme qui pour tout système à agents mobiles permet ou bien de résoudre le problème du rendez-vous, ou bien de détecter qu'il est impossible de résoudre ce problème.

Il est important de noter que les conditions nécessaires pour résoudre dans le modèle de Barrière *et al.* sont équivalentes à celles obtenues dans le modèle que j'ai étudié. J'ai ainsi répondu positivement à la question posée par Barrière *et al.*

Ces résultats ont été publiés dans [8].

1.5 Complexité

Puisqu'on sait quels sont les graphes admettant un algorithme d'élection (ou de nommage) dans les différents modèles, il est naturel de chercher à déterminer quelle est la complexité de décider si un graphe G donné admet un algorithme d'élection (ou de nommage) dans un modèle fixé.

Dans certains modèles de [YK99] où les processus communiquent par échange de messages, Boldi et Vigna [BV02] ont montré que les problèmes de décision correspondant aux caractérisations existantes pouvaient être résolus en temps polynomial. Dans le modèle étudié dans [YK96a], Yamashita et Kameda [YK96b] ont montré qu'il était co-NP-complet de décider si un graphe donné admettait un algorithme d'élection.

Résultats. Avec D. Paulusma, nous nous sommes intéressés au dernier modèle où les processus communiquent par échange de messages étudié dans [YK99] pour lequel aucun résultat n'était connu. Nous nous sommes également intéressés à tous les modèles de calculs locaux. Nous avons ainsi étudié la complexité de six problèmes de décision correspondant aux caractérisations existantes.

Pour chacun de ces modèles, nous avons montré qu'il était co-NP-complet de décider si un graphe donné admettait un algorithme de nommage ou d'élection dans ce modèle.

Il est intéressant de noter que dans certains de ces modèles, les algorithmes existants sont polynomiaux : cela permet de mettre en évidence l'aspect non-déterministe d'une exécution distribuée.

Ces résultats ont été publiés dans [7] et correspondent au Chapitre 9 de ma thèse.

2 Théorie des graphes

Avec D. Gonçalves et P. Ochem, nous nous sommes intéressés à certaines classes de graphes définies par des modèles d'intersection. Dans les modèles d'intersection, un graphe est défini par un ensemble d'objets correspondant aux sommets et il existe une arête entre deux sommets si et seulement si les objets correspondants s'intersectent. Par exemple, les graphes de cordes sont les graphes définis par intersection de cordes dans un cercle : à chaque sommet du graphe correspond une corde du cercle et

deux sommets sont voisins si et seulement si les cordes correspondantes s'intersectent. Un exemple est présenté sur la Figure 4.

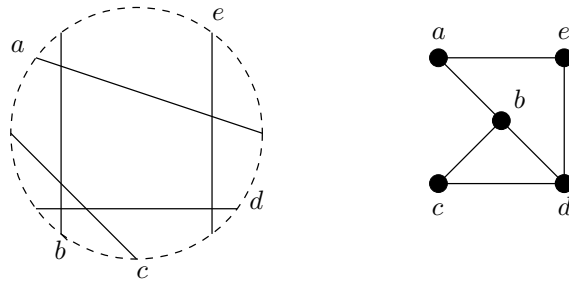


FIG. 4 – Un graphe de cordes et son modèle d'intersection.

La famille *String* est la famille des graphes d'intersection de courbes dans le plan. Ehrlich *et al.* [EET76] ont montré en 1976 que tout graphe planaire appartient à la classe *String*. En 1984, Scheinerman [Sch84] a conjecturé dans sa thèse que tout graphe planaire était le graphe d'intersection de segments dans le plan.

Résultats. Nous avons montré une version plus faible de cette conjecture en prouvant que tout graphe planaire est le graphe d'intersection d'une famille de courbes du plan qui se coupent au plus une fois (dans la construction de Ehrlich *et al.*, les courbes correspondant à deux sommets adjacents s'intersectaient deux fois). Ce résultat répond à un problème étudié par d'éminents chercheurs (cf. <http://www.ehess.fr/centres/cams/person/pom/langfr/research.html> parmi lesquels : H. de Fraysseix (Paris), J. Kratochvil (Prague), B. Mohar (Ljubljana), P. Ossona de Mendez (Paris), J. Pach (Budapest, New-York City) et C. Thomassen (Copenhage).

Ce résultat a été publié dans [13].

Nous avons aussi étudié les relations entre certaines classes de graphes définies par modèle d'intersection et des classes définies par modèle de chevauchement. Dans les modèles de chevauchement, deux sommets v_1, v_2 sont adjacents si les objets correspondants o_1, o_2 s'intersectent et si $o_1 \setminus o_2$ et $o_2 \setminus o_1$ sont non-vides. Par exemple, tout graphe de cordes est défini par chevauchement de sous-arbres dans un chemin. Il est aussi connu que la classe des graphes de co-comparabilité correspond à la classe de graphes défini par chevauchement de sous-arbres dans une étoile.

La classe *Interval Filament* est l'ensemble des graphes pouvant être définis par intersection de courbes de fonctions continues positives définies sur des intervalles fermés de \mathbb{R} et s'annulant aux extrémités de leurs intervalles de définition. Il est connu que cette classe est contenu dans *String* et qu'elle contient les graphes de cordes et les graphes de co-comparabilité.

La classe *Subtree overlap* est l'ensemble des graphes pouvant être définis par chevauchement de sous-arbres dans un arbre. Il est aussi connu que cette classe est contenue dans *String* et par définition, on voit qu'elle contient les graphes de co-comparabilité et les graphes de cordes.

Résultats. Nous avons montré que la classe *Interval Filament* coïncide avec la classe des graphes définis par chevauchement de sous-arbres dans une chenille (caterpillar en anglais). Un corollaire de ce résultat est que la classe *Interval Filament* est incluse dans la classe *Subtree overlap* (cette inclusion est stricte).

Nous avons aussi montré que la classe *Subtree overlap* est close par contraction d'arêtes. La transformation de modèle présenté dans cette preuve permet aussi de montrer que les classes *Interval Filament* et *Co-Comparabilité* sont aussi closes par contraction d'arêtes (ces résultats étaient connus).

Ces résultats n'ont pour l'instant pas été publiés.

3 Combinatoire des mots

Il est bien connu qu'il n'existe pas de mot infini sur deux lettres ne contenant aucun carré (i.e., ne contenant aucun facteur de la forme uu où u est un mot) et on sait depuis les travaux de Thue [Thu06, Thu10] qu'il existe un mot infini sur trois lettres évitant les carrés.

Un carré est une répétition d'exposant 2 et plus généralement, une répétition d'exposant rationnel est définie de la manière suivante. Un mot u est une répétition d'exposant α (où α est un nombre rationnel strictement supérieur à 1) s'il peut s'écrire sous la forme $u = v^n w$ où w est un préfixe de v et où $|u| = \alpha|v|$. Par exemple, le mot aba est une répétition d'exposant $\frac{3}{2}$.

Dejean [Dej72] a conjecturé que pour tout entier $k \geq 5$, il existe un mot infini sur un alphabet à k lettres qui ne contient pas de répétition d'exposant strictement supérieur à $\frac{k}{k-1}$. Un tel mot est dit $(\frac{k}{k-1}^+)$ -libre. Cette conjecture a été prouvée pour $k \in [5, 11]$ par Moulin-Ollagnier [MO92], pour $k \in [12, 14]$ par Currie et Mohammad-Noori [CMN07], et pour $k \geq 38$ par Carpi [Car06].

Dans [Och05], P. Ochem a proposé une version plus forte de cette conjecture où une lettre doit apparaître avec une fréquence minimale (ou maximale) dans le mot $\frac{k}{k-1}$ -libre.

Il a conjecturé que pour tout entier $k \geq 5$, il existe un mot infini à k lettres qui est $(\frac{k}{k-1}^+)$ -libre dans lequel une lettre a une fréquence égale à $\frac{1}{k+1}$.

Il a aussi conjecturé que pour tout $k \geq 6$, il existe un mot infini à k lettres qui est $(\frac{k}{k-1}^+)$ -libre dans lequel une lettre a une fréquence égale à $\frac{1}{k-1}$.

Résultats. Avec P. Ochem, nous avons montré que le premier cas de chaque conjecture était vrai.

Ainsi, il existe un mot infini à 5 lettres qui est $(\frac{5}{4}^+)$ -libre dans lequel une lettre a une fréquence égale à $\frac{1}{6}$ et il existe un mot infini à 6 lettres qui est $(\frac{6}{5}^+)$ -libre dans lequel une lettre a une fréquence égale à $\frac{1}{5}$.

Pour obtenir ces résultats, on a introduit une méthode originale de génération de mots reposant sur un codage sur un alphabet à deux lettres des mots $(\frac{5}{4}^+)$ -libre (resp. $(\frac{6}{5}^+)$ -libre) sur 5 (resp. 6) lettres minimisant (resp. maximisant) la fréquence d'une lettre.

Ces résultats ont été publiés dans [9].

4 Théorie des semigroupes

Dans le cadre d'un stage de maîtrise effectué avec H. Leung, j'ai travaillé sur la théorie des semigroupes. Nous avons étudié la notion de factorisations ramseyennes de mots finis, introduite par Simon [Sim90].

Étant donné un alphabet A , une forêt de factorisations sur A peut être définie par une fonction de factorisation d qui à tout mot $w \in A^*$ associe une suite finie de mots $d(w) = (w_1, w_2, \dots, w_n)$ telle que $w = w_1 w_2 \dots w_n$. De plus, il faut que les seuls mots w tels que $d(w) = w$ soient les lettres de A .

À partir de la fonction d , il est facile d'associer à tout mot $w \in A^*$ un arbre $T(w)$ défini de la manière suivante. Si $|d(w)| = 1$, $T(w)$ est un arbre réduit à un sommet étiqueté w . Si $d(w) = (w_1, w_2, \dots, w_n)$ avec $n \geq 2$, alors la racine de $T(w)$ est étiqueté par w et à n fils $T(w_1), T(w_2), \dots, T(w_n)$. La hauteur d'une forêt de factorisations sur A définie par une factorisation d est la hauteur maximale des arbres associés aux mots de A^* par d .

Étant donné un morphisme f de A^* dans un semigroupe S , une forêt de factorisations sur A définie par une factorisation d est ramseyenne si pour tout mot w tel que $d(w) = (w_1, w_2, \dots, w_n)$ où $n \geq 3$, il existe un idempotent e de S tel que $f(w) = f(w_1) = f(w_2) = \dots = f(w_n) = e$.

En 1990, Simon [Sim90] a montré que tout morphisme f du monoïde libre A^* dans un semigroupe S admet une forêt ramseyenne de factorisations de hauteur au plus $9|S|$.

Résultats. Avec H. Leung, nous avons présenté une preuve plus directe du résultat de Simon qui a permis de réduire la borne supérieure à $7|S|$. Nous avons par ailleurs exhibé plusieurs exemples de familles de semigroupes S pour lesquels il existait des morphismes $f : A^* \rightarrow S$ dont toutes les forêts

ramseyennes de factorisations associées avaient une hauteur supérieure ou égale à $|S|$; ce qui montre l’optimalité à un facteur constant près du résultat de Simon. Ces résultats ont été publiés dans [1].

Bibliographie personnelle

- [1] J. Chalopin et H. Leung. On factorization forests of finite height. *Theoretical Computer Science*, 310(1–3):489–499, 2004.
- [2] J. Chalopin, Y. Métivier et W. Zielonka. Local computations in graphs : the case of cellular edge local computations. *Fundamenta Informaticae*, 74(1):85–114, 2006.
- [3] J. Chalopin et Y. Métivier. Election and local computations on edges. In *Proc. of Foundations of Software Science and Computation Structures, 7th International Conference (FOSSACS’04)*, LNCS 2987, pp. 90–104, 2004.
- [4] J. Chalopin, Y. Métivier et W. Zielonka. Election, naming and cellular edge local computations. In *Proc. of the 2nd International Conference on Graph Transformations (ICGT’04)*, LNCS 3256, pp. 242–256, 2004.
- [5] J. Chalopin. Local computations on closed unlabelled edges : the election problem and the naming problem. In *Proc. of the 31st Conference on Current Trends in Theory and Practice of Informatics (SOFSEM’05)*, LNCS 3381, pp. 81–90, 2005.
- [6] J. Chalopin et Y. Métivier. A bridge between the asynchronous message passing model and local computations in graphs. In *Proc. of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS’05)*, LNCS 3618, pp. 212–223, 2005.
- [7] J. Chalopin et D. Paulusma. Graph labelings derived from models in distributed computing. In *Proc. of Graph-Theoretic Concepts in Computer Science, 32nd International Workshop (WG’06)*, LNCS 4271, pp. 301–312, 2006.
- [8] J. Chalopin. Election in the qualitative world. In *Proc. of Structural Information and Communication Complexity, 13th International Colloquium (SIROCCO’06)*, LNCS 4056, pp. 85–99, 2006.
- [9] J. Chalopin et P. Ochem. Dejean’s conjecture and letter frequency. *Journées Montoises d’Informatique Théorique (JM’06)*, pp. 121–124, 2006.
- [10] J. Chalopin, S. Das et N. Santoro. Groupings and pairings in anonymous networks. In *Proc. of Distributed Computing, 20th International Conference (DISC’06)*, LNCS 4167, pp. 105–119, 2006.
- [11] J. Chalopin, E. Godard, Y. Métivier et R. Ossamy. Mobile agent algorithms versus message passing algorithms. In *Proc. of the 10th International Conference on Principles of Distributed Systems (OPODIS’06)*, LNCS 4305, pp. 187–201, 2006.
- [12] J. Chalopin, E. Godard, Y. Métivier et G. Tel. About the termination detection in the asynchronous message passing model. In *Proc. of the 33rd Conference on Current Trends in Theory and Practice of Informatics (SOFSEM’07)*, LNCS 4362, pp. 200–211, 2007.
- [13] J. Chalopin, D. Gonçalves et P. Ochem. Planar graphs are in 1-string. In *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’07)*, à paraître en 2007.

Bibliographie

- [ABRS95] B. Awerbuch, M. Betke, R. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot (extended abstract). In *Proc. of the 8th annual conference on Computational learning theory (COLT 1995)*, pages 321–328. ACM Press, 1995.
- [Ang80] D. Angluin. Local and global properties in networks of processors. In *Proc. of the 12th Symposium on Theory of Computing (STOC 1980)*, pages 82–93, 1980.

- [BCG⁺96] P. Boldi, B. Codenotti, P. Gemmell, S. Shammah, J. Simon, and S. Vigna. Symmetry breaking in anonymous networks : characterizations. In *Proc. of the 4th Israeli Symposium on Theory of Computing and Systems (ISTCS 1996)*, pages 16–26. IEEE Press, 1996.
- [BFFS03a] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Can we elect if we cannot compare? In *Proc. of the 15th annual ACM Symposium on Parallel Algorithms and Architectures, (SPAA 2003)*, pages 324–332. ACM Press, 2003.
- [BFFS03b] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Election and rendezvous in fully anonymous systems with sense of direction. In *Proc. of the 10th International Colloquium on Structural Information Complexity (SIROCCO 2003)*, volume 17, pages 17–32. Carleton Scientific, 2003.
- [BFFS06] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Rendezvous and election of mobile agents : impact of sense of direction. *Theory of Computing Systems*, to appear, 2006.
- [BR05] P. Braun and W. Rossak. *Mobile agents : basic concepts, mobility models and the tracy toolkit*. Morgan Kaufman, 2005.
- [BS94] M. Bender and D. Slonim. The power of team exploration : two robots can learn unlabeled directed graphs. In *Proc. of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994)*, pages 75–85, 1994.
- [BV02] P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Mathematics*, 243(1-3) :21–66, 2002.
- [Car06] A. Carpi. On the repetition threshold for large alphabets. In *Proc. of the 31st International Symposium on Mathematical Foundations of Computer Science (MFCS 2006)*, volume 4162 of *Lecture Notes in Computer Science*, pages 226–237. Springer, 2006.
- [CMN07] J.D. Currie and M. Mohammad-Noori. Dejean’s conjecture and sturmian words. *European Journal of Combinatorics*, 28(3) :876–890, 2007.
- [Dej72] F. Dejean. Sur un théorème de thue. *J. Combin. Theory, Ser. B*, 13 :90–99, 1972.
- [DFNS05] S. Das, P. Flocchini, A. Nayak, and N. Santoro. Distributed exploration of an unknown graph. In *Proc. of Structural Information and Communication Complexity, 12th International Colloquium (SIROCCO 2005)*, volume 3499 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2005.
- [DFNS06] S. Das, P. Flocchini, A. Nayak, and N. Santoro. Effective elections for anonymous mobile agents. In *Proc. of Algorithms and Computation, 17th International Symposium (ISAAC 2006)*, Lecture Notes in Computer Science. Springer, 2006.
- [DFP03] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In *Proc. of the 11th Annual European Symposium (ESA 2003)*, volume 2832 of *Lecture Notes in Computer Science*, pages 184–195, 2003.
- [EET76] G. Ehrlich, S. Even, and R.E. Tarjan. Intersection graphs of curves in the plane. *J. Combin. Theory, Ser. B*, 21 :8–20, 1976.
- [FRS03] P. Flocchini, A. Roncato, and N. Santoro. Computing on anonymous networks with sense of direction. *Theoretical Computer Science*, 301(1-3) :355–379, 2003.
- [GM02] E. Godard and Y. Métivier. A characterization of families of graphs in which election is possible (*ext. abstract*). In *Proc. of Foundations of Software Science and Computation Structures, 5th International Conference (FOSSACS 2002)*, volume 2303 of *Lecture Notes in Computer Science*, pages 159–172. Springer-Verlag, 2002.
- [GM03] E. Godard and Y. Métivier. Deducible and equivalent structural knowledges in distributed algorithms. *Theory of Computing Systems*, 36(2) :631–654, 2003.
- [GMM04] E. Godard, Y. Métivier, and A. Muscholl. Characterization of classes of graphs recognizable by local computations. *Theory of Computing Systems*, 37(2) :249–293, 2004.

- [KKR06] E. Kranakis, D. Krizanc, and S. Rajsbaum. Mobile agent rendezvous : A survey. In *Proc. of Structural Information and Communication Complexity, 13th International Colloquium (SIROCCO 2006)*, volume 4056 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2006.
- [Mat87] F. Mattern. Algorithms for distributed termination detection. *Distributed computing*, 2(3) :161–175, 1987.
- [Maz97] A. Mazurkiewicz. Distributed enumeration. *Information Processing Letters*, 61(5) :233–239, 1997.
- [MO92] J. Moulin-Ollagnier. Proof of dejean’s conjecture for alphabets with 5, 6, 7, 8, 9, 10 and 11 letters. *Theoretical Computer Science*, 95(2) :187–205, 1992.
- [MT00] Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *Proc. of 7th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2000)*, Proceedings in Informatics, pages 237–251. Carleton Scientific, 2000.
- [Nor95] N. Norris. Universal covers of graphs : isomorphism to depth $n - 1$ implies isomorphism to all depths. *Discrete Applied Math.*, 56(1) :61–74, 1995.
- [Och05] P. Ochem. Letter frequency in infinite repetition-free words. In *Proc. of the 5th International Conference on Words (Words 2005)*, volume 36 of *Publications du LACIM*, 2005.
- [RFH72] P. Rosenstiehl, J.-R. Fiksel, and A. Holliger. Intelligent graphs. In R. Read, editor, *Graph theory and computing*, pages 219–265. Academic Press (New York), 1972.
- [Sch84] E.R. Scheinerman. *Intersection classes and multiple intersection parameters of graphs*. PhD thesis, Princeton University, 1984.
- [Sim90] I. Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1) :65–94, 1990.
- [SSP85] B. Szymanski, Y. Shy, and N. Prywes. Terminating iterative solutions of simultaneous equations in distributed message passing systems. In *Proc. of the 4th Annual ACM Symposium on Principles of Distributed Computing (PODC 1985)*, pages 287–292. ACM Press, 1985.
- [Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.
- [Thu06] A. Thue. Über unendliche zeichenreihen. *Norske Vid. Selsk. Skr. I. Mat. Nat. Kl.*, 7 :1–22, 1906.
- [Thu10] A. Thue. Über die gegenseitige lage gleicher teile gewisser zeichenreihen. *Norske Vid. Selsk. Skr. I. Mat. Nat. Kl.*, 10 :1–67, 1910.
- [YK96a] M. Yamashita and T. Kameda. Computing on anonymous networks : Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1) :69–89, 1996.
- [YK96b] M. Yamashita and T. Kameda. Computing on anonymous networks : Part ii - decision and membership problems. *IEEE Transactions on parallel and distributed systems*, 7(1) :90–96, 1996.
- [YK99] M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transactions on parallel and distributed systems*, 10(9) :878–887, 1999.