Modélisation et traitement décentralisé des graphes dynamiquesApplication aux réseaux mobiles ad hoc

Yoann Pigné

UNIVERSITÉ DU HAVRE

UFR Sciences et Techniques

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DU HAVRE

Discipline : Informatique

Yoann PIGNÉ

le 4 décembre 2008

MODÉLISATION ET TRAITEMENT DÉCENTRALISÉ DES GRAPHES DYNAMIQUES

–

APPLICATION AUX RÉSEAUX MOBILES *AD HOC*

Directeur :		
	Frédéric GUINAND	Professeur à l'université du Havre
Raporteurs :		
	Pascal BOUVRY	Professeur à l'université du Luxembourg
	Serge CHAUMETTE	Professeur à l'université de Bordeaux 1
Examinatrices :		
	Isabelle GUÉRIN-LASSOUS	Professeure à l'université de Lyon 1
	Clémence MAGNIEN	Chargée de recherche CNRS à l'université de Paris 6
Invité :		
	Alain CARDON	Professeur de première classe retraité de l'université du Havre

REMERCIEMENTS

Vous commencez ici la lecture de cette thèse, moi, c'est par ces remerciements que j'en termine la rédaction. Difficile en une page d'énumérer, sans en oublier, toutes les personnes qui ont, de près ou de loin, eu une importance dans cette aventure, et à qui je dois un "Merci !". L'exercice est d'autant plus délicat que c'est peut-être la seule page que vous lirez (c'est la seule page que je lis d'ordinaire dans une thèse... enfin presque).

Je commencerai donc de manière assez conventionnelle mais avec un réel plaisir par remercier Frédéric GUINAND pour m'avoir permis de travailler avec lui. Notre collaboration a commencé avant cette thèse et se prolongera encore après, je l'espère. Merci Fred pour ta disponibilité, pour ta générosité, pour ton optimisme infaillible et pour ton goût pour la bonne chère.

Merci à Pascal BOUVRY et à Serge CHAUMETTE pour avoir accepté de rapporter sur cette thèse. Par votre travail de lecture et d'analyse vous représentez la première validation de mon travail hors de l'équipe et je vous en suis très reconnaissant. Merci à Isabelle GUÉRIN-LASSOUS, à Clémence MAGNIEN et à Alain CARDON pour avoir accepté de participer au jury.

Je tiens à remercier les membres de l'équipe RI_2C qui m'ont accueilli et avec qui j'ai pu travailler. Damien, merci pour ton esprit de groupe et pour ta vision noble (et remotivante quand il le faut) de la recherche. Cyrille, merci pour tout ce que tu fais pour l'équipe et pour le labo et bravo pour tes goûts en pâtisseries orientales et autres tabacs à narguilé. Stefan, tu vas bientôt savoir mieux parler havrais que moi. Véronique, grâce à toi j'ai découvert que j'étais baryton, la classe ! Merci pour les relectures de mon travail et bravo pour ta maîtrise spectaculaire des lettres. Jean-Luc, si tu ne devais en choisir qu'une (lettre), je sais laquelle ce serait. Mon respect indéfectible pour ta blague de la petite fille et de sa grand-mère. Antoine, mon maître *geek*, je suis très fier de notre projet commun mais le truc le plus classe est quand même de t'avoir converti à *Gentoo*. Pierrick, bravo pour ta quête de la-connaissance-de-tout-sur-absolument-tout, tu t'en sors pas mal. Tu as rendu nos trajets en train moins ennuyeux. Merci pour ça et merci pour le Métal. Merci à la *team Urban Terror* : Butcher, Obama, Razi3l, Ichitaka, Master-Kekete, Virus, Kamikaze, Astra, Ciao16Coups et Mobutu pour cette ambiance de travail si harmonieuse et paisible qui a toujours garanti une productivité optimale dans mon travail.

Un peu plus loin de l'équipe et du labo, il y a toutes les rencontres faites au détour d'une conférence ou plus simplement d'un couloir. Merci Djamila, Nathalie, Arnaud, Luc, Apivadee, Louisa, *you all are so great, my friends*.

Il y a les potes de toujours, ceux que j'ai saoulés et que je saoulerai encore en essayant d'expliquer ce que je fais. Merci Anelise, Quentin, Antoine, Nicolas, Yannis, Valéry, Mélanie, Alexia, Lucie, Matthieu.

Il y a aussi la famille qui m'a toujours fait confiance, Gilbert, Marie, Tony, Max, Christine, Frédo, Hélène, Vinie, Romu et bien-sûr mes parents qui m'ont toujours soutenu et encouragé.

Enfin, il y a Toi. Merci pour tout ce qu'on a eu.

À mes parents.

TABLE DES MATIÈRES

TABLE DES FIGURES

Liste des définitions

CHAPITRE 1

INTRODUCTION

1.1 Systèmes complexes et graphes dynamiques

Le monde réel, tel que peut le voir un observateur est chargé de diversité et de complexité. On trouve la complexité aussi bien dans la nature que dans nos sociétés. Elle réside dans l'organisation de la matière et des processus physiques tout autant que dans le monde vivant ou dans les interactions entre individus. De ces observations est née la notion de « systèmes complexes ». Cette notion est transversale à un vaste ensemble de domaines scientifiques. Biologistes, physiciens, linguistes, géographes ou informaticiens, tous étudient, à leur manière et avec leurs outils de tels systèmes. Un système complexe peut se définir simplement comme un ensemble d'entités, éventuellement hétérogènes, en interaction, plongées dans un environnement. Les entités ont des interactions localisées entre elles et avec l'environnement. Le nombre et la nature des interactions ainsi que le comportement des entités sont sujets à évolution dans le temps, ils sont « dynamiques ». Des interactions émergent une ou plusieurs propriétés globales qui ne peuvent être déduites de la simple analyse des entités constituant le système. Ces propriétés ont souvent la capacité de rétroagir sur les entités du système lui-même.

Trois points de vue différents peuvent être adoptés pour l'étude de ces systèmes.

- Point de vue *structurel*. Les éléments constitutifs du système, entités et interactions, sont au centre de l'étude. Les caractéristiques quantifiables telles que le nombre, la taille, la nature ou la durée sont analysés.
- Point de vue *fonctionnel*. L'étude porte principalement sur le comportement global du système en général et des entités en particulier. La forme et les conditions des interactions avec l'environnement sont également visées.

– Point de vue *dynamique*. L'analyse concerne essentiellement la trajectoire du système. L'étude détermine par exemple si, à un instant donné, le système s'adapte à une perturbation, s'il croît, ou s'il est en position d'équilibre.

Parmi les multiples ambitions des recherches menées dans ce cadre, l'une des première est de comprendre le fonctionnement des systèmes observés. De cette compréhension on espère estimer les futurs états du système. Enfin, cette compréhension peut servir de source d'inspiration pour concevoir de nouveaux systèmes, artificiels cette fois, conservant certaines des propriétés des systèmes originaux.

La démarche scientifique se focalise d'abord sur la modélisation des interactions et des entités d'un point de vue structurel afin de produire des modèles à base d'entités qui schématisent le comportement des véritables entités du système. La démarche est itérative, les modèles produits sont simulés et analysés. Les résultats produits sont ensuite reconsidérés dans une nouvelle phase de modélisation.

La mise en œuvre de la modélisation de tels systèmes pourrait s'appuyer sur des objets bien connus et maîtrisés que sont les graphes avec son cortège de résultats théoriques et d'algorithmes performants. En effet, il existe une analogie simple entre les systèmes complexes définis ici et les graphes. Les entités d'un système sont associés aux nœuds d'un graphe et les interactions entre entités sont associées aux arêtes. Cependant, les graphes, tels qu'ils sont communément manipulés sont des structures discrètes dont la structure, la topologie, est statique, or, les systèmes complexes sont, par essence, dynamiques. Ils évoluent dans le temps. L'un des objectifs de ce travail est d'ajouter à l'objet graphe, une notion de temporalité qui lui permette de prendre pleinement en compte la nature des systèmes formés d'entités en interactions en général et des systèmes complexes en particulier. Ce sont donc les *graphes dynamiques* que nous proposons d'étudier.

FIGURE 1.1: *Quelea Flock* par Alastair Rae, license (cc-by-sa-2.0).

Considérons un exemple de système complexe naturel, celui que forment les oiseaux d'une nuée. Chaque individu dans la nuée est unique et indépendant. Il a des interactions avec son environnement (vent, obstacles, *etc.*). Il a également des interactions avec les autres individus de son voisinage. Son but en cherchant à voler dans la nuée est probablement de garantir sa sécurité. Pour cela il maintient une certaine distance avec ses voisins, sans trop s'en éloigner, il ajuste aussi sa vitesse et sa direction par rapport à celles de ses congénères. Chaque individu à sensiblement le même comportement et un voisinage différent qui correspond aux congénères proches de lui. La nuée que l'on peut observer possède un comportement, elle évolue tout en restant cohérente. Elle semble posséder sa propre existence or elle n'est que le résultat d'interactions et de comportements simples et localisés entre les individus. On observe effectivement un

comportement de cohérence mais on peut aussi lui attribuer d'autres caractéristiques. Si un prédateur surgit, les oiseaux en fuyant risquent de ne plus maintenir l'organisation, la nuée va par exemple se diviser en deux. Ensuite, après l'agitation, il y a de fortes chances pour que la nuée se reforme. On dit que ce phénomène est un cas d'*auto-organisation*. On peut parler de *robustesse* quand un système est capable de s'adapter aux variations de son environnement. De même si la nuée ne se reforme pas et reste divisée en deux parties, chaque partie va continuer à exister et à conserver des propriétés similaires, c'est une preuve de *flexibilité*.

À partir de l'observation de ce système complexe on peut par exemple produire un modèle qui tente de mimer le comportement des oiseaux et reproduire le phénomène de nuées. C. REYNOLDS [Rey87] à proposé un modèle très largement utilisé en infographie, les *Boids*, pour réaliser des mouvements de foules, des bancs de poissons, et des nuées d'oiseaux artificiels. Mais l'observation des nuées, et d'autres phénomènes collectifs reposant sur des comportements localisés et individuels, a aussi inspiré certaines techniques heuristiques pour l'optimisation de problèmes combinatoires : les algorithmes fourmis décrits dans l'ouvrage de BONABEAU, M. DORIGO et THÉRAULAZ [BDT99] et l'optimisation par essaims de particules (*Particle Swarm Optimization*) proposée par R. EBERHART et J. KENNEDY [EK95], deux techniques qui rentrent aujourd'hui dans la catégorie plus large des méthodes dites d'*intelligence collective*.

L'étude, la modélisation et le traitement des systèmes complexes par des méthodes d'intelligence collective, est le but que s'est fixé l'équipe *RI2C* (Réseaux d'Interaction et Intelligence Collective) du Laboratoire LITIS (Laboratoire d'Informatique, de Traitement de L'information et des Systèmes). Les domaines d'applications de cette étude sont variés : modélisation de flux de populations, épidémiologie, génomique, distribution adaptative de charge, réseaux mobiles *ad hoc*. Ce dernier domaine d'application a servi de fil rouge au déroulement de cette thèse.

1.2 Les réseaux mobiles *ad hoc*

Conjointement au développement d'Internet durant la dernière décennie, l'avènement des normes *IEEE802.11a/b/g*, et *bluetooth*, et des interfaces matérielles associées ont permis le développement de nouveaux équipements mobiles qui héritent à la fois de l'ordinateur portable, du téléphone mobile, des consoles de jeux ou des lecteurs multimédia. Ces nouveaux terminaux partagent tous deux points communs : la mobilité et des moyens variés de communication sans fil.

En outre, certaines de ces technologies radio offrent des possibilités nouvelles de communication, elles permettent notamment de s'affranchir des infrastructures existantes et de mettre en œuvre une communication dite *ad hoc*[1].

1. Locution latine qui peut être traduite par adéquat ou appropriée. Dans le contexte des réseaux sans fil, elle signifie plutôt « créé pour l'occasion ».

Cette avancée technologique s'accompagne d'un besoin important pour de nouvelles connaissances théoriques et de nouvelles approches algorithmiques. Si celles-ci sont largement éprouvées pour les graphes statiques, modèles naturels des réseaux infrastructurés, il n'en est pas de même des graphes dynamiques qui modélisent ces nouveaux réseaux, qui sont à la fois mobiles, imprévisibles et dont le fonctionnement est par essence décentralisé.

Ces équipements mobiles, souvent nommés stations, peuvent être des téléphones, de petits ordinateurs, des *PDA*[2] (ou assistants personnels numériques) ou encore des capteurs. Ces stations sont équipées de moyens de communication radio de normes différentes : *Wifi*[3], *2G*[4], *3G*[5] ou encore *Bluetooth*. Certains de ces moyens de communication permettent de s'affranchir des infrastructures existantes et de communiquer selon un mode pair-à-pair. Dans ces conditions ce sont les stations elles-mêmes qui assurent le transit de la communication dans le réseau qu'elles constituent. Ce réseau est dit *ad hoc* puisqu'il est créé localement pour un besoin précis et disparaîtra lorsque le besoin de communication sera terminé.

Outre l'absence d'infrastructure, c'est aussi la volatilité et la mobilité des machines qui caractérisent ce genre de réseau. En effet, les stations considérées sont souvent des machines de petite dimension, transportées par leur propriétaire ou encore des capteurs embarqués dans de plus grosses installations elles aussi mobiles. Le voisinage de ces machines évolue au cours de leur mobilité. La volatilité, quant à elle, représente le fait que ces stations ne sont pas continuellement en train de communiquer. Elles peuvent être éteintes ou allumées à tout moment, se révélant ainsi à leur voisinage. Au niveau du réseau que constituent les stations, la volatilité et la mobilité s'expriment en terme de dynamique et on parle alors de réseau mobile *ad hoc* ou de *MANET* (*Mobile Ad hoc Network*).

Puisque les *MANETs* sont des réseaux, sans infrastructure, qui résultent des communications entre différentes stations, ils ne sont pas identifiables simplement et ne peuvent être considérés de manière centralisée. L'absence d'infrastructure ne permet pas d'offrir simplement les mêmes services que les réseaux classiques. Prenons l'exemple d'un *MANET* où deux stations qui n'appartiennent pas au voisinage l'une de l'autre cherchent à communiquer. Elles doivent s'en remettre aux autres stations pour faire le lien entre elles et assurer le transit de l'information via un chemin composé de stations s'échangeant l'information. Cela implique la mise en place de mécanismes de découverte de voisinage et de routage locaux. La dynamique du réseau doit également être prise en compte, et les chemins construits dans ce réseau nécessitent de continuelles

2. *Personal Digital Assistant*

3. Le *Wifi* (contraction de *Wireless Fidelity*) était le nom de la certification délivrée aux dispositifs répondant à la norme IEEE 802.11. Maintenant le terme désigne la norme elle-même.

4. *2G* représente les normes de téléphonie mobile de seconde génération comme le *GSM* (*Global System for Mobile Communications*).

5. *3G* représente l'ensemble des normes de communication de troisième génération. En Europe, c'est la norme *UMTS* (*Universal Mobile Telecommunications System*) qui prévaut.

mises à jour. Cela suppose également que les algorithmes présentent des propriétés de robustesse et de flexibilité leur permettant de s'adapter à la fois à l'apparition et à la disparition de stations et des liens de communication associées.

1.3 Objectifs

La thèse que nous défendons est celle des graphes dynamiques. Elle représente l'une des meilleures solutions pour modéliser des systèmes à base d'entités en interactions. Il nous semble donc essentiel de les définir de manière claire et explicite ainsi que de proposer un ensemble de métriques afférentes. Cette formulation passe par l'étude de différentes définitions proposées dans la littérature. Cette définition ainsi que la description des métriques associées se justifient d'autant plus que l'étude des modèles existants ne permet pas de mettre en avant un modèle incluant toutes les fonctionnalités des autres. La partie I de cette thèse leur est dédiée.

Nous nous intéressons également à la résolution de problèmes particuliers intervenant au sein de systèmes formés d'entités en interaction que nous considérons être modélisables par des graphes dynamiques. La dynamique et l'incertitude caractérisent ces problèmes. Ainsi les méthodes de traitement qui leur sont associées doivent être, tout à la fois, robustes et flexibles, deux propriétés très souvent rencontrées dans les systèmes complexes naturels qui ont inspiré le principe à la base du fonctionnement des colonies de fourmis artificielles. Dans la partie II de ce document, nous exposons un modèle de résolution général décentralisé pour les problèmes d'optimisation au sein des graphes dynamiques.

Nous savons que les réseaux mobiles *ad hoc*, qui font l'objet d'études dans notre équipe, sont naturellement modélisables à l'aide de graphes dynamiques. Nous savons également que les problèmes dont souffrent les *MANETs*, tels que le routage, ne sont plus solubles de la manière dont ils l'étaient dans le cadre des réseaux infrastructurés. Dans ce contexte décentralisé et dynamique qui les caractérisent, notre point de vue est que ce sont les méthodes de résolutions dédiées aux graphes dynamiques qui offrent les meilleures chances de résolution. La partie III s'attache à la résolution de problèmes dans les *MANETs*. Ces méthodes, relevant tout à la fois du domaine de l'algorithmique distribuée et du domaine de l'intelligence collective sont adaptées aux graphes dynamiques.

Première partie

Graphes dynamiques

INTRODUCTION

Dans [Ber05] H. BERSINI propose une définition de graphe dynamique en s'appuyant sur l'étude de différents systèmes en interactions dans des domaines scientifiques variés. Il s'appuie sur un vocabulaire particulier. Pour lui, la notion de graphe correspond à une structure mathématique figée et reliée à la théorie des graphes. Selon lui toujours, les graphes ne permettent pas de modéliser les entités et les interactions qui existent dans les différents domaines que sont par exemple la physique, la sociologie, la chimie, ou la biologie. Il leur manque en effet un paramètre fondamental qu'est le temps, et la possibilité d'évoluer dans ce temps. H. BERSINI nomme ces graphes inscrits dans le temps des « réseaux ». La possibilité d'évoluer est définie comme étant la dynamique des réseaux. Une distinction nette est faite entre deux types de dynamiques. D'abord il définit la « dynamique simple » ou « par défaut », qui caractérise dans le temps l'évolution des valuations des composants des réseaux que sont les nœuds et les arêtes. Ensuite, il considère la dynamique structurelle des réseaux qui consiste en l'ajout et la suppression de nœuds et d'arêtes. Elle est nommée « métadynamique ». L'une des justifications de cette distinction entre la dynamique, qui concerne les changements de valuations des éléments, et la métadynamique, qui concerne les modifications structurelles du réseau, est que leur évolution dans le temps est souvent différente dans les exemples étudiés. Les événements de la dynamique se produiraient plus souvent que les événements de la métadynamique. Notons également que la dynamique est souvent définie de manière locale, sur chaque élément du réseau à l'aide d'« équations dynamiques » qui régissent l'évolution des valuations des éléments.

Suite à la présentation de ce modèle, l'auteur note deux remarques importantes. La première est que la métadynamique, en créant et en supprimant des éléments, peut entraîner la modification d'équations dynamiques sur les éléments existants, et entraîne la définition de nouvelles équations dynamiques sur les nouveaux éléments. Métadynamique implique donc dynamique. Une autre remarque fondamentale est que les équations de la dynamique sont souvent à l'origine de la création de nouveaux éléments. La dynamique est donc à son tour responsable de la création d'événements métadynamiques. L'auteur illustre son propos avec l'exemple de réactions chimiques. Des molécules modélisées par des nœuds dans un réseau possèdent des interactions (des liens dans le réseau) dont la dynamique est définie par des « équations dynamiques ». L'appli-

cation de ces équations dynamiques peut provoquer la création de nouvelles molécules qui sont autant de nœuds dans le réseau. Cet exemple illustre donc le fonctionnement d'équations dynamiques qui provoquent des événements métadynamiques.

Notre intérêt le plus général dans cette thèse, et qui est commun à toute l'équipe RI_2C (Réseaux d'Interaction et Intelligence Collective), est celui de l'étude des systèmes complexes. Comme nous l'avons vu, notre propos est de modéliser ces systèmes à l'aide de graphes dynamiques. Si certains modèles existants permettent de prendre en compte la dynamique structurelle des graphes, la dynamique du graphe, elle, est souvent vue comme une collection d'événements appliqués sur le graphe. La raison qui a poussé à la production de ces événements n'est pas connue du modèle. Nous nous intéressons ici à la définition et à la formalisation des règles qui sont à l'origine de l'évolution d'un graphe dynamique. La définition de H. BERSINI et sa notion d'« équation dynamique » vont dans le sens de nos préoccupations. Nous cherchons ici à formaliser un modèle de graphe où toute la dynamique (métadynamique et dynamique simple) est caractérisée. La description de cette dynamique peut être faite d'un point de vue global, à l'échelle du graphe, ou local, à l'échelle des entités du graphe. C'est donc dans cet objectif que nous proposons dans cette partie un modèle intégrant la notion de « processus d'évolution ».

Dans le premier chapitre de cette partie différents modèles de graphes dynamiques proposés dans la littérature sont présentés. La plupart des modèles de graphes dynamiques présentés sont le résultat d'une nécessité applicative. Les travaux qui proposent ces modèles le font pour modéliser et résoudre leurs problèmes spécifiques. Le formalisme de ces modèles est donc souvent dépendant de l'application traitée. Rares sont les contributions qui traitent de graphes dynamiques en tant que sujet d'étude à part entière. Dans la majorité des cas ces modèles sont des outils au service de problèmes applicatifs particuliers.

Nous nous attachons ici à présenter les différents modèles en les détachant le plus possible de leur contexte applicatif, afin de faciliter leur formalisation, leur analyse, ainsi que la comparaison des différents modèles entre eux. Cette analyse est menée dans le chapitre 2.

À la lumière de cette analyse, le chapitre 3 propose un formalisme précis incluant la délicate notion du temps ainsi que celle du processus d'évolution du système observé. Des notions et des métriques adaptées aux graphes dynamiques sont également proposées.

CHAPITRE 2

ÉTAT DE L'ART

Dans la littérature, le terme de graphe dynamique n'est pas utilisé de manière consensuelle. D'une part, il est souvent fait mention de *graphe dynamique* pour des travaux qui s'apparentent aux *Complex Networks* (« réseaux complexes » ou « graphes de terrain » [1]), aux graphes évolutifs, aux *Space-Time Networks*, mais également relevant de domaines aussi variés que l'écologie, la biologie ou les sciences sociales pour l'ensemble desquels on parlera plus fréquemment de réseaux d'interactions. D'autre part, la notion même de

1. Terme proposé par C. MAGNIEN et M. LATAPY pour traduire le terme *Complex Networks*.

graphe dynamique n'est souvent définie que par défaut, pour nommer un objet qui ne constitue pas à proprement parler le sujet central d'un travail ou d'une étude mais simplement le modèle servant de support implicite à l'étude d'un système particulier. Ainsi, il existe un corpus assez important de travaux se rapportant à des méthodes de réoptimisation dans un graphe lorsque celui-ci varie [FMSN96, RR96, DI07, DI06]. Mais, dans ce cas, la dynamique du graphe est réduite à une suite de modifications sur un graphe statique, et chaque graphe appartenant à cette liste est traité comme un nouveau graphe dans lequel subsistent quelques traces du traitement du précédent.

La notion de graphe dynamique est centrale dans le travail décrit dans ce document, c'est pourquoi il nous a semblé important de passer en revue, sans prétendre à une quelconque exhaustivité, quelques uns parmi les différents modèles qualifiés de graphes dynamiques ou traités comme tels par leurs auteurs. Bien que ces travaux relèvent de différentes thématiques scientifiques, cette étude tente de faire abstraction du contexte applicatif, lorsque cela est possible, pour ne retenir que les éléments essentiels de ces graphes. La seconde partie de ce chapitre propose une analyse des modèles présentés. Enfin la troisième section du chapitre montre les relations qui peuvent exister entre ces modèles.

2.1 Quelques modèles de graphes dynamiques

Dans cette section, nous décrivons quelques modèles de graphes dynamiques parmi les principaux afin de mettre en évidence les éléments qui les caractérisent. Nous montrons notamment qu'il n'existe actuellement aucun formalisme qui soit suffisamment complet pour recouvrir l'ensemble des éléments qui les caractérisent.

2.1.1 Réseaux Complexes

Les réseaux complexes[2] forment une très vaste famille de graphes pour lesquels le terme de dynamique prend tout son sens. Selon le périmètre définit par S. BOCCALETTI et ses collègues [BLM+06], on y retrouve, pêle-mêle, les graphes du Web, les réseaux d'interactions protéiques, les réseaux sociaux, etc. Les systèmes formés d'un grand nombre d'entités dynamiques fortement interconnectées sont à la base de ces réseaux. Ils relèvent donc naturellement de domaines aussi variés que la biologie, la chimie, la technologie, la physique ou les sciences sociales. Cependant, bien que l'imposant travail de S. BOCCALETTI traite de graphes et de leur rapport à la dynamique, nulle définition explicite de la notion de graphe dynamique n'est proposée. Parmi le vaste ensemble des réseaux complexes, pour les besoins de notre exposé, nous nous restreindrons aux graphes du Web.

Le graphe du Web est un graphe dont les sommets correspondent aux documents accessibles sur Internet et reliés entre eux par des liens hypertextes, qui eux corres-

2. *Complex Networks*

pondent aux arêtes du graphe. Chaque sommet de ce graphe est donc uniquement identifié par une *URL*. Un tel graphe fait partie de la famille des réseaux possédant un grand nombre d'éléments et des topologies non triviales. Les graphes qui caractérisent ces réseaux ne sont ni réguliers ni aléatoires, ils possèdent des propriétés bien particulières en terme de diamètre, de répartition des degrés, ou de densité.

Les recherches relatives aux réseaux complexes, dans leur majorité, peuvent être classées en deux catégories. D'une part, on note les travaux qui explorent des instances réelles de réseaux complexes (tel que le *World Wide Web*) afin d'en extraire des graphes sur lesquels des propriétés sont étudiées. D'autre part, on note les travaux dont la vocation est de construire des graphes synthétiques (sans existence réelle *a priori*) reproduisant certaines des propriétés desdits réseaux.

Si l'on examine en particulier le graphe du Web, mais ceci reste valable pour les autres réseaux complexes, ce type de graphe peut prétendre à la mention dynamique. En effet, sa structure évolue, des pages et des liens hypertextes apparaissent tous les jours. Cette dynamique structurelle est néanmoins lente en comparaison de la dynamique que l'on peut observer dans le réseau d'interaction d'une application de type *Boids* [Rey87] par exemple. La dynamique peut également être observée dans les mécanismes d'exploration du graphe comme nous allons le voir ci-dessous. Enfin les modèles aléatoires de construction de graphes présentent également des mécanismes dynamiques de création comme nous le verrons dans la dernière partie de cette section dédiée au graphe du Web.

Exploration d'instances réelles de réseaux complexes

Le *World Wide Web* représente un ensemble de documents (souvent nommés pages Web) accessibles depuis Internet et liés entre eux par des liens hypertextes. Le graphe du Web qui représente cette structure ne peut être obtenu que par un parcours itératif des éléments du réseau réel. Pour parvenir à une représentation de ce graphe on utilise des *crawlers* ou *web spiders* qui parcourent les pages et leurs liens. Ce mécanisme fournit une vue instantanée plus ou moins fidèle du Web. La qualité de cette vue dépend à la fois de la manière dont l'exploration est faite, du nombre de pages visitées, et de la fréquence à laquelle les pages sont revisitées. Les *crawlers* construisent donc une représentation statique et globale. La dynamique du graphe du Web est souvent ignorée comme mécanisme, mais simplement considérée pour la mise à jour du graphe.

Les représentations construites ont permis de mettre en évidence certaines propriétés remarquables. Cependant, la taille de ce graphe, ainsi que sa dynamique entrainent parfois quelques désaccords concernant ces propriétés. Ainsi, l'idée que le Web est un graphe « petit monde » émise en 1999 dans [Ada99] fut réfutée un an plus tard par A. Z. BRODER *et al.* [BKM+00]. En revanche, il semble aujourd'hui unanimement reconnu que la répartition des degrés des sommets suit une loi de puissance ce qui correspond bien à une propriété observable dans les graphes de terrain.

Beaucoup de publications ont porté sur l'allure générale du graphe. Ainsi, le travail de A. Z. BRODER *et al.* [BKM+00] présente une organisation structurelle globale du Web dont l'allure générale prend la forme d'un nœud papillon (figure 2.1). Cette thèse peut néanmoins essuyer la critique que c'est la manière dont l'exploration fut menée qui confère au graphe cette allure et que cette étude ne tient pas compte du *Web invisible*[3].

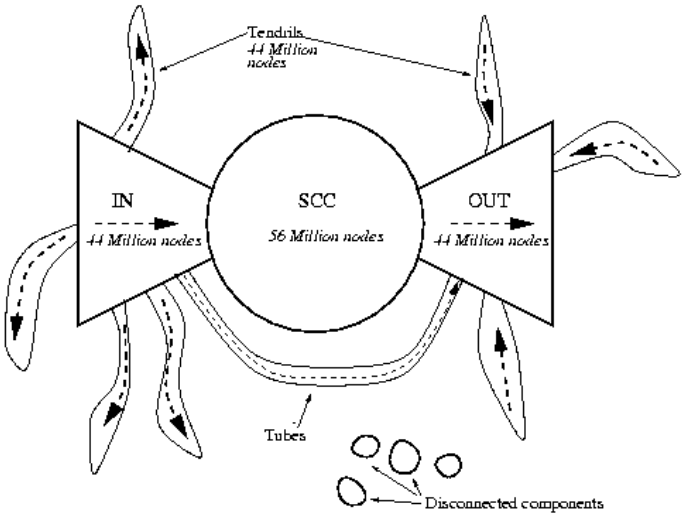


FIGURE 2.1: L'allure générale du Web selon A. Z. BRODER *et al.* en forme de nœud papillon. Image extraite de [BKM+00].

L'exploration des réseaux complexes se fait de manière itérative, des éléments sont ajoutés de manière successive. Le graphe construit à partir de ces explorations évolue donc au fil de l'exploration ; on peut le considérer comme un graphe dynamique et on qualifie cette évolution en terme de dynamique de construction. Certains travaux exploitent ce phénomène, ainsi, dans [LM06], M. LATAPY et C. MAGNIEN proposent d'étudier les graphes pendant leur création (pendant l'exploration du phénomène naturel) et d'observer l'évolution de certaines métriques dans ces graphes. Leur but est d'observer une certaine régularité ou une stabilité dans les observations faites et ainsi déterminer expérimentalement le volume suffisant d'information (la taille de graphe suffisante) pour estimer que les mesures sont représentatives et pertinentes pour le phénomène observé.

3. Le Web invisible est constitué des pages ne pouvant pas être indexées par les moteurs de recherche comme les résultats de requêtes de bases de données.

Construction de graphes synthétiques

La difficulté d'observation d'instances réelles de réseaux complexes, en particulier de réseaux qui relèvent de la biologie ou de la chimie pour lesquels la dynamique est très importante, mais également certains réseaux sociaux ou technologiques de tailles imposantes et fortement décentralisés par nature, a incité de nombreuses équipes de recherche à investir le champ de la création de réseaux complexes synthétiques.

Pour le graphe du Web, ces travaux, pour une part importante d'entre eux, se sont focalisés sur la conception de modèles de construction de graphes aléatoires qui reproduisent les propriétés observées, par l'intermédiaire des *crawlers*, dans les instances réelles. Tous domaines confondus, les premiers travaux en la matière datent des années 60 avec P. ERDŌS et A. RÉNYI [ER61] mais ne sont pas suffisamment représentatifs des propriétés du Web. Le modèle par « attachement préférentiel » de A. L. BARABÁSI et R. ALBERT [BA99] et le modèle par « recopie » de R. KUMAR *et al.* [KRS00] construisent des graphes aléatoires ayant des répartitions des degrés en loi de puissance.

Le graphe du Web est un graphe dynamique où des pages et des liens hypertextes apparaissent et disparaissent à chaque instant. Cette dynamique est néanmoins relative et peut sembler négligeable dans certains cas. Les modèles de conceptions précédents n'incluent pas cette dynamique. En effet, leur construction est itérative, ils construisent des sommets et des liens, ils sont donc dynamiques, mais pour la plupart, ils ne prévoient pas la suppression d'éléments.

Dans [DC07] les auteurs, N. DEO et A. CAMI cherchent avant tout à reproduire l'ensemble des modifications dynamiques du Web dans leur modèle. Le mode de construction contient donc des mécanismes d'ajout mais aussi de suppression de sommets et d'arcs. L'algorithme général est très simple. Le graphe est construit de manière itérative, étape par étape :

- à chaque étape, deux possibilités, 1) un nœud est ajouté avec un lien pour le relier au graphe ou 2) un nœud est retiré ainsi que ses arcs incidents ;
- les ajouts et suppressions de nœuds dépendent d'une loi de probabilité.

L'originalité du modèle de N. DEO et A. CAMI réside dans la proposition d'un mécanisme « préférentiel » pour les ajouts et pour les suppressions de sommets.

Ce modèle propose donc la création de graphes dynamiques aléatoires possédant certaines caractéristiques observées dans le graphe du Web et en particulier celle de mimer sa dynamique.

2.1.2 Graphes évolutifs

Dans [Fer02] puis dans [BXFJ03] A. FERREIRA et ses collègues proposent un modèle qui mémorise les différents événements modifiant un graphe au cours de son évolution. Le résultat est un graphe statique contenant toutes les arêtes et tous les sommets ayant existé pendant l'évolution du graphe observé. Les arêtes sont étiquetées par des informations relatives à des événements d'évolution.

Définition 1 (GRAPHE ÉVOLUTIF)
Soit $G = (V, E)$ un graphe orienté avec V un ensemble de sommets et E un ensemble d'arcs dont les extrémités appartiennent à V. Soit $S_G = \{G_1, G_2, \ldots, G_T\}$, un ensemble de sous graphes de G. Le système $\mathcal{G}_E = (G, S_G)$ est appelé « graphe évolutif ».

Cette définition correspond à un modèle de graphe permettant l'agrégation d'une liste de sous-graphes statiques. Chaque graphe représente l'état du réseau étudié à un moment donné. Chaque sous-graphe peut être composé d'un nombre quelconque de sommets et d'arêtes. Le modèle gère la dynamique topologique du réseau (ajout ou suppression de sommets et d'arêtes) et il permet aussi de représenter les modifications de valuations sur les sommets et les arêtes. La figure 2.2 montre l'exemple de graphes statiques qui représentent les différentes étapes de l'évolution d'un graphe dynamique.

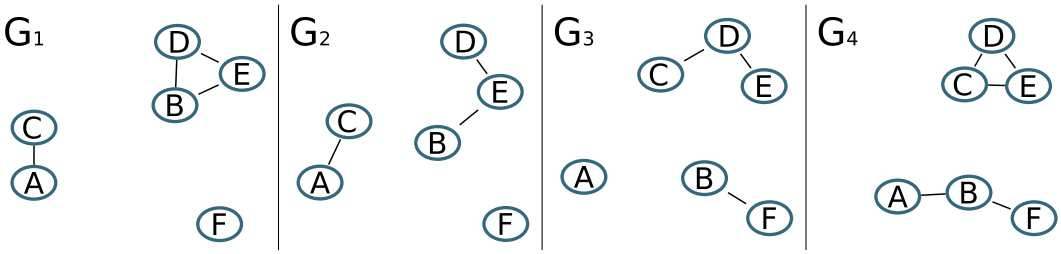


FIGURE 2.2: Exemple de quatre graphes statiques représentant les étapes de l'évolution d'un graphe dynamique.

Le graphe évolutif construit à partir de cette liste de sous-graphes G_i contient donc toutes les arêtes et tous les sommets des sous-graphes considérés. Les arêtes sont étiquetées en fonction des sous-graphes dans lesquels elles apparaissent. La figure 2.3 représente le graphe évolutif correspondant aux étapes d'évolution de la figure 2.2.

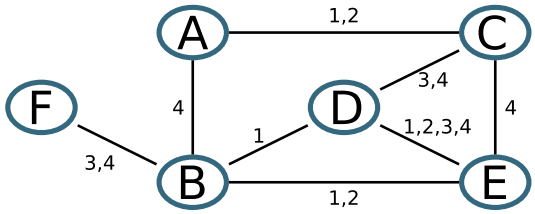


FIGURE 2.3: Le graphe évolutif construit à partir des graphes de la figure 2.2. La notation utilisée value les arêtes du graphe par les indices des graphes dans lesquels elles apparaissent.

Gestion du temps

Un graphe évolutif contient la dynamique de l'évolution sous forme d'étapes d'un graphe dynamique. Ces étapes correspondent aux sous-graphes du graphe évolutif. Dans ce modèle aucune notion temporelle n'est encore introduite et la dynamique est celle des étapes. Sans perte de généralité il est possible d'inclure une notion temporelle dans les étapes en affectant une date à chaque sous-graphe du graphe dynamique observé. Ainsi, chaque étape se trouve comprise entre deux dates. Une étape associée à un sous-graphe possède donc une date de début, qui est aussi la fin de l'étape précédente, et une date de fin qui marque le début de l'étape suivante.

La figure 2.4 montre une ligne de temps ainsi que des dates qui définissent les étapes de la vie d'un graphe dynamique. On remarque que les dates ne se suivent pas obligatoirement à intervalles réguliers et que les étapes ne sont donc pas obligatoirement associées à des durées identiques.

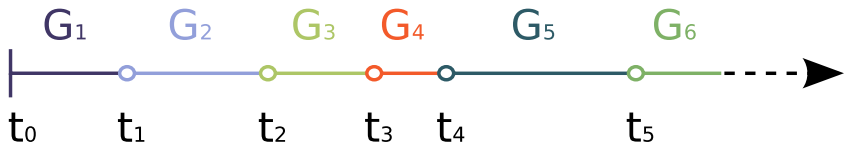


FIGURE 2.4: Ligne du temps dans l'évolution d'un réseau dynamique. Les t_i sont des dates qui correspondent aux changements d'étapes. L'étape i correspond au sous-graphe G_i, elle débute à la date t_{i-1} et se termine à la date t_i ($[t_{i-1}, t_i[$).

Poids des arêtes

Dans [BXFJ03], dans le cadre de l'étude d'algorithmes de plus courts chemins dans les réseaux mobiles *ad hoc*[4], les auteurs définissent un temps de trajet sur chaque arête du graphe évolutif. Ce temps de trajet correspond au temps nécessaire à un paquet pour être transmis entre deux stations. Ce temps de trajet est modélisé dans le graphe avec des poids sur les arêtes, $\zeta(e)$ étant le poids de l'arête e. Cette fonction impose que le poids d'une arête soit constant. Nous verrons dans la section 2.3 page 25 que ce modèle peut être étendu sans être dénaturé en permettant d'associer à une arête un poids dépendant de l'étape avec la fonction $\zeta(e,t)$. Cette proposition est également présentée dans la thèse d'A. CASTEIGTS [Cas07]. En d'autres termes, cette modification permet d'introduire de l'évolution dans la valuation des éléments du graphe.

En résumé le modèle proposé par A. FERREIRA et ses collègues est une construction à traitement différé[5] de l'évolution d'un graphe dynamique. Le graphe construit est cumulatif. Tous les sommets et tous les liens créés pendant la période d'observation du

4. Ces réseaux sont présentés en détail dans le chapitre 6.
5. Le terme *post mortem* et également utiliser pour parler de traitement différé.

système modélisé, apparaissent dans le graphe évolutif. Chaque arête est étiquetée avec les indices des étapes durant lesquelles elle est présente. Ce type de graphe est particulièrement bien adapté pour l'étude différée d'un réseau en utilisant une version adaptée des algorithmes et des outils de la théorie des graphes.

2.1.3 Graphes pour la ré-optimisation

Les problèmes d'optimisation dans les graphes consistent généralement à déterminer une structure, une valeur ou une configuration qui correspond à la minimisation d'une fonction de coût. Dans le contexte d'un graphe dynamique, les techniques de ré-optimisation consistent à déterminer les méthodes qui recalculent un optimum sans recalculer la totalité de la solution.

C'est au problème du plus court chemin dynamique que les méthodes de ré-optimisation s'appliquent le plus souvent. Les travaux les plus connus sont ceux de C. DEMETRESCU et G. F. ITALIANO [DI06] mais aussi de G. RAMALINGAM et T. W. REPS [RR96]. Le but pour ces méthodes est de permettre une mise à jour de la structure (arbre) ou le calcul d'une valeur (plus court chemin), après changement du graphe, qui soit plus rapide qu'un calcul qui ne tiendrait pas compte de la structure (ou de la valeur) courante. D'une manière générale, et sans entrer dans les détails d'implémentation, la plupart de ces méthodes consistent à adapter le célèbre algorithme de E. W. DIJKSTRA [Dij59] aux environnements dynamiques. Les méthodes de ré-optimisation nous intéressent dans le cadre de l'étude des graphes dynamiques car elles manipulent des graphes qualifiés de dynamiques par leurs auteurs.

Définition 2 (GRAPHE POUR LA RÉ-OPTIMISATION)
Soit $G = (V, E)$ un graphe statique composé d'un ensemble de sommets V et d'arêtes E. Soit $\sigma = \{u_1, u_2, \ldots, u_n\}$ un ensemble ordonné d'événements. $u_i \in \sigma$ est un événement du type ajout, suppression ou modification d'une arête. Le graphe G_i est construit à partir du graphe G_{i-1} en lui appliquant l'événement u_i.
Nous noterons $\mathcal{G}_R$ le graphe dynamique pour la ré-optimisation construit de manière itérative à partir de G avec les événements de l'ensemble σ.

Le temps dans les graphes pour la ré-optimisation

Dans les graphes pour la ré-optimisation le temps n'est pas explicite. Ces graphes évoluent de manière discrète, étape après étape. À chaque étape, le graphe courant subit un ensemble de modifications (ajout/suppression/modification d'éléments). À chaque étape correspond une représentation instantanée et statique du graphe dynamique. La suite de graphes statiques constitue le graphe dynamique. Cette suite de graphes ne sert pas à produire un graphe cumulatif comme le font les modèles de A. FERREIRA (page 15) ou de C. Cortes *et al.* (page 19), mais il existe cependant un lien entre ces graphes et les graphes évolutifs, lien qui sera explicité dans la section 2.3. S'il est vrai que l'étape i

donne une représentation de graphe antérieure à l'étape $i+1$, la notion de temps n'apparaît pas dans cette construction. Il n'existe pas de durée associée à une étape.

Concernant l'application de ré-optimisation, ce processus s'exécute à chaque étape de l'évolution du graphe. Le but étant de minimiser le temps nécessaire à la ré-optimisation, la complexité des méthodes est finement analysée. Le temps nécessaire à la mise à jour de la structure (ou de la valeur) est proportionnel aux dimensions du graphe et aux modifications effectuées à chaque étape. Ce temps nécessaire au calcul est totalement découplé de l'évolution du graphe qui est atemporelle. En effet, puisque le temps n'intervient pas dans le modèle de ce graphe dynamique, l'application dispose d'un temps potentiellement non limité avant la prochaine étape.

Contraintes topologiques

Les graphes proposés pour les méthodes de ré-optimisation sont dits dynamiques dans le sens ou ils permettent à chaque étape l'ajout, la suppression et la modification d'une arête dans le graphe. Seules les arêtes du graphe peuvent être modifiées, le nombre de sommets reste constant pendant l'évolution. Dans la pratique, cette contrainte est souvent contournée. Dans le contexte des algorithmes de plus court chemin, il est possible de virtuellement retirer un sommet du graphe en assignant un poids infini à toutes ses arêtes incidentes. Il peut aussi être déconnecté du graphe en supprimant ses arêtes. En effet, un sommet isolé ne peut être considéré pour et par la recherche d'un plus court chemin et le fait de le connecter au reste du graphe le rend présent au regard de l'application de plus court chemin. Le graphe considéré par l'application correspond de fait à la plus grande composante connexe.

Certains algorithmes sont capables de gérer des insertions ainsi que des suppressions d'arêtes, le modèle de graphe associé ainsi que la méthode sont dits « pleinement dynamiques ». Certains ne permettent que des suppressions d'arêtes, on parle alors de graphes « décrémentaux ». Enfin des méthodes ne prennent en charge que les ajouts d'arêtes, on parle alors de graphes et de méthodes « incrémentaux » [6].

2.1.4 Autres modèles de graphes dynamiques

Graphes cumulatifs

Dans [CPV03] les auteurs proposent l'analyse d'un grand réseau de télécommunication et s'intéressent au graphe construit à partir des appels téléphoniques entre abonnés d'une compagnie de téléphone. L'étude porte sur l'activité des communications téléphoniques pour une période de temps prolongée (plusieurs mois). Pour mener à bien cette analyse, ils définissent un modèle de graphe incluant la dimension temporelle en créant une accumulation des événements ayant modifié le graphe des appels pendant une période d'observation donnée.

6. On retrouve ces notions dans les différentes contributions sitées plus haut.

Les auteurs n'ont pas donné de nom à ce modèle, nous prendrons néanmoins la liberté dans ce document de le nommer « graphe cumulatif » pour y faire plus facilement référence. Nous présentons d'abord le modèle dans sa forme originale puis nous présentons des reformulations, proposées par les auteurs eux-mêmes. En supprimant judicieusement certaines informations, ils permettent au graphe de conserver une taille raisonnable dans le contexte de l'application. En outre, ces reformulations nous permettent de faire le lien avec d'autres graphes considérés dans ce chapitre.

Modèle original. Le temps d'observation est discrétisé en intervalles réguliers (un mois, une semaine, un jour, *etc.*). Les informations relatives au réseau de communication sont vues comme des événements datés et pondérés. Ces événements sont des appels téléphoniques, leur date correspond à l'intervalle de temps dans lequel ils apparaissent et leur poids correspond à la durée de la communication dans cet intervalle. Il est sous-entendu que la durée d'un appel est plus courte qu'un intervalle de temps d'observation.

Le modèle de « graphe cumulatif » propose une projection de ces événements sous forme d'un graphe statique contenant une agrégation des informations. Pour construire cette projection, une fenêtre de temps est considérée et les événements dont la date est incluse dans la fenêtre de temps sont agrégés. Dans ce modèle les sommets sont les abonnés et les liens sont les appels et pour chaque lien, le poids correspond à la somme des poids des événements correspondant à ce lien dans l'intervalle de temps considéré. Cette agrégation d'événements produit donc un graphe statique qui contient dans ses éléments les informations relatives à la période d'observation. La définition 3 proposée pour définir un graphe cumulatif n'est pas extraite du travail de CORTES. Nous l'avons construite à partir de la description qu'en ont fait les auteurs.

Définition 3 (GRAPHE CUMULATIF)
Soit $I = \{1, 2, \ldots, i, \ldots, n\}$ *un ensemble d'entiers correspondant à des intervalles de temps de durées égales (un jour, une heure, une seconde,* etc.*). Une étape* i *correspond à l'intervalle de temps* $]t_{i-1}, t_i]$*. Soit* g_i $(i \in I)$ *un ensemble d'événements se produisant à l'étape* i *(c'est-à-dire dans l'intervalle* $]t_{i-1}, t_i]$*).*
$\mathcal{G}_C(t) = g_1 \oplus g_2 \oplus \cdots \oplus g_t$ *est un graphe cumulatif. L'opérateur* $\oplus$ *permet de faire l'union de deux ensembles d'événements correspondant à deux étapes successives, et permet de faire la somme des poids des arêtes quand celles-ci apparaissent dans les deux étapes.*

C'est donc plus dans la datation et la valuation de ses éléments que ce modèle de graphe est « dynamique » que dans l'évolution structurelle de ses composants. Cette évolution structurelle existe mais elle est limitée à la création de sommets et d'arêtes, il n'y a pas de suppression d'éléments.

Reformulation du modèle. Lorsque ce graphe est construit de manière itérative, sa densité augmente irrémédiablement. Les auteurs déplorent ce phénomène d'autant que le contexte applicatif (l'étude d'appels téléphoniques entre abonnés) fait que sa densité peut demeurer relativement faible et peut permettre la manipulation et l'analyse de grands graphes. Pour pallier à ce problème les auteurs proposent de reformuler le modèle.

Une première solution consiste à ne considérer qu'un certain nombre d'étapes (k) précédant la date observée. Ainsi $\mathscr{G}_C(t) = g_{t-k} \oplus g_{t-k+1} \oplus \cdots \oplus g_t$. Le graphe n'est construit que des k derniers ensembles d'événements. Le problème de cette approche est son coût en mémoire puisqu'elle impose, à chaque instant de l'étude du graphe, de posséder en mémoire les k dernières étapes (pour pouvoir faire glisser la « fenêtre » historique).

Les auteurs remarquent de plus qu'avec ce modèle un lien très âgé est considéré de la même manière qu'un lien très récent. Ils proposent donc de pondérer l'importance des liens plus âgés en agissant sur leur poids. Le moyen pour le faire est une construction récursive des graphes dynamiques qui minimise aussi la consommation de mémoire :

$$\mathscr{G}_C(t) = \theta \mathscr{G}_C(t-1) \oplus (1-\theta) g_t \tag{2.1}$$

Où $\theta, (0 \leq \theta < 1)$ permet d'accorder plus ou moins d'importance aux anciens événements.

Cette reformulation ne diminue pas le nombre de liens du graphe, elle permet simplement de pondérer le poids des arcs. Les auteurs proposent ensuite des mécanismes de seuillage afin d'éliminer les liens de poids faible qui sont les plus anciens (*a priori*).

Ils proposent dans un premier temps un seuillage global sur tout le graphe. À chaque étape, toute arête de poids inférieur à ϵ est retirée. Puis ils proposent un seuillage local sur le degré des nœuds. Seuls les k meilleurs arcs sont conservés pour chaque nœud.

En résumé, les auteurs proposent un formalisme qui retranscrit la dynamique d'un réseau de communications téléphoniques dans un graphe statique. Les reformulations permettent de conserver une densité limitée pour le graphe. Ce modèle ne rend pas vraiment compte de la dynamique du réseau et propose finalement la construction de graphes statiques. Cette méthode pourrait sembler hors de notre sujet d'étude néanmoins la construction de ces graphes statiques repose sur une formulation récursive (Équation 2.1). Le graphe $\mathscr{G}_C(t+1)$ est construit à partir de $\mathscr{G}_C(t)$ et la construction étape par étape de ces graphes statiques s'inscrit bel et bien dans un processus dynamique. De plus les éléments du graphe cumulatif contiennent les informations relatives à son évolution temporelle.

Space-Time Networks

Dans [PS97] S. PALLOTTINO et M. G. SCUTELLÁ s'intéressent aux problèmes de durées de trajets dans un graphe valué et de dates de départ pour un trajet allant d'un

sommet source à un sommet destination. Ils considèrent le cas de graphes dont la structure n'évolue pas mais dont les poids des arêtes qui définissent le temps de trajet sur celles-ci est variable dans le temps. Pour résoudre ce problème ils proposent un modèle de graphe spatio-temporel qui agrège les informations sur la dynamique et produit une représentation statique permettant l'utilisation d'outils classiques de la théorie des graphes.

Notons qu'un trajet de s à d est un chemin qui dure dans le temps et tel que la connexion de bout en bout entre s et d n'est pas assurée à chaque étape [7]. Dans le cas de graphes dynamiques, il est possible qu'un trajet ne soit valide que dans le sens source vers destination.

Le modèle de graphe utilisé ici pour résoudre les problèmes de temps de trajets et de dates de départ est un modèle où la dynamique est connue à l'avance, soit parce que le traitement est différé et la dynamique est observée, soit parce que la dynamique est prévisible comme dans les réseaux de communication avec des satellites orbitaux [WDV+97].

Définition 4 (*Space-Time Network*)
Soit $G = (V, E)$ un graphe représentant un réseau avec V l'ensemble des sommets du graphe et E l'ensemble de ses arêtes. $|V| = n$ et $|E| = m$. Soit $[1, q]$ un ensemble d'étapes, soit e et (u, v) deux arêtes ($e, (u, v) \in E, u, v \in V$), et soit $P_e(t) \in \mathbb{N}$ (avec $t \in [1, q]$) le poids de chaque arête pouvant évoluer discrètement en fonction du temps, étape après étape. Le poids $P_e(t)$ de l'arête e correspond au temps de traversée de cette arête. $\mathcal{G}_{ST} = (V_{ST}, E_{ST})$ est le Space-Time Network *de G tel que :*

$$V_{ST} = \{v_i | v \in V, i \in [1, q]\},$$

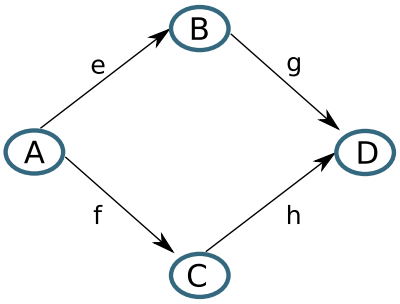
$$E_{ST} = \{(v_{i-1}, v_i) | v \in V, i \in [2, q]\} \cup \{(u_i, v_j) | (u, v) \in E, 1 \leq i \leq j \leq q, i + P_{(u,v)}(i) = j\},$$

$$|V_{ST}| = n \times q,$$

$$|E_{ST}| \leq (n + m) \times q.$$

On remarque d'emblée que la notion d'événement présente dans d'autres définitions ne l'est pas dans le *Space-Time Network*. En effet, une fois développé, ce graphe est purement statique. De fait, ce modèle permet de construire un graphe statique qui contient la dynamique des poids des arêtes du réseau. Pour construire ce graphe à partir de G, le graphe du réseau étudié, il faut dupliquer chaque nœud q fois (pour les q étapes). Les liens entre un sommet à l'étape i et le même sommet à l'étape $i + 1$ permettent de définir des parcours avec des attentes. Les autres liens représentent le poids des arêtes en fonction du temps. Ainsi, si un lien entre u et v à l'étape i vaut $P_{(u,v)}(i) = k$, alors le lien partant de u_i aboutira en v_{i+k}. La figure 2.5 propose un exemple de réseau à topologie statique associé à la table des poids des liens en fonction des étapes. La figure 2.6 correspond au graphe espace temps du réseau de la figure 2.5.

7. Cette notion de trajet sera plus formellement définie page 37.



	1	2	3	4
e	1	2	1	1
f	2	2	2	2
g	0	1	0	1
h	3	2	1	1

FIGURE 2.5: Un réseau à topologie fixe et table des poids des arêtes en fonction des étapes.

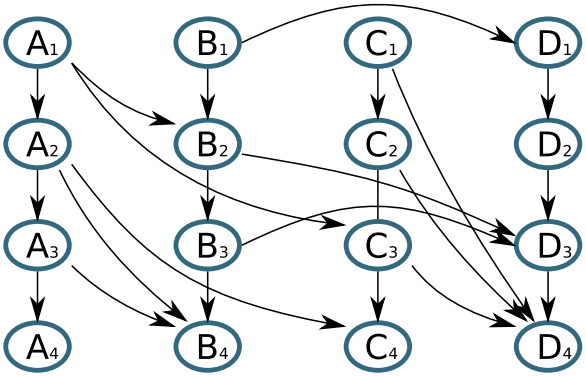


FIGURE 2.6: Le graphe *Space-Time* du graphe de la figure 2.5.

Le modèle proposé construit un graphe statique, mais cette représentation est le fruit de l'analyse et de l'agrégation d'événements dynamiques. C'est donc en cela que nous jugeons ce modèle de graphe comme dynamique. La nature de la dynamique quant à elle ne concerne pas la structure du graphe, aucun nœud, aucune arête n'apparaît ou ne disparaît. Seuls les poids des arêtes sont sujets à modifications dans le temps.

2.2 Analyse des différents modèles

Nous proposons dans cette section d'analyser les différents modèles présentés dans les sections précédentes. Ces modèles sont l'aboutissement de travaux menés dans différentes communautés, dans différents contextes scientifiques et, pour certains, ont été conçus pour des applications bien spécifiques. La comparaison n'est donc pas toujours aisée. On tente ci-dessous de les analyser en fonction de plusieurs critères : les types de modifications applicables aux graphes, le niveau de connaissance *a priori* de l'évolution

et l'expressivité de cette évolution.

2.2.1 Opérations de modification

Les approches étudiées sont classées en fonction de leur propre définition de la dynamique d'un graphe. On dit que la notion de dynamique est faible quand seuls quelques éléments du graphe évoluent. C'est le cas par exemple quand seuls les poids des arêtes évoluent et que la structure du graphe reste inchangée. À l'inverse, la dynamique est forte quand toutes les combinaisons d'ajouts, de suppressions, de modifications d'arêtes et de sommets sont prises en compte. L'énumération suivante ordonne les modèles considérés en fonction de leur dynamique, de la plus faible à la plus forte.

Space-Time Networks. Ce modèle, le plus contraint de tous ne permet aucune évolution topologique, les sommets et les arêtes sont les mêmes du début à la fin de l'évolution. Seuls les poids des arêtes changent.

Graphes cumulatifs. Dans sa définition de base, ce modèle ne permet que l'ajout d'arêtes et de sommets, pas de suppression.

Graphes pour la ré-optimisation. Ce modèle se veut pleinement dynamique dans sa structure, néanmoins cette dynamique ne concerne que les arêtes, le nombre de sommets ne varie pas.

Graphes cumulatifs, dans leur reformulation. Le modèle prévoit un mécanisme de suppression des arcs et des sommets basé sur un seuil sur les poids des arêtes. La suppression d'éléments existe donc mais elle est conditionnée par la valeur des poids des éléments.

Graphes aléatoires dynamiques. Le modèle de construction de graphe du Web prévoit une dynamique totale dans les sommets et les arêtes.

Graphes évolutifs. La dynamique est totale, toutes les combinaisons d'ajouts, de suppressions, de modifications d'arêtes et de sommets sont envisageables.

Des opérations de modifications permises sur les modèles dépendront les applications modélisables. Pour un problème de transport routier entre différentes villes et pour un ensemble de routes connues qui ne changent pas, mais dont la charge est variable et connue au cours du temps, le modèle *Space-Time Network* est tout indiqué. Si certaines routes deviennent impraticables au cours du trajet, alors ce modèle n'est plus adapté, mais le modèle de graphe pour la ré-optimisation convient. Enfin si on souhaite *a posteriori* connaître le meilleur itinéraire qu'il aurait été possible d'emprunter, le modèle de graphe évolutif permet de répondre à cette question.

2.2.2 Connaissance des événements d'évolution

Du point de vue du traitement des graphes dynamiques, les modèles se partagent en deux catégories. Ceux pour lesquels l'évolution du graphe dynamique est entièrement

connue au départ, et ceux pour lesquels les modifications se dévoilent au fur et à mesure.

Les approches étant destinées à l'analyse différée d'un graphe dynamique nécessitent une connaissance complète de son évolution. Lorsque l'évolution du graphe dynamique n'est pas connue *a priori* ; on ne sait pas à l'étape i quels sont les événements de l'étape $i+1$, seules les méthodes opérant sur le graphe courant et éventuellement l'historique du graphe courant sont applicables. Cette distinction permet de classer les modèles : *Space-Time Networks*, graphes évolutifs d'une part et graphes aléatoires dynamiques, graphes pour la ré-optimisation et graphes cumulatifs d'autre part.

Considérons l'exemple d'un dispositif de navigation auquel il est demandé de planifier un itinéraire. Celui-ci, à partir d'une représentation statique de l'environnement calcule un plus court chemin dans un graphe. Si lors du trajet nous empruntons une route qui ne fait pas partie de l'itinéraire et que nous nous éloignons légèrement de la route prédéfinie, le navigateur va adapter la route sans recalculer tout l'itinéraire. Pour modéliser ce problème un graphe pour la ré-optimisation peut être utilisé. Dans le même ordre d'idée, le site `www.sytadin.fr` propose un état global du trafic routier en temps réel. Pour mettre au point un algorithme de prévision de l'évolution du trafic, qui permette de rendre compte des tendances de fréquentation des différentes routes, le problème peut être modélisé par un graphe cumulatif. Si ce site cherche à effectuer une analyse des événements de la journée, alors l'utilisation d'un graphe évolutif s'impose.

2.2.3 Règles d'évolution

Il existe un troisième critère d'analyse, de notre point de vue majeur, celui qui permet de distinguer les modèles en fonction de la façon dont sont générés les événements d'évolution. Soit le processus de génération des événements est décrit dans le modèle, comme cela est le cas pour le processus d'attachement préférentiel proposé par BARABÁSI et ses collègues, soit la suite des événements d'évolution n'est pas le produit d'un processus de génération explicite, que ces événements soient le résultat de la dynamique d'une application ou non. Dans les modèles de graphes dynamiques aléatoires, le processus de génération des événements est inclus dans le modèle de graphe. Dans tous les autres modèles étudiés ici la génération des événements n'est simplement pas décrite.

2.3 Relations entre modèles

Sur la base de l'analyse précédente, il est possible de dégager certaines relations entre les différents modèles de graphes. On note cependant que le modèle des graphes dynamiques aléatoires ne peut pas être relié à l'un des autres modèles car son formalisme inclut le processus de génération des événements d'évolution et qu'il est, parmi l'ensemble des modèles répertoriés le seul à proposer explicitement un tel processus.

2.3.1 Graphes pour la ré-optimisation ⊂ graphes cumulatifs

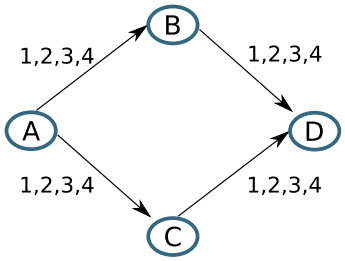
Il existe une surjection entre l'ensemble des graphes cumulatifs d'une part et l'ensemble des graphes pour la ré-optimisation d'autre part. En effet les graphes pour la ré-optimisation ne considèrent que l'instance courante du graphe dynamique et il est possible de réduire un graphe cumulatif à l'instance courante en fixant $\theta = 0$ dans l'équation 2.1 page 21. En conséquence tout algorithme résolvant un problème pour un graphe cumulatif peut résoudre ce même problème sur un graphe pour la ré-optimisation.

2.3.2 *Space-Time Network* ⊂ graphes évolutifs

Il existe également une surjection entre l'ensemble des graphes évolutifs d'une part et l'ensemble des *Space-Time Networks* d'autre part. En effet, le modèle *Space-Time Network* transforme un graphe à topologie statique et à poids variable dans le temps en un graphe statique possédant toutes les informations relatives aux variations de poids en dupliquant les sommets, donc, comme cela a été noté au paragraphe 2.2.2 l'ensemble des événements d'évolution est connu *a priori*. Par conséquent, tout *Space-Time Network* peut être modélisé par un graphe évolutif. La figure 2.7 montre le graphe évolutif correspondant à l'exemple de *Space-Time Network* de la Figure 2.5 page 2.5. Le réseau est statique et toutes les arêtes apparaissent à chaque étape. La topologie ne change pas, seuls les poids des arêtes évoluent avec la fonction $\zeta(e,t)$.

La conséquence principale de cette inclusion est que tout algorithme résolvant un problème sur un graphe évolutif le résoudra également sur un *Space-Time Network*.

Notons par exemple l'algorithme de calcul de plus court chemin proposé dans [BXFJ03] pour des graphes évolutifs. Comme le *Space-Time Network* peut être transformé en graphe évolutif, résoudre le problème pour un graphe évolutif permet de le résoudre pour un *Space-Time Network* [PS97].



	1	2	3	4
e	1	2	1	1
f	2	2	2	2
g	0	1	0	1
h	3	2	1	1

FIGURE 2.7: Le graphe évolutif correspondant au réseau de la figure 2.5.

La figure 2.8 montre les inclusions entre modèles. On voit que les graphes dynamiques aléatoires ne sont inclus dans aucun autre modèle du fait de leur définition des règles d'évolution.

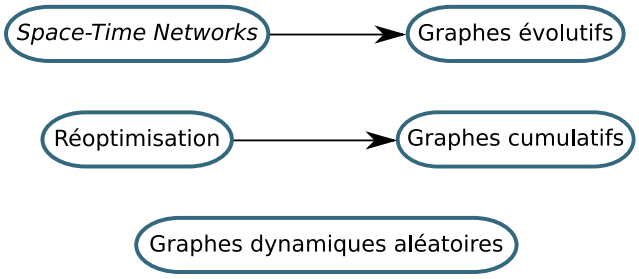


FIGURE 2.8: Inclusions entre les différents modèles de graphes dynamiques.

2.3.3 Analyse différée : l'universalité des graphes évolutifs

Quel que soit le modèle considéré, la totalité de l'évolution du graphe dynamique peut se ramener à la donnée d'un graphe initial et d'un ensemble d'événements impliquant les sommets, les arêtes et les valuations de ces éléments. Or les graphes évolutifs permettent de représenter de manière agrégée cette évolution. Cependant dans le modèle de graphe évolutif tel que défini dans [Fer02], le poids d'une arête e est fixé à $\zeta(e)$. cette restriction peut être levée en substituant à $\zeta(e)$ la fonction $\zeta(e,t)$ sans que la nature du modèle n'en soit altérée.

En conséquence, tous les modèles présentés peuvent être analysés de manière différée en ayant recours à un graphe évolutif.

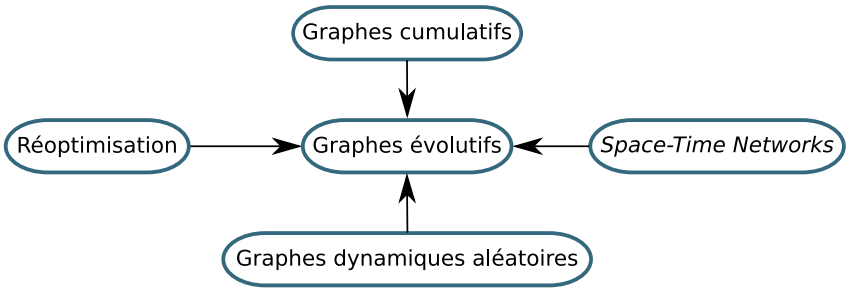


FIGURE 2.9: Tous les modèles de graphes peuvent être analysés de manière diférée avec un graphe évolutif.

Ainsi, pour un problème P donné, les performances atteintes par les algorithmes développés pour les modèles de graphes cumulatifs et de graphes dynamiques aléatoires, pourront être comparés avec le meilleur algorithme traitant du problème P développé pour les graphes évolutifs.

2.3.4 Conclusion

À la lumière de ce chapitre, deux points importants se dégagent. L'ensemble des modèles, à l'exception des modèles de graphes dynamiques aléatoires, définissent implicitement *un graphe dynamique comme une suite de graphes ou comme une suite d'événements produisant le nouveau graphe à partir du graphe courant*. Seul le modèle de graphes dynamiques aléatoires propose implicitement d'inclure le *processus d'évolution du graphe* dans la description de celui-ci. Au chapitre 3, nous proposons une définition des graphes dynamiques qui considère explicitement le processus d'évolution du graphe comme l'un des éléments de sa définition. Enfin, dans le contexte de l'analyse différée d'algorithmes de traitement de graphes dynamiques, et cela quel que soit le modèle envisagé, le modèle des graphes évolutifs semble s'imposer comme le formalisme le plus adapté.

CHAPITRE 3

MODÈLES ET MÉTRIQUES POUR LES GRAPHES DYNAMIQUES

Dans ce chapitre, nous tentons de définir la notion de graphe dynamique dans son acception la plus large qu'il nous soit possible de considérer, ainsi qu'un ensemble de métriques s'y rapportant.

Définir un graphe dynamique par la donnée d'un graphe initial et d'une suite d'événements nous semble restrictif. En effet, dans le cas des réseaux d'entités en interactions par exemple, les événements ne sont pas le résultat d'un processus exogène, mais résultent de l'évolution du système modélisé par le graphe dynamique. Lorsque l'ob-

servation du système est effectuée *a posteriori*, la suite des événements est connue et peut être restituée par une représentation compacte et adaptée à une analyse différée ; c'est ce que réalisent les graphes évolutifs. *A contrario*, un graphe évolutif ne capture que ce que l'on pourrait appeler la « trace » d'un graphe dynamique mais ne capture pas son essence, le processus qui est à la base de son évolution. Par ailleurs la plupart des contributions au domaine des réseaux complexes présentées dans le chapitre précédent décrivent souvent les réseaux étudiés sous la forme d'un processus de construction, mais sans jamais s'intéresser au temps. En effet, ils partent du principe, vrai mais limitatif, que les réseaux complexes sont des structures discrètes et considèrent leur construction davantage comme un processus itératif atemporel que comme un processus qui pourrait être assujetti à une base temporelle réelle ou plus contraignante.

La première partie s'attache à la présentation de notre définition et à sa mise en perspective au travers de quelques exemples. La seconde partie du chapitre est dédiée à la présentation de définitions complémentaires et à la discussion d'un ensemble de métriques. Celles-ci nous semblent pertinentes dans le cadre de l'étude des graphes dynamiques en général et sont nouvelles par rapport à celles communément utilisées dans le domaine de la théorie des graphes. Dans la suite de ce document, certaines d'entre elles seront utilisées pour mesurer la validité et l'efficacité de certaines approches de traitement des graphes dynamiques.

3.1 Définition générale

De notre point de vue, un graphe dynamique est un graphe en évolution dans le temps. Ainsi, il nous semble incontournable de considérer que le processus de l'évolution du graphe est l'un des éléments de sa définition, tout autant que la base temporelle dans laquelle ce processus s'exécute.

Définition 5 (GRAPHE DYNAMIQUE)
Un graphe dynamique $\mathcal{G}$ est défini par un triplet (G_0, T, P) tel que :
- *$G_0 = (S_0, A_0)$ est un graphe statique initial, éventuellement vide ;*
- *T est une base temporelle continue ou discrète ;*
- *P est un processus d'évolution.*

On note G_t le graphe statique image du graphe dynamique $\mathcal{G}$ à la date t.

3.1.1 Notion de base temporelle

La base temporelle peut être continue ou discrète. En effet, dans l'hypothèse où le graphe étudié modélise un réseau spatialisé composé d'entités en déplacement, comme peuvent l'être par exemple les réseaux mobiles *ad hoc*, la position des sommets change à chaque instant. Or, les interactions entre ces entités, qu'il s'agisse d'interactions de contact ou d'interactions à distance, dépendent de la distance spatiale entre ces entités. Si l'interaction dépend de la distance entre les entités, celle-ci est une fonction continue

de la distance et donc indirectement du temps. La base temporelle d'un tel graphe doit donc, en toute généralité, être continue. Notons malgré tout que ce formalisme n'exclut en rien la possibilité de considérer une base temporelle discrète, comme cela est fait classiquement dans le domaine de la simulation à temps discret. Cependant, toutes choses égales par ailleurs, le graphe dynamique définit par une base temporelle discrète peut alors être considéré comme inclus dans le graphe dynamique définit par une base temporelle continue. De la même façon, plusieurs bases temporelles discrètes peuvent être envisagées pour la définition d'un graphe dynamique. Pour une propriété donnée existante dans le modèle de graphe dynamique à base temporelle continue, la meilleure « granularité temporelle » est celle qui correspond au pas de temps le plus important tout en permettant l'observation de la propriété.

3.1.2 Processus d'évolution

Le processus d'évolution peut être décrit selon un formalisme algorithmique. Il comprend également les « équations dynamiques », qui correspondent à des processus d'évolution locaux, dont le principe a été proposé et décrit par H. BERSINI en 2002 dans [Ber02] puis dans son livre [Ber05].

Si la base temporelle est discrète, alors le processus d'évolution peut s'exprimer par :

$$\begin{aligned} P: \quad & \mathbb{N} \times \mathbb{G} \longrightarrow \mathbb{G} \\ & t, G_{t-1} \longmapsto G_t = P(t, G_{t-1}) \end{aligned}$$

où $\mathbb{N}$ représente l'ensemble des entiers naturels et $\mathbb{G}$ l'ensemble des graphes statiques. Une telle base temporelle permet de définir des étapes durant lesquelles le processus d'évolution agit sur le graphe courant. Si la base temporelle est continue, alors le processus d'évolution peut s'exprimer par :

$$\begin{aligned} P: \quad & \mathbb{R}^+ \times \mathbb{G} \longrightarrow \mathbb{G} \\ & t, G_{courant} \longmapsto G_{nouveau} = P(t, G_{courant}) \end{aligned}$$

où $\mathbb{R}$ représente l'ensemble des réels et $\mathbb{G}$ l'ensemble des graphes statiques.

Dans cette dernière formulation, rien n'empêche de considérer que P est un processus d'évolution continu.

3.2 Exemples de graphes dynamiques

Nous proposons maintenant de reconsidérer la formulation de plusieurs familles de graphes dynamiques rencontrés dans la littérature.

3.2.1 Processus d'évolution pour les réseaux complexes à attachement préférentiel

Dans le modèle à « attachement préférentiel » de A.-L. BARABÁSI et R. ALBERT [BA99], le graphe statique initial est vide, la base temporelle est discrète. L'algorithme 1 en décrit le principe.

Algorithme 1 : Processus d'évolution du modèle d'attachement préférentiel

1 Ajouter un sommet s_1;
2 **pour** $t \leftarrow 2$ *à* N **faire**
3 Ajouter un sommet s_t ;
Choix du sommet s_i selon la probabilité $Proba(s_i) \leftarrow \frac{degré(s_i)}{\sum_{j=1}^{t-1} degré(s_j)}$;
4
5 Ajout de l'arête (s_i, s_t);
6 **fin**

Il est possible de produire un algorithme plus général qui inclut la majorité des modèles de construction de graphes dynamiques aléatoires. L'algorithme 2 correspond au modèle de construction de N. DEO et A. CAMI [DC07] vu dans le chapitre 2 mais il inclut aussi le modèle « attachement préférentiel » de A.-L. BARABÁSI et R. ALBERT.

Algorithme 2 : Processus d'évolution pour la famille des graphes dynamiques aléatoires

1 **pour** $t \leftarrow 1$ *à* N **faire**
`/* Π est un tirage aléatoire et p un paramètre. */`
2 **si** $\Pi < p$ **alors**
3 Création de sommets et d'arêtes avec choix probabiliste;
4 **sinon**
5 Suppression de sommets et d'arêtes avec choix probabiliste;
6 **fin**
7 **fin**

3.2.2 Processus d'évolution pour les réseaux d'entités indépendantes spatialisées

Dans l'exemple précédent, les sommets et les arêtes n'ont aucune existence autre que dans le graphe dynamique. Il est cependant des modèles pour lesquels les sommets et les arêtes ne peuvent pas être considérés sans qu'un minimum d'information leur

soit attachée. C'est le cas des réseaux d'entités spatialisées. Chaque sommet est associé à une entité qui possède une position dans l'espace euclidien. De cette position peut dépendre l'existence ou la non existence de certains liens avec d'autres entités, c'est-à-dire l'apparition ou la disparition d'arêtes dans le graphe dynamique. Nous proposons un schéma général de processus d'évolution pour ce type de réseaux.

La principale difficulté vient de l'ajout du temps. Il est possible de considérer le processus d'évolution comme un processus continu (algorithme 3).

Algorithme 3 : Processus d'évolution des graphes dynamiques issus de réseaux d'entités indépendantes spatialisées

```
1  tant que Vrai faire
2      Choisir Δt ;
3      Ajout/suppression de sommets;
4      pour chaque sommet s faire
           /* La fonction m définit l'évolution des propriétés du
              sommet. */
5          s.props(t+Δt) ← m(s.props(t));
6      fin
7      pour chaque sommet s faire
8          pour chaque sommet s' faire
               /* La fonction condition examine la possibilité pour un
                  sommet de se lier à d'autres sommet. */
               /* La fonction p examine si les conditions pour une
                  liaison symétrique entre deux sommets donnés sont
                  réunies. */
9              si ((condition(s) ET condition(s')) ET p(s,s') alors
10                 si (s,s') ∉ A alors
11                     A ← A ∪ {(s, s')};
12                 fin
13             sinon
14                 si (s,s') ∈ A alors
15                     A ← A\{(s, s')};
16                 fin
17             fin
18         fin
19     fin
20 fin
```

Dans la description du processus, le pas (Δt) peut être fixé à chaque nouvelle itération, la seule contrainte étant que sa valeur soit strictement supérieure à 0. Ainsi, si on

note P_c le processus continu, et $P_d(\Delta t)$ le processus discret, alors :

$$\lim_{\Delta t \to 0} P_d(\Delta t) = P_c$$

Les entités possèdent un ensemble de propriétés (`props`) qui est défini pour chaque modèle considéré. La fonction `m()` définit les modifications à apporter aux propriétés en fonction du temps. Le comportement des entités en terme de liaison est précisé par les fonctions `p()` et `condition()`. Dans le contexte de l'épidémiologie, `condition()` peut définir la susceptibilité des entités. Dans le contexte des réseaux mobiles, la fonction `p()` peut indiquer que les deux stations s'échangent des données.

3.2.3 Processus d'évolution pour les réseaux d'entités en interaction

Nous proposons cette fois une description d'un processus d'évolution général pour les graphes dynamiques qui modélisent des réseaux d'entités en interaction relevant des domaines des sciences du vivant et des sciences humaines (algorithme 4). Un processus de ce type régit les réseaux d'interactions de type *Boids*, bancs de poissons, mouvements de foules, ou tout autre système caractérisé par une mobilité de groupe. Il permet aussi de modéliser des systèmes non spatialisés mais au sein desquels le voisinage d'une entité agit sur son comportement.

Algorithme 4 : Processus d'évolution des graphes dynamiques issus de réseaux d'entités en interaction

```
1  tant que Vrai faire
2      Choisir Δt ;
3      pour chaque sommet s faire
4          s.newProps ← f(s.props(t), v_e(t), Δt);
5      fin
6      pour chaque sommet s faire
7          s.props( t + Δt ) ← s.newProps;
8      fin
9      Ajouter/Supprimer de sommets;
10     pour chaque sommet s faire
11         pour chaque sommet s' ≠ s faire
12             si (s, s') ∉ A alors
13                 si condition(s,s') vérifiée alors
14                     A ← A ∪ {(s, s')};
15                 fin
16             sinon
17                 si condition(s,s') non vérifiée alors
18                     A ← A\{(s, s')};
19                 fin
20             fin
21         fin
22     fin
23 fin
```

3.3 Métriques pour les graphes dynamiques

Dans cette section sont proposées plusieurs définitions et métriques pour les graphes dynamiques. La suite du document fait régulièrement référence à ces définitions. Le modèle de graphe dynamique ($\mathcal{G}$) proposé ci-dessus (définition 5) est utilisé et toutes les notations utilisées sont définies dans la définition du modèle. On suppose que les sommets et les arêtes de tels graphes sont identifiables de manière unique et qu'ils peuvent disparaître du graphe puis réapparaître plus tard avec le même identifient. Il est aussi à noter que le terme élément peut être utilisé pour désigner un sommet ou une arête. On considère un graphe dynamique $\mathcal{G} = (G_0, T, P)$.

3.3.1 Définition complémentaires

Définition 6 (PRÉSENCE)
Dans un graphe dynamique $\mathcal{G}$, un élément est présent à l'instant t s'il appartient à G_t.

Définition 7 (ABSENCE)
Dans un graphe dynamique $\mathcal{G}$, un élément est absent à l'instant t s'il n'appartient pas à G_t.

Définition 8 (APPARITION)
Un élément apparaît dans le graphes dynamique $\mathcal{G}$ quand il passe de l'état absent à l'état présent.

Définition 9 (DISPARITION)
Un élément disparaît du graphe dynamique $\mathcal{G}$ quand il passe de l'état présent à l'état absent.

Définition 10 (NAISSANCE)
La naissance d'un élément est la date correspondant à sa première apparition dans le graphe dynamique $\mathcal{G}$.

Définition 11 (MORT)
La mort d'un élément est la date correspondant à la dernière disparition de cet élément dans le graphe dynamique $\mathcal{G}$.

Définition 12 (STRUCTURE)
Une structure S dans un graphe dynamique est constituée d'un ensemble d'éléments qui vérifie une propriété. Un chemin, une clique, une composante connexe, un trajet, sont des exemples de structures.

Les éléments d'une structure peuvent changer au cours du temps, mais la structure conserve sa propriété.

Définition 13 (CHEMIN)
Un chemin $C_t = \{a_1, a_2, \ldots, a_k\}$ est une structure composée d'un ensemble d'arêtes dans G_t. Le sommet s est la source du chemin, il est aussi le sommet source de l'arête a_1. Le sommet d est la destination du chemin ainsi que de l'arête a_k. Pour toute arête $a_j, 1 \leq j \leq k-1$ le sommet destination de a_j est le sommet source de a_{j+1}.

Toutes les arêtes du chemin C_t sont présentes simultanément dans G_t. Lorsqu'elles ne sont pas simultanément présentes, la notion de trajet définie dans [BXFJ03] pour un graphe évolutif remplace la notion de chemin.

Définition 14 (TRAJET)
Soit $T = \{a_1, a_2, \ldots, a_k\}$ une structure composée d'un ensemble d'arêtes dans un graphe dynamique $\mathcal{G}$. Soit $E(a_j)$ l'ensemble des intervalles de temps durant lesquels l'arête a_j est présente dans $\mathcal{G}$. T est un trajet s'il existe un ensemble $\{t_0, t_1, t_2, \ldots, t_k\}$ de dates tel que $t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_k$ et $t_j \in E(a_j)$, $\forall j\ 0 \leq j \leq k$.

Cette définition vaut pour le modèle de graphe dynamique défini plus haut mais inclue également la définition de la notion de trajet dans les graphes évolutifs.

3.3.2 Métriques

Définition 15 (ÂGE)
L'âge d'un élément présent est la différence entre la date courante et la date de sa dernière apparition. L'âge d'un élément absent est zéro.

Soit $t_{courant}$ la date courante, celle à laquelle on se pose la question de l'âge de l'élément s. Soit $t_{apparition}$ la dernière date d'apparition de s. $t_{apparition} = max\{t \mid t \geq 0, s \in G_t, s \notin G_{t-1}\}$. Finalement, l'âge de s est la différence entre $t_{courant}$ et $t_{apparition}$.

Définition 16 (ÂGE CUMULÉ)
L'âge cumulé d'un élément est la somme des durées des intervalles de temps durant lesquels il est présent dans le graphe dynamique $\mathcal{G}$.

L'âge cumulé est une grandeur qui peut être observée à n'importe quel moment de l'évolution du graphe. L'âge cumulé ne donne aucune indication qualitative sur cette présence, c'est à dire sur la répartition de cette présente au cours du temps dans le graphe.

Définition 17 (VOLATILITÉ)
La volatilité d'un élément est le ratio entre son nombre d'apparitions et son âge cumulé.

$$volatilité = \frac{nombre\ d'apparitions}{âge\ cumulé}$$

L'intérêt de cette mesure est qu'en complément de l'âge cumulé qui donne une grandeur quantitative, elle donne une indication qualitative de la vie de l'élément observé. La figure 3.1 illustre le rapport entre le nombre d'apparitions et l'âge cumulé d'un élément.

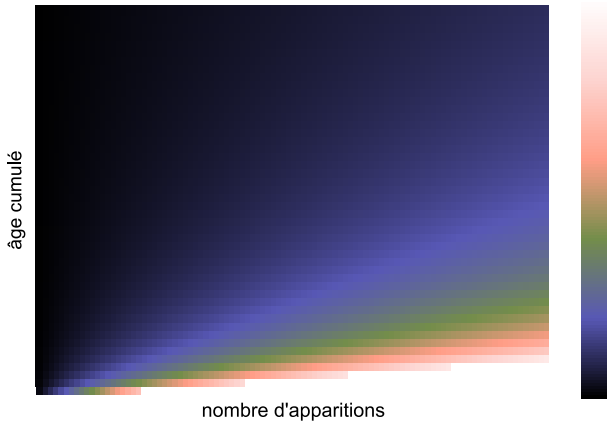


FIGURE 3.1: Représentation de la volatilité d'un élément en fonction du nombre d'apparitions et de son âge cumulé. Les couleurs sombres tendent vers zéro. Les couleurs claires tendent vers plus l'infini.

Définition 18 (VOLATILITÉ D'UNE STRUCTURE)
La volatilité d'une structure dans un graphe dynamique à un instant donné est la valeur moyenne de la volatilité des éléments de cette structure à cet instant.

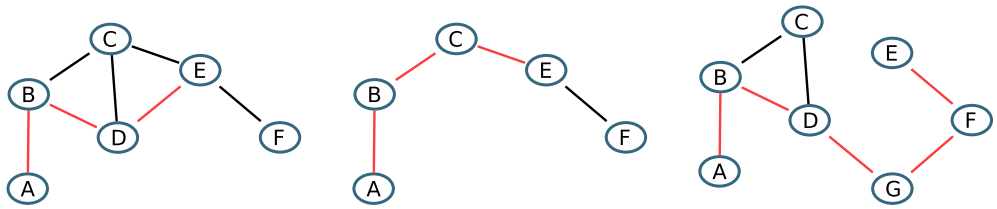
Définition 19 (ÂGE MOYEN)
L'âge moyen d'un élément correspond à l'inverse de sa volatilité.

Définition 20 (TAUX DE RENOUVELLEMENT)
Soit une structure S dans un graphe dynamique observée à deux dates différentes de l'évolution du graphe (t_1 et t_2). S_{t_1} est la structure au temps t_1 et S_{t_2} est la structure au temps t_2. Le taux de renouvellement $t_r(S_{t_1}, S_{t_2})$ correspond au nombre M de modifications (ajouts et suppressions d'éléments) dans la structure entre ces deux dates par rapport au nombre d'éléments $|S_{t_1}|$ dans la structure de départ :

$$t_r(S_{t_1}, S_{t_2}) = \frac{M}{|S_{t_1}|} \tag{3.1}$$

3.3.3 Exemple illustratif

Nous proposons d'illustrer les différentes définitions précédentes à l'aide d'un exemple. La figure 3.2 présente 6 états instantanés d'un graphe dynamique $\mathscr{G}$ pris à des dates quelconques et croissantes. Nous considérons que les évènements topologiques se produisent au dates indiquées et qu'entre ces dates aucune modification ne survient dans le graphe. Une structure dont la propriété est d'être un plus court chemin entre les sommets A et E est maintenue. La structure en question est représentée par l'ensemble d'arêtes de couleur rouge (claire) qui forme un plus court chemin entre A et E. La table 3.1 montre la liste, entre chaque date, des évènements topologiques survenus.



(a) date $t_1 = 0$ (b) date $t_2 = 12$ (c) date $t_3 = 40$

(d) date $t_4 = 45$ (e) date $t_5 = 50$ (f) date $t_6 = 100$

FIGURE 3.2: Exemple de Graphe Dynamique $\mathscr{G}$.

Métriques

Considérons maintenant quelques uns des éléments ainsi que la structure construite dans le graphe et observons certaines des métriques présentées plus hauts.

Le sommet D. Il apparaît et disparaît à plusieurs reprises pendant l'évolution. Son âge cumulé peut être calculé : il est présent entre t_1 et t_2 ($t_2 - t_1 = 12 - 0 = 12$), puis disparaît en t_2 avant de réapparaître en t_3 pour rester présent jusqu'en t_6 ($t_6 - t_3 = 100 - 40 = 60$). L'âge cumulé de d vaut donc 72 dans l'intervalle $[t_0, t_6]$. Il apparaît deux fois, en t_0 et en t_3. Son âge moyen vaut donc $\frac{nombre\ d'apparitions}{\text{âge cumulé}} = 72/2 = 36$ et sa volatilité est l'inverse de son âge moyen : $1/36 \simeq 0.027$.

TABLE 3.1: Événements dynamiques entre chaque date du graphe dynamique d'exemple.

t_1	apparition de A, B, C, D, E et F apparition de (B,A), (B,C), (B,D), (C,D), (D,E), (C,E), (E,F)	t_2	disparition de (B,D), (C,D), (E,D), D	t_3	disparition de (C,E) apparition de D, G, (B,D), (C,D), (D,G), (G,F)
t_4	disparition de C , (B,C) , (C,D) , (E,F) apparition de (D,E)	t_5	disparition de (A,B), (B,D), (D,G), (F,G), G, F apparition de (A,D), C, (C,E)	t_6	disparition de (A,D), (E,D), D apparition de (A,B), (B,C), F, G , (E,F), (F,G)

L'arête (C,E)**.** Elle apparaît dès la date t_0 pour disparaître à la date t_3 ($t_3 - t_0 = 40 - 0 = 40$).Puis elle réapparaît à la date t_5 ($t_6 - t_5 = 100 - 50 = 50$). Son âge cumulé est donc 90. Elle apparaît deux fois (en t_0 et en t_5), son âge moyen vaut donc $90/2 = 45$ et sa volatilité $1/45 \simeq 0.022$.

La structure de plus court chemin. Observons maintenant des métriques sur cette structure de plus court chemin. Elle est constituée des arêtes d'un plus court chemin et évolue dans le temps. Il est possible d'observer des valeurs moyennes pour les métriques des éléments qui la composent.

La volatilité de la structure correspond à la volatilité moyenne de ses éléments.

Arête	Âge cumulé	Apparitions	Volatilité
(A,B)	45	2	0.044
(B,D)	17	2	0.117
(E,D)	17	2	0.117
(B,C)	40	2	0.05
(C,E)	62	2	0.032
(D,G)	10	1	0.1
(G,F)	10	2	0.2
(F,E)	40	2	0.05
(A,D)	50	1	0.04

La volatilité de la structure de plus court chemin illustrée dans la figure 3.2 vaut environ 0.083.

Le taux de renouvellement de la structure peut être observé entre chacune des dates présentées et une valeur moyenne de ce taux peut être calculée pour toute la durée de l'évolution du graphe. Rappelons que le taux de renouvellement entre deux structures (ou entre deux états d'une même structure) est le rapport entre la somme des apparitions et des suppressions d'une part, et du cardinal de la structure de départ d'autre

part.

Intervalle	Apparitions	Disparitions	Cardinal de départ	Taux de renouvellement
$t_1 \rightarrow t_2$	2	2	3	$4/3 \simeq 1.33$
$t_2 \rightarrow t_3$	4	2	3	$6/3 = 2$
$t_3 \rightarrow t_4$	1	3	5	$4/5 = 0.8$
$t_4 \rightarrow t_5$	1	2	3	$3/3 = 1$
$t_5 \rightarrow t_6$	3	2	2	$6/2 = 3$

Le taux moyen de renouvellement de la structure est alors environs 1.626.

Deuxième partie

Graphes dynamiques : traitement et optimisation décentralisés

INTRODUCTION

Le thème central abordé dans cette partie est la construction et le maintien de solution pour des problèmes modélisés par des graphes dynamiques.

Cette partie du document est divisée en deux chapitres. Le premier réalise un état de l'art des différentes méthodes inspirées de la métaphore des fourmis. Ce chapitre s'intéresse également aux applications traitées. Le second chapitre présente une approche générale de traitement de problèmes d'optimisation dans les graphes dynamiques. Celle-ci est fidèle à la métaphore des fourmis, s'inspire des approches présentées dans le chapitre d'état de l'art et repose sur une approche distribuée.

Quel que soit le domaine considéré, physique, biologie, mathématiques, sciences sociales, chimie, informatique, *etc.*, on peut identifier des systèmes qui présentent des propriétés globales émergentes issues des interactions locales entre les entités constituant ces systèmes. Les bancs de poissons, les essaims d'oiseaux [HC04], les colonies de bactéries [EBJ04], les tas de sable [Bak96], les automates cellulaires [Wol84, Heu98], les réseaux d'interaction protéiques [ACKN04], l'organisation des villes, les langages humains [Wal94] sont autant d'exemples. Ils sont nommés « systèmes complexes » [Zwi03]. Ces systèmes sont généralement ouverts, traversés par différents flux et leurs composants entretiennent des relations entre eux ainsi qu'avec l'environnement. De ces interactions émergent des propriétés dont celle « d'auto-organisation » qui peut être définie comme un processus holistique et dynamique qui permet au système de s'adapter aussi bien aux changements de l'environnement qu'aux modifications des entités elles-mêmes.

Le but de ce travail est d'explorer cette propriété d'auto-organisation pour résoudre des problèmes modélisés à l'aide de graphes. L'idée centrale consiste à construire un système complexe artificiel qui présente des propriétés d'auto-organisation et dont les entités évoluent dans un environnement artificiel. Cet environnement est un graphe dynamique dans lequel on recherche des structures ou des sous-ensembles qui représentent des solutions pour les problèmes que l'on se pose. Pour construire des structures, les entités doivent être capables de se déplacer, de communiquer ensemble et de marquer leur environnement. En d'autres termes, on cherche à observer une projection de l'organisation du système à l'intérieur du graphe.

En éthologie, les structures et les organisations produites par les animaux sociaux

sont étudiées de deux manières complémentaires [The97]. D'une part on cherche à expliquer « pourquoi » ces structures et ces organisations sont créées. D'autre part on cherche à comprendre « comment » celles-ci sont créées. Pour cette dernière question, le but est de faire le lien entre les comportements individuels des différents animaux et les structures ou les décisions collectives qui émergent des groupes. La notion d'auto-organisation est centrale ici. Les phénomènes collectifs chez les animaux sont différenciés en trois classes. On note d'abord les phénomènes d'organisation spatiotemporelle des activités des individus, ensuite la différenciation des individus et enfin les phénomènes menant à la structuration collective de l'environnement. Le dernier point se produit essentiellement chez les guêpes, les termites et les fourmis. C'est probablement l'une des raisons pour lesquelles les fourmis font l'objet d'études éthologiques et technologiques [BDT99].

On s'intéresse également à la question « comment » dans ce travail et en particulier au phénomène collectif de structuration de l'environnement. Les travaux précédemment menés par différents auteurs autour du problème sont autant de connaissances à étudier et à utiliser. Dans le contexte de l'optimisation, on verra que la contribution de M. DORIGO et de ses co-auteurs autour du paradigme des *ACOs* (*Ant Colony Optimization*) est majeure.

Les fourmis dans la nature sont capables de communications directes, par le biais des antennes et par la trophalaxie, mais aussi de communications indirectes, via l'environnement, dont le principe se nomme « stigmergie » [Gra59].

Le comportement des fourmis a été largement étudié et appliqué avec succès à de nombreux problèmes d'optimisation modélisés à l'aide de graphes. Le principe général est le suivant :

1. chaque fourmi se déplace et dépose des phéromones dans l'environnement sur son passage ;
2. Le choix du déplacement des fourmis est partiellement guidé par les quantités de phéromones présentes sur les pistes et ce choix est probabiliste ;
3. les phéromones s'évaporent avec le temps.

Le mécanisme général est itératif, les étapes précédentes sont répétées. Durant l'exécution, les chemins voient leur quantité de phéromones augmenter proportionnellement au nombre de fourmis qui les parcourent. À l'échelle de l'environnement qui peut être un graphe, on observe des chemins, des groupes d'arêtes, des structures.

Dans le domaine informatique l'étude des colonies de fourmis a donné naissance à une classe générale d'algorithmes d'optimisation basés sur l'exploration efficace d'un espace de recherche. Cette classe d'algorithmes qui produit des heuristiques pour différents problèmes appartient à la famille des méta-heuristiques. La méta-heuristique inspirée des fourmis la plus connue est celle des *ACOs*. Elle repose sur un processus stochastique et itératif de construction de solutions qui sont évaluées grâce à une fonction d'évaluation générale. Les solutions sont donc produites et évaluées de manière centra-

lisée par l'algorithme. Les meilleures solutions produites participent au renforcement des pistes de phéromones.

Une autre classe de méta-heuristiques basée sur des phénomènes de comportements collectifs est *PSO* (*Particle Swarm Optimization*) [JK01]. Il s'agit d'un processus stochastique et itératif qui fait évoluer et interagir entre elles des entités dans un environnement. De manière itérative les entités se déplacent dans un environnement à n dimensions (chaque dimension est une contrainte à satisfaire). Chaque entité ajuste son vecteur de déplacement en fonction des entités de son voisinage et de l'évaluation de leur position par une fonction d'évaluation globale. Là encore, ce mécanisme est centralisé et nécessite une fonction d'évaluation.

Notre proposition est proche des *ACOs* et de *PSO*, elle est inspirée des mêmes principes et utilise les même approches à base de populations d'entités. Cependant, les modèles sur lesquels doivent opérer les entités sont dynamiques, ce qui dans bien des cas rend problématique la définition d'une fonction d'évaluation. Ainsi, si on considère la mise au point de méthodes de traitement dans le contexte d'environnements distribués et dynamiques comme peuvent l'être les réseaux mobiles *ad hoc*, l'évaluation centralisée des solutions produites est difficilement envisageable.

Les mécanismes d'intelligence collective sont naturellement adaptés aux environnements distribués, les *ACOs* et *PSO* ne le sont pas. C'est pourquoi cette partie du document s'attache à la description d'une approche de résolution sans évaluation globale. En cela notre approche est plus proche de la métaphore des fourmis et de l'approche *ABC* (*Ant-Based Control*) de R. SCHOONDERWOERD *et al.* [SHBR97]

CHAPITRE 4

ÉTAT DE L'ART DES MÉTHODES À BASE DE FOURMIS [1]

Ce chapitre s'intéresse dans un premier temps aux différentes contributions scientifiques ayant utilisé la métaphore des fourmis pour résoudre des problèmes. Nous constatons qu'une filiation existe entre les différentes propositions qui s'enrichissent des fonctionnalités décrites dans les premières pour proposer des améliorations s'appliquant parfois mieux aux problèmes considérés.

1. Ce chapitre a fait l'objet d'une publication sous le titre « Tour d'horizon des problèmes combinatoires traités par les fourmis artificielles » [Dutot and Pigné(à paraître)], dans l'ouvrage « Fourmis artificielles, Des bases algorithmiques aux concepts et réalisations avancés » [MGS09].

Les différentes contributions ont vocation à résoudre des problèmes variés que l'on peut classer en différentes familles présentés dans la seconde partie du chapitre.

Dans le cadre du traitement de problèmes réels, des contraintes moins formalisées doivent être prises en compte telles que les environnements dynamiques et décentralisés, les incertitudes, les erreurs dans les données, *etc.* D'autre part, pour certains de ces problèmes, la formulation d'une fonction d'objectif peut s'avérer délicate voire impossible. De plus, les objectifs peuvent être multiples, on parlera alors d'optimisation multi-objectifs. Ces contraintes définissant des problèmes toujours ouverts sont traitées dans la troisième partie de ce chapitre.

4.1 Les principales approches à base de fourmis

Des centaines de contributions utilisant la métaphore des fourmis ont été proposées ces deux dernières décennies. Certaines propositions sont restées proches du modèle naturel, d'autres s'en sont éloignées afin d'améliorer les résultats obtenus pour les problèmes traités. La méthode originale, apparue pour la première fois en 1991 dans la thèse de M. DORIGO [Dor92], consiste à proposer une méta-heuristique pour l'optimisation combinatoire. Cette approche plus tard formalisée sous le nom de *ACO* (*Ant Colony Optimization*) [DC97] mêle l'idée de collaboration entre entités (les fourmis) explorant un espace de recherche et l'évaluation des solutions basée sur une fonction d'objectif. À partir de l'algorithme original, une multitude de variantes furent proposées et beaucoup de problèmes furent traités avec succès. Cette section revient sur quelques unes des contributions fondatrices de l'approche *ACO*; puis elle se termine sur d'autres propositions originales plus proches du modèle naturel.

4.1.1 La famille des *ACOs*

La méthode proposée dans les travaux de M. DORIGO *et al.* [DMC91, Dor92, DMC96] sous le nom de *Ant System* est une méta-heuristique conçue pour l'optimisation de problèmes combinatoires. L'idée d'une population d'agents qui évoluent dans un environnement et coopèrent par le biais de communications asynchrones est clairement inspirée des colonies d'insectes sociaux comme les fourmis. La méthode hérite aussi des mécanismes classiques de recherche opérationnelle et d'intelligence artificielle tels que l'apprentissage par renforcement ou la construction gloutonne d'une solution guidée par une fonction d'objectif. *Ant System* fut la première des contributions du genre. Toutes les méthodes utilisant ce paradigme sont maintenant appelées des *ACOs*. Chaque nouvelle contribution a apporté des améliorations au modèle original ou des spécialisations pour des problèmes particuliers.

Ant System

Ant System a été initialement conçu pour le problème du voyageur de commerce mais peut facilement être adapté à d'autres problèmes combinatoires. Il consiste à déterminer l'itinéraire le plus court d'un voyageur de commerce lui permettant de traverser chacune des n villes de sa tournée une fois et une seule. Ce problème est généralement modélisé par un graphe complet dont l'ensemble N des sommets représente les villes et l'ensemble E des arêtes les routes entre ces villes. Chaque arête est valuée par la distance d_{ij} qui sépare les villes i et j.

En théorie des graphes, ce problème correspond à la recherche d'un cycle hamiltonien de poids minimum qui est connue pour être NP-difficile au sens fort.

Dans la méthode *Ant System*, le graphe est parcouru par des fourmis artificielles qui se déplacent de manière à construire des cycles hamiltoniens. Chaque fourmi possède un comportement simple et une vision localisée. Le fonctionnement d'une colonie peut être schématisé ainsi :

- De manière itérative chaque fourmi choisit son prochain sommet. Son choix dépend de la distance entre son sommet courant et le suivant, et de l'attractivité des liens qui est fonction de la quantité de phéromone associée au lien.
- Les villes déjà visitées sont supprimées du voisinage des fourmis de manière à ce qu'elles visitent une seule fois chaque ville.
- Quand les fourmis ont terminé leur cycle elles modifient l'attractivité des liens qu'elles ont visités en déposant des phéromones.

La méthode est itérative, une fourmi choisit au temps t quelle sera la ville visitée au temps $t+1$. Ces temps sont des itérations. À chaque itération toutes les fourmis du système se déplacent d'une ville à une autre. Au bout de n itérations (n est le nombre de villes) les fourmis ont construit un cycle et sont revenues à la ville de départ parce que le graphe est complet. Afin de produire des solutions réalisables (des cycles hamiltoniens), chaque fourmi maintient une liste ($tabu_k$) des villes qu'elle a déjà visitées et les retire de son voisinage lors du choix de la prochaine ville.

L'attractivité des liens repose sur le dépôt de phéromones. Soit $\tau_{ij}(t)$ la quantité de phéromone sur le lien entre les villes i et j à l'instant t. La mise à jour de l'attractivité des liens est effectuée grâce à la formule suivante :

$$\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} \tag{4.1}$$

ρ est un paramètre qui règle la conservation des phéromones d'un cycle au suivant. $(1-\rho)$ correspond donc au facteur d'évaporation des phéromones sur les pistes.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{4.2}$$

$\Delta\tau_{ij}^{k}$ est la quantité de phéromone que dépose la fourmi k sur l'arête (i,j) pendant le

cycle courant. m est le nombre de fourmis dans le système.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{si la fourmi } k \text{ est passée par } (i,j) \text{ pendant le cycle} \\ 0 & \text{sinon} \end{cases} \tag{4.3}$$

avec L_k la longueur du cycle effectué par la fourmi k et Q une constante.

Le choix de la prochaine ville pour chaque fourmi se fait à chaque itération en fonction de l'attractivité des liens et de la distance entre la ville courante et les villes voisines. Pour permettre aux fourmis de maximiser la qualité de leurs solutions la distance est prise en compte dans la *visibilité* : $\eta_{ij} = 1/d_{ij}$. Le choix du prochain sommet est soumis à une loi de probabilité aléatoire et proportionnelle aux critères présentés ci-dessus.

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in voisins_k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} & \text{si } j \in voisins_k \\ 0 & \text{sinon} \end{cases} \tag{4.4}$$

avec $voisins_k$ l'ensemble des villes privé de celles déjà visitées par la fourmi k. α et β sont deux paramètres qui permettent de modifier l'importance relative des pistes de phéromone par rapport à la visibilité. Cette formule représente donc la probabilité à l'itération t que la fourmi k choisisse j comme prochaine destination à $(t+1)$.

Une fois un cycle effectué, le système dispose de solutions au problème donné. Ces solutions peuvent être améliorées. D'autres cycles peuvent être effectués et tirer profit des cycles précédents grâce au marquage des pistes. L'algorithme général de *Ant System* pour la résolution du problème du voyageur de commerce peut alors être décrit :

1. Initialisation :
   - $t = 0$
   - $\tau_{ij}(t) = c$
2. pour k allant de 1 à m
   - positionner la fourmi k sur le sommet source S.
   - $tabu_k = \{S\}$
3. Pour t allant de 1 à $n-1$ et k allant de 1 à m
   - Choix du prochain sommet S' en fonction de la formule 4.4.
   - Déplacement de la fourmi k vers le sommet S'.
   - Ajout de S' à la liste tabou : $tabu_k = tabu_k \cup \{S'\}$
4. À la fin du cycle, mise à jour des pistes de phéromone avec la formule 4.1
5. Si une condition d'arrêt ou un certain nombre de cycles ne sont pas atteints alors retour à l'étape 2.

Les auteurs ont appliqué *Ant System* à des instances répertoriées du problème du voyageur de commerce pour lesquelles des solutions sont connues. Ils ont ainsi pu comparer les résultats produits par leur algorithme avec des résultats optimaux ainsi qu'avec des résultats obtenus par d'autres heuristiques. L'algorithme montre son efficacité et la qualité des solutions produites. Ultérieurement, des améliorations ont été apportées et des variantes ont été proposées, notamment pour traiter d'autres problèmes. Nous en décrivons quelques unes dans ce qui suit.

Renforcement des meilleures solutions

Dans le but d'accélérer la convergence de la méthode, les auteurs de *Ant System* proposent dans les mêmes contributions une modification du comportement du système : *Elitist Ant*. À la fin de chaque cycle, pendant l'étape de renforcement des pistes, ils proposent d'ajouter un second renforcement sur les liens de la meilleure solution jamais réalisée depuis le début de la recherche (*global best solution* S_{gb}). La formule de renforcement à la fin de chaque cycle est alors :

$$\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau^{*}_{ij} \tag{4.5}$$

Seul le troisième terme diffère de la formule de renforcement classique [4.1].

$$\Delta\tau^{*}_{ij} = \begin{cases} \sigma \frac{Q}{L_{gb}} & \text{si } (i,j) \text{ appartient à } S_{gb} \\ 0 & \text{sinon} \end{cases} \tag{4.6}$$

avec L_{gb} la longueur de la meilleure solution globale S_{gb} et σ un paramètre supérieur à 1 qui permet d'ajuster l'importance du renforcement. L'idée est de rendre encore plus attractifs les liens participant aux meilleures solutions afin qu'ils soient utilisés dans la construction de nouvelles solutions.

Dans le prolongement de cette idée, l'algorithme *Rank-Based Ant System* [BHS99] propose de classer les fourmis après chaque cycle en fonction de la qualité de leur solution pour sélectionner les individus qui vont participer au renforcement. Ainsi la liste ordonnée des ω meilleures fourmis va être construite, le renforcement va être proportionnel à leur classement (meilleur est le classement d'une fourmi, plus important sera le renforcement associé à sa solution). Pour ne pas interférer avec le renforcement élitiste précédent, les auteurs notent que ω ne doit pas être supérieur à σ qui est le facteur multiplicatif appliqué à la meilleure solution globale et rien ne devrait être supérieur à cette solution. ω est fixé à $\sigma - 1$. La formule de renforcement dans cet algorithme ressemble à la formule de renforcement de *Elitist Ant* (4.5) à la différence du second terme $\Delta\tau_{ij}$ qui vaut maintenant :

$$\begin{aligned} \Delta\tau_{ij} &= \sum_{\mu=1}^{\sigma-1} \Delta\tau^{\mu}_{ij} \\ \Delta\tau^{\mu}_{ij} &= \begin{cases} (\sigma-\mu)\frac{Q}{L_{\mu}} & \text{si } (i,j) \text{ appartient au cycle de la } \mu^{ème} \text{ fourmi.} \\ 0 & \text{sinon} \end{cases} \end{aligned} \tag{4.7}$$

Dans [GD95] les auteurs proposent un algorithme proche de *Ant System* mais se focalisant sur la partie apprentissage par renforcement. *Ant-Q* propose de n'appliquer la formule de renforcement des pistes que sur les liens utilisés lors du cycle courant, affin d'accélérer la convergence de la recherche. Les liens non utilisés ne sont pas mis à jour et seules les solutions générées sont renforcées. De plus comme dans *Elitist Ant*, *Ant-Q* propose un second renforcement basé sur les meilleures solutions déjà produites. Deux variantes concernant le choix des pistes à renforcer sont proposées. La version *global-best* qui correspond à *Elitist Ant* en sélectionnant la meilleure solution depuis le début de la recherche (S_{gb}) et la version *iteration-best* où la solution utilisée pour le second renforcement est la meilleure du cycle qui vient d'être effectué (*iteration best solution* S_{ib}).

L'algorithme $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System* [SH+00] inspiré de *Ant System* propose entre autre d'utiliser la formule de renforcement 4.1 en modifiant $\Delta\tau_{ij}$ de manière à ce que seule la meilleure solution (globale ou locale) soit renforcée :

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L_{best}} & \text{si } (i,j) \text{ appartient à la solution } S_{best}. \\ 0 & \text{sinon} \end{cases} \tag{4.8}$$

où L_{best} est la longueur de la meilleure solution globale (S_{gb}) ou locale (S_{ib}). Ce mécanisme utilisé seul tend à ne renforcer que les bonnes solutions et à faire disparaître l'attractivité des liens plus rarement visités. Ceci permet de faire converger rapidement vers une solution mais provoque une mauvaise exploration de l'espace de recherche. $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System* règle ce problème en utilisant un mécanisme de bornes inférieures et supérieures empêchant la disparition de l'attractivité de certaines arêtes ou la sur-représentation d'autres. Notons que d'une manière générale, la dichotomie entre exploration de l'espace de recherche et exploitation des solutions déjà construites est un problème classique dans les méthodes heuristiques en général.

Le paramètre d'évaporation des pistes

Plusieurs contributions s'intéressent au mécanisme d'évaporation des pistes, sensé éviter la convergence prématurée de la méthode tout en assurant une bonne exploitation de l'espace de recherche. En effet l'évaporation des pistes lisse les différences d'attraction entre les liens, augmente la probabilité de visite de certains liens peu utilisés, améliore l'exploration de l'espace de recherche et tend à éviter la convergence prématurée de l'algorithme vers les premières solutions trouvées. L'effet secondaire indésirable est qu'une trop forte évaporation annule l'effet attractif des phéromones, fait perdre la mémoire collective des bonnes solutions trouvées auparavant et réduit le processus de recherche des fourmis à un parcours aléatoire de l'environnement ou à une recherche uniquement guidée par l'heuristique locale (la visibilité). L'évaporation dans les *ACOs* est ajustée grâce au paramètre ρ de la formule 4.1 page 51.

Dans *ANTS* [Man99] les auteurs n'utilisent pas le paramètre d'évaporation ρ et proposent un autre mécanisme pour modifier l'attractivité des pistes. Le mécanisme n'entre

en jeu qu'après un certain nombre d'étapes de calcul qui permettent de construire plusieurs (k) solutions et d'obtenir une estimation de la longueur moyenne d'une solution $L_{\overline{S}}$. Après cette phase d'initialisation, les pistes des nouvelles solutions générées sont renforcées proportionnellement à leur écart par rapport à la solution moyenne $\overline{S}$. Après avoir été comparée à la solution moyenne, chaque nouvelle solution est utilisée pour mettre à jour $L_{\overline{S}}$ qui est la longueur moyenne des k dernières solutions produites. Cette valeur évolue donc au cours du traitement. Seules les arêtes appartenant à des solutions générées pendant le cycle courant sont renforcées. Ainsi pour chaque solution s le renforcement est :

$$\tau_{ij}(t+n) = \tau_{ij}(t) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij} = \tau_0.\left(1 - \frac{L_s - LB}{L_{\overline{S}} - LB}\right) \tag{4.9}$$

où L_s est la solution courante et LB une borne inférieure (*Lower Bound*) connue pour le problème considéré. Ce mécanisme permet de modifier les quantités de phéromone par une valeur comprise entre τ_0 et $-\tau_0$.

Dans *Best-Worst Ant System* [CdVHM00], les auteurs s'intéressent aussi à la sensibilité du paramètre ρ. Ici le mécanisme d'évaporation n'est pas supprimé mais il est restreint de manière à minimiser son impact. Dans ce modèle seules les arêtes de la meilleure solution locale (S_{ib}) sont renforcées, et seules les arêtes de la plus mauvaise solution ne faisant pas partie de S_{ib} subissent l'évaporation $\tau_{ij} = \rho.\tau_{ij}$.

Renforcement global et local des pistes de phéromone

La solution apportée par l'algorithme *Ant Colony System* [DG97a] [DG97b] pour éviter le problème de mauvaise exploration tout en gardant les améliorations de convergence du renforcement élitiste est de définir deux types de renforcements, un local et un global. Le renforcement global est celui classiquement défini par *Ant System* et repris par les autres méthodes pour favoriser les liens appartenant à de bonnes solutions déjà trouvées. Il est élitiste et propose qu'à chaque cycle, seule la fourmi ayant produit la meilleure solution renforce sa piste de manière inversement proportionnelle à la longueur de celle-ci. C'est donc la meilleure solution locale qui est privilégiée (S_{ib}) de sorte que les liens de cette solution soient renforcés de manière inversement proportionnelle à la longueur L_{ib} de la meilleure solution locale S_{ib} ($\Delta\tau_{ij} = 1/L_{ib}$). Ainsi plus les solutions sont bonnes, plus elles sont renforcées. Le renforcement local quant à lui a pour vocation de redonner de l'attractivité aux liens peu visités. Il consiste en un renforcement systématique et constant des liens traversés par toutes les fourmis. Ce renforcement est dit local car les fourmis renforcent les liens à l'instant où elles les traversent. De plus ce renforcement est constant (τ_0), il n'est proportionnel à aucune solution et ne peut pas l'être car au moment de ce renforcement la fourmi n'a pas encore construit de solution.

Exploration *vs.* exploitation

Toujours dans le souci d'améliorer l'exploration de l'espace de recherche tout en tirant profit des bonnes solutions générées, les auteurs de *Ant-Q* [GD95] s'intéressent à d'autres manières de mettre en œuvre la procédure de choix du prochain sommet pour chaque fourmi. La formule classique (équation 4.4 page 52) est avant tout basée sur une exploration aléatoire influencée par les solutions déjà trouvées et par l'heuristique locale. La proposition ici est de mieux exploiter les solutions déjà produites. Ainsi les auteurs proposent la formule suivante pour déterminer le sommet s que la fourmi k située sur le sommet r au temps t visitera au temps $t+1$:

$$s = \begin{cases} \max\limits_{j \in voisins_k} \{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}\} & \text{si } p \leq p_0 \\ S & \text{sinon} \end{cases} \tag{4.10}$$

avec p une valeur choisie aléatoirement de manière uniforme dans $[0,1]$ et p_0 un paramètre ($0 \leq p_0 \leq 1$). S est une variable aléatoire ayant pour but de déterminer un sommet suivant s. Cette formule permet donc grâce au paramètre p_0 d'équilibrer l'exploitation des résultats précédemment trouvés et l'exploration aléatoire de l'environnement. La formule $\max\limits_{j \in voisins_k} \{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}\}$ détermine en effet le sommet le plus attractif alors que S est relatif à l'exploration aléatoire de l'environnement. S et p_0 doivent être définis et pour cela les auteurs proposent trois versions.

1. S est une variable aléatoire uniforme dans $[0,1]$, dans ce cas, le choix de s est dit *pseudo-aléatoire* et il est partagé entre une exploitation des meilleurs résultats et une exploration aléatoire pure.
2. S correspond à la formule 4.4 de *Ant System* et le choix de s est partagé entre l'exploitation des meilleurs résultats et une exploration aléatoire biaisée par les résultats existants. Ce mécanisme est dit *pseudo-aléatoire proportionnel.*
3. S correspond à la formule 4.4 et $p_0 = 0$. Cela signifie que S est systématiquement choisie pour déterminer le prochain sommet s. Cette formule correspond au mécanisme de sélection de *Ant System*, on le dit *aléatoire proportionnel.*

Les tests effectués par les auteurs montrent que le modèle *pseudo-aléatoire proportionnel* est supérieur aux deux autres pour certaines valeurs de p_0. C'est ce modèle de sélection qui est repris dans l'algorithme *Ant Colony System* [DG97a] [DG97b].

Pour aider à la compréhension des différents modèles présentés ici, la figure 4.1 montre un graphe d'héritage qui illustre les relations entre méthodes.

4.1.2 *Ant-Based Control*, fidèle à la métaphore fourmi

Ant-Based Control [SHBR97] est un algorithme très proche de la métaphore fourmi qui exploite l'idée d'un système composé d'entités autonomes au fonctionnement to-

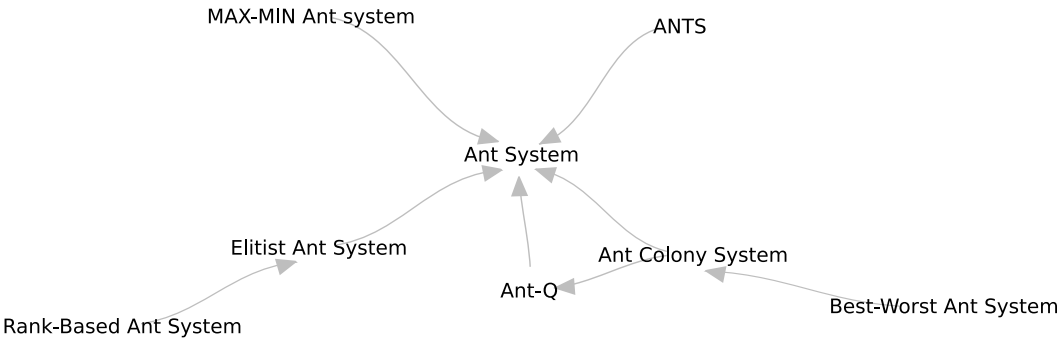


FIGURE 4.1: Graphe d'héritage entre les *ACOs*. Un lien orienté de B vers A signifie que B hérite de l'algorithme A pour construire sa propre contribution.

talement décentralisé. Cet algorithme est appliqué à un problème naturellement distribué de répartition de charge dans un réseau de télécommunication. Ce réseau est composé de routeurs sensés assurer la communication de bout en bout entre des paires de clients. Chaque routeur possède une capacité (un nombre maximum de communications simultanées possibles). Le but est non seulement de trouver des routes relativement courtes mais aussi d'équilibrer la charge et d'éviter les congestions.

L'idée générale est que des agents (fourmis) sont dispersés dans ce réseau, qu'ils s'y déplacent et qu'ils mettent collectivement à jour les tables de routage dans les routeurs du réseau. Ces tables servent à router les appels téléphoniques. Les appels modifient la charge du réseau et influencent indirectement le comportement des fourmis. Du fait de la nature distribuée du réseau, les décisions de routage et les modifications de tables de routage se font localement, sans évaluation centralisée.

Tables de phéromone

Dans les problèmes de routage classiques, chaque nœud possède une table qui permet de déterminer pour chaque sommet du réseau quelle est la route à prendre. Cette route est déterminée la plupart du temps de proche en proche : la table de routage d'un sommet A indique lequel des voisins de A doit être traversé pour atteindre un sommet destination.

Dans *ABC* (*Ant-Based Control*), les tables de routage sont remplacées par des tables de phéromone. Celles-ci n'indiquent plus pour chaque nœud vers quel voisin aller mais la probabilité pour chaque voisin d'être le suivant. Les tables de phéromone ne contiennent donc pas explicitement des valeurs numériques quantitatives comme dans les *ACOs* mais l'effet produit reste identique puisque les phéromones des *ACOs* sont exprimées en probabilités lors de la procédure de choix des fourmis.

Ainsi chaque sommet possède pour chaque nœud du réseau une table de probabilités avec autant d'entrées que le sommet a de voisins. La somme des probabilités des voisins d'un sommet pour une destination donnée est 1. Les fourmis choisissent leur prochain nœud à l'aide d'un tirage aléatoire proportionnel aux probabilités de la table de routage.

Les appels téléphoniques sont routés d'une manière (plus) classique en utilisant les tables de phéromone et en sélectionnant systématiquement les voisins avec les meilleures probabilités.

Renforcement des tables de routage

Les fourmis en se déplaçant mettent à jour les tables de phéromone. Une fourmi choisissant d'emprunter le voisin v du sommet s pour se rendre à la destination d va mettre à jour la table de phéromone de d dans s de la manière suivante :

$$\begin{aligned} p_d^s(v) &= \frac{p_d^s(v) + \Delta p}{1 + \Delta p} \\ p_d^s(u) &= \frac{p_d^s(u)}{1 + \Delta p} \qquad \forall u \in voisins(s) \backslash \{v\} \end{aligned} \tag{4.11}$$

avec $voisins(s)$ l'ensemble des voisins de s et Δp le renforcement des probabilités à définir. Cette formule assure que la somme des probabilités des voisins de s pour la destination d vaut 1.

Plus court chemin et équilibrage de charge

Pour permettre la construction de plus courts chemins il faut que les tables de phéromone guident les fourmis de préférence vers les plus courtes pistes. Pour faire transparaître cette idée de longueur des chemins, le dépôt de phéromone des fourmis est proportionnel au trajet déjà effectué par la fourmi, de sorte que celles ayant parcouru un long chemin influencent moins les tables de routage que les fourmis ayant parcouru de plus courts chemins. Si a est l'âge en nombre de sauts d'une fourmi, les auteurs proposent, après expérimentations, la valeur de renforcement de phéromone suivante : $\Delta p = \frac{0.08}{a} + 0.005$. Ce mécanisme favorise donc les chemins les plus courts.

L'équilibrage de la charge est mis en œuvre en ralentissant les fourmis sur les sommets congestionnés (dont la bande passante arrive à saturation). Le fait de bloquer une fourmi sur un sommet congestionné empêche le dépôt des phéromones, n'améliore donc pas l'attactivité de ce nœud et fait aussi diminuer l'affluence des communications via celui-ci. Expérimentalement le délai pendant lequel les fourmis sont bloquées sur les sommets congestionnés est fixé à $\lfloor 80.e^{-0.075.s} \rfloor$ avec s la capacité restante sur le sommet.

4.2 Les différentes classes de problèmes abordés

4.2.1 Les problèmes d'affectation

Les problèmes d'affectation consistent en la recherche de couples dans un graphe bipartite de manière à optimiser (maximiser ou minimiser) la somme des poids des arêtes sélectionnées dans le graphe. Une illustration de ce problème est l'affectation d'un ensemble de tâches à un ensemble d'employés dans une entreprise. Chaque employé peut effectuer une seule tâche à la fois et possède des compétences différentes pour chaque tâche. Le but est de maximiser la somme de toutes les compétences utilisées.

On peut aussi formaliser le problème avec 2 ensembles A et B et une matrice M de coûts entre les éléments de A et B. Le problème consiste alors à trouver la fonction $f : a \mapsto b$ qui maximise ou minimise la somme :

$$\sum_{a \in A} M[a, f(a)] \tag{4.12}$$

Plusieurs problèmes connus d'affectation ont été traités à l'aide d'algorithmes fourmis, parmi ceux-ci le problème d'affectation quadratique ou le problème de coloration de graphe.

Affectation quadratique

Le problème d'affectation quadratique considère un ensemble U de n unités et un ensemble S de n sites. Il existe des flux entre les différentes unités et une distance entre les sites. La fonction $f : U \times U \rightarrow \mathbb{R}$ définit le flux entre deux unités de U. La fonction $d : S \times S \rightarrow \mathbb{R}$ donne la distance entre les sites. Soit $\psi : U \rightarrow S$ la fonction qui a chaque unité associe un site. Le problème est de définir quel site associer à quelle unité (quelle fonction ψ) de façon à minimiser la somme des produits des distances et des flux :

$$\sum_{u \in U} \sum_{v \in U} f(u, v).d(\psi(u), \psi(v)) \tag{4.13}$$

Les algorithmes à base de fourmis utilisés pour le problème d'affectation quadratique le modélisent sous forme d'un graphe $G = (V, E)$ où V est l'ensemble des sites et des unités. E est l'ensemble des liens. Le graphe est complet.

La famille des ACOs est largement utilisée pour résoudre ce problème. L'idée générale de tous les ACOs tient en quelques principes. Le sens de l'affectation est fixé par avance : les unités sont affectées aux sites, ou le contraire. Dans la suite on suppose que les unités sont affectées aux sites. Ensuite l'ordre dans lequel les unités vont être traitées est défini. Cet ordre peut être aléatoire, fixé par avance ou pas. Puis, les fourmis construisent des chemins dans le graphe G. Chaque chemin représente une affectation valide (une solution réalisable) dans l'espace de recherche. Enfin, les solutions générées

sont améliorées grâce à des heuristiques locales. Ces solutions sont finalement marquées de phéromone et permettent d'influencer la construction des futures solutions générées. Le processus est itératif autour de ce mécanisme.

Parmi les différents ACOs proposés, on retrouve une adaptation de *Ant System* nommée *AS-QAP* [MC99] qui est la première adaptation du modèle au problème. Les algorithmes *ANTS* [Man99] et $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System* [SH+00] déjà présentés traitent également ce problème.

HAS-QAP [GTD99] est une proposition inspirée des *ACOs* mais fonctionnant différemment. Les pistes de phéromone ne sont pas utilisées pour fabriquer de nouvelles solutions mais pour améliorer des solutions déjà existantes. À chaque fourmi est associée une solution complète. Le but est d'améliorer cette solution en effectuant des permutations de sites entre les unités. À chaque cycle, sur chaque solution, deux unités sont choisies en fonction d'une formule aléatoire proportionnelle aux phéromones présentes. Les sites de ces deux unités sont échangés. La solution ainsi générée est souvent moins bonne que la solution d'origine mais elle est ensuite optimisée en utilisant une heuristique locale. Ce mécanisme vise à faire sortir la solution d'un optimum local grâce à l'échange de sites dans un premier temps, puis à l'optimiser grâce à l'heuristique locale dans un second temps.

Coloration de graphe

Ce problème consiste à affecter à chaque nœud d'un réseau une couleur de sorte que deux voisins n'aient pas la même couleur et que le nombre total de couleurs utilisées dans le graphe soit minimal. Ce nombre ne peut être inférieur à la taille de la clique maximale (plus grand sous-graphe complet) du graphe. Ce nombre minimal de couleurs est appelé nombre chromatique du graphe, il est apparenté à la notion de stable. Un stable est un sous-ensemble d'un graphe dont les sommets ne sont pas adjacents. La coloration d'un graphe est donc une partition de l'ensemble des sommets en stables. La recherche d'une coloration minimale consiste à minimiser le nombre de stables.

Dans *ANTCOL* [CH97] les auteurs considèrent que les nœuds du réseau à colorer sont des unités et les couleurs sont des ressources. La qualité d'une solution dépend à la fois du choix des couleurs pour les nœuds mais aussi de l'ordre de sélection de ceux-ci. La construction d'une solution se fait donc en deux temps, d'abord la sélection du prochain sommet à colorer puis la sélection de la couleur.

Les auteurs observent qu'une procédure de choix de la séquence de nœuds à colorer s'adaptant à la coloration partiellement réalisée donne de meilleurs résultats qu'une liste fixée à l'avance. Ainsi ils utilisent deux heuristiques de choix. La première sélectionne de préférence les nœuds avec le degré de saturation le plus élevé. Le degré de saturation d'un sommet est le nombre de couleurs différentes utilisées dans son voisinage. La seconde heuristique tente de maximiser la taille des stables. Quand un sommet est sélectionné et que sa couleur est attribuée, la méthode sélectionne d'autres nœuds

de manière à maximiser la taille du stable courant.

Le choix des couleurs est conditionné par les solutions déjà proposées. Ainsi la probabilité de choix d'une couleur donnée (d'un stable) pour un sommet est inversement proportionnelle au nombre chromatique des solutions passées où le sommet en question était affecté de cette couleur.

ANTCOL est un algorithme qui calcule une coloration exacte, les solutions produites respectent les contraintes de coloration (pas de voisins de couleur identique). D'autres auteurs se sont intéressés à une version moins rigide du problème de coloration. Ainsi J. SHAWE-TAYLOR et J. ZEROVNIK [STZ01] proposent un algorithme dont le but n'est pas de construire des solutions mais d'améliorer itérativement une solution non valide jusqu'à ce qu'elle le devienne.

4.2.2 Les problèmes d'ordonnancement

Les problèmes d'ordonnancement consistent à organiser l'exécution d'un ensemble de tâches sur un ensemble de ressources. Les tâches ont des durées et des contraintes de précédence entre elles. Le but dans ces problèmes est de minimiser la date de fin d'exécution de l'ensemble des tâches.

Le problème d'atelier à cheminements multiples (*Job-shop problem*) se pose lorsque plusieurs tâches ont besoin d'être usinées sur plusieurs machines dans un ordre donné. Le but est de trouver une séquence qui minimise la durée totale de production. Dans [vdZM99] les auteurs appliquent l'algorithme *Ant System* en proposant une modélisation du problème à l'aide d'un graphe. Si n est le nombre de tâches et m le nombre de machines, la matrice $T(n \times m)$ représente pour chaque tâche son ordre d'utilisation des machines. Les cellules de la matrice correspondent aux nœuds du graphe. Les nœuds sont reliés pour former un graphe complet. Les fourmis construisent ensuite des chemins dans ce graphe qui correspondent à des séquences. Une liste tabou des nœuds déjà visités ainsi qu'une liste des nœuds disponibles sont maintenues pour garantir la production de solutions réalisables.

Le problème de durée pondérée (*weighted tardiness*) dans le contexte d'ateliers à cheminement unique concerne la définition de l'ordre d'exécution de tâches sur une seule machine de sorte que la somme des durées pondérées des tâches soit minimisée. Dans [dBSD00] les auteurs s'attaquent à ce problème en utilisant *Ant Colony System*.

Beaucoup d'autres problèmes d'ordonnancement ont été traités avec des algorithmes fourmis.

4.2.3 Les problèmes d'ensembles

Les problèmes d'ensembles en général désignent des problèmes qui consistent à sélectionner un ou des sous-ensembles à partir d'un ensemble de départ de manière à optimiser une fonction d'évaluation. Le choix d'un élément de l'ensemble a une conséquence sur les autres, il y a donc des relations de dépendances entre les éléments.

Le problème du sac à dos

Ce problème d'ensemble consiste à sélectionner des objets possédant un poids p ainsi qu'une valeur v. Le but est de maximiser la valeur totale du sous-ensemble sélectionné tout en considérant une contrainte sur le poids total à ne pas dépasser. Ce problème est comparé au remplissage d'un sac à dos par des objets. Ceux-ci ont un poids et une valeur et le poids total qu'il est possible de mettre dans le sac à dos est limité. Les objets o susceptibles d'être sélectionnés sont numérotés de 1 à n. Une sélection d'objets peut être codée sous la forme d'un vecteur binaire. Ainsi l'objet o_i vaut 1 s'il est sélectionné et 0 sinon. La valeur totale d'une sélection qui est à maximiser vaut $V = \sum_{i=1}^{n} o_i.v_i$. Cette valeur est contrainte par le poids de la sélection $P = \sum_{i=1}^{n} o_i.p_i$ qui ne doit pas dépasser la capacité du sac à dos P_c.

Le problème peut être généralisé à plusieurs contraintes. Si l'on considère m contraintes par objet, le poids d'un objet o_i devient $P_i = \sum_{j=1}^{m} p_i^j$. Le problème peut donc être formulé ainsi :

$$\begin{aligned} &\text{maximisation de } \sum_{i=1}^{n} o_i.v_i \\ &\text{sous la contrainte } \sum_{i=1}^{n}\sum_{j=1}^{m} o_i.p_i^j \end{aligned} \tag{4.14}$$

La principale contribution concernant les fourmis pour le problème du sac à dos est celle de G. LEGUIZAMON et Z. MICHALEWICZ [LM99]. Leur proposition mime celle de *Ant System* avec le problème du voyageur de commerce. La principale différence avec le problème du voyageur de commerce est que dans celui-ci le nombre d'éléments (de villes) à sélectionner est constant, il faut sélectionner toutes les villes et seul l'ordre de sélection compte. Dans le problème du sac à dos, le nombre d'éléments est variable et l'ordre de sélection n'importe pas. Dans *Ant System* pour le problème du voyageur de commerce la notion de cycle est importante. Après la sélection de tous les éléments, l'évaluation des solutions ainsi que le renforcement des pistes de phéromone sont effectués. Pour coller au comportement de *Ant System* les auteurs proposent une notion de cycle avec un nombre d'itérations fixé en paramètre. Après ce nombre d'itérations, chaque fourmi évalue sa solution. Une solution est un remplissage itératif de sac à dos. Le choix des objets dépend des phéromones déposées sur les objets ainsi que de l'heuristique locale qui donne la « désirabilité » d'un objet en fonction du profit qu'il génère et des coûts qu'il engendre. La principale innovation consiste à effectuer le dépôt de phéromone sur les sommets du graphe qui représentent les objets plutôt que sur les liens comme cela est fait avec le problème du voyageur de commerce.

Le problème d'ensemble indépendant maximal

Le problème d'ensemble indépendant maximal consiste dans un graphe $G = (V, E)$ à trouver le sous ensemble maximal de V tel que deux sommets quelconques de cet ensemble ne soient pas reliés par une arête dans le graphe. Soit $V^* \subset V$, $\forall u, v \in V^*, (u, v) \notin E$ et $|V^*|$ est maximal. Les mêmes auteurs que précédemment pour le problème du sac à dos proposent la même adaptation de *Ant System* [LM01]. Pour ce nouveau problème seule l'heuristique locale est différente.

Clique maximale dans un graphe

Une clique dans un graphe est un sous-ensemble de sommets tous connectés les uns aux autres. Une clique est donc un sous-graphe complet du graphe étudié. Le problème de la clique maximale consiste à trouver le plus grand sous-graphe complet dans un graphe. Le cardinal de ce sous-graphe correspond d'ailleurs au nombre chromatique vu plus haut.

Dans [FS03], S. FENET et C. SOLNON proposent *Ant-Clique*, un algorithme dérivé de $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System* [SH+00] qui construit itérativement des cliques. Chaque fourmi de la colonie construit une clique qui est évaluée et la plus grande clique construite est renforcée. L'originalité de cette approche est qu'aucune heuristique locale n'est utilisée pour construire les cliques et seules les pistes de phéromone influencent les fourmis pour la sélection des sommets. De ce fait l'initialisation des pistes de phéromone est cruciale. Les auteurs proposent deux méthodes d'initialisation. D'abord ils reprennent l'une des méthodes de $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System* qui consiste à initialiser les arêtes avec la quantité maximale de phéromone pour assurer une diffusion maximale de l'exploration. L'autre solution consiste à initialiser les arêtes en utilisant les cliques obtenues lors d'un premier échantillonage de l'espace de recherche.

4.3 Les problèmes ouverts

Les grandes familles de problèmes présentés dans la section précédente montrent des algorithmes aux résultats intéressants d'un point de vue théorique mais aussi d'un point de vue pratique. Néanmoins, les problèmes réels sont souvent soumis à plusieurs contraintes qui ne peuvent pas toujours être formulées simplement.

Prenons l'exemple du routage de véhicules. Ce problème peut être résolu par un processus construit de manière centralisée qui cherche à optimiser une contrainte (minimiser les retards par exemple) tout en connaissant les différentes contraintes (durées de trajets, nature des demandes). Dans la réalité, un tel service de routage soulève d'autres problèmes. En effet, il est souvent pertinent de prendre en compte plusieurs objectifs à optimiser (répartition de charge entre les véhicules, minimisation du temps de tournée, minimisation du nombre de véhicules). Ensuite l'environnement dans lequel évoluent

les véhicules est naturellement distribué, et des contraintes locales (embouteillages) imprévisibles et invisibles de manière globale, peuvent localement remettre en question la solution et forcer une décision locale (au niveau du véhicule). Enfin, des demandes aléatoires ou mal définies peuvent remettre en cause des fonctions d'optimisation statiques.

Cet exemple recèle de nombreuses contraintes non triviales, difficiles à modéliser et à traiter. La notion de contraintes multiples nous entraîne vers l'optimisation multi-objectifs. Le caractère aléatoire de certaines demandes introduit la notion d'incertitude dans le modèle. Enfin l'environnement changeant localement de manière imprévisible est distribué et dynamique. Dans ce contexte l'optimisation globale à l'aide d'une fonction objectif semble difficile à mettre en œuvre. Ces contraintes sont étudiées dans la suite.

4.3.1 Les problèmes multi-objectifs

Les problèmes multi-objectifs caractérisent des problèmes dans lesquels plusieurs contraintes doivent être prises en compte pour construire une solution. L'existence de plusieurs contraintes implique que toutes les solutions ne peuvent pas être comparées sur la base d'une fonction d'objectif simple. En effet il est possible qu'une solution soit plus efficace qu'une autre pour certains objectifs mais pas pour d'autres. On définit alors une relation d'ordre partielle entre les solutions d'un tel problème. Cette relation définit la notion de dominance. Une solution en domine une autre si elle améliore au moins un objectif sans pour autant détériorer les autres.

Deux approches principales existent. Une méthode générale consiste à reformuler le problème multi-objectif pour le transformer en problème mono-objectif. Une fonction qui pondère les différents objectifs est alors créée et est utilisée pour évaluer les solutions produites. L'autre approche consiste à optimiser tous les objectifs indépendamment, en construisant un ensemble de solutions répondant aux contraintes de dominance, c'est un ensemble ou front de *Pareto*.

Le problème de routage de véhicules

La première contribution est le résultat du travail de L. M. GAMBARDELLA, É. TAILLARD et G. AGAZZI [GETA99] pour le problème de routage de véhicule avec fenêtre de temps. Dans ce problème, un dépôt contient des véhicules ainsi que des ressources qui doivent êtres distribuées à des clients à l'aide des véhicules. Le problème est de définir des tournées pour les véhicules qui optimisent plusieurs critères et respectent certaines contraintes comme la capacité des véhicules ou leur nombre. Parmi les critères il y a la minimisation du nombre de véhicules (ou du nombre de tournées), la minimisation du temps total de trajet ou encore la minimisation des délais de livraison.

Dans [GETA99] les auteurs proposent un algorithme fourmi (*MACS-VRPTW*) qui optimise deux critères, le nombre de véhicules et le temps total de trajet. La construction est itérative et deux colonies de fourmis différentes sont utilisées, une pour chaque cri-

tère. Les auteurs utilisent les mêmes mécanismes que *Ant System* dans chaque colonie. Les deux colonies utilisent donc des matrices de phéromones différentes mais partagent une même meilleure solution globale pour les mises à jour de pistes de phéromone. Les solutions sont générées et améliorées de manière itérative. Pour la colonie chargée d'optimiser le nombre de véhicules chaque itération fait diminuer le nombre maximum de véhicules utilisés. La seconde colonie tente d'améliorer les chemins trouvés par la précédente colonie en diminuant le temps total de trajet. Cette méthode n'est néanmoins pas réellement une approche multi-objectif car dans ce mécanisme c'est la première optimisation (le nombre de véhicules) qui guide la recherche. C'est d'abord ce critère qui est optimisé puis l'autre en fonction des solutions du précédent.

Dans [BS03], les auteurs proposent une amélioration de *MACS-VRPTW* qui cherche à optimiser trois critères de manière indépendante. Ces critères sont le nombre de véhicules ($F1$), le temps total de trajet ($F2$) et les dates de livraison ($F3$). Une seule colonie est utilisée et les critères $F2$ et $F3$ sont considérés dans la formule de choix du sommet suivant. Le nombre de véhicules est modélisé en multipliant artificiellement le nombre de sommets correspondant aux dépôts. Le mécanisme est réellement multi-objectif même si le critère $F1$ est fixé à l'avance et ne peut être modifié qu'en réinitialisant la colonie et en retirant au graphe un sommet correspondant à un véhicule.

Le problème de sélection de portefeuille financier

Le problème de portefeuille financier consiste à sélectionner un sous-ensemble de projets pour constituer un portefeuille d'investissement. Plusieurs contraintes doivent être respectées et plusieurs objectifs sont considérés. Dans [DGH+01, DGH+04] les auteurs proposent d'utiliser une adaptation de *Ant Colony System* pour construire des solutions sous forme de fronts de *Pareto* avec K critères pour ce problème.

La modélisation du problème diffère du *TSP* pour *Ant Colony System* car dans le problème de portefeuille le nombre de projets sélectionnés n'est pas constant alors que dans le *TSP* le nombre de villes à visiter est constant. Pour pouvoir adapter l'algorithme fourmi au problème, un choix aléatoire du nombre de projets à sélectionner Ξ est fait à chaque cycle pour chaque fourmi. De même, l'importance relative χ accordée aux K critères est définie pour chacun d'eux. L'algorithme est similaire à *Ant Colony System*, si ce n'est l'utilisation de K matrices de phéromones (une pour chaque critère) ainsi que la distribution de probabilité du choix du prochain sommet qui change avec la formule

4.15 :

$$P(\Psi) = \begin{cases} \dfrac{\left[\sum_{k=1}^{K} \chi_k \cdot (\sum_{j\in\Psi} \pi_{ij}^k)\right]^{\alpha} \cdot [\eta_i(\Psi)]^{\beta}}{\sum_{h\in\Omega}\left(\left[\sum_{k=1}^{K} \chi_k \cdot (\sum_{j\in\Psi} \pi_{hj}^k)\right]^{\alpha} \cdot [\eta_h(\Psi)]^{\beta}\right)} & \text{si } i \in \Omega \\ 0 & \text{sinon} \end{cases} \tag{4.15}$$

avec Ψ la solution en cours de production, α et β des paramètres qui relativisent l'importance des pistes de phéromone par rapport à la visibilité, et Ω l'ensemble des sommets disponibles (n'ayant pas encore été sélectionnés).

4.3.2 Incertitude et dynamique

Les problèmes considérés jusqu'à présent dans ce chapitre sont caractérisés par des espaces de recherche où les informations nécessaires à la construction d'une solution sont bien identifiées et ne sont pas modifiées pendant cette construction. L'approche *ABC* (*Ant-Based Control*) utilise la nature décentralisée des individus pour disperser des fourmis dans un véritable réseau de communication décentralisé. L'une des caractéristiques qui est rarement utilisée est la capacité qu'ont les colonies de fourmis à s'adapter aux changements de leur environnement. En effet, même après la construction d'un chemin, les fourmis sont capables dans une certaine mesure d'améliorer ou de modifier ce chemin si les conditions de l'environnement l'imposent. Il est donc possible d'utiliser des algorithmes fourmi pour des problèmes modélisés par des environnements qui évoluent, c'est-à-dire dont les informations changent pendant l'exécution de l'algorithme. Les graphes dynamiques sont un support adapté à la modélisation des problèmes qui nous intéressent ici.

Les algorithmes vus plus haut dans ce chapitre utilisent une modélisation de l'environnement sous forme de graphes. Les graphes dynamiques, tels qu'ils ont été définis au chapitre 3, permettent de modéliser des environnements dynamiques.

Un *ACO* pour le problème dynamique du voyageur de commerce

Le *Dynamic TSP* (*Dynamic Traveling Salesman Problem*) ou problème dynamique du voyageur de commerce est une modification du problème traditionnel dans lequel les temps de parcours peuvent évoluer pendant l'exécution de l'algorithme de recherche d'une solution. Certaines versions du problème prennent également en compte le fait qu'une ville soit ajoutée ou retirée ou encore que certaines routes disparaissent ou apparaissent. En termes de graphes, il faut distinguer le cas où seules des modifications de temps de parcours sont prises en compte du cas où des villes et des routes apparaissent

et disparaissent. Dans le premier cas la dynamique du problème réside dans des changements de valeurs des poids des arêtes du graphe. Dans le second cas la dynamique est structurelle et le graphe qui modélise le problème voit son nombre d'arêtes et de sommets changer au cours de l'exécution. La différence entre ces deux cas est importante car dans le premier, toutes les solutions générées sont comparables entre elles alors que dans le second cas, une solution générée dans une configuration structurelle du graphe ne peut être comparée avec une solution construite dans une autre configuration.

Dans [ES02] les auteurs s'intéressent à la première version du problème où seuls les temps de parcours sont modifiés. Ils traitent le problème des embouteillages qui se produisent sur les routes trop fréquentées en augmentant les temps de trajets. Leur proposition reprend l'algorithme *Ant Colony System* en ajoutant un mécanisme de lissage des pistes de phéromone nommé *shaking*. Ce mécanisme est appliqué de manière localisée autour des routes encombrées et traite les routes avoisinantes. L'idée est de diminuer l'attraction des routes encombrées ainsi que des routes qui y mènent sans pour autant annuler l'effet des phéromones.

Dans [GM02b, GM02a] M. GUNTSCH et M. MIDDENDORF considèrent la seconde version du problème, où les routes et les villes peuvent disparaître ou apparaître. Ils présentent une méthode également dérivée de *Ant Colony System*. Ils proposent de supprimer le mécanisme d'évaporation au profit d'un mécanisme de mémorisation des meilleures solutions à fenêtre glissante. A chaque cycle, après la construction et l'évaluation des solutions, la meilleure est sauvegardée dans une file de taille fixe k. Si la file est pleine, la solution générée $k+1$ étapes auparavant est supprimée. C'est l'ensemble des solutions de la file qui participent au renforcement des pistes.

L'héritage de *ABC*

L'approche *ABC* (*Ant-Based Control*) présentée page 56 est conçue pour s'exécuter dans l'environnement décentralisé et dynamique que forme un réseau de communication. Comme il a été dit plus haut l'approche est restée fidèle au modèle naturel. Beaucoup de travaux héritent de ce modèle.

Citons *AntNet* [DD98] ainsi que *AntHocNet* [DCDG04] par G. DI CARO et ses collègues. Même si ces approches sont apparentées aux *ACOs* elles sont néanmoins très proches de *ABC* de par leur fonctionnement distribué et leur adaptation à la dynamique. *AntNet* est un algorithme de routage pour les réseaux infrastructurés tels qu'Internet. *AntHocNet* quant à lui est un protocole de routage hybride pour les réseaux sans fil et mobiles *ad hoc*.

Toujours dans la lignée de l'approche *ABC*, $AntCO_2$ [Dut05, CDGO06, BDGO07] par A. DUTOT est un algorithme de répartition de charge pour des applications distribuées sur une grappe de machines. L'algorithme a la particularité de s'exécuter dans le même environnement que l'application qu'il doit distribuer. L'approche est totalement décentralisée et s'adapte dynamiquement aux variations de l'application ainsi qu'aux modifi-

cations du réseau de machines sur lequel elle s'exécute.

4.4 Conclusion

Les algorithmes fourmis ont montré leur capacité à traiter avec succès un ensemble très conséquent de problèmes d'optimisation traditionnels caractérisés par des informations fiables et constantes. La résolution de ces problèmes consiste dans la plupart des cas à minimiser une fonction de coût.

Lorsque la dynamique, l'incertitude ou l'absence de vision globale sont au cœur de la définition des problèmes, certains travaux d'une inspiration proche de *ABC* ont été proposés ces dernières années. La majorité de ces travaux repose sur une approche décentralisée dépourvue de fonction de coût globale et les méthodes proposées sont capables de s'adapter aux changements qui se produisent en cours de résolution.

Le chapitre suivant s'attache à la description d'une approche générale pour la résolution de ce type de problèmes.

CHAPITRE 5

APPROCHE GÉNÉRALE D'OPTIMISATION DÉCENTRALISÉE POUR LES GRAPHES DYNAMIQUES [1]

A notre connaissance il n'a pas encore été décrit d'approche générale ou systématique pour le traitement de problèmes modélisés par des graphes dynamiques comme il en existe pour les problèmes statiques (comme les *ACO* par exemple). Les approches proposées jusqu'à présent pour la résolution de problèmes dynamiques sont conçues sur mesure. Elles ne s'insèrent pas dans un schéma général de résolution et sont, de ce fait, très peu adaptables à de nouveaux problèmes. Ce constat rejoint la remarque que nous avions faite dans la partie I à propos des graphes dynamiques qui à l'exception des graphes évolutifs sont des modèles *ad hoc*.

1. Ce chapitre a fait l'objet d'une publication sous le titre « *Problem Solving and Complex Systems* » [Guinand and Pigné(2006)], dans l'ouvrage « *Emergent Properties in Natural and Artificial Dynamical Systems* » [AB06].

L'ambition de ce chapitre est de faire le lien entre les systèmes complexes, les colonies de fourmis et l'optimisation décentralisée de problèmes modélisés par des graphes dynamiques. L'approche que nous décrivons considère les colonies de fourmis comme des systèmes complexes dont les propriétés obtenues par auto-organisation se matérialisent par la mise en évidence de structures dans le graphe dynamique.

5.1 Approche générale

Dans la majorité des méthodes méta-heuristiques telles que les algorithmes génétiques, les processus de recherche tabou, les méthodes *PSO* ou encore les *ACOs*, les solutions aux problèmes posés sont explicitement construites puis évaluées afin de diriger la recherche vers de meilleures solutions. Lorsque les individus ne construisent pas une solution, ils sont une solution ou représentent l'information nécessaire à sa construction (*e.g.* une particule dans la méta-heuristique *PSO* ou un chromosome dans les algorithmes génétiques). Nous sommes convaincus que ce mécanisme n'est pas le seul qui puisse être mis en œuvre pour la résolution de problèmes d'optimisation. Nous pensons que la construction d'une solution ne doit pas forcément être réalisée par une seule entité ni que cette construction doit être connue des entités du système.

Dans les problèmes modélisés par des graphes, une solution à un problème d'optimisation peut souvent être associée à une structure dans ce graphe. Une structure peut être un chemin, un ensemble de sommets, un sous-graphe, *etc.* Le principe général de l'approche que nous proposons repose sur la conception de méthodes de construction et de maintien de structures particulières dans les graphes dynamiques. Ainsi lorsque le graphe change, la structure est mise à jour. L'une des difficultés dans cette approche réside dans le choix de méthodes qui puissent s'adapter à la dynamique du problème, et les résultats de *ABC*, *AntHocNet*, *AntNet* et $AntCO_2$ semblent indiquer que les approches à base de fourmis constituent un choix pertinent.

Les contraintes associées à cette approche sont les suivantes :

- le problème doit être modélisé par un graphe dynamique ;
- les solutions doivent pouvoir s'exprimer sous forme de structures dans le graphe : chemin, arbre, sous-graphe, *etc.*

Les entités quant à elles, les fourmis artificielles, sont créées, évoluent et parfois meurent. Leur comportement est simple et localisé dans le graphe. Elles exécutent deux types d'actions conditionnées par leur perception locale. En premier lieu elles peuvent se déplacer dans le graphe de manière probabiliste. Ces mouvements peuvent être influencés par les informations que recèle l'environnement. En second lieu, elles peuvent modifier l'environnement de deux manières, soit en déposant des marques comme les phéromones, soit en modifiant la structure. Ces deux types de modifications constituent les moyens de communication asynchrone dont elles disposent. La modification de l'environnement poursuit deux objectifs, la communication asynchrone entre les entités et la formation de structures représentant les solutions aux problèmes posés. D'une ma-

nière générale la modification de l'environnement a un effet rétroactif sur le comportement des colonies de fourmis comme l'illustre la figure 5.1.

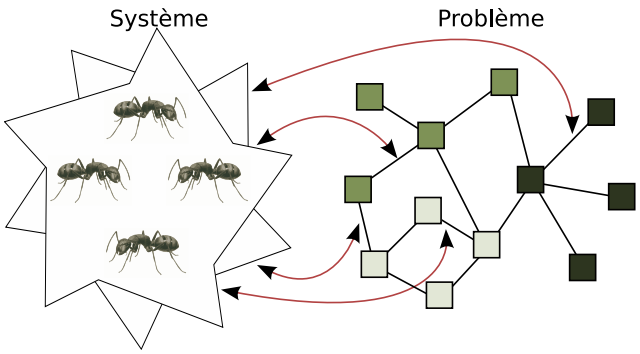


FIGURE 5.1: Principe général de résolution de problèmes à l'aide d'un système complexe matérialisé par une colonie de fourmis. Le système agit sur l'environnement qui rétroagit sur le comportement du système.

Les colonies de fourmis artificielles sont vues comme des systèmes complexes qui exhibent des propriétés émergentes dues au mécanisme d'auto-organisation. Ces propriétés se manifestent dans le graphe par la mise en évidence de structures qui représentent les solutions aux problèmes que l'on se pose.

Le processus général de résolution de problèmes à l'aide de systèmes complexes qui est décrit dans la suite comprend 3 étapes :

1. identification de structures dans le graphe pouvant être associées à des solutions au problème posé ;
2. configuration des colonies de fourmis pour qu'elles produisent les structures recherchées ;
3. paramétrage du système.

5.2 Identification des structures

La première étape avant la construction du système de résolution est la formalisation du problème sous forme de graphe ainsi que l'identification d'une solution sous forme de structure dans ce graphe. La table 5.1 et la figure 5.2 présentent quelques exemples de structures solutions pour certains problèmes d'optimisation classiques.

Les structures solutions dépendent entièrement de la modélisation du problème. Ainsi, à différentes modélisations d'un problème donné, correspondront des structures solutions également différentes.

TABLE 5.1: Exemples de problèmes et de structures solutions.

Problème	**Structure solution**
problème de clique maximale [FS03]	ensemble de sommets
Routage [DD98]	ensemble de chemins
Répartition dynamique de charge [BDGO07]	ensembles de sommets
TSP [DMC96]	circuits
Alignement multiple de séquences [Guinand and Pigné(2008)]	ensembles d'arêtes
Plus court chemin (voir le chapitre 7)	chemins ou arbres

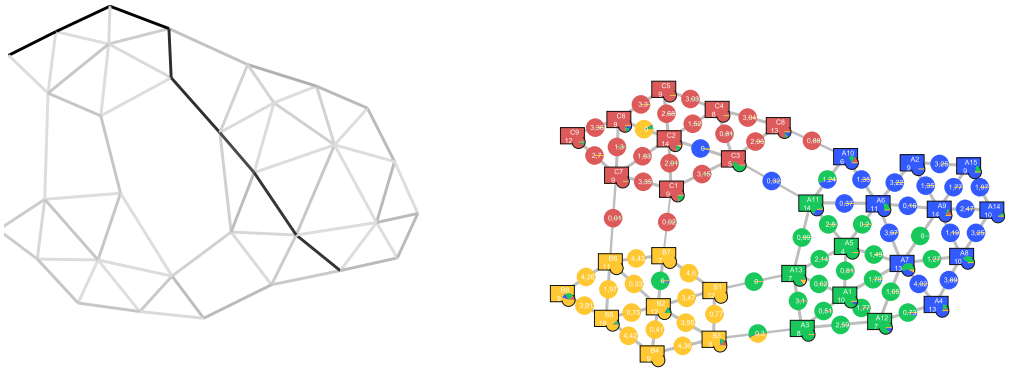


FIGURE 5.2: Exemple de deux structures différentes dans un même graphes qui sont aussi deux solutions pour différents problèmes. La structure de droite est une répartition de la charge d'une application sur plusieurs machines construite par A. DUTOT, celle de gauche est un chemin dans ce graphe.

5.3 Le modèle

Nous avons rappelé que l'une des caractéristiques des colonies de fourmis est leur capacité à modifier l'environnement qui leur sert de support, à la fois pour communiquer de façon asynchrone, et pour construire collectivement des structures. Ces modifications de l'environnement rétroagissent sur leur comportement. Soulignons également que l'indépendance des individus autorise la conception d'algorithmes fourmis distribués.

Nous considérons trois types d'éléments dans les systèmes à base de fourmis artificielles : les nids, les colonies et les fourmis. Deux échelles d'observation différentes sont envisageables, le niveau des colonies et le niveau des fourmis. Les nids sont des éléments (généralement des sommets) au sein desquels sont créées les fourmis.

5.3.1 Les nids et les colonies

Une colonie représente un ensemble de fourmis qui partagent des caractéristiques communes. Même si les individus restent autonomes leur comportement et leurs affinités sont identiques. L'utilisation de plusieurs colonies permet de définir plusieurs comportements de recherche dans l'environnement et donc de conduire différentes heuristiques pour un problème donné. Ceci peut être utile dans le contexte d'une recherche multi-objectif où chaque objectif sera confié à une colonie particulière. La multiplication de colonies permet également de mettre en œuvre un mécanisme de compétition entre elles qui conduit à d'autres phénomènes émergents utiles pour la répartition de charge par exemple.

Les nids quant à eux sont des points de départ pour les fourmis dans le graphe. Une colonie peut disposer d'un ou plusieurs nids. Dans le cas de problèmes où tous les sommets doivent générer des fourmis, comme les problèmes de routage, chaque sommet du graphe possède un nid. Si le problème est lié à la construction d'un chemin, il est alors vraissemblable qu'un seul nid soit utilisé comme point de départ du chemin recherché.

Plusieurs configurations sont donc envisageables, entre plusieurs colonies avec un seul nid en compétition pour une certaine ressource et une seule colonie disposant de plusieurs nids comme on l'observe dans la nature avec les supercolonies de la fourmi argentine [GPK02].

5.3.2 Les fourmis

Les fourmis sont caractérisées par trois traits essentiels : leur comportement, leur mode de déplacement, et la manière dont elles procèdent au dépôt de phéromones.

Caractéristiques générales

Les fourmis sont autonomes et indépendantes les unes des autres, elles peuvent donc être exécutées de manière concurrente et asynchrone. Chaque entité réagit aux modifications de son environnement local défini par un voisinage constitué d'arêtes et de sommets. Les informations locales dont chaque fourmi dispose sont de deux types. D'abord une valuation des arêtes et des sommets est possible si le problème considéré le permet. Ces informations sont spécifiques au problème abordé. Des métriques locales sur le graphe (comme le degré d'un sommet) peuvent également être une indication mesurable et utilisable localement par les fourmis. Le second type d'information est celui généré par les fourmis elles-mêmes : les phéromones et autres modifications opérées par les fourmis sur le graphe.

En fonction des problèmes considérés les fourmis se voient attribuer des caractéristiques particulières. Par exemple :

– la définition d'une couleur associée à une colonie permet d'introduire de la compétition ;

– une liste tabou permet d'empêcher une fourmi de visiter de manière répétée un même ensemble de sommets, elle assure ainsi une meilleure exploration de l'espace de recherche et permet d'éviter les phénomènes d'agglomération ;
– un paramètre de sensibilité aux pistes de phéromones est souvent défini pour équilibrer l'attraction des pistes par rapport aux heuristiques locales.

Déplacement et dépôt de phéromones

Le but de chaque individu à tout moment est de se déplacer et de marquer son environnement. Le déplacement est influencé par des informations et des contraintes locales.

Le renforcement local des pistes est effectué à chaque déplacement d'une fourmi, il représente une valeur constante ou calculée à partir des informations locales. Ce renforcement est local et ne correspond à aucune évaluation globale. Ce mécanisme a été proposé dans *Ant Colony system* [DG97a] [DG97b] par M. DORIGO et L. M. GAMBARDELLA mais aussi par R. SCHOONDERWOERD et ses collègues dans *Ant-Based Control* [SHBR97].

On parle parfois, de manière ambiguë, de renforcement global, lorsque le mécanisme de renforcement est déclenché par une fourmi sur la base d'informations glanées pendant son déplacement, informations qui ne sont globales mais collectées localement. Dans l'exemple de la recherche d'un plus court chemin, une fourmi arrivée à destination peut avec ses connaissances faire le chemin à l'envers vers le nid en déposant des phéromones. Le volume de ce dépôt peut dépendre du chemin réalisé. L'information utilisée ici est à l'échelle du chemin entier, néanmoins cette connaissance existe dans la fourmi qui l'a construit et ce mécanisme est opérationnel de manière distribuée.

Les phéromones jouent leur rôle dans la stigmergie. Déposées par certaines fourmis elles en influencent d'autres. Sur un sommet donné, les quantités de phéromones sont attractives et l'attraction est proportionelle aux quantités présentes. Le choix d'un nouveau sommet pour une fourmi suit une loi pseudo-aléatoire proportionnelle aux quantités de phéromones présentes dans le voisinage et aux heuristiques locales. Le mécanisme d'abord formalisé dans l'algorithme *Ant-Q* [GD95] puis repris dans *Ant Colony System* est tout à fait en accord avec nos préoccupations.

D'une manière générale le mode de dépôt de phéromones et le processus de choix du prochain sommet vont être influencés par le type de problème traité et par le type de structure solution souhaité. Si le problème consiste à construire des chemins dans le graphe, alors le mode de dépôt au retour de la fourmi vers le nid est préféré. Si la solution cherchée est un ensemble de sommets ou un sous-graphe, un dépôt constant et itératif semble plus approprié.

5.3.3 Considérations générales sur le modèle

La motivation pour développer un tel système tient en trois propriétés, auto-organisation, robustesse et flexibilité.

Le système ne construit pas explicitement de structures, celles-ci apparaissent comme le résultat des interactions individus/individus et individus/environnement. Ceci est une conséquence de la propriété d'auto-organisation.

Les entités sont capables de s'adapter aux variations de l'environnement et à la dynamique. Les solutions construites s'adaptent aux changements de contexte grâce aux compétences du système, il fait donc preuve de robustesse.

Enfin, la notion de flexibilité peut être observée lorsque certaines entités du système disparaissent soit parce qu'elles l'ont décidé soit parce qu'elles ont disparu avec une partie de l'environnement dynamique. Malgré cette perte, le système reste opérationnel mais peut fonctionner en mode dégradé.

5.4 Conclusion

Dans ce chapitre nous avons proposé une approche générale de résolution de problèmes modélisés par des graphes dynamiques. L'approche inspirée des mécanismes d'intelligence collective observés chez les insectes sociaux adapte naturellement son comportement à la dynamique du graphe. Les solutions à de tels problèmes sont vues comme des structures dans les graphes dynamiques que notre approche construit et maintient pendant les événements dynamiques des graphes.

Contrairement aux *ACOs* dédiés à l'optimisation combinatoire, notre proposition n'utilise que des mécanismes locaux et peut donc fonctionner dans des environnements distribués sans contrôle centralisé. En cela elle s'apparente aux approches telles que *ABC*.

L'applicabilité de l'approche dans un contexte dynamique et décentralisé en fait une méthode de choix pour résoudre des problèmes dans des contextes tels que les réseaux mobiles *ad hoc*.

Troisième partie

Applications aux réseaux mobiles *ad hoc*

INTRODUCTION

En 2008, les réseaux mobiles *ad hoc* constituent une technologie proche de la maturité. Cependant au moins deux raisons font qu'ils ne sont pas encore très répandus au niveau du grand public et qu'aucune publicité n'en vante les mérites. La première raison, qui n'a rien de scientifique, est économique. Il n'existe pas à l'heure actuelle un modèle économique pour ces réseaux dont l'utilisation échappe, par essence, à tout contrôle. La seconde raison qui nous intéresse au premier chef est que leur mise en œuvre effective reste délicate dans la plupart des cas, même si quelques tentatives commencent à être couronnées de succès. Pourtant, il suffit d'observer à la fois l'évolution du taux d'équipement en téléphones portables des populations des pays développés ainsi que l'évolution des technologies qu'ils embarquent pour comprendre et prédire que ces appareils envahiront notre quotidien dans un futur très proche. En 2007 déjà, un sondage publié par TNS-Sofres [2] indiquait un taux d'équipement en téléphones mobiles de 91% pour les 12-24 ans. En 2008 la majorité des téléphones vendus sont équipés d'une interface *Bluetooth* et la presque totalité des *PDAs* sont capables de communications *WiFi* et peuvent donc participer à la constitution d'un réseau mobile *ad hoc*. Il est à noter toutefois que les réseaux mobiles *ad hoc* se présentent dès à présent comme une solution complémentaire dans de nombreuses situations pour lesquelles la mise en place d'une infrastructure serait trop coûteuse (équipement des routes et des autoroutes) ou s'avérerait impossible à prévoir (utilisation sur des sites ou d'autres moyens de communications serait temporairement indisponibles), auquel cas le caractère *ad hoc* prend tout son sens.

Nous nous plaçons donc dans l'hypothèse de l'existence de tels réseaux et nous proposons de nous intéresser à leur fonctionnement. Le principal constat qui peut être fait est que les réseaux mobiles *ad hoc* ne peuvent être mis en œuvre en utilisant les mécanismes des réseaux classiques de communication. En effet l'absence d'infrastructure ainsi que la dynamique rendent inutilisables la plupart des outils et des algorithmes qui opèrent d'ordinaire. Les besoins principaux de ces réseaux se situent donc dans une remise en question des mécanismes classiques, comme le routage, en tenant compte des nouvelles contraintes que sont l'absence d'infrastructure, la dynamique, la volatilité des stations et des liens de communication, les faibles débits ou encore les problèmes

2. http://www.tns-sofres.com/etudes/pol/171007_afom.htm

d'énergie.

Notre thèse est que les algorithmes extraits de l'observation des systèmes complexes naturels présentent d'avantageuses propriétés utilisables dans le contexte des réseaux mobiles *ad hoc*. En effet, le modèle général d'optimisation décentralisé par colonies de fourmis présenté au chapitre 5 fonctionne dans un graphe dynamique de manière décentralisée. Or, les réseaux mobiles *ad hoc* qui sont des environnements totalement décentralisés peuvent êtres modélisés à l'aide de tels graphes dynamiques. La définition de graphe dynamique proposée au chapitre 3 fournit le modèle idéal pour modéliser de tels réseaux ainsi que leur comportement.

Le premier chapitre de cette partie introduit en détails la notion de réseau mobile *ad hoc*. Il montre comment modéliser et simuler ces réseaux mais aussi comment les inclure entièrement (état et comportement) dans notre modèle de graphes dynamiques.

Le second chapitre de cette partie (chapitre 7) traite du besoin de routage dans les réseaux mobiles *ad hoc* et investit le problème de la découverte et du maintien de chemins de communication dans ces environnements aux contraintes multiples. La contrainte de plus court chemin, ainsi que celle de la construction de chemins robustes, explicités dans le chapitre, font de notre proposition une approche multi-objectifs.

Le troisième chapitre de cette partie traite de la construction est du maintien d'une forêt d'arbres couvrants dans un réseau mobile *ad hoc*. La notion d'arbre couvrant est très importante dans ce contexte et est liée au maintien de chemins de communication optimisant les ressources de communication.

CHAPITRE 6

DES RÉSEAUX MOBILES *ad hoc* AUX GRAPHES DYNAMIQUES

6.1 Introduction

Formellement, les réseaux mobiles *ad hoc* sont définis dans les spécifications des normes de la famille *IEEE 802.11* [Com99], dont les *802.11b* et *802.11g* sont les standards de communication implémentés sur la plupart des machines à ce jour. La spécification définit un réseau *ad hoc* comme un réseau composé uniquement de stations à portée

les unes des autres et communicant via un medium sans fil [1]. Cette définition implique qu'une communication de type *ad hoc* sans fil, entre deux stations, ne peut avoir lieu que si celles-ci sont directement à portée radio l'une de l'autre. Autrement dit, la topologie d'un réseau *ad hoc* dans l'esprit de la norme est un graphe complet. En d'autre termes, selon le comité *IEEE Computer Society LAN/MAN Standards*, la communication multi-sauts, via un ou plusieurs tiers n'est pas prévue. Ceci constitue une contrainte majeure pour leur mise en œuvre ainsi que pour les applications envisageables. En conséquence, la plupart des travaux dans ce domaine relâchent cette contrainte et considèrent une version étendue de la définition d'un *MANET* en autorisant les échanges de données entre stations qui ne sont pas à portée radio, voire même entre stations entre lesquelles il n'existe pas toujours un chemin permettant à une donnée de transiter. Pour ce dernier type de réseaux, on parle alors de *DT-MANETs* (*Delay-Tolerant MANET*) [SAR] ou de *réseaux mobiles ad hoc discontinus*.

Dans la section suivante, nous nous intéressons à la définition d'un réseau mobile *ad hoc* en nous appuyant sur certaines propositions extraites de la littérature, nous présentons notre propre définition à partir de laquelle le lien avec les graphes dynamiques est plus direct. La section 6.3 fait le point sur quelques technologies permettant la mise en œuvre de réseaux mobiles *ad hoc*, puis quelques unes des applications parmi les plus citées pour les *MANETs* sont passées en revue, enfin quelques outils permettant de réaliser des simulations de ces réseaux sont présentés succintement, ainsi que le simulateur utilisé dans ce travail. Les modèles d'environnement et de mobilité font l'objet de la section 6.4. Enfin, la dernière section de ce chapitre s'attache à faire le lien entre d'une part les réseaux mobiles *ad hoc* et les graphes dynamiques d'autre part et présente quelques métriques pertinentes dans le cadre des *MANETs*.

6.2 Définition

Si l'on met de côté la définition d'un réseau *ad hoc* inscrite dans la norme *802.11*, de multiples propositions de définitions ont été formulées ces dernières années pour les réseaux mobiles *ad hoc*. Chacune de ces définitions efface le caractère complet de la topologie du réseau, mais presque toutes conservent *a minima* le caractère décentralisé et sans fil.

Dans le travail de M. GERLA et de ses collègues [GLR05], les auteurs proposent une définition dans laquelle la dynamique du réseau apparaît explicitement : « Un réseau mobile *ad hoc* (*MANET*) est un ensemble de nœuds mobiles et sans fil qui peuvent dynamiquement former un réseau sans infrastructure pré-existante » [2]. Les *MANETs* sont créés et maintenus sans infrastructure, et leur formation tout comme leur évolution,

1. « *A network composed solely of stations within mutual communication range of each other via the wireless medium* »

2. « *A mobile ad hoc network (MANET) is a collection of mobile wireless nodes that can dynamically form a network without any pre-existing infrastructure* »

dynamique, ne dépend d'aucun service centralisé. Les auteurs insistent sur la nature pair-à-pair des communications.

L'aspect dynamique était déjà présent dans le travail de I. CHLAMTAC, M. CONTI et J. J.-M. LIU [CCL03] qui en 2003 proposent dans un très long article de faire le point sur les réseaux mobiles *ad hoc*. Sans qu'une définition formelle soit donnée, la notion de réseau mobile *ad hoc* est précisée au début de l'article et dans le corps du document [3] [4]. Les principaux aspects qui sont mis en avant sont à la fois l'aspect dynamique, leur capacité à s'auto-organiser et l'absence de mécanisme central de contrôle.

D'autres définitions ont été proposées, parfois pour préciser le contexte dans lequel ces réseaux sont construits comme par exemple dans le cadre des réseaux mobiles *ad hoc* en environnements métropolitains [CGMT03, BG03].

La diversité des sources permet également de trouver des définitions plus détaillées, qui vont au-delà de la description simple d'un réseau mobile *ad hoc* en introduisant des éléments qui rendent plus évidentes les questions relatives à la mobilité. Ainsi dans le *RFC2501* de l'*IETF*, on peut lire [5] que les terminaux peuvent être localisés sur divers supports allant des avions aux personnes physiques, en passant par toute autre forme de véhicule mobile. Il est bien évident que le mode de déplacement du « support » impacte de manière critique la topologie du réseau et rend les études sur les modèles de mobilité extrêmement pertinentes lorsque l'objet d'étude est le graphe dynamique résultant des connexions entre machines. D'autres éléments sont également à prendre en compte, comme le souligne le même document [6] qui esquisse la question de la modélisation de ces réseaux par des *graphes aléatoires multi-sauts*.

Une autre vision des réseaux mobiles *ad hoc* est proposée par les travaux précurseurs sur les communications inter-planétaires pour lesquelles les délais de communication et la permanence des liens de communication sont hors normes. K. FALL dans un rapport de 2003 [Fal03] parle de *DTN* pour qualifier des réseaux *tolérants les délais* [7], et déclare que dans certains réseaux, notamment les réseaux mobiles sans fil, les dé-

3. « *Mobile ad hoc networks (MANETs) represent complex distributed systems that comprise wireless mobile nodes that can freely and dynamically self-organize into arbitrary and temporary, "ad-hoc" network topologies, allowing people and devices to seamlessly internetwork in areas with no pre-existing communication infrastructure [...]* »

4. « *In general, mobile ad hoc networks are formed dynamically by an autonomous system of mobile nodes that are connected via wireless links without using the existing network infrastructure or centralized administration. The nodes are free to move randomly and organize themselves arbitrarily* »

5. « *A MANET consists of mobile platforms [...] which are free to move about arbitrarily. The nodes may be located in or on airplanes, ships, trucks, cars, perhaps even on people or very small devices, and there may be multiple hosts per router.* »

6. « *MANET nodes are equipped with wireless transmitters and receivers using antennas which may be omnidirectional (broadcast), highly-directional (point-to-point), possibly steerable, or some combination thereof. At a given point in time, depending on the nodes' positions and their transmitter and receiver coverage patterns, transmission power levels and co-channel interference levels, a wireless connectivity in the form of a random, multihop graph or "ad hoc" network exists between the nodes. This ad hoc topology may change with time as the nodes move or adjust their transmission and reception parameters.* »

7. *Delay-Tolerant Networks*

connexions peuvent être plus fréquentes que les connexions, des déconnexions dues pour l'essentiel à la mobilité ainsi qu'à l'économie d'énergie [8]. Pour qualifier les changements topologiques qui sont le fait de la mobilité, il utilise le terme *opportuniste*, un terme qu'empruntent également L. PELUSI, A. PASSARELLA et M. CONTI dans un rapport de 2006 [PPC06] pour qualifier les réseaux mobiles *ad hoc* et pour motiver la recherche de nouvelles approches plus opportunistes pour faire transiter des informations sur les réseaux, sur la base de communications entre voisins [9].

Sur la base de l'ensemble de ces contributions, il nous est possible de proposer une définition sur laquelle nous pourrons nous appuyer pour définir, dans la section 6.5, les graphes dynamiques correspondants à divers réseaux mobiles *ad hoc*.

Définition 21 (RÉSEAU MOBILE *ad hoc*)
Un réseau mobile ad hoc ou MANET (Mobile Ad Hoc Network) est un réseau de communication constitué d'un ensemble de stations généralement mobiles dans leur environnement et capables de communiquer entre elles via des liaisons sans fil. Un tel réseau résulte des interconnexions entre les stations, possède une topologie qui change dynamiquement, ne nécessite ni infrastructure ni contrôle centralisé et peut ne pas être connexe. En outre, un réseau mobile ad hoc est constitué de quatre principaux éléments :
- *les stations avec leurs caractéristiques ;*
- *un modèle de propagation du signal ;*
- *un modèle de comportement des stations, incluant un modèle de mobilité et un modèle de volatilité ;*
- *le modèle d'environnement qui contraint à la fois la propagation du signal et la mobilité des stations.*

Remarque. Très souvent, les modèles de propagation du signal, de mobilité et d'environnement sont regroupés sous le terme « modèle de mobilité », mais nous avons volontairement choisi de les traiter de manière séparée, simplement parce que ce découpage en quatre composantes des réseaux mobiles *ad hoc* se révèle pertinent dès lors

8. « *In many challenged networks, end-to-end disconnection may be more common than connection. [...] Non-faulty disconnections arise most frequently in wireless environments, from primarily two sources : motion and low-duty-cycle system operation. Disconnection due to motion may be highly predictable (e.g. satellite passes, busses that act as data routers, etc) or opportunistic (nodes arrive in communication range due to random walk) and may arise due to motion of either end-nodes, routers, or some other object or signal that obscures the communication.* »

9. « *As the application environments of MANETs increase, their traditional communication paradigms need adequacy. Specifically, today's ad hoc networks are evolving towards opportunistic networks where the proactive and reactive approaches for routing management are integrated, or definitely substituted, with techniques that exploit communication opportunities, whenever they arise, to forward messages on a hop-by-hop basis. In fact, in opportunistic networking no assumption is made on the existence of a complete path between couple of nodes wishing to communicate. This well suits many of the new application scenarios because intermittent connectivity is likely to occur.* »

que l'on se place du point de vue des graphes de connexions. En effet, chacune de ces composante affecte le graphe dynamique d'une certaine façon comme nous l'exposons ci-après.

Les stations. Le nombre et la densité de stations sont deux paramètres du réseau mobile *ad hoc*, auxquels s'ajoutent des caractéristiques propres à chaque station : l'énergie disponible qui peut influer sur la présence de la station dans le réseau, la puissance d'émission dont dépend le rayon de transmission à partir duquel est défini, au moins partiellement, le voisinage de la station, le type d'antenne, qui peut être mono- ou omni-directionnelle, la technologie de l'interface de communication sans fil, *bluetooth* ou de la famille *802.11*, *etc.* En terme de graphes de connexions, les caractéristiques des stations peuvent avoir une influence sur le contour du voisinage, c'est-à-dire ses limites en terme de distance géographique, et donc indirectement sur les propriétés de connexité du graphe dans son ensemble, comme le degré moyen, la distribution des degrés, le nombre de composantes connexes, ou d'autres propriétés globales.

Modèle de propagation du signal. Le modèle de propagation du signal décrit la manière dont le signal se propage dans l'environnement. Ce modèle peut simplement prendre en compte une propagation de type LOS [10], ou bien considérer également les effets de diffraction, d'amortissement ou de réflexion sur les obstacles, voire d'interférences entre les signaux émis par différentes stations. De même que pour les caractéristiques des stations, le modèle de propagation du signal influe directement sur la composition du voisinage des stations.

Modèle de mobilité. Le *modèle de comportement* des stations se réduit très souvent à un *modèle de mobilité*, mais il peut également décrire d'autres aspects du comportement de la station, comme sa propension à s'allumer ou s'éteindre, sa tendance à se désynchroniser, etc. Au niveau du graphe des connexions, le modèle de mobilité joue le rôle du moteur de l'évolution du graphe. Il n'est que marginalement impliqué dans la composition du voisinage des stations. Par contre, de ce modèle peuvent découler un ensemble de propriétés importantes du graphe dynamique comme sa connexité, son hétérogénéité, ainsi que son évolution dans le temps et éventuellement l'apparition de structures au sein du graphe. De fait, la distribution spatiale des stations dépend directement de ce modèle. Il est également important de noter que ces propriétés ne dépendent pas exclusivement du modèle de mobilité, mais également du modèle d'environnement dans lequel les stations évoluent. La section 6.4 est consacrée à un rapide tour d'horizon des modèles de mobilité proposées dans la littérature.

10. *Line of Sight* ou ligne de vue

Modèle d'environnement. Le *modèle d'environnement* contraint à la fois la propagation du signal et la mobilité des stations. Les modèles d'environnements ouverts sont parmi les plus souvent considérés et les plus simples à prendre en compte puisqu'ils n'entravent ni la propagation du signal ni la mobilité des stations. Au contraire, les environnements avec obstacles imposent des contraintes à la propagation du signal mais surtout à la mobilité des stations qui ne peuvent plus se déplacer en ligne droite. Cependant, à la différence des modèles précédents, les modèles d'environnements n'ont pas une influence directe sur les graphes obtenus, mais imposent des contraintes aux autres modèles et donc façonnent indirectement les graphes de connexions. Les modèles d'environnements seront de nouveau abordés dans la section 6.4.

6.3 Généralités

6.3.1 Technologies

Cette section fait un très rapide tour d'horizon des différentes technologies employées en 2008, technologies qui resteront probablement d'actualité dans un futur proche pour mettre en œuvre les réseaux mobiles *ad hoc*.

IEEE802.11

L'ensemble de normes *IEEE802.11* [IEE] contient plusieurs spécifications qui représentent la majorité des implémentations actuelles et futures. Les normes *IEEE802.11* concernent les modes de communication mais aussi la qualité de service (*IEEE802.11i*) et la sécurité (*IEEE802.11h*). Ces normes ont des couches physiques différentes mais reposent sur le même protocole de liaison de données, *CSMA/CA* (*Carrier Sense Multiple Access with Collision Avoidance*). Ce protocole repose sur une approche probabiliste dont le principe consiste à réserver le support radio avant d'émettre des paquets. En effet, dans un réseau sans fil, une seule machine peut émettre à la fois et du fait de la particularité du medium, les collisions plutôt qu'être détectées doivent être évitées. Il est en effet impossible à une station d'écouter le medium et d'émettre sur ce même medium simultanément. L'efficacité du protocole de communication repose alors sur la qualité des stratégies d'accès au medium. Les protocoles utilisés sont basés sur des demandes de réservation du medium avec indication du délai de transmission associé. *DCF* (*Distributed Coordination Function*), permet, en mode *ad hoc* de réaliser la coordination nécessaire entre les stations. D'autre part, en mode *ad hoc*, pour que le transit de données soit effectué il faut que l'émetteur et le récepteur soient synchronisés. Le mécanisme de recherche de voisinage et de synchronisation se révèle relativement rapide en pratique (généralement inférieur à la seconde). Par contre lorsque des stations sont synchronisées, elles n'explorent plus leur voisinage. Il se peut donc que deux stations déjà synchronisées par ailleurs, se retrouvent à portée de communication radio l'une de

l'autre sans en avoir connaissance. Ainsi, le mécanisme de transit ne permet pas d'assurer une qualité de service suffisante des applications de type *VOIP*. L'intérêt principal de cette norme est de permettre la conception de réseaux autour de points d'accès, mais aussi en mode *ad hoc*, de pair-à-pair. Parmi toutes les normes, les trois suivantes nous semblent les plus intéressantes dans le cadre des réseaux mobiles *ad hoc*.

IEEE802.11b. Cette norme définie en 1999 est implémentée, en 2008, sur l'immense majorité des dispositifs dédiés aux réseaux sans fil. Elle permet un débit théorique de $11Mbit/s$ sur la fréquence de $2.4GHz$. La couverture est de l'ordre de 30 mètres pour le débit théorique et 90 mètres pour un débit de $1Mbit/s$.

IEEE802.11g. Cette norme est compatible avec la précédente et propose un meilleur débit ($54Mbit/s$ au maximum) grâce à une méthode qui multiplexe les communications de manière plus efficace mais aussi plus sensible aux échos. Elle est aussi très largement implémentée sur les différents équipements aujourd'hui disponibles.

IEEE802.11n. Cette norme propose un débit théorique encore supérieur aux deux précédents avec $248Mbit/s$ en théorie. Ce débit est atteint entre autre grâce à la technologie des antennes *MIMO* (*Multiple-Input Multiple-Output*) qui en plus de multiplexer les communications sur plusieurs bandes de fréquences, utilise plusieurs antennes à l'émission et à la réception.

Bluetooth

Cette spécification publiée en 1999 utilise la même plage de fréquences que l'*IEEE802.11*. Avec une portée de 10 mètres et un débit théorique de $1Mbit/s$ elle a pour but la communication sans fil entre un ordinateur et ses périphériques. Elle est également destinée aux applications domotiques. Son mode de communication basé sur des sauts de fréquences très rapides (1600 sauts par seconde) la rend peu sensible aux collisions et permet une meilleure qualité de service que la norme *IEEE802.11b/g*. Les applications de type *VOIP* y sont envisageables. L'inconvénient est que la communication ne fonctionne qu'après une phase de découverte qui peut prendre plusieurs secondes. Il est néanmoins possible de créer de petits réseaux de 8 machines (1 maître et 7 esclaves) dans un même voisinage réseau : les *piconets*. La communication à plusieurs sauts est également envisageable grâce à une interconnexion des *piconets* : les *scatternet*. Lorsque le maître ou un esclave d'un *piconet* est contacté par un maître d'un autre *piconet*, il peut s'associer à ce dernier en plus de son réseau d'origine et peut ainsi, par intermittence, communiquer avec les stations des deux réseaux et éventuellement, jouer le rôle de passerelle. Les communications multi-sauts sont alors envisageables.

Cependant, dans le cadre des *MANETs*, la portée ainsi que la durée nécessaire à la découverte du voisinage semblent deux obstacles majeurs à l'utilisation de cette technologie pour des réseaux *mobiles*.

6.3.2 Simulateurs

Dans le but de concevoir et d'analyser des algorithmes avant leur implémentation sur des plateformes réelles de réseaux mobiles *ad hoc*, il est d'usage d'avoir recours, dans un premier temps, aux simulations. En effet, les simulateurs offrent un moyen simple et rapide de prototyper une application avant son implémentation et d'identifier des points qui pourraient s'avérer problématiques sur plateformes réelles. Trois moyens différents permettent de simuler les véritables réseaux. Il est possible de recréer de tels réseaux, à une moindre échelle, en utilisant de véritables machines mobiles et en mobilisant des personnes afin de mimer un comportement donné en jouant un scénario. C'est le cas par exemple du projet *Roofnet* [BABM05]. Cette approche, qui présente l'avantage d'être une image fidèle de la réalité, au moins pour certains aspects, est néanmoins très coûteuse en temps, en énergie et en argent.

Une seconde approche consiste à réutiliser des *log*, des traces de connexions et autres informations parcellaires glanées par divers moyens. Ces traces peuvent, sous certaines conditions, permettre de reconstruire les mouvements et les connexions d'un véritable réseau mobile. Le projet *CRAWDAD*[CRA] de l'université de Dartmouth a choisi cette option. En contrepartie, cette approche ne mesure que partiellement l'activité des stations, ce qui constitue sa principale limitation. Pour connaître l'activité des stations instant par instant avec ces données il faut procéder à une « interpolation » de l'activité des stations entre deux instants observés, ce qui est toujours délicat hors contexte.

Enfin la troisième approche relève de la simulation pure de réseaux mobiles *ad hoc*. C'est de loin la technique la plus employée. La suite de la section présente quelques uns parmi les principaux simulateurs actuellement utilisés, ainsi que le simulateur utilisé dans cette thèse, simulateur qui a été conçu et mis au point dans le cadre d'une collaboration entre l'universités du Luxembourg et l'université du Havre.

***NS-2* [NS-].** C'est un simulateur basé sur une évolution événementielle et discrète de l'environnement modélisé. Développé à l'*Information Sciences Institute* de Californie, il a le soutien de la *DARPA*[11] depuis 1995. Il est très largement utilisé et reconnu par la communauté scientifique. Il donne accès a une multitude de renseignements et à une multitude de réglages. Il possède donc une très fine granularité. Le noyau monolithique est implémenté en C++ et des modules complémentaires sont intégrables. Sa configuration se fait en utilisant un langage de script (*OTCL*, dérivé de *TCL*). C'est un projet libre. Au départ le simulateur fonctionnait sur un seul *CPU* et souffrait de mauvaises performances avec l'incapacité de simuler plus de quelques centaines de stations. Ce

11. *Defense Advanced Research Projects Agency*

problème est maintenant résolu, un module chargé de distribuer la simulation lance sur différents processeurs des instances du simulateur en partitionnant le réseau simulé.

***GloMoSim* [Glo].** Un projet conçu et développé pour une exécution parallèle, grâce à *PARSEC* [PAR], un environnement dédié à la programmation et l'exécution parallèle de simulations à événements discrets. Le simulateur utilise le modèle en couches de la norme *OSI*. Chaque couche possède sa propre *API*, il est donc très facile d'intégrer du code écrit pour de véritables applications réseaux et *vice versa*.

***OPNet* [OPN].** C'est également un simulateur à changement d'états discrets. C'est une suite commerciale très reconnue dans le milieu professionnel pour analyser et prévoir le comportement d'un réseau donné. Il permet de simuler un grand nombre de protocoles et de services. Ses résultats sont comparables aux deux précédents.

***Madhoc* [Hog08, Hog07].** C'est un simulateur de réseaux mobiles destiné à la conception et à l'analyse d'algorithmes distribués. Il est développé en parallèle par l'université du Luxembourg et par l'université du Havre par L. HOGIE. Il est utilisé dans les applications de cette thèse notamment pour la génération de graphes dynamiques, en intégrant des contraintes liées aux *MANETs* (modèles de comportement des stations, modèles de mobilité, modèles d'environnement, *etc.*).

6.4 Environnements et modèles de mobilité

L'objet de cette thèse ne concerne pas spécifiquement les réseaux mobiles *ad hoc*, mais plutôt les graphes dynamiques, c'est la raison pour laquelle cette section ne prétend pas à l'exhaustivité des travaux dans le thème des modèles de mobilité. Notre ambition est simplement de présenter quelques modèles de mobilité, d'environnement et de propagation de signal de manière à discuter la façon dont ils influent sur les graphes de connexions entre stations, c'est-à-dire indirectement sur la définition des graphes dynamiques qui leur sont associés.

Nous présentons dans la suite de cette section un ensemble représentatif de modèles de mobilité, où le sens de l'expression « modèles de mobilité » correspond à celui discuté dans la remarque page 84. Les modèles sont regroupés par catégorie, selon la classification proposée par F. BAI et A. HELMY [BH04], qui dans le contexte des *MANETs* ont proposé quatre catégories principales basées sur des caractéristiques spécifiques : 1) les modèles basés sur des règles aléatoires ; 2) les modèles avec une dépendance temporelle, quand le mouvement d'une station dépend des mouvements des étapes précédentes ; 3) les modèles avec des dépendances spatiales, quand les stations sont contraintes par les mouvements d'autres stations ; 4) les modèles avec des restrictions géographiques comme le déplacement rectiligne sur une autoroute. La suite de

la section visite les différents modèles de mobilité couramment utilisés en fonction de cette classification. Notons que les noms des modèles sont conservés en anglais car il y est fait référence ainsi dans les publications francophones.

6.4.1 Modèles aléatoires

Random Walk Mobility Model. Ce modèle fut proposé pour simuler le mouvement imprévisible de particules en physique. Les physiciens y font d'ailleurs référence en parlant de « mouvement brownien ». C'est un modèle itératif qui fait évoluer toutes les stations à chaque étape. À son tour, une station choisit une direction et une vitesse de manière aléatoire et se rend à sa nouvelle destination. La distribution du choix de l'angle (entre 0 et 2π) et de la vitesse (entre 0 et v_{max}) suit une distribution gaussienne.

Random Waypoint Mobility Model. Ce modèle proposé par J. BROCH *et al.* [BMJ+98] est devenu le principal modèle de mobilité utilisé pour les simulations dans les publications académiques, en particulier pour tester des algorithmes de routage. Le principe dérive du *Random Walk*. Itérativement les stations se déplacent avec une vitesse constante choisie aléatoirement de manière uniforme, vers une destination elle aussi choisie aléatoirement dans l'espace de simulation. Une fois à destination, la station attend pendant une durée (un nombre d'étapes) choisie entre 0 et W_{max}. Ce modèle est implémenté dans tous les simulateurs présentés ci-dessus. Il est souvent fait référence à ce modèle par l'acronyme *RWP*.

6.4.2 Modèles avec dépendances temporelles

Les modèles aléatoires ne reflétant pas assez la réalité, on cherche à s'en approcher en s'intéressant à la rémanence des vitesses et directions de déplacement. Pour cela, d'une étape à l'autre, le système conserve une mémoire des caractéristiques de ses déplacements antérieurs.

Gauss-Markov Mobility Model. Ce modèle [LH99] propose, pour calculer la vitesse au temps t, de s 'appuyer sur la vitesse au temps $t-1$ mais aussi sur la vitesse moyenne de la station considérée et sur une valeur aléatoire uniforme. Un paramètre α permet de donner plus ou moins d'importance à ces trois composantes pour calculer la nouvelle vitesse de la station. Les auteurs notent que si le paramètre α est réglé à zéro, alors la formule ne tient plus compte de la vitesse à $t-1$ et le modèle est équivalent au *Random Walk*. Si au contraire α vaut 1, la vitesse de la station est constante tout au long de la simulation et le modèle se rapproche des mécanismes de diffusion en physique. Tout réglage de α entre 0 et 1 mène à un comportement plus réaliste que le *Random Walk* évitant les cas extrêmes où la vitesse est très rapide à l'étape i et subitement très lente à l'étape $i+1$.

Smooth Random Mobility Model. Dans ce modèle [Bet01a], C. BETTSTETTER propose une amélioration du *Random Waypoint* en rendant les changements de vitesse et de direction plus réalistes. Il propose de lisser ces changements en incluant un mécanisme incrémental de variation de la vitesse et du changement de direction. Pour définir sa nouvelle vitesse, une station modifie légèrement sa vitesse antérieure. De même sa nouvelle direction est obtenue en modifiant légèrement son précédent angle de direction.

6.4.3 Modèles avec dépendances spatiales

Dans les précédents modèles présentés les stations agissent indépendamment les unes des autres. La classe des modèles avec dépendances spatiales s'intéresse aux situations au cours desquelles le déplacement des stations s'effectue en groupe. Ce type de déplacement peut être induit par certaines contraintes liées à l'environnement, soit le résultat d'interactions entre les stations elles-mêmes.

Reference Point Group Mobility Model. Ce modèle de mobilité cherche à mimer le mouvement de stations qui se déplacent en groupe dans la même direction [HGPC99]. Ce modèle s'inspire des phénomènes de foules lors de grandes manifestations, des phénomènes de fuites lors de catastrophes ou encore des déplacements d'une armée au combat.

Autres modèles avec dépendances spatiales. Dans [SM01] les auteurs proposent un simulateur équipé de trois modèles de mobilité avec interdépendances entre les stations. Le premier modèle reproduit le mode de déplacement en ligne de secouristes à la recherche de survivants lors d'une avalanche. Il est nommé « *scanning an area* ». Les auteurs font également référence à un groupe de robots à la recherche de mines antipersonnelles à désamorcer. Les entités dans ce modèle se réfèrent aux déplacements de leurs congénères pour définir leur position. Le second modèle, nommé « *pursue a target* » fait évoluer les entités vers une destination commune. Les stations se déplacent avec une vitesse limitée et un certain aléa dans leur direction. Le point de mire change aussi de manière aléatoire, remettant ainsi en question la trajectoire des stations. Les auteurs prennent pour exemple les mouches attirées par la pomme que je suis en train de manger. Elles se dirigent vers la pomme. Quant je bouge la pomme elles changent subitement de cap. Enfin le dernier modèle, nommé « *nomadic community* », reprend l'idée de la nuée d'oiseaux. Toutes les entités du groupe se déplacent globalement dans la même direction en maintenant une certaine distance entre elles pour éviter les collisions. Les déplacements dans le groupe sont relativement aléatoires même si le comportement global du groupe est cohérent.

6.4.4 Modèles avec dépendances géographiques

Cette dernière catégorie de modèles s'intéresse aux environnements qui contraignent la mobilité ainsi que la propagation du signal. Il permettent, par exemple, de modéliser les déplacements de véhicules sur une autoroute ou de piétons dans une rue.

Pathway Mobility Model. Dans ce modèle ([THB+02]) l'environnement n'est plus un espace ouvert, il est constitué de chemins qui mènent à différentes destinations. Le comportement des stations est similaire au *Random Waypoint Mobility Model*. Les stations choisissent une vitesse et une destination et s'y rendent. Le déplacement ne se fait plus en ligne droite d'un point à l'autre mais en utilisant le plus court des chemins qui relie la station à sa destination. Les chemins peuvent être choisis de manière à modéliser des autoroutes ou les rues d'une ville. Dans ce type de modèle, les graphes sont assez naturellement employés pour modéliser l'environnement.

Obstacle Mobility Model. Dans [JBRAS05] les auteurs présentent deux modèles : un modèle de propagation des signaux et un modèle d'environnement urbain. Ce modèle intégrant à la fois les contraintes de propagation et les contraintes de mobilité est nommé *Obstacle Mobility Model (OM)*. Leur travail est motivé par la constatation que les schémas de déplacement des modèles existants ne sont pas nécessairement comparables à des déplacements réels. Ainsi, les auteurs proposent de créer un environnement dans lequel les mouvements seraient contraints. Pour y parvenir, ils génèrent un ensemble d'obstacles (des polygones) qui peuvent correspondre à des constructions ou à tout autre élément de l'environnement qui entrave la mobilité (lac, trou, construction, zone interdite d'accès, *etc.*). À partir des sommets de chaque obstacle, l'algorithme construit un diagramme de VORONOÏ. Les cellules de ce diagramme partagent des arêtes qui constituent le graphe de déplacement des stations. Les points correspondant aux intersections entre les arêtes du diagramme et VORONOÏ et les côtés des obstacles sont considérés comme des accès vers l'intérieur des bâtiments. Le scénario de mobilité retenu est similaire au *RWP*, à ceci près que les destinations choisies doivent être des obstacles et que les chemins pour s'y rendre empruntent les arêtes du diagramme de VORONOÏ.

6.4.5 Impact des modèles de mobilité sur les graphes dynamiques

Dans les modèles aléatoires, en général, la distribution des stations dans l'environnement est initialement uniforme. Cependant, les règles de déplacement aléatoire, couplées aux contraintes de l'environnement, induisent des biais dans les graphes de connexion comme cela a été souligné dans de nombreuses publications. En particulier, C. BETTSTETTER et ses collègues [BRS03] montrent que la probabilité de présence des stations lorsque l'environnement est borné n'est pas uniforme avec pour conséquence un degré moyen plus important pour les sommets situés au centre de l'espace. Lorsque l'environnement considéré est torique [Bet01b], la probabilité de présence des stations devient

uniforme. En comparaison, l'impact des dépendances temporelles est faible au regard de l'importance à la fois des règles choisies pour le déplacement et des contraintes induites par l'environnement. En effet, si les déplacements sont régis par des règles aléatoires celles-ci priment sur l'effet mémoire introduit dans les modèles.

Dans les modèles de déplacement de groupes, les stations se déplacent selon des règles qui tendent à les agréger. La conséquence sur les graphes dynamiques qui en dérivent est l'apparition de groupes de sommets fortement connectés.

Enfin, l'impact sur les graphes dynamiques dérivés des modèles de mobilités en présence d'obstacles n'a, à notre connaissance, pas encore été étudié en détail. Cependant, les résultats préliminaires d'études que nous avons menées nous incitent a penser que la présence d'obstacles aurait un impact sur la distribution des degrés. On observe en effet une augmentation significative de la densité des stations autour des obstacles, impliquant la présence de sous-graphes induits à forts coefficients de *clustering*.

6.5 Des *MANETs* aux graphes dynamiques

En se basant sur l'ensemble des contributions précédentes, nous pouvons définir le périmètre d'un modèle de mobilité comme l'ensemble des éléments qui régissent le déplacement des stations en respectant les contraintes liées à leur environnement. Nous en proposons une nouvelle définition, en accord avec la définition que nous avons retenu pour les *MANETs* (définition 21).

Définition 22 (MODÈLE DE MOBILITÉ)
Dans le contexte des MANETs, un modèle de mobilité est un processus, propre à chaque station ou commun à toutes, qui en régit le déplacement. Il est caractérisé par un ensemble de règles dont les conditions d'application sont précisées par des paramètres et contraintes dans le modèle d'environnement.

En suivant cette définition, le très connu *Random Waypoint Mobility Model* ou *RWP* décrit précédemment peut se résumer à :

1. Un processus qui consiste à appliquer itérativement les quatre règles suivantes :
   - Règle 1 : chaque station choisit aléatoirement une destination d et une vitesse v ;
   - Règle 2 : la station se déplace en ligne droite depuis sa position courante vers la destination d à la vitesse v ;
   - Règle 3 : arrivée à destination, la station choisit aléatoirement un temps de pause p ;
   - Règle 4 : la station reste immobile durant un temps égal à p.
2. Un ensemble de paramètres qui précise les conditions d'application des règles 1 et 3 :
   - Paramètre V : intervalle de vitesses $[V_{min}, V_{max}]$;

– Paramètre P : intervalle pour le temps de pause $[P_{min}, P_{max}]$.

3. Des contraintes liées au modèle d'environnement : l'application de la règle 1 est contrainte par l'environnement. Dans le *RWP*, l'environnement est libre de tout obstacle, mais peut être une surface carrée, ronde, ou torique, sans que le modèle de mobilité n'en soit modifié. Nous pouvons noter également que le modèle de propagation du signal n'a aucune influence sur la manière dont les stations se déplacent.

Le rapport entre modèles de mobilité et graphes dynamiques apparaît maintenant clairement. Relativement à la définition 5 d'un graphe dynamique, le modèle de mobilité précisé par ses paramètres et contraint par le modèle d'environnement participe au processus d'évolution de ce graphe en déterminant les propriétés des sommets. Les liens sont quant à eux définis par le modèle de propagation du signal. Pour les modèles aléatoires, les modèles avec dépendances temporelles et les modèles avec dépendances géographiques, les graphes dynamiques qui en dérivent sont caractérisés par un processus d'évolution qui peut être décrit selon le principe de l'algorithme 3 (page 33) dans lequel la fonction $m()$ correspond au modèle de déplacement des stations et dans lequel la fonction $p()$ dépend à la fois du modèle de propagation du signal et des caractéristiques des stations (couverture radio en particulier). Pour des modèles avec dépendances spatiales, c'est-à-dire des modèles dans lesquels le déplacement d'une station dépend de la position et du déplacement d'autres stations, les graphes dynamiques qui en dérivent sont caractérisés par un processus d'évolution qui peut être décrit selon le schéma de l'algorithme 4.

6.5.1 Métriques dans les *MANETs*

Le rapport entre *MANETs* et graphes dynamiques étant établi, il reste à identifier parmi les métriques définies au chapitre 3 celles dont la pertinence est avérée dans le contexte des réseaux mobiles *ad hoc* et si d'autres mesures propres aux *MANETs* sont à définir.

Il est important de noter que les réseaux mobiles *ad hoc*, à la différence des graphes dynamiques pris dans toute leur généralité, sont spatialisés. Ainsi certaines mesures inexistantes dans les graphes dynamiques se révèlent pertinentes dans le contexte des *MANETs*, par exemple, la distribution spatiale des stations.

Définition 23 (DISTRIBUTION SPATIALE)
La distribution spatiale décrit la manière dont les stations sont réparties dans l'environnement.

Cette métrique est importante pour les réseaux mobiles car elle permet de détecter certaines propriétés des modèles de mobilité. Par exemple C. BETTSTETTER a montré que le *RWP* présentait une distribution en cloche. Ce qui peut entraîner des biais sur les ré-

sultats d'applications simulées. En outre, de la distribution spatiale vont dépendre la répartition des degrés et le nombre de composantes connexes.

Définition 24 (DURÉE DES LIENS)
La durée d'un lien entre deux stations correspond au temps moyen que dure la connexion entre ces stations.

Elle correspond à l'âge moyen [12] de l'arête qui lui est associée dans le graphe dynamique. Elle est importante pour les algorithmes distribués qui doivent disposer de fenêtres de temps suffisantes pour pouvoir faire transiter de l'information.

Revenons à notre objectif de départ qui est la construction et le maintien de structures correspondant à des solutions dans les graphes dynamiques. Dans le contexte des réseaux mobiles *ad hoc*, la contrainte du maintien impose que la méthode de traitement s'exécute à la fois de manière décentralisée et en continu sur les stations. On peut donc admettre, sans perte de généralité que cette méthode est itérative. Les stations correspondent aux ressources de calcul, et nous considérons que le schéma général de traitement se compose de trois étapes :

1. la collecte d'information ;
2. le calcul des nouveaux états ;
3. la diffusion d'information.

Nous appelons l'ensemble de ces trois étapes une itération. L'efficacité d'une méthode de traitement dépend de la capacité des stations composant le système à exécuter un nombre plus ou moins grand d'itérations relativement aux changements du graphe dynamique. Nous définissons donc une métrique globale que nous appelons la vitesse relative.

Définition 25 (VITESSE RELATIVE)
Soit $V_s(A)$ la vitesse d'un algorithme A sur la station s, elle correspond au nombre d'itérations de cet algorithme par unité de temps. La vitesse relative VR d'un tel algorithme sur l'intervalle de temps $I = [t, t']$ s'exécutant sur un réseau mobile ad hoc R est définie par :

$$VR(A,R,I) = \frac{\sum_{s \in R} V_s(A) \times d}{apparitions(I) + disparitions(I)}$$

avec $d = t' - t$, $apparitions(I)$ le nombre d'apparitions d'éléments dans l'intervalle I et $disparitions(I)$ le nombre des disparitions dans I, dans le graphe dynamique correspondant au réseau mobile ad hoc R.

Ce que nous nommons vitesse de l'algorithme dépend de la puissance de calcul de la station, mais également de la capacité de communication de cette station avec ses

12. définition 19, page 38.

voisines. Si la vitesse de l'algorithme est égale à v cela implique en particulier que pendant une unité de temps, une donnée traitée par une station ne pourra atteindre une autre station située à une distance supérieure à $\lfloor v \rfloor$.

Considérons un système S composé de 100 stations qui au cours d'un intervalle de temps I de 10 secondes subit 50 événements d'apparitions et de disparitions d'éléments (stations, liens). L'unité de temps considérée est la seconde. Considérons que la vitesse de l'algorithme sur chaque station soit égale à 4, alors la vitesse relative $VR(A,S,I) = \frac{4\times100\times10}{50} = 80$.

CHAPITRE 7

LE PROBLEME DU CHEMIN LE PLUS COURT ET LE PLUS ROBUSTE

Le problème de base que nous souhaitons traiter dans ce chapitre est le suivant : étant donné un graphe dynamique défini par un graphe initial, une base temporelle discrète et un processus d'évolution, et étant donnés deux sommets a et b appartenant à ce graphe, déterminer et maintenir un chemin entre a et b qui soit à la fois le plus court et le plus robuste possible. Si la notion de plus court chemin est bien connue, celle de robustesse nécessite quelques précisions. Nous exprimons l'objectif de robustesse en définissant ce que nous nommons le « taux de renouvellement » d'une structure. Ce

taux défini formellement par la définition 20 page 38 exprime le pourcentage de modifications nécessaires pour passer d'une structure à une autre entre deux étapes.

Ce problème se pose de manière critique dans le contexte des *MANETs* et s'apparente au problème du routage déjà riche de nombreux travaux. En outre, dans le cadre des *MANETs*, certaines contraintes supplémentaires viennent s'ajouter à la difficulté du problème. Cette construction de chemin ne peut être faite que de manière locale et décentralisée du fait de l'absence d'infrastructure, de la volatilité des stations et de la dynamique de la topologie du réseau. Pour parvenir à un résultat, il est proposé ici de mettre en œuvre une méthode décentralisée basée sur le principe de l'intelligence collective.

Ce chapitre est structuré en trois parties. La première partie discute du problème de routage dans les *MANETs*. La seconde partie propose un état de l'art des méthodes de plus courts chemins en contexte dynamique. Enfin, la troisième partie est consacrée à la présentation de notre contribution.

7.1 Problématique

Les problèmes que pose le routage dans les réseaux mobiles sont liés à leur nature imprévisible et dynamique ainsi qu'aux faibles ressources (débit et autonomie) des stations qui les composent. Les algorithmes de routage proposés doivent tenir compte de certaines contraintes :

- l'absence d'infrastructure qui empêche tout contrôle centralisé et impose une résolution distribuée ;
- la dynamique du réseau et la perte rapide de validité des routes qui imposent des mécanismes de mise à jour adaptés ;
- la faiblesse des ressources qui oblige à limiter le volume et la fréquence des données de routage échangées.

Les routes construites doivent elles aussi se soumettre aux contraintes des *MANETs*. En effet, les faibles débits poussent à limiter le nombre d'intermédiaires entre deux stations de façon à minimiser le temps de transit de l'information. Le plus court chemin en nombre de sauts a des chances d'être également le plus rapide en temps de transmission. De plus, le chemin qui utilise le moins de stations est, vraisemblablement, le plus économe en énergie. Il y a donc un intérêt à déterminer des plus courts chemins pour construire des routes.

Il faut aussi considérer les problèmes de latence lors de connexions et re-connexions. En effet, plus que la communication elle-même, c'est la synchronisation entre les stations qui est coûteuse en temps. Dans un chemin de communication, si un des liens casse, tout un processus est nécessaire pour reconstruire tout ou partie du chemin. On cherchera donc à minimiser le nombre de changements de nouvelles connexions dans un chemin lors de modifications du réseau.

Définition 26 (TAUX DE RENOUVELLEMENT DANS UN *MANET*)
On définit le taux de renouvellement dans un MANET à partir de la définition 20 de « taux de renouvellement » (cf. page 38) en restreignant les éléments aux seules arêtes.

L'hypothèse posée ici, qui reste à expérimenter sur des plateformes réelles, est qu'il est plus pertinent de construire des routes en maximisant le nombre de liens qui se déconnectent moins souvent. Cependant, du fait de la nature presque imprévisible des réseaux mobiles il est difficile de prédire si un lien donné va bientôt disparaître. Il est possible en revanche de se baser sur l'activité passée d'un lien donné et d'extrapoler son comportement dans un futur proche. La métrique de volatilité définie page 37 définit l'inverse de l'âge moyen d'une arête dans un graphe dynamique. Appliquée au cas des réseaux mobiles *ad hoc*, cette métrique mesure la durée de vie moyenne d'une connexion entre deux stations. De plus, cette métrique présente l'avantage de pouvoir être construite de manière décentralisée par les stations. Grâce à elle il devient envisageable d'effectuer des choix locaux basés sur l'historique des connexions.

Cependant, le plus court chemin n'est pas forcément le plus stable, ainsi, le problème que nous nous proposons de traiter s'apparente à un problème d'optimisation bi-objectif pour lequel il convient de minimiser à la fois la longueur des chemins et leur taux de renouvellement.

7.2 Quel service pour quel réseau ?

Les contraintes de routage ci-dessus dépendent à la fois de la mobilité des réseaux et du type d'information que l'on cherche à router. En effet certains réseaux sont plus mobiles que d'autres et on sait que les déconnexions coûtent très cher. La nature des informations dépend des applications que l'on cherche à exécuter. Celles-ci sont plus ou moins sensibles à la latence dans les communications.

7.2.1 *Streaming* audio ou vidéo

Le type de service le plus difficile à prendre en charge est celui des applications de *streaming* audio ou vidéo qui consistent à envoyer un flux de données d'un serveur vers un ou plusieurs clients. Le volume de données est très important et le flux des données envoyées ne supporte pas d'être interrompu. La contrainte de flux renvoie à la notion de qualité de service. On sait par exemple que les différentes normes IEEE 802.11 ne peuvent pas assurer de qualité de service à cause de l'implémentation probabiliste de la couche physique. La norme *Bluetooth* semble plus adaptée au problème. De plus, les *Scatternets* (vus page 87) permettraient d'envisager ce service en multi-sauts. Néanmoins, les faiblesses de débit et de couverture du *Bluetooth* ne sont pas attractifs et à ce jour aucune solution de ce genre n'existe. Concernant le *WiFi*, malgré les problèmes

de qualité de service, plusieurs propositions sont faites en traitant le problème de diverses manières. Dans [KWG99], les auteurs proposent un mécanisme d'adaptation du débit en fonction du taux de perte d'information perçu. Ils mènent à bien des simulations pour vérifier leurs hypothèses. Le routage et la mobilité ne sont pas traités dans cette proposition. Dans [XC07], les auteurs proposent un algorithme de routage pour un réseau de stations mobiles. Des simulations sont faites et l'algorithme est comparé à *AODV*, un protocole de routage classique dédié aux *MANETs*. Enfin, dans [MLPW03], les auteurs reportent de véritables expérimentations sur des ordinateurs portables et un réseau *IEEE802.11b*. Dans ces tests les stations ne sont pas mobiles.

En résumé les applications de type *streaming* audio ou vidéo s'accommodent difficilement des contraintes des réseaux *ad hoc* sans fil et d'autant moins que les stations sont mobiles.

7.2.2 Applications pour les réseaux mobiles *ad hoc* discontinus

Les réseaux mobiles *ad hoc* tolérant les délais ne permettent pas de garantir l'existence de chemins de communication de bout en bout. La notion de trajet (définie page 37) se substitue à celle de chemin. Le principal mécanisme mis en œuvre est de type *store-and-forward*[1], qui plutôt que de subir les déconnexions, s'appuie sur la mobilité et la capacité de stockage des stations pour transmettre de l'information.

Ainsi, les applications utilisables dans ces réseaux sont de type diffusion et partage de documents comme le proposent H. ROUSSAIN et F. GUIDEC dans [GR05], mais également de type collecte d'informations dans les réseaux de capteurs qui peuvent présenter des délais de connexions importants [Pat05].

7.3 État de l'art

Le problème présenté précédemment peut être considéré de deux points de vues distincts : d'abord du point de vue d'un observateur, en considérant que le réseau est observable dans sa globalité de manière centralisée, et que la suite des événements est éventuellement connue à l'avance. Parmi les différentes contributions, nous présentons quelques uns des travaux de la communauté dite de la ré-optimisation ainsi que ceux de S. Pallottino et M. G. Scutellà [PS97] sur les *Space-Time Networks*. Ces deux approches ont déjà été abordées au chapitre 2 pour leur modèle de graphe dynamique original.

Du point de vue des stations, la détermination d'un plus court chemin dans un *MANET* est un processus décentralisé. Chaque station utilise des informations locales, son voisinage par exemple, pour effectuer ses traitements et prendre ses décisions. Les contributions sur le sujet se rapportent à des problèmes de routage dans les *MANET*. Ces approches seront abordées dans la seconde partie de cet état de l'art.

1. La communication entre deux stations se fait de proche en proche avec stockage sur les stations intermédiaires.

7.3.1 Approches centralisées pour la construction de plus courts chemins dans les graphes dynamiques

Ré-optimisation

La ré-optimisation est une technique qui consiste à construire et maintenir une propriété dans une structure dynamique telle qu'un graphe dynamique. L'intérêt annoncé de cette approche est que le maintien de la structure après modification de l'environnement est d'une complexité algorithmique moindre que le recalcul complet.

Un plus court chemin peut être une telle structure que l'on cherche à maintenir. On s'intéresse ici au problème du plus court chemin entre une source et toutes les destinations d'un graphe dynamique. L'idée générale consiste à construire et maintenir un arbre couvrant de plus courts chemins, dont la racine est la source. À chaque étape, l'arbre de plus courts chemins est mis à jour de manière à ne pas recalculer l'ensemble des plus courts chemins. Le graphe dynamique utilisé est le modèle de « graphe pour la ré-optimisation » étudié page 18 dans le chapitre 2. Il permet des ajouts, suppressions et modifications d'arêtes ; le nombre de sommets reste constant. Son évolution est itérative sans notion temporelle.

Chaque arête dans le graphe dispose d'un poids w sur lequel s'appuie le calcul du plus court chemin. Si le poids des arêtes est unitaire, les plus courts chemins comptent alors le nombre d'arêtes entre la source et une destination. La solution est donc un arbre couvrant dans le graphe. Les arêtes appartenant à cet arbre sont marquées et référencées dans l'ensemble noté $ArbrePCC$. À chaque destination d est associé sa distance à la source $D(d)$.

Le problème du plus court chemin entre une source et plusieurs destinations dans un graphe statique est facilement résolu à l'aide de l'algorithme de E. W. DIJKSTRA [Dij59]. Le même problème plongé dans un environnement dynamique peut être traité et maintenu d'une manière similaire, ce qui implique un recalcul complet entre deux étapes d'évolution du graphe dynamique. Les différentes approches proposées pour traiter le problème reposent toutes sur une adaptation de l'algorithme de E. W. DIJKSTRA aux graphes dynamiques. Les principales contributions sont dues à D. FRIGIONI *et al.* [FMSN96] ainsi qu'à G. RAMALINGAM et T. W. REPS [RR96] pour leurs innovations dans les structures de données utilisées qui réduisent la complexité de l'algorithme.

L'algorithme 5 illustre le processus de mise à jour des plus courts chemins lors de l'insertion d'une arête. L'algorithme 6 montre le mécanisme de mise à jour suivant la suppression d'une arête. La mise à jour du poids d'une arête ne fait pas l'objet d'un algorithme particulier car elle peut être réduite à l'un des deux cas précédents. En effet la diminution du poids d'une arête revient à considérer cette arête comme nouvelle et doit être traitée avec l'algorithme 5. Dans le cas d'une augmentation du poids d'une arête, deux cas sont à considérer. Si l'arête fait partie de l'arbre de plus courts chemins, il faut considérer l'événement comme une suppression sans que l'arête soit réellement retirée et l'algorithme 6 est exécuté à l'exception de la suppression de l'arête (ligne 2).

Enfin si l'augmentation concerne une arête n'appartenant pas à l'arbre de plus courts chemins, cette modification n'a aucune incidence sur le résultat.

Algorithme 5 : Insertion d'une arête (x, y, w_{xy})

Entrées : (x, y), w_{xy}; /* (x, y) l'arête à ajouter et w_{xy} son poids. */
Données : $C \leftarrow \emptyset$; /* Ensemble de triplets $\langle z, p(z), x\rangle$ avec z un sommet, $p(z)$ sa distance potentielle et x son nouveau père. */

```
1  début
2      si D(y) ≤ D(x) + w_xy alors Fin ; /* S'il n'y a pas de diminution de la
       distance, il n'y a rien à modifier. */
3      C ← C ∪ {⟨y, D(y) + w_xy, x⟩};
4      tant que C ≠ ∅ faire
5          ⟨z, p(z), x⟩ ← Min(C) ;   /* Triplet avec la plus petite distance
           potentielle. */
6          q ← RecherchePère(z) ; /* Recherche un père potentiel avec la
           distance la plus proche. */
7          Père(z) ← q;
8          D(z) ← D(q) + w_zq;
9          pour tous les voisins h de z faire
10             si D(h) > D(z) + w_zh alors
11                 C ← C ∪ {⟨h, D(z) + w_zh, z⟩};
12             fin
13         fin
14     fin
15 fin
```

ChronoSPT

Dans [PS97], S. PALLOTTINO et M. G. SCUTELLÀ s'intéressent aux problèmes de plus courts chemins lorsque les durées de traversées des arêtes dépendent du temps. Cette approche ne construit pas exactement des plus courts chemins mais des plus courts trajets comme défini page 37. Un trajet commence donc à une date donnée et en un lieu donné (un sommet du graphe) puis se prolonge dans le temps, sur d'autres sommets.

La proposition originale de S. PALLOTTINO et M. G. SCUTELLÀ est de construire un modèle de graphe à partir du graphe dynamique originalement étudié et d'en construire une représentation statique. Ce modèle de graphe nommé *Space-Time Network* est présenté en détail page 21 dans le cadre de l'étude des différents modèles de graphes dynamiques. Le modèle sous-entend une évolution discrète du graphe dynamique. Les

Algorithme 6 : Suppression d'une arête (x, y)

Entrées : (x, y); /* (x, y) l'arête à supprimer. */
Données : $M \leftarrow \emptyset$; /* Ensemble de couples $\langle z, p(z)\rangle$ avec z un sommet et $p(z)$ sa distance potentielle. */

début
  Suppression$((x, y))$;
  si $(x, y) \notin$ ArbrePCC **alors** FIN ; /* Si l'arête n'est pas dans l'arbre de plus courts chemins, il n'y a rien à modifier. */
  $M \leftarrow M \cup \{\langle y, D(y)\rangle\}$;
  tant que $M \neq \emptyset$ **faire**
    $\langle z, p(z)\rangle \leftarrow$ Min(M); /* Couple avec la plus petite distance potentielle. */
    $q \leftarrow$ RecherchePère(z) ; /* Recherche un père potentiel avec la distance la plus proche. */
    si $q = null$ **alors** $D(z) \leftarrow \infty$ Père$(z) \leftarrow q$;
    si $D(z) \neq D(q) + w_{qz}$ **alors**
      $D(z) \leftarrow D(q) + w_{qz}$;
      pour tous les *fils h de z* **faire**
        $M \leftarrow M \cup \{\langle h, D(z) + w_{zh}\rangle\}$;
      fin
    fin
  fin
fin

différents événements qui modifient le graphe ont lieu à des dates connues et recensées dans un ensemble fini. Une arête (i, j) possède un délai $d_{i,j}(t)$ qui indique son temps de traversée et qui dépend de la date de début de la traversée. Chaque arête possède également un coût $c_{ij}(t)$ dépendant également de la date et proportionnel à $d_{i,j}(t)$.

Cette construction impose de connaître à l'avance toute la dynamique du graphe, en revanche, une fois ce graphe construit, un simple parcours à l'aide d'un algorithme comme celui de DIJKSTRA permet de construire des plus courts chemins qui sont des plus courts trajets dans le problème original.

Ce modèle produit des graphes statiques d'une taille supérieure au graphe original. Si n et m sont respectivement le nombre de sommets et d'arêtes d'un graphe original et k, le nombre d'étapes, alors les dimensions du *Space-Time Network* sont de l'ordre de $n \times k$ sommets et $(n + m) \times k$ arêtes. Toute cette information n'est pas utile, en effet, lorsque l'on connaît la source du plus court chemin. Par exemple, un certain nombre de sommets devient inaccessible depuis ce sommet source et alourdit la structure inutilement. Les auteurs proposent un algorithme qui parcourt la structure et ne garde

que les parties utiles d'un *Space-Time Network* connaissant le sommet source de la recherche. L'algorithme nommé *ChronoSPT* utilise une liste $B = \{B_1, B_2, \ldots, B_k\}$ composée de k ensembles. Dans chaque ensemble figurent les sommets qui apparaissent à l'étape donnée. Comme indiqué dans la définition du *Space-Time Network* (page 22) un sommet s à l'étape t est noté s_t. À chaque sommet s_t est associé un coût depuis la source $C_s(t)$. Les auteurs définissent également $P_s(t)$ le meilleur prédécesseur de s_t ainsi que $w_s(t)$ le coût unitaire de l'attente sur s. L'algorithme 7 illustre une étape k de la procédure *ChronoSPT* qui sélectionne le sous graphe utile pour un sommet source donné et éventuellement un temps de départ différé donné.

Algorithme 7 : Procédure ChronoSPT pour l'étape k.

```
   /* k^ème itération */
1  tant que B_k ≠ ∅ faire
2      i ← un élément de B_k;
3      B_k ← B_k\{i} ;                                  /* On extrait i de B_k */
4      pour tous les successeurs j de i faire
5          t_l ← t_k + d_ij(t_k);
6          si C_i(t_k) + c_ij(t_k) < C_j(t_l) alors
7              C_j(t_l) ← C_i(t_k) + c_ij(t_k);
8              P_j(t_l) ← i_k;
9              si j ∉ B_l alors B_l ← B_l ∪ {j}
10         fin
11     fin
       /* Si l'attente est autorisée : */
12     si C_i(t_k) + w_i(t_k) × (t_{k+1} − t_k) < C_i(t_{k+1}) alors
13         C_i(t_{k+1}) ← C_i(t_k) + w_i(t_k) × (t_{k+1} − t_k);
14         P_i(t_{k+1}) ← i_k;
15         si i ∉ B_{k+1} alors B_{k+1} ← B_{k+1} ∪ {i}
16     fin
17 fin
```

La figure 7.1 illustre l'algorithme *ChronoSPT* en utilisant l'exemple de *Space-Time Network* de la figure 2.6 page 23.

7.3.2 Approches décentralisées pour la construction de plus courts chemins dans les graphes dynamiques

La plupart des méthodes de plus courts chemins dans le contexte des *MANETs* sont des algorithmes de routage.

Plusieurs techniques sont employées et plusieurs catégories peuvent être définies.

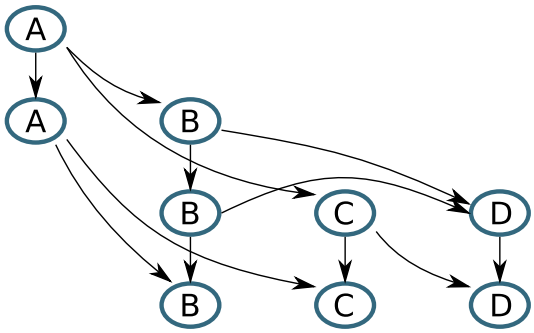


FIGURE 7.1: *ChronoSPT* produit à partir du *Space-Time Network* de la figure 2.6. *A* est le sommet source. Le départ est autorisé à être différé d'une étape. L'attente sur tous les autres sommets est autorisée.

Les algorithmes sont classés en fonction de leur pro-activité ou de leur réactivité dans la recherche de routes. Un algorithme pro-actif tente de maintenir un catalogue exhaustif des différentes routes possibles à tout moment alors qu'un algorithme réactif ne cherche de chemin entre une source et une destination qu'à l'occasion d'une communication entre ces deux stations. Les algorithmes pro-actifs ont l'inconvénient de générer beaucoup de trafic radio en plus des communications elles-mêmes mais bénéficient d'un temps de réponse aux requêtes de routage bien meilleur que les algorithmes réactifs qui doivent construire une route pour chaque nouvelle demande. En revanche, ces derniers n'utilisent que peu de bande passante lorsque le nombre de communications n'est pas important. Enfin, il existe la catégorie des algorithmes hybrides qui peuvent adapter leur politique au contexte.

Routage pro-actif

Dans [Jac01], les auteurs présentent *OLSR* (*Optimized Link State Routing protocol*), un protocole de routage dit à « état de liens ». La notion d'état de liens se réfère au fait que chaque nœud chargé du routage dans le réseau (mobile ou non) maintient à tout moment une table de routage qui représente l'état du réseau selon ce nœud. Les algorithmes à « état de liens » sont donc des algorithmes pro-actifs puisque le maintien des tables de routage nécessite de nombreux échanges d'informations. *OLSR* fait l'objet d'une RFC[2] auprès de l'*IETF*[3] [CJL+01]. Il n'échappe pas aux critiques faites aux algorithmes pro-actifs mais tente de minimiser le volume des communications dédiées au maintien des tables de routage en optimisant la diffusion d'information. Ainsi les mises

2. Les *Request For Comments* sont des documents et des normes soumis à l'approbation de L'*IETF*, concernant l'Internet qui aboutissent parfois à des standards.

3. L'*Internet Engineering Task Force* est un groupe ouvert et informel de personnes qui produit la plupart des standards de l'Internet (HTTP, FTP, POP, SSH, ARP, *etc.*) via les *Request For Comments*.

à jour des tables de routage ainsi que les chemins calculés utilisent un sous ensemble de nœuds choisis : les relais multi-points. L'idée des relais multi-points ou *MPR* (*Multipoint Relay*) repose sur le fait qu'un nœud donné va sélectionner un sous-ensemble des stations de son voisinage pour propager l'information. Les autres noeuds, non sélectionnés, se contentent de recevoir et de traiter l'information sans la propager. Toute l'efficacité de la technique repose dans la sélection judicieuse des ensembles *MPR*. En effet, il faut que les relais d'une station donnée soient capables d'atteindre toutes les stations du voisinage à deux sauts de la station considérée. Pour parvenir à cette sélection chaque nœud envoie à ses voisins (et reçoit d'eux de manière symétrique) des paquets *HELLO* contenant des informations sur son voisinage. Les paquets *HELLO* contiennent deux listes distinctes ; une liste des stations voisines pour lesquelles la station courante possède un lien bi-directionnel et une liste des stations avec liens uni-directionnels (liens par lesquels elle reçoit des paquets *HELLO* dans lesquels elle n'est pas listée). Cet échange de paquets *HELLO* permet à chaque station de connaître son voisinage à deux sauts et ainsi de construire son ensemble *MPR*. La figure 7.2 montre le voisinage d'un sommet (S) ainsi que ses relais sélectionnés. Dans l'exemple le sommet A est choisi plutôt que B car il permet d'atteindre les sommets à deux sauts de B mais aussi le sommet D. L'intérêt est que les nœuds U, V, A et C ne recevront qu'une fois les messages diffusés par S alors qu'ils l'auraient reçu deux fois lors d'une diffusion classique. Une fois sélectionnés les relais sont indiqués dans les paquets *HELLO* de manière à les prévenir qu'ils sont élus par un voisin. Grâce aux paquets *HELLO* chaque sommet peut savoir s'il a été sélectionné par un ou plusieurs de ses voisins pour être *MPR*. Il construit et maintient donc la liste des voisins qui l'ont sélectionné, ce sont ses « sélecteurs *MPR* ».

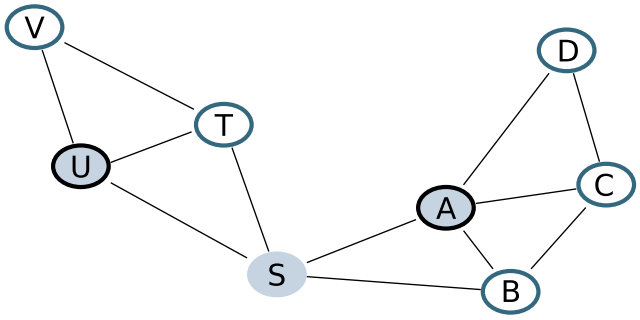
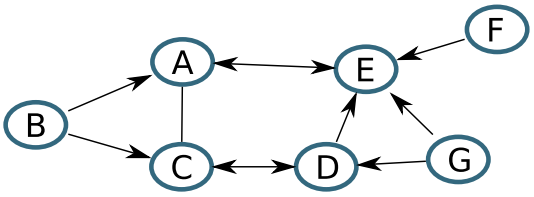


FIGURE 7.2: Voisinage d'un nœud S. Les nœuds A et U sont les *MPR* de S car ils permettent d'atteindre tout le voisinage à deux sauts de S.

Afin de mettre en œuvre l'algorithme de routage, il faut propager de l'information dans le réseau (les paquets *HELLO* ne sont envoyés qu'aux voisins proches à un saut). Chaque nœud va propager sa liste de sélecteurs *MPR* dans le réseau en profitant des avantages des *MPR*. Ces paquets sont les « Messages de contrôle de topologie » ou *TCM* (*Topology Control Messages*). À la réception d'un *TCM* un nœud dispose de couples ($dernier\_saut$,

destination) représentant le sommet émetteur du message *TCM* et ses sélecteurs *MPR*. Chaque nœud construit et maintient une table de topologie contenant ces couples de manière à reconstruire un chemin. La figure 7.3 montre un réseau ainsi que le tableau des messages *TCM* que chaque nœud envoie.



nœud	sélecteurs *MPR*
A	$\{B, E\}$
B	$\emptyset$
C	$\{B, D\}$
D	$\{C, G\}$
E	$\{A, D, F, G\}$
F	$\emptyset$
G	$\emptyset$

FIGURE 7.3: Un réseau dont l'orientation des liens indique que le sommet pointé est sélectionné pour être *MPR*. Le tableau contient l'ensemble des sélecteurs *MPR* pour chaque sommet.

Enfin pour parvenir à construire une route d'une source à une destination, un sommet source doit posséder des couples où tour à tour le « dernier saut » d'un couple devient la destination de l'autre. Par exemple, dans la figure 7.3, si le nœud B désire communiquer avec F il lui faudra les messages *TCM* des nœuds A et E pour construire le chemin $[F, E, A, B]$ à partir des couples (E, F) et (A, E). Il pourra également construire le chemin $[F, E, D, C, B]$ grâce aux couples (E, F), (D, E) et (C, D) venant des messages *TCM* de E, D et C. On observe par ailleurs que le dernier chemin est plus long et que l'algorithme est capable de produire localement des plus courts chemins.

Routage à la demande

Dans [PR99b], les auteurs présentent *AODV (Ad hoc On Demand Distance Vector)* un protocole de routage à la demande, lui aussi reconnu par une *RFC* ([PR99a]), qui ne construit de chemin entre les nœuds d'un réseau que si une demande provenant de la source est émise. Comme vu plus haut, l'avantage de ce genre de protocole est le faible volume de données échangées pour contrôler le mécanisme. En revanche, le temps de réponse aux requêtes est bien supérieur à celui des algorithmes pro-actifs.

Pour assurer la cohérence et la fraîcheur des données échangées l'algorithme utilise un système de numéros de séquences. Lors d'une demande d'émission d'une source vers une destination, un paquet (une requête *RREQ (route request)*) est diffusé dans le réseau, toute station le recevant met à jour sa table de routage vers la source et propage le paquet. Quand il atteint sa destination ou une station possédant une route vers la destination avec un numéro de séquence supérieur ou égal à celui du *RREQ*, la station atteinte génère un nouveau paquet (une réponse *RREP (Route Reply)*) en direction

de la source. Ce paquet est transmis en *unicast*. Une fois le paquet *RREQ* reçu par la source, celle-ci commence à envoyer les données. Avec ce processus, rien n'assure que le chemin utilisé soit le meilleur, néanmoins la station source peut modifier sa route si elle reçoit d'autres informations (paquets *RREP*) plus pertinents (nombre de sauts inférieur).

En cas de rupture d'un lien utilisé dans une transmission, la station, extrémité du lien cassé, située du côté de la source envoie un paquet *RERR (Route Error)* en direction de la source. Une fois ce paquet reçu, la source peut recommencer le processus de recherche de route. Le protocole peut également produire des routes vers des groupes *multicasts*.

Routage Hybride

Pour tenter de pallier aux problèmes rencontrés par les algorithmes de routage pro-actifs et réactifs, les approches hybrides tentent de tirer profit des deux approches tout en limitant les inconvénients. Ainsi, *ZRP (Zone Routing Protocol)* [Haa97] utilise indépendamment les deux méthodes en faisant un découpage du réseau en deux zones distinctes. Une où s'exécute un algorithme pro-actif et une autre où un algorithme réactif prend le relais.

L'argument à la base de cette approche est que les requêtes de communication proches sont prioritaires aux demandes distantes ; elles ont aussi par nature de plus grandes chances d'aboutir. L'idée est donc d'offrir la rapidité de réponse d'un algorithme pro-actif pour les requêtes vers des nœuds proches de la source et de mettre en œuvre un algorithme moins gourmand en ressources comme un algorithme réactif pour les demandes plus lointaines. *ZRP* divise donc le réseau en deux zones. Les zones sont définies de manière décentralisée pour chaque nœud. La limite entre les zones est établie en fonction d'un nombre de sauts depuis le nœud source. Ainsi chaque sommet aura dans une première zone tous les nœuds accessibles à n sauts et dans une autre zone les autres nœuds du réseau. Le routage dans la zone proche du nœud source est appelé *IARP (Intra-zone Routing Protocol)*. Le routage dans le reste du réseau est appelé *IERP (Inter-zone Routing Protocol)*. Les auteurs proposent l'utilisation du protocole pro-actif *OLSR* pour l'*IARP* (la zone proche) et d'utiliser *AODV* (routage réactif) pour les parties lointaines du réseau.

AntHocNet : un algorithme fourmis pour le routage hybride

AntHocNet [DCDG04] est un algorithme de routage hybride pour les réseaux mobiles *ad hoc* construit autour du modèle des *ACOs* (présentés au chapitre 4). Il est constitué à la fois d'un mécanisme proactif et d'un mécanisme réactif. Il ne maintient pas les routes entre toutes les paires de stations du réseau à tout instant comme le fait *AntNet* [DD98] (dont il est fortement inspiré), mais maintient seulement les routes vers un ensemble de stations quand une demande d'échange de données est initiée. Dans une phase d'ini-

tialisation, l'algorithme adopte un comportement réactif où des agents « fourmis réactives » sont propagés dans le réseau à partir des sources, dans le but de trouver des chemins multiples vers les destinations. En accord avec le mécanisme des *ACOs*, les tables de routage des différentes stations sont considérées comme des pistes de phéromone dont les valeurs quantitatives rendent plus ou moins attractifs des chemins en fonction de leur qualité. Après la phase d'initialisation, les paquets de données sont transférés en choisissant des chemins de manière probabiliste et ce choix influencé par les tables de phéromone. En parallèle du transfert, une autre classe d'agents, les « fourmis proactives » est propagée dans le but de maintenir et de mettre à jour les tables de routage.

Construction réactive de chemins. Quand un nœud source s initie une communication vers un nœud destination d, une « fourmi réactive aller » F_d^s est créée pour rechercher cette destination. Le mode de déplacement de cette fourmi dépend de l'information locale que possède son nœud courant à propos de la destination. En effet, si le nœud courant ne possède pas d'information sur d, la fourmi F_d^s est clonée et diffusée sur les voisins du nœud courant. Si au contraire celui-ci possède une entrée pour d dans sa table de routage, la fourmi est simplement envoyée à un voisin. Les fourmis clonées à partir d'une même fourmi sont considérées comme appartenant à une même « génération ». Elles disposent d'un nombre limité de sauts afin d'éviter les chemins trop longs et inutiles. Une limitation existe également concernant la diffusion. En effet, un sommet recevant deux fourmis ou plus de la même génération dans un même intervalle de temps, choisira de ne laisser continuer sa recherche qu'à la fourmi ayant parcouru le chemin le plus court jusqu'à présent et détruira les autres. Le choix local d'un prochain nœud se fait de manière aléatoire et proportionelle aux pistes de phéromone locales de la même manière que cela est défini dans les *ACOs*, mais sans l'heuristique locale. Ainsi la probabilité P_{nd} pour une fourmi de choisir le voisin n du nœud courant i pour atteindre d vaut :

$$P_{nd} = \frac{(\mathcal{T}_{nd}^{i})^{\beta_1}}{\sum_{j \in \mathcal{N}_{jd}^{i}} (\mathcal{T}_{jd}^{i})^{\beta_1}} \tag{7.1}$$

avec $\mathcal{T}_{nd}^{i}$ la quantité de phéromones sur le lien (i,n) concernant la destination d, $\mathcal{N}_{jd}$ l'ensemble des voisins de i et β_1 une constante supérieure à 1 qui modifie l'importance des pistes de phéromone.

Quand la destination d est trouvée par une fourmi, celle-ci est convertie en « fourmi retour », et parcours son chemin à l'envers en marquant les tables de routage locales.

Routage stochastique des données. Quand la phase d'initialisation est passée, des chemins existent pour le routage de données. Ces données utilisent le même mécanisme aléatoire que les fourmis pour choisir les prochains sommets. La formule utilisée

est identique à l'équation 7.1 à la seule différence que la constante β_1 est remplacée par une constante β_2 avec $\beta_2 > \beta_1$ de manière à réduire l'exploration au profit de l'exploitation.

Maintien proactif des chemins. Pendant l'envoi de données, le nœud source envoie également des « fourmis proactives » dans le réseau dont le travail est de maintenir les chemins en mettant à jour les tables de routage. De même que pour les fourmis réactives, les fourmis proactives possèdent un mode « aller » et un mode « retour ». Le comportement normal d'une « fourmi proactive aller » est un déplacement de nœud en nœud en suivant la même formule probabiliste guidée par les pistes de phéromone. Une fois la destination atteinte, elle est transformée en fourmi « retour » et met à jour les informations (temps de parcours) relatives au chemin.

Il est également possible qu'une « fourmi proactive aller » soit clonée et diffusée dans le voisinage du chemin qu'elle parcourt. Dans ce cas, la fourmi explore de nouveaux chemins. La probabilité pour qu'une telle fourmi soit clonée et diffusée est faible (0.1 dans les expériences), et le nombre de ces mécanismes est limité (deux dans les expériences). Le but de ce mécanisme est de forcer l'exploration autour de solutions établies afin d'y apporter des variations et des améliorations.

Pour aider les « fourmis proactives aller », un mécanisme d'échange local et périodique d'information est mis en place. Avec un intervalle donné, les nœuds du réseau diffusent dans leur voisinage leur propre identifiant. Ce mécanisme permet à chaque nœud de connaître son voisinage : si un nouveau nœud apparaît il est ajouté dans les tables de routage de ses voisins et si un noeud disparaît, l'absence de message reçu par ses voisins permet à ceux-ci d'en déduire sa disparition.

Réparation des routes cassées. Quand un nœud ne reçoit plus d'information de la part de l'un de ses voisins, il en conclut sa disparition et met à jour sa table de routage. S'il possède une route alternative pour toutes les destinations que pouvait atteindre ce nœud alors il met à jour sa table de routage et informe ses voisins par diffusion de la modification. Si une (ou plusieurs) des destinations proposées par le nœud supprimé ne peut pas être atteinte, alors le nœud courant va créer une « fourmi réactive aller » vers cette destination. Si la fourmi trouve une route, la table de routage est mise à jour et la nouvelle route diffusée. Si la fourmi ne trouve pas de route, l'entrée pour cette destination est supprimée et toute requête de transfert de données vers cette destination retournera un message d'erreur à son expéditeur.

7.4 Approche fourmi pour le plus court chemin robuste

Cette section présente notre contribution au problème du plus court chemin robuste dans les *MANETs*.

Dans un premier temps une analyse théorique est menée afin de déterminer des bornes sur le nombre minimum de plus courts chemins et le nombre minimum de chemins robustes pour un graphe dynamique donné. Cette analyse centralisée est permise dans le cadre de simulations de réseaux mobiles *ad hoc* où le graphe de connection peut être analysé de manière différée (à l'aide d'un graphe évolutif par exemple). Cette analyse permet d'explorer le problème et de trouver mettre en évidence les valuers optimales au objectifs recherché. Ces valeurs sont ensuite utiliser pour comparer les résultats fournis par l'approche décentralisée et heuristique à base de fourmis artificiels proposée.

7.4.1 Analyse des graphes dynamiques

Pour cette analyse, seules les communications entre des stations appartenant à la même composante connexe sont considérées.

Rappelons que le problème qui nous intéresse est celui de la recherche d'une structure de plus courts chemins [4] d'une source à une destination qui minimise son taux de renouvellement.

Notons qu'en toute généralité il est peu probable qu'un plus court chemin reste valide tout au long de la vie d'un graphe dynamique. Ainsi plusieurs structures de plus courts chemins doivent se succéder. Cette remarque est également valable pour des structures robustes.

Nous proposons pour commencer l'analyse, de construire pour une paire de sommets donnés le nombre minimal de plus courts chemins. Si d'une étape à l'autre une structure de plus courts chemins est conservée, son taux de renouvellement est nul. Dans un premier temps, nous proposons un algorithme, EMPCC (Ensemble Minimum de Plus Courts Chemins), qui construit l'ensemble minimal des structures plus courts chemins entre deux sommets pendant l'évolution du graphe dynamique. Puis, dans un second temps nous relâchons la contrainte de plus court chemin et nous nous intéressons simplement à la contrainte d'existence d'un chemin entre deux sommets. Nous proposons de rechercher les structures (sous-graphes) reliant source et destination qui perdurent dans le temps le plus longtemps possible ; on parle alors de structures robustes. L'algorithme EMSR (Ensemble Minimum de Structures Robustes) permet de construire le nombre minimum de structures différentes reliant source et destination.

Ces deux algorithmes donnent des structures pour lesquelles il est possible de calculer un taux de renouvellement. Le taux de renouvellement des structures robustes est considéré comme une référence dans la recherche d'un taux de renouvellement minimum pour le plus court chemin.

4. Une telle structure est composée par l'ensemble des arêtes et des sommets qui participent à un plus court chemin entre source et destination.

Nombre minimum de structures plus courts chemins

Nous proposons EMPCC (algorithme 8) qui, de manière itérative, construit à chaque étape d'évolution la structure de plus courts chemins entre source et destination, et établit la liste minimale des plus courts chemins nécessaires pour relier source et destination. Voici le fonctionnement général de l'algorithme : à partir de la première étape $i = 0$, le sous-graphe de plus courts chemins est calculé et vaut comme structure de référence. De manière itérative, les événements des étapes suivantes sont appliqués et à chaque étape t une nouvelle structure de plus courts chemins est calculée. L'intersection entre la structure de plus courts chemins de référence et la nouvelle est calculée de manière à savoir si le plus court chemin qui vaut pour le nouveau graphe existe dans cette intersection. S'il existe bien un tel chemin, alors cette nouvelle structure devient la structure de référence. Le nombre de plus courts chemins n'augmente pas car la nouvelle structure de référence est incluse dans la précédente. Si au contraire, l'intersection entre le nouvelle structure de plus court chemin et la structure de référence ne permet pas de relier source et destination, cela signifie que la structure de référence n'est plus valide pour l'étape t. On sait en revanche que cette structure est valable jusqu'à l'étape $t-1$. Dans ce cas, la structure de référence invalide est mémorisée, la nouvelle structure de référence correspond alors à la structure de plus courts chemins de l'étape t et le nombre total de structures de plus courts chemins est incrémenté.

Remarque 1 *Notons que les structures de référence sont construites de manière itérative et mises à jour à chaque étape. Lors d'une étape ne provoquant pas de rupture de cheminement entre la source et la destination, la structure de référence est mise à jour de sorte que les arêtes disparaissant du graphe disparaissent aussi de la structure. Ainsi, durant sa construction, le cardinal (en nombre d'arête) d'une structure de plus courts chemins ne peut que diminuer.*

Théorème 1 *L'algorithme EMPCC construit un ensemble minimal de structures de plus courts chemins reliant une source à une destination données dans un graphe dynamique.*

Démonstration 1 *Soit A l'algorithme EMPCC produisant l'ensemble* $ES = \{S_0, S_1, S_2, \ldots, S_k\}$ *des structures de plus courts chemins, défini sur l'intervalle de temps* $[0, t_{fin}]$ *et* $ES^* = \{S_0^*, S_1^*, S_2^*, \ldots, S_p^*\}$ *un ensemble optimal (minimal) des structures de plus courts chemins produit par un algorithme optimal* A^* *pour le même graphe et le même intervalle de temps. Prouver que l'algorithme EMPCC produit l'ensemble minimal de structures de plus courts chemins, revient à prouver que* $|ES| \leq |ES^*|$.

Supposons qu'il existe un t $(0 \leq t \leq t_{fin})$ *tel que* $|ES| \leq |ES^*|$, *ce qui est vrai trivialement pour* t_0. *S'il existe un* t' $(t < t' < t_{fin})$ *tel que* $|ES| = |ES^*| + 1$, *cela signifie que la structure de référence* S^* *à* t' *construite par l'algorithme* A^* *perdure après la disparition de la structure courante S de notre algorithme A.*

Algorithme 8 : *EMPCC* (ensemble minimal de structures de plus courts chemins)

```
1  G_courant ← G_0;
2  s ← nœud source;
3  d ← nœud destination;
4  nombrePCC ← 0 ; /* nombre de structures de plus courts chemins */
5  ensemblePCC ← ∅ ;        /* ensemble des structures de plus courts
   chemins */
6  PCC ← plusCourtChemin(G_courant, s, d) ;     /* calcule la structure de
   plus courts chemins entre s et d dans G_courant en utilisant
   l'algorithme de DIJKSTRA. */
7  pour chaque étape i du graphe dynamique faire
8      G_courant ← G_i;
9      nouveauPCC ← PCC ∩ plusCourtChemin(G_courant, s, d);
       /* s'il existe un chemin entre s et d dans nouveauPCC */
10     si ∃chemin(s, d) ∈ nouveauPCC alors
11         PCC ← nouveauPCC
12     sinon
           /* la nouvelle structure de plus court chemin n'est plus
              valide, on conserve la précédente */
13         nombrePCC ← nombrePCC + 1;
14         ensemblePCC ← ensemblePCC ∪ PCC;
15         PCC ← plusCourtChemin(G_courant, s, d);
16     fin
17 fin
```

Or, si avant t' $|ES| = |ES^*|$ alors la formation de S^* a eut lieu à une date antérieure ou égale à t. Or à t, par construction, la structure S inclut l'ensemble des sommets et des arêtes participants à un plus court chemin. Donc à t $S^* \subseteq S$. De plus, pour tout t'' $(t < t'' < t')$, chaque suppression d'un élément e (sommet ou arête) dans S correspond à la disparition de e dans le graphe. Donc si à t' il n'existe plus de chemin dans S depuis la source vers la destination, c'est également le cas pour S^*. Donc $|ES| \leq |ES^*|$ et l'algorithme EMPCC est optimal.

Nombre minimal de structures robustes

Quand on ne s'intéresse plus à la notion de plus court chemin mais seulement à l'existence d'un chemin de bout en bout entre la source et la destination, il devient intéressant de rechercher l'ensemble minimal de structures ou chemins qui relient source et destination dans le graphe dynamique. L'algorithme 9, EMSR (Ensemble Minimal de Structures Robustes), propose avec le même mécanisme de construire l'ensemble mi-

nimal de structures reliant deux sommets. Le mécanisme est similaire à l'algorithme précédent. Au départ la structure de référence est le graphe entier. À chaque étape t une nouvelle structure est construite avec l'intersection de la structure de référence valide à l'étape précédente et le graphe courant. Si un chemin existe dans cette nouvelle structure, cela signifie que la structure valide à l'étape précédente est toujours valide à cette étape. Si au contraire l'intersection ne contient plus de chemin reliant source et destination, la structure de référence n'est plus valide. Elle l'est néanmoins jusqu'à $t-1$. L'ancienne structure de référence est mémorisée, une nouvelle structure de référence est construite et correspond au graphe courant et le nombre de structures est incrémenté.

Algorithme 9 : *EMSR* (ensemble minimal de structures robustes)

```
1  G_courant ← G_0;
2  s ← nœud source;
3  d ← nœud destination;
4  nombreSR ← 0 ;                    /* nombre de structures robustes */
5  ensembleSR ← ∅ ;                 /* ensemble des structures robustes */
6  SR ← G_courant ;      /* la structure robuste est le graphe courant */
7  pour chaque étape i du graphe dynamique faire
8      G_courant ← G_i;
9      nouvelleSR ← SR ∩ G_courant;
       /* s'il existe un chemin entre s et d dans nouveauSR */
10     si ∃chemin(s,d) ∈ nouvelleSR alors
11         SR ← nouvelleSR
12     sinon
           /* la nouvelle structure n'est plus valide, on conserve la
              précédente. */
13         nombreSR ← nombreSR + 1;
14         ensembleSR ← ensembleSR ∪ SR;
15         SR ← G_courant;
16     fin
17 fin
```

Théorème 2 *L'algorithme EMSR construit un ensemble minimal de structures robustes reliant une source et une destination données dans un graphe dynamique.*

Démonstration 2 *Le cheminement utilisé dans la preuve du théorème 1 de l'algorithme EMPCC est le même pour l'algorithme EMSR. L'algorithme produit l'ensemble minimal de structures liant la source et la destination dans un graphe dynamique.*

La figure 7.4 illustre cette notion de structure robuste.

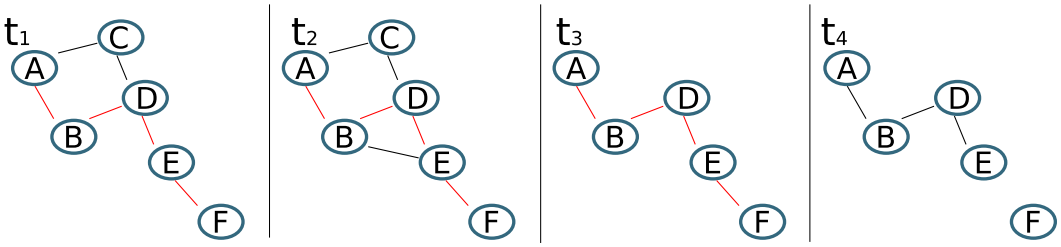


FIGURE 7.4: Exemple de structure robuste. Voici un graphe dynamique à quatre étapes de temps successives. La structure robuste correspond au chemin $[A,B,D,E,F]$. La structure créée au temps t_1 prévaut jusqu'à t_3 même si en t_2 elle ne représente pas le plus court chemin entre A et F.

L'ensemble minimal des structures robustes ne possède pas un taux moyen de renouvellement optimal. Néanmoins, dans la grande majorité des expérimentations cet ensemble possède un taux moyen de renouvellement bien inférieur à celui de l'ensemble minimal des plus courts chemins ou même à celui de toutes les expérimentations heuristiques présentées par la suite. Ce taux servira donc de borne inférieure de référence dans les expérimentations d'algorithmes décentralisés qui suivent.

En conclusion, l'analyse par les algorithmes EMPCC et EMSR nous permettent de déterminer pour des graphes dynamiques extraits de simulations, des bornes inférieures pour les deux contraintes que l'on cherche à minimiser. La question qui est posée maintenant est de savoir si ces bornes peuvent être approchées dans un contexte décentralisé et dynamique, pendant l'évolution du réseau. La partie suivante propose un algorithme inspiré des colonies de fourmis qui tente de répondre à cette question.

7.4.2 Approche inspirée des colonies de fourmis

Hypothèses et paramètres de simulation

Pour la partie expérimentale, des simulations sont réalisées à l'aide du simulateur *Madhoc* [Hog08]. Ce simulateur fait évoluer de manière itérative un réseau mobile. Il gère le déplacement, les connexions ainsi que la volatilité des stations considérées. Il est entièrement paramétrable. Parmi les paramétrages possibles, citons la surface de simulation, le modèle de mobilité des stations, leur nombre maximum de connexions, le modèle d'environnement, mais aussi les technologies de communication utilisées et les caractéristiques des interfaces réseau.

Le simulateur est également dirigé par des règles probabilistes car la mobilité des stations est couramment simulée en utilisant des modèles tels que le *Random Waypoint Mobility Model*. Ce fonctionnement probabiliste ne permet donc pas d'assurer qu'il existera à chaque pas de simulation un chemin de bout en bout entre deux stations. Ce qui est réaliste dans un véritable réseau mobile et décentralisé.

L'algorithme décentralisé proposé peut fonctionner en dépit de pertes de connexité (les messages ne sont pas délivrés, mais l'algorithme le prévoit). Par contre, l'analyse différée par les algorithmes EMPCC et EMSR des graphes dynamiques résultant des simulations gère la perte de connexité en ignorant les étapes correspondantes.

Pour les simulations un contexte doit être défini. Nous avons vu que les applications de type *streaming* n'étaient quasiment pas envisageables dans un contexte dynamique. À l'opposé, les services de diffusion d'informations utilisant le modèle *store and forward* dans des réseaux tolérant les délais ne nous intéressent pas non plus car la notion de plus court chemin y est inexistante. Notre proposition se situe donc dans le cas d'applications nécessitant des chemins de connexions de bout en bout mais qui tolèrent également des latences d'acheminement. Les applications de type transfert de fichiers ou même de type Web nous semblent envisageables.

Concernant les réseaux, c'est la mobilité des stations qui est le point important et qui définit les possibilités de services. Nous souhaitons considérer une mobilité raisonnable permettant l'échange d'un minimum d'informations. Ainsi la mobilité des piétons dans un environnement urbain nous semble un scénario pertinent dans le cadre de notre étude.

Dans le cas de la simulation de dispositifs radio portés par des piétons il nous faut également prendre en compte le fait que ces dispositifs ne sont pas constamment allumés. Nous considérons donc un modèle de volatilité des stations.

Description de l'algorithme

L'idée est de concevoir un algorithme au fonctionnement décentralisé à base de fourmis, qui utilisent leur capacité naturelle à trouver des plus courts chemins entre une source et une destination en se déplaçant dans le réseau.

La première contrainte à considérer avec cette approche est celle du nombre d'individus. En effet, les algorithmes de type colonie de fourmis sont connus pour leur efficacité à la condition d'utiliser un nombre suffisant d'entités. Cela implique une certaine puissance de calcul et un volume important d'informations échangées. Or la puissance de calcul et la bande passante sont les principaux points faibles des *MANETs*. Nous sommes néanmoins convaincus qu'il existe un compromis entre l'efficacité de la recherche et le volume des communications qui passe à la fois par l'optimisation du nombre d'individus et par l'optimisation des structures de données échangées.

Philosophie

L'idée générale exploitée dans cette proposition reprend les idées exposées dans le chapitre 5 concernant l'utilisation de colonies de fourmis sur la base d'informations purement locales dans un environnement décentralisé. En cela, l'algorithme que nous proposons n'est pas un algorithme d'optimisation classique, guidé par sa production de solutions évaluées de manière centralisée comme le sont les *ACOs*, vus dans le chapitre

4. Les fourmis de notre système n'ont pas pour objectif explicite de construire des solutions au problème considéré même si elles construisent effectivement des chemins. C'est le dépôt de phéromone dans l'environnement (ici les stations du réseau mobile) qui définit les solutions que l'on cherche. En effet, les solutions sont des routes au sens classique des réseaux et comme dans les réseaux classiques, la route menant d'un point à un autre est découverte au fur et à mesure du parcours des stations.

Le fonctionnement décentralisé de notre proposition hérite de travaux antérieurs tels que *Ant-Based Control* [SHBR97] pour la répartition de charge dans les réseaux de télécommunications, *AntHocNet* [DCDG04] pour le routage dans les réseaux mobiles *ad hoc* ou plus proche de nous, dans la même équipe, $AntCO^2$ [Dut05]. Fidèle à ces travaux, notre approche utilise les mêmes mécanismes locaux de sélection de voisinage. Ainsi, la formule de choix d'un prochain voisin reprend l'équation 4.4 page 52 de *Ant System* avec l'attraction conjointe des pistes de phéromone et de la visibilité.

Concernant la visibilité, nous proposons l'utilisation de la métrique de volatilité définie dans le chapitre 3, définition 17. Plus précisément, c'est l'inverse de la volatilité, à savoir l'âge moyen des arêtes qui est utilisé pour mesurer la robustesse. Cette métrique est calculée localement. Ainsi le processus de recherche est guidé par les pistes de phéromone ainsi que par les arêtes dont l'âge moyen est plus élevé. Notre intuition est que l'utilisation de liens qui sont en moyenne plus âgés dans les routes permet de diminuer le taux de renouvellement moyen de ces chemins. L'utilisation de la volatilité se justifie aussi par le fait qu'elle soit une métrique construite de manière décentralisée par les stations.

Modèle

Le modèle proposé est constitué d'agents mobiles qui se déplacent de station en station dans le réseau. Les agents ont tous une même station d'origine qui est aussi la station source pour laquelle une route est recherchée. Cette station est initiatrice de la recherche, elle crée donc les agents, on la nomme aussi le nid. Les agents partent donc de cette station source à la recherche de la destination. Une fois la destination trouvée, ils reviennent vers la source.

La présentation de la méthode se réfère au champ lexical des graphes et des algorithmes fourmis. Les agents du paragraphe précédent correspondent aux fourmis mentionnées dans la suite de cette section. Les sommets correspondent aux stations et les arêtes aux liens de communication entre les stations.

Comportement. Les fourmis se déplacent dans le graphe de sommet en sommet selon deux modes, un mode « aller », à la recherche de la destination et un mode « retour », vers la source quand la destination est trouvée. On retrouve ce comportement dans *AntHocNet* où les auteurs parlent de fourmis *forward* et *backward*.

Mode « aller ». La fourmi se déplace itérativement de station en station à la recherche de la destination tout en mémorisant le chemin qu'elle emprunte. À chaque itération le choix du prochain sommet suit une formule aléatoire proportionnelle similaire à celle définie dans *Ant System*. Pour une fourmi k située sur le sommet i à l'étape t, la probabilité de choisir le sommet j à la prochaine itération est :

(7.2)

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in voisins(i) \setminus chemin(k)} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} & \text{si } j \in voisins(i) \setminus chemin(k) \\ 0 & \text{sinon} \end{cases}$$

- $\tau_{ij}(t)$ est la quantité de phéromone sur l'arête (i, j) à l'étape t ;
- η_{ij} est la visibilité sur l'arête (i, j) elle est égale à l'âge moyen de l'arête et correspond à l'inverse de la volatilité : $\eta_{ij} = \frac{1}{volatilité(i,j)}$;
- $voisins(i)$ est l'ensemble des sommets adjacents à i ;
- $chemin(k)$ est la liste des sommets appartenant au chemin parcouru par la fourmi k ;
- $voisins(i) \setminus chemin(k)$ correspond à l'ensemble des voisins de i privé de ceux figurant dans le chemin de k ;
- α et β sont deux paramètres permettant de relativiser l'importance des pistes de phéromone par rapport à la visibilité.

Mode « retour ». Il survient quand la fourmi a trouvé la destination. Elle parcourt en sens inverse le chemin parcouru à l'aller. Sur chaque sommet elle renforce la piste de phéromone. De même que dans *Ant System*, le renforcement $\Delta\tau_{ij}^{k}$ est proportionnel à la longueur du chemin qu'elle a réalisé. $\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$, avec Q une constante correspondant également à la quantité initiale de phéromone sur l'arête et L_k est la longueur du chemin effectué par la fourmi k.

Contrôle de la population. Puisqu'aucun contrôle sur le comportement des fourmis ne peut être effectué de manière centralisé, une durée de vie limitée exprimée en nombre de sauts est définie. À l'image du mécanisme de *TTL* (*Time To Live*) des paquets IP dans les réseaux classiques, les fourmis décident de disparaître au bout d'un certain nombre de sauts. Ce paramètre permet d'éviter que des fourmis parcourent indéfiniment des zones éloignées du graphe. En contrepartie le nid (sur le sommet source) tente de maintenir un nombre constant de fourmis dans le graphe. Pour cela un mécanisme de mémorisation des dates de départ des fourmis peut être mis en œuvre de manière à générer une fourmi de remplacement quand une fourmi n'est pas repassée par la source au delà de son *TTL*.

Gestion des impasses. Lors de son parcours il est possible qu'une fourmi se retrouve dans une situation ou plus aucun voisin n'est disponible pour continuer. Cela se produit par exemple lorsqu'elle arrive sur une « feuille », un sommet avec une seule arête. La fourmi venant de cette arête ne la considère pas dans sa formule de choix du prochain sommet et ne dispose plus d'alternative. Dans ce cas un dispositif de « marche arrière » est mis en œuvre. Le dernier sommet du chemin est considéré et la fourmi revient sur ce dernier sommet pour y appliquer de nouveau la formule de choix classique. Pour que son choix ne la mène pas de nouveau dans cette impasse, le sommet en question, même s'il a été retiré du chemin, est mémorisé dans une liste tabou. Ainsi la sélection des voisins disponibles dans la formule de choix du prochain sommet prend maintenant en compte les sommets figurant dans cette liste tabou en plus de ceux du chemin en cours de construction. Dans un souci d'optimisation il est concevable d'utiliser une liste tabou de taille fixe à fenêtre glissante. La figure 7.5 illustre ce phénomène : au temps t_1 la fourmi est dans une impasse, elle revient en arrière et ajoute le dernier sommet visité à sa liste tabou. Au temps t_2 sa liste de voisins possibles est toujours vide. Elle applique de nouveau la procédure de retour arrière en mettant à jour sa liste tabou. Enfin au temps t_3 sa liste de voisins n'est plus vide, elle peut continuer dans sa progression.

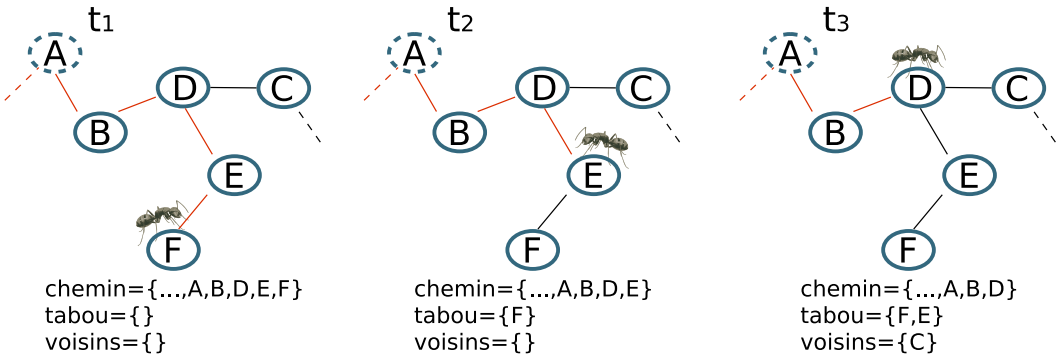


FIGURE 7.5: Illustration du phénomène d'impasse dans le parcours d'une fourmi.

Phéromone orientée. Nous proposons l'utilisation sur chaque arête de deux quantités de phéromone différentes. Une marquant le passage dans un sens (de A vers B) et l'autre marquant le passage dans l'autre sens (de B vers A). Ce mécanisme a deux avantages. Il permet de différencier les sens de circulation. Ainsi une fourmi croisant une piste de phéromone fortement marquée ne risque pas de se diriger vers la source au lieu de la destination. L'autre avantage concerne l'implémentation dans un réseau réel. Les liens de communication n'ont pas d'existence tangible et l'image du dépôt de phéromone sur les arêtes impose aux deux stations concernées par une connexion de prendre en charge cette quantité et de se synchroniser régulièrement. Or dans notre proposition de

phéromone orientée, chaque sommet a deux valeurs de phéromone pour chaque arête, une correspondant au sens sortant et une correspondant au sens entrant. En terme de réseau chaque station tient à jour la quantité de phéromone qui la concerne et n'a plus besoin d'échanger d'informations avec ses voisines.

Effet mémoire des pistes de phéromone. Dans la même idée que la métrique de volatilité qui est construite et maintenue par les stations qui se souviennent des différents états de leurs connexions, les stations peuvent également tenir à jour un historique des quantités de phéromone associées aux connexions quand celle-ci disparaissent. Ce mécanisme permet lors de la réapparition d'une connexion de ré-allouer la quantité de phéromone qu'elle possédait jadis. Le but de ce mécanisme est encore une fois de tirer profit de l'expérience acquise et de mettre éventuellement en œvre la supposition qu'une arête attractive qui disparaît puis revient doit rapidement retrouver son attractivité. L'efficacité de ce mécanisme qui peut ne pas sembler évident s'avère déterminant dans les environnements à forte volatilité comme cela est corroboré par les résultats de simulations.

7.4.3 Simulations et résultats

Cette partie présente les différentes simulations menées pour l'analyse des performances de la méthode. Nous présentons dans un premier temps les scénarios de simulation générés avec *Madhoc*. Ensuite les résultats obtenus pour le problème exprimé sous la forme d'un problème multi-objectifs sont présentés. Ces résultats sont comparés à ceux obtenus lors de l'analyse ainsi qu'à des résultats obtenus avec une méthode de recherche aléatoire décentralisée. Enfin une présentation des différents paramètres du système est faite et une analyse des résultats est proposée.

Scénario de simulation de réseaux mobiles. Nous considérons des réseaux mobiles *ad hoc* avec mobilité limitée tels que peuvent l'être des réseaux formés par des matériels de type *PDA* et téléphones mobiles équipés d'interfaces supportant une technologie conforme à la norme *IEEE 802.11b*. Des piétons dans un espace urbain, ville ou centre commercial assurent la mobilité de ces matériels. Deux scénarios ont été construits pour les besoins de la simulation.

Le premier scénario nommé « couloir » est caractérisé par une mobilité forte et par des stations non volatiles, c'est-à-dire des stations qui restent actives en permanence. Les stations se déplacent en deux groupes circulant en sens opposé. Ce modèle s'inspire des modes de déplacement des piétons dans les couloirs du métro. L'ensemble des paramètres, densité, rayon de propagation, vitesse de déplacement, etc, qui précisent le scénario sont regroupés dans la table 7.1. La caractéristique principale de ce scénario est l'hétérogénéité des liens qui sont de deux types principaux : des liens stables entre sta-

tions appartenant au même groupe et des liens à durée de vie très limitée entre stations appartenant à des groupes distincts.

Le second scénario nommé « volatilité » est caractérisé par une faible mobilité et par des stations volatiles, c'est-à-dire des stations dont l'activité varie au cours du temps. Les stations se déplacent dans un environnement ouvert selon un modèle de type *RWP*. Ce modèle s'inspire des modes de déplacement des individus dans un supermarché. Tout comme pour le précédent scénario, l'ensemble des paramètres qui précisent le scénario, pourcentage de stations volatiles, portée radio, surface de l'espace ouvert, etc, sont regroupés dans la table 7.1. Dans ce scénario, au cours du temps, la présence de certains liens est discontinue.

Pour les deux scénarios, les simulations sont effectuées sur une base temporelle discrète. Cela signifie que chaque itération du simulateur fait évoluer le temps simulé d'une même durée. Le choix de ce pas de temps influence la dynamique du graphe des connexions et joue un rôle également dans l'exécution de l'algorithme fourmi. En effet, dans le chapitre 5 nous avons défini la vitesse relative de l'application par rapport à la vitesse du graphe (définition 25 page 95) et dans un souci de cohérence avec les contraintes réelles des réseaux mobiles, nous avons fixé ce pas de temps à 1 seconde. Durant cette seconde, les stations se déplacent dans leur environnement d'une distance proportionnelle à leur vitesse et chaque fourmi peut traverser jusqu'à k stations, k est un paramètre variable de la simulation dont l'impact sera analysé plus loin.

TABLE 7.1: Paramètres de simulation pour les scénarios « couloir » et « volatilité ».

	scénario « couloir »	scénario « volatilité »
Déplacement	d'un *hot spot* à un autre	*RWP*
Espace	fermé (couloir)	ouvert
Surface	couloir de $500m$	$1km^2$
Nombre de stations	150	400
Vitesse de déplacement	$1,4$ à $2m/s$ (5 à $7km/h$)	0 à $1m/s$
Volatilité	pas de volatilité	40% des stations
Portée radio	$20m$	$40m$

Notons que les portées radio sont volontairement réduites par rapport aux chiffres couramment considérés dans la littérature. Sur la base d'expériences menées au sein de l'équipe, nous avons constatés que les portées habituellement considérées étaient inatteignables en conditions réelles. En particulier, un environnement fermé comme une pièce ou un couloir diminue de manière significative à la fois la portée et le nombre de voisins détectés par les stations.

La figure 7.6 représente un sous-graphe du graphe de connexions obtenu par simulation du scénario « volatilité ». Les sommets représentent les stations et sont valués par l'identifiant de la station correspondante. Les liens de communication sont des-

sinés avec un diagramme circulaire qui indique leur volatilité. Les stations de couleur plus foncée sont celles qui présentent un comportement volatile. La volatilité est implémentée de manière probabiliste dans la simulation. À chaque pas de temps les stations choisies sont déconnectées de leurs voisins avec une probabilité non nulle (20%) puis reconnectées à l'étape suivante (avec une probabilité de 90%). Cette information de volatilité n'est pas connue de notre algorithme décentralisé qui ne connaît que la métrique de volatilité calculable localement.

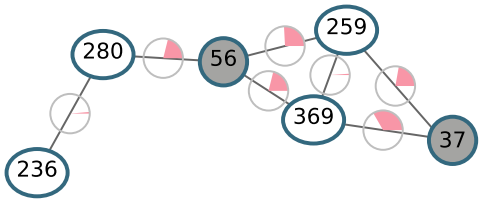


FIGURE 7.6: Échantillon du réseau de communication obtenu par la simulation du scénario « volatilité ». Les stations sont représentées par une ellipse contenant un numéro identifiant. Les liens de communication sont dessinés avec un diagramme circulaire qui indique leur volatilité. Les stations de couleur plus foncée sont celles qui ont un comportement volatile. Une vidéo de cette simulation est disponible à l'adresse http://litis.univ-lehavre.fr/~pigne/these/.

La figure 7.7 donne une idée de la simulation du scénario « couloir ». Une vidéo également disponible à l'adresse http://litis.univ-lehavre.fr/~pigne/these/ permet d'illustrer le phénomène de groupe que nous cherchons à mettre en évidence. Les stations se déplaçant dans la même direction ont des liens de communication caractérisés par une faible volatilité. Les liens qui s'établissent entre des stations se déplaçant à contresens ne durent pas et n'ont aucune chance de réapparaître.

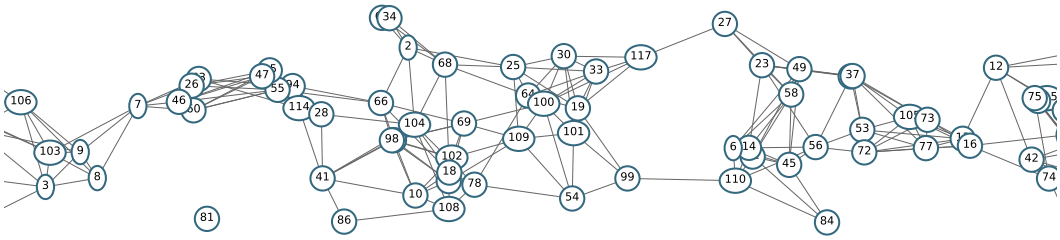


FIGURE 7.7: Simulation du scénario « couloir ». Une vidéo disponible à l'adresse donnée ci-dessus illustre le phénomène de groupes qui avancent ensemble et se croisent.

Optimisation multi-objectifs. L'objectif recherché ici est double : minimiser la taille des chemins construits et minimiser le taux de renouvellement des liens de ces che-

mins. La figure 7.8 donne une vue d'ensemble des résultats que nous avons obtenus dans ce travail concernant les graphes dynamiques extraits du scénario « volatilité ». Les données sont des moyennes prises pour plusieurs paires différentes de sommets source destination ainsi que pour plusieurs graines aléatoires. Quatre valeurs sont représentées dans l'espace multi-objectifs des solutions :

- la droite horizontale nommée « Borne inférieure plus courts chemins » représente la borne inférieure en nombre de sauts des plus courts chemins moyens entre 8 paires des sommets différents durant toute la simulation (100 étapes) ;
- la droite verticale représente la borne inférieure de référence pour les taux de renouvellement obtenus par l'algorithme EMSR (page 114) ;
- le point « + » nommé « Moyenne méthode fourmis » représente la moyenne de toutes les simulations effectuées avec différents paramètres pour les différentes paires de sommets et les graines aléatoires ;
- enfin, le point « × » nommé « Moyenne méthode aléatoire » donne la moyenne des résultats obtenus dans les mêmes conditions avec une méthode de recherche aléatoire de la destination qui utilise néanmoins les mécanismes de liste tabou et de détection d'impasse explicités ci-dessus. Cette méthode aléatoire peut être comparée à un parcours de graphe en profondeur.

Nous observons que l'approche proposée ici, bien qu'offrant des solutions avec des chemins plus longs que la méthode aléatoire, produit des structures dont les taux de renouvellement sont bien inférieurs à ceux obtenus par la méthode aléatoire.

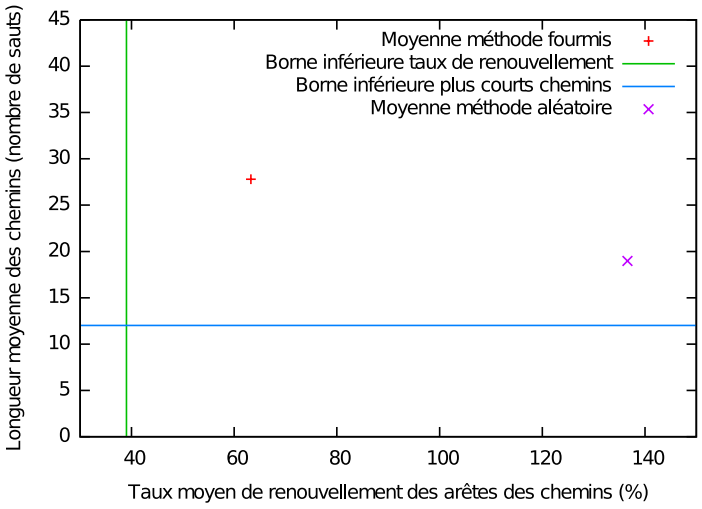


FIGURE 7.8: Répartition générale des solutions dans l'espace multi-objectifs.

Analyse des paramètres. Le reproche communément fait aux approches méta-heuristiques dédiées à l'optimisation est que la polyvalence de leur utilisation dans plusieurs problèmes est aussi accompagnée d'un grand nombre de paramètres à ajuster. Notre proposition n'échappe pas à cette critique et possède également quelques paramètres qui peuvent changer radicalement son comportement. Néanmoins, les tests effectués sur les deux scénarios semblent montrer une certaine stabilité du système.

Commençons par différencier deux types de paramètres, ceux qui agissent sur des grandeurs quantitatives et les paramètres qui modifient le comportement du système sans engager de variation quantitative. Par grandeur quantitative nous entendons par exemple des temps de calcul, des volumes de données à échanger ou des temps d'attente. Dans le contexte des *MANETs* toutes ces grandeurs correspondent à des contraintes à minimiser au même titre que la longueur des chemins ou le taux de renouvellement. En général plus la grandeur considérée est importante plus l'algorithme est performant. Dans le cas présent, deux paramètres sont de ce type : le nombre d'agents à propager dans le réseau et le nombre de sommets que chaque fourmi peut parcourir entre chaque étape d'évolution du graphe. Cette grandeur est directement reliée à la notion de vitesse relative définie page 25.

Dans les expérimentations, le nombre de fourmis à été fixé à 40 individus et k est le nombre de sommets visités par chaque fourmi à chaque étape d'évolution du graphe (vitesse relative). Ce paramètre de vitesse relative fait l'objet d'une analyse page 127 et est fixé à 100 pour les simulations suivantes.

D'autres paramètres sont plus simplement liés au fonctionnement de l'algorithme. Parmi ceux-ci, considérons les deux paramètres de la règle de choix du prochain sommet, α et β qui donnent plus ou moins d'importance aux pistes de phéromone et à la visibilité. Ces deux paramètres étant dépendants l'un de l'autre, α est fixé à 1 et l'évolution de β est analysée. D'autre part le taux d'évaporation ρ est également une valeur clé dans le fonctionnement de l'algorithme. Finalement ce sont deux paramètres β et l'évaporation ρ qui nécessitent d'être ajustés.

La figure 7.9 montre le résultat des simulations menées sur le scénario « volatilité ». Les simulations consistent en l'exécution de notre algorithme sur un graphe dynamique de 100 étapes. Chaque point dans le graphique correspond aux résultats moyens obtenus avec 8 paires de sommets source/destination et 6 graines aléatoires différentes, soit 48 simulations. D'un point à l'autre, les paramètres β et ρ sont différents. Ces points sont le résultat de la recherche d'un front de Pareto visant à minimiser les deux contraintes. Le tableau 7.2 donne le front de Pareto trouvé lors de ces simulations avec les valeurs de β et ρ associées.

L'observation des résultats montre de très faibles valeurs pour les paramètres. Sur le front, le paramètre β qui influe sur la visibilité dans la formule de choix du prochain sommet (équation 7.2 page 118) varie entre 0 et 0.3 alors que le paramètre α de la même formule est fixé à 1. Cela implique que la visibilité qui exprime l'âge moyen des arêtes est très peu représentée par rapport aux pistes de phéromone. Notons qu'une valeur de 0

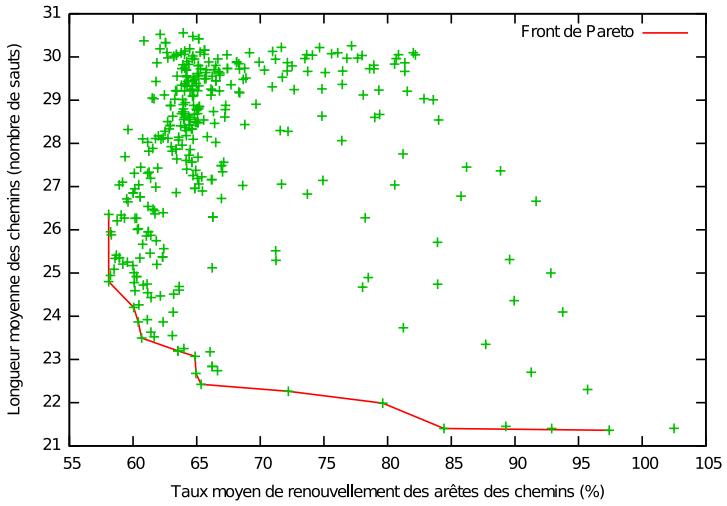


FIGURE 7.9: Répartition des solutions dans le scénario « volatilité ». Chaque point donne la valeur moyenne du taux de renouvellement et de la longueur du chemin pour 8 couples source-destination et 6 graines aléatoires différentes. Pour chaque point les paramètres ρ et β varient.

TABLE 7.2: Valeurs des paramètres ρ et β produisant le front de Pareto de la figure 7.9 entre le taux de renouvellement et la longueur des chemins pour le scénario « volatilité ».

ρ	β	renouvellement	longueur
0.0	0.3	**58.08**	26.35
0.05	0.03	60.41	23.86
0.05	0.05	60.05	24.20
0.1	0.02	60.67	23.49
0.1	0.0	63.52	23.19
0.1	0.08	58.08	24.79
0.15	0.03	64.87	23.07
0.15	0.0	65.35	22.42
0.18	0.0	64.94	22.67
0.3	0.0	72.19	22.26
0.4	0.0	79.61	21.98
0.5	0.0	84.43	21.40
0.8	0.0	97.40	**21.36**

pour le paramètre β implique que l'âge moyen des arêtes n'est pas pris en compte dans la formule de choix 7.2 (page 118). Concernant l'évaporation ρ, qui peut varier entre 0 et 1, on trouve dans le front de Pareto des valeurs variant de 0 à 0.9. Une valeur de 0 pour ce paramètre signifie qu'aucune évaporation n'est effectuée sur les pistes. À l'inverse une évaporation de 1 effacerait toutes les pistes de phéromone, et transformerait

l'algorithme fourmi en algorithme de recherche aléatoire.

Le front de Pareto nous montre que la meilleure solution concernant le taux de renouvellement est de 58,05% avec $\rho = 0$ et $\beta = 0,3$, ce qui signifie que le taux de renouvellement minimum est atteint en utilisant au maximum l'âge moyen des arêtes et en conservant au maximum les pistes de phéromone. La meilleure solution pour l'objectif de longueur des chemins est de 21,36 avec $\rho = 0.9$ et $\beta = 0$, ce qui correspond au comportement classiquement décrit concernant les algorithmes fourmis et leur capacité à trouver des plus courts chemins.

Notons que les résultats d'analyse de paramètres du scénario « couloir », ne différant pas notablement, ne sont pas présentés ici.

Analyse de la propriété de mémoire des pistes de phéromone. La technique de mémorisation des valeurs des pistes de phéromone présentée précédemment consiste à conserver la valeur des phéromones dans les nœuds quand une arête disparaît puis de la réaffecter quand elle réapparaît. Dans le scénario « volatilité » les arêtes qui disparaissent ont une probabilité très forte de réapparaître et cette heuristique semble tout à fait à propos pour accélérer le processus de recherche et redonner rapidement de l'attrait aux « bons chemins ». La figure 7.10 montre l'efficacité de l'heuristique sur les simulations faites sur ce scénario.

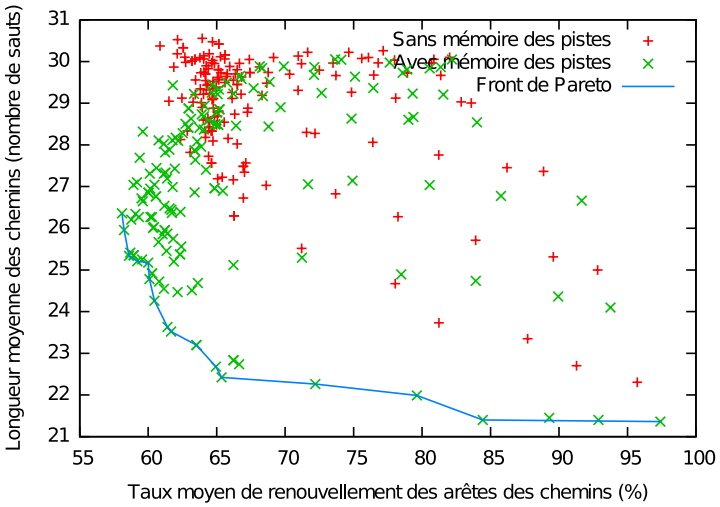


FIGURE 7.10: Comparaison des résultats obtenus avec et sans conservation des pistes de phéromone dans le scénario « volatilité ».

Dans le scénario « couloir », les liens qui existent entre les stations d'un même groupe ont une durée de vie longue et une faible probabilité de déconnexion, ils ne sont donc pas concernés par les cas de reconnexion. Les liens entre stations qui se croisent ont

une durée de vie très courte et une très faible probabilité de reconnexion. Il n'y a donc *a priori* pas grand intérêt à utiliser cette heuristique dans le scénario « couloir ». Les résultats de simulation reportés dans la figure 7.11 illustrent ce fait.

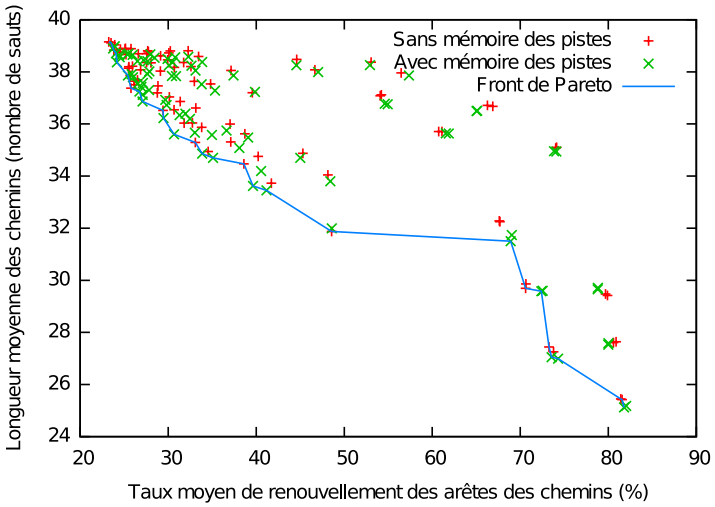


FIGURE 7.11: Efficacité de l'heuristique de mémoire des pistes de phéromone dans le scénario « couloir ».

Analyse de la vitesse relative. Un paramètre dans cette simulation régit la vitesse de l'algorithme par rapport à l'évolution du graphe dynamique comme nous l'avons défini page 95. L'algorithme est itératif et est exécuté k fois entre chaque étape de modification du graphe. La figure 7.12 montre l'évolution du taux moyen de renouvellement des liens des chemins construits pour une simulation donnée en fonction du nombre k d'itérations par étape d'évolution du graphe (la vitesse relative).

La figure 7.13 montre l'évolution de la longueur moyenne des chemins en fonction de la vitesse relative.

Dans les deux observations l'augmentation de la vitesse relative améliore notablement les résultats obtenus, pour les deux objectifs.

Conclusion sur cette approche. La volonté de trouver des chemins robustes dans des graphes dynamiques, dans le contexte des *MANETs*, nous a amené à proposer un algorithme à base de fourmis. Cet algorithme décentralisé utilise pour son fonctionnement des informations locales. Certaines des idées proposées pourraient être exploitées pour la mise en oeuvre d'un service de routage pour les réseaux mobiles *ad hoc*. Deux propositions originales se dégagent de ce travail. L'utilisation de la métrique de volatilité des arêtes, calculée localement, permet, comme le montrent les simulations, de faire di-

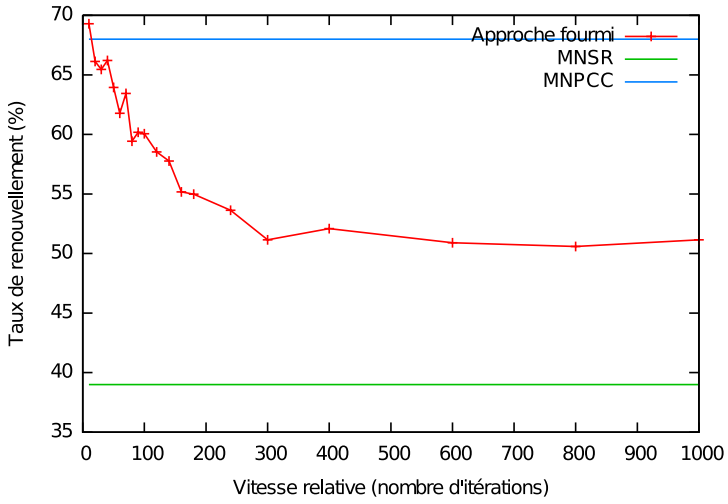


FIGURE 7.12: Taux de renouvellement moyen des liens des chemins construits par les fourmis en fonction de la vitesse relative de l'algorithme. Le scénario « volatilité » est utilisé, ρ vaut 0.05 et β vaut 0.3. Par rapport aux simulations précédentes, les autres paramètres restent inchangés. Les droites montrent la valeur du taux de renouvellement pour les algorithmes EMSR et EMPCC.

minuer le taux de renouvellement des arêtes. Cette propriété est très intéressante dans le contexte des *MANETs* puisque les changements de chemins impliquent des synchronisations coûteuses en ressources entre les stations. La seconde proposition consiste à conserver les valeurs de phéromone des arêtes qui disparaissent pour les leur réaffecter en cas de réapparition. Cette dernière proposition a montré son efficacité lorsque les connexions qui disparaissent ont de bonnes chances de réapparaître comme dans le scénario de « volatilité ».

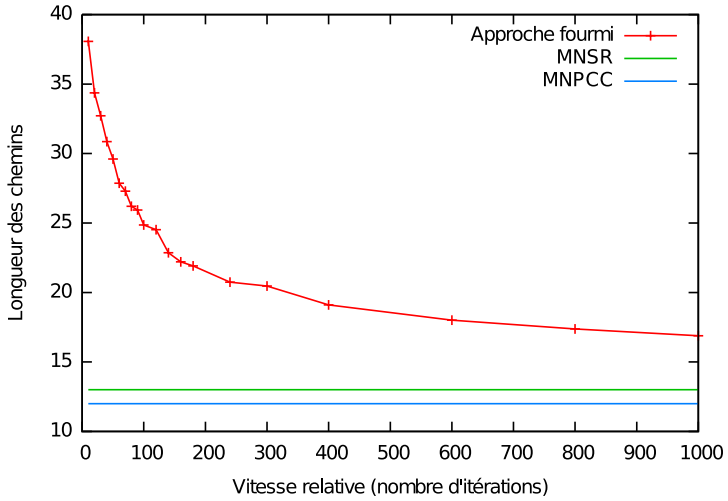


FIGURE 7.13: Longueur moyenne des chemins construits par les fourmis en fonction de la vitesse relative de l'algorithme. Le scénario « volatilité » est utilisé, ρ vaut 0.8 et β vaut 0. Par rapport aux simulations précédentes, les autres paramètres restent inchangés. Les deux droites montrent la valeur moyenne du plus court chemin dans les structures construites par les algorithmes EMPCC et EMSR pour ces mêmes simulations.

CHAPITRE 8

FORÊT COUVRANTE DANS UN MANET

Dans ce chapitre nous présentons un algorithme décentralisé qui construit et maintient une forêt couvrante dans un graphe dynamique. Notre contribution à ce problème est double. Nous présentons d'une part, une amélioration signigficative, en terme de performance, d'un algorithme décentralisé à jeton basé sur le modèle *DA-GRS* proposé A. CASTEIGTS et S. CHAUMETTE [CC06, Cas06, Cas07] pour la construction et le maintien d'un arbre couvrant dans les *MANETs*. D'autre part, nous proposons une analyse probabiliste de l'algorithme à jeton initial et de l'algorithme amélioré. Après avoir discuté des motivations de ce travail, nous décrivons en détail l'algorithme, puis les différentes simulations sont présentées ainsi qu'une proposition d'amélioration des performances. Enfin, la dernière section étudie les phénomènes observés en simulation à l'aide d'une analyse probabiliste.

8.1 Motivations

Une forêt couvrante dans un graphe est un ensemble d'arbres couvrants, tel que chaque sommet appartienne à un arbre et un seul. Dans un graphe dynamique la structure de la forêt évolue avec le graphe de façon à conserver la propriété de couverture. L'objectif à atteindre dans un contexte dynamique est souvent de minimiser le nombre d'arbres dans la forêt. Lorsque le graphe est connexe, l'objectif est atteint lorsqu'un seul arbre est obtenu. Dans le cas contraire il est atteint lorsqu'un seul arbre couvre chaque composante connexe. Dans le contexte des réseaux mobiles *ad hoc* la forêt est construite et maintenue par les stations.

Les forêts couvrantes, dans le contexte des *MANETs* servent de support à un ensemble de services réseaux tels que le routage, le *multicast*, la diffusion, ou la sécurité [PBGK08]. Pour le routage citons notamment le *Spanning Tree Protocol* (*RFC 1493*). Un ensemble d'autres applications peuvent tirer avantage de la pré-existence d'un arbre couvrant comme le souligne F. C. GÄRTNER [Gär03] dans un document de synthèse.

L'intérêt de l'algorithme à jeton utilisé ici est qu'il n'exige ni infrastructure ni identification unique des stations. Enfin, cet algorithme s'adapte à la dynamique du réseau. Il est auto-stabilisant, la dynamique ne fait qu'augmenter le nombre d'arbres au contraire de l'algorithme qui a pour but de le diminuer. En outre, l'algorithme permet de garantir l'existence d'une forêt couvrante à chaque instant dans le graphe.

Dans l'objectif d'une implémentation sur plate-forme réelle, nous avons décidé de simuler le comportement de cet algorithme. Nous nous sommes intéressés en particulier au temps nécessaire au jeton pour la construction d'un unique arbre par composante connexe. En cohérence avec la métrique de vitesse relative que nous avons défini chapitre 6, nous avons décidé de mesurer ce temps en nombre de mouvements de jetons dans le graphe.

8.2 Algorithme initial

Définissons pour commencer le graphe dans lequel s'applique l'algorithme. Il représente un réseau mobile *ad hoc* de stations qui communiquent. Dans ce graphe $G = (V, E)$, l'ensemble V des sommets représente les stations et l'ensemble E des arêtes représente les liens de communication. Une arête (a, b) est considérée comme un lien de communication bidirectionnel si a et b peuvent échanger des informations. Le graphe est donc non orienté. Dans un contexte de mobilité des stations, le graphe qui modélise ce réseau est dynamique. Son évolution est considérée comme discrète et peut être ramenée à une succession d'étapes autour desquelles le graphe subit des modifications structurelles (apparition, disparition). L'algorithme s'exécute dans ce contexte dynamique et les événements ne sont pas connus à l'avance. Dans le travail de A. CASTEIGTS, est introduite la notion de brin d'arête. Un brin d'arête est un couple (v, e) tel que $v \in V$, $e \in E$ et v est une extrémité de e. A tout instant dans le graphe, les sommets et

les arêtes sont étiquetés. Un sommet possède donc une étiquette qui lui est propre mais également une étiquette pour chacune de ses arêtes. Une arête est donc étiquetée dans chacune de ses extrémités.

L'algorithme est basé sur l'idée de jetons disposant de trois opérations : « circulation », « fusion » et « régénération ». Les états possibles pour un sommet sont J si le sommet courant est la racine de son arbre, et N dans le cas contraire. Pour les brins d'arêtes l'étiquette 0 signifie que l'arête correspondante n'est pas dans un arbre couvrant. Lorsque l'étiquette d'un brin d'arête (v,e) est à 1, cela signifie que l'arête e appartient à un arbre couvrant et que le sommet v est des deux extrémités de e le plus éloigné de la racine de l'arbre. L'étiquette 2 marque l'extrémité, de l'arête appartenant à l'arbre couvrant, la plus proche de la racine de l'arbre. Initialement tous les sommets sont étiquetés avec J et les arêtes avec 0, autrement dit, tous les sommets sont les racines de leur propre arbre de cardinal 1. L'algorithme peut être formulé sous forme de règles de ré-étiquetage au nombre de quatre comme le montre la figure 8.1.

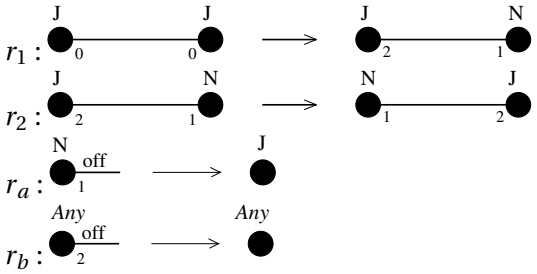


FIGURE 8.1: Formulation de l'algorithme de forêt couvrante sous forme de règles de ré-étiquetage.

Quand deux sommets étiquetés J se rencontrent, grâce à la règle r_1, ces deux racines d'arbres fusionnent pour ne former qu'un seul arbre. Une extrémité de l'arête en question est étiquetée 2 et son sommet garde l'étiquette J, il devient la racine des deux arbres. L'autre extrémité est étiquetée 1 et son sommet est marqué N. Cette règle assure l'unicité du jeton dans un arbre.

La règle r_2 code pour la circulation du jeton dans le réseau. Dans cette règle, les ré-étiquetages font que le jeton passe itérativement d'un sommet à l'autre.

Quand un lien appartenant à un arbre disparaît, l'arbre se trouve alors coupé en deux. Pour la partie de l'arbre qui contient le jeton (repéré par l'extrémité 2), aucun problème n'apparaît, la règle r_b est appliquée. Pour l'autre partie de l'arbre en revanche, il n'y a plus de jeton, il faut donc en créer un nouveau. C'est le sommet en rapport avec l'extrémité étiquetée 1 qui devient la racine du nouvel arbre et est étiqueté J. Ce mécanisme est maintenu par la règle r_a.

Les deux dernières règles r_a et r_b sont appliquées en réponse aux changements topologiques du graphe. Ainsi quand une arête est supprimée, l'étiquetage de ses extrémi-

tés dans les deux sommets concernés est mis à jour.

L'algorithme traite également des problématiques de synchronisation entre les stations. En effet, la manière la plus efficace de diffuser de l'information dans ces réseaux est d'utiliser la diffusion car elle se fait sans nécessiter de processus de synchronisation. Chaque station diffuse périodiquement son statut et écoute son voisinage pour connaître l'état de ses voisins. Dans certains cas, la communication doit être synchrone et deux stations doivent s'accorder pour communiquer. C'est le cas pour le mécanisme de déplacement du jeton. La station d'où part le jeton doit s'assurer qu'il est bien arrivé. C'est également le cas pour la fusion de deux arbres, quand deux stations avec jetons se rencontrent. Ce mode de communication est beaucoup plus coûteux en temps et en énergie mais n'est nécessaire que dans les deux cas précédents (mouvement et fusion de jeton). Les autres formes de communication peuvent se faire en mode asynchrone, c'est-à-dire en diffusion. En conclusion, seules les stations pourvues de jetons son susceptibles d'initier une communication synchrone.

8.3 Simulations et améliorations

L'algorithme original est décrit comme un systèmes à base de règles. Pour les besoins de la simulation, nous adoptons un autre point de vue qui consiste à considérer un système à base d'entités. Les jetons possèdent leur propre comportement et évoluent dans le réseau en le modifiant. Les règles r_1 et r_2 de la figure 8.1 définissent le comportement itératif du jeton alors que les règles r_a et r_b définissent le comportement réactif des stations du réseau face à certains événements structurels.

La connaissance du voisinage est assurée localement par les stations elles-mêmes qui diffusent leur état et écoutent leur voisinage pour connaître l'état de leurs voisins. Ces états sont à disposition des jetons qui sont des agents se déplaçant de station en station. L'étiquetage des brins d'arêtes est conservé pour continuer à indiquer la direction de la racine. L'étiquetage des stations est rendu obsolète par l'utilisation des jetons. L'algorithme 10 décrit le comportement général d'un jeton.

L'approche orientée entité mobile utilisée pour présenter l'algorithme en fait une méthode distribuée capable de fonctionner dans un réseau. Des simulations de réseaux mobiles sont mises en place pour tester l'algorithme. Une fois encore c'est le simulateur *Madhoc* qui est utilisé pour simuler les réseaux. Comme pour le problème de plus court chemin, nous extrayons des graphes dynamiques de connexions à partir du simulateur, puis nous exécutons l'algorithme sur ces graphes dynamiques.

Intéressons nous maintenant au comportement de l'algorithme. La gestion de la dynamique du graphe nous permet de faire varier la vitesse relative (voir page 95) de l'algorithme par rapport à celle du graphe. Commençons par observer le comportement de l'algorithme sur un graphe statique puis sur un graphe dynamique.

Algorithme 10 : Comportement d'un jeton (algorithme de forêt couvrante)

```
1  S_courant;                    /* le sommet sur lequel se situe le jeton */
2  répéter
       /* ----Première partie--- */
3      si il y a un jeton sur un sommet S_voisin voisin de S_courant alors
4          Tentative de synchronisation avec S_voisin;
5          si la synchronisation réussit alors
6              Modification des étiquettes du lien (S_courant, S_voisin);
7              Disparition du jeton courant;      /* Il ne subsiste qu'un seul
                   jeton sur S_voisin. */
8          fin
9      fin
       /* ----Seconde partie--- */
10     Choix d'un sommet voisin S_voisin de S_courant appartenant à l'arbre couvrant;
11     Tentative de synchronisation avec S_voisin;
12     si la synchronisation réussit alors
13         Modification des étiquettes du lien (S_courant, S_voisin);
14         Migration du jeton vers s_voisin;
15         s_courant ← s_voisin;
16     fin
17 jusqu'à disparition du jeton ;
```

8.3.1 Construction d'un arbre couvrant dans un graphe statique

Dans le cadre de la simulation il est possible de faire varier la vitesse relative le l'algorithme par rapport à celle du graphe dynamique. Nous étudions le cas limite d'une vitesse relative non bornée, ce qui revient à considérer un graphe statique. Ce cas de figure peut s'appliquer pour des réseaux *ad hoc* de capteurs ou pour des réseaux mobiles à mobilité très réduite. Ce scénario devient irréaliste pour des modèles de mobilité généraux. Nous considérons ce cas de figure pour capturer le fonctionnement limite de la méthode. Définissons une itération de l'algorithme comme le fait que chaque jeton du graphe se déplace d'une station, on parle aussi de mouvement de jeton.

La figure 8.2 illustre la stabilisation de l'algorithme. Le graphe considéré est connexe. Dans la figure, le nombre de mouvements de jetons est compté entre chaque fusion de jeton. Ainsi au début de cette simulation il y avait 10 arbres dans la composante connexe. Lors des premières fusions il a fallu moins de 10 mouvements aux jetons pour se rencontrer. Quand il ne reste plus que 7 arbres, le nombre de mouvements de jetons nécessaires pour diminuer à nouveau le nombre d'arbres augmente de manière exponentielle (on notera l'échelle logarithmique en ordonnée). Dans cet exemple, il a fallu près de 60000 mouvements aux deux derniers jetons pour se rencontrer.

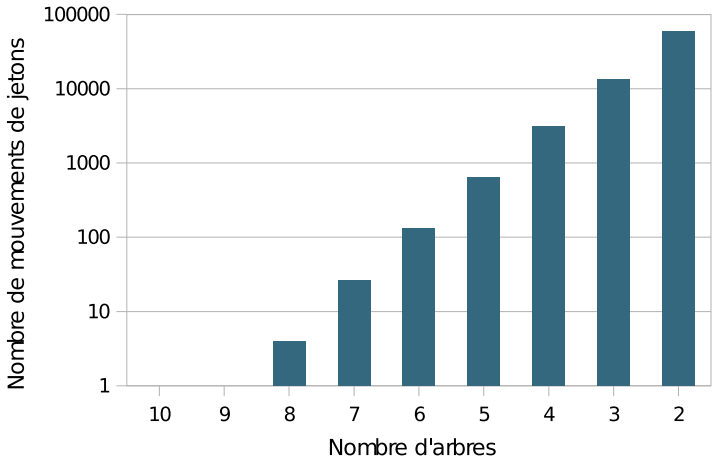


FIGURE 8.2: Étude de la stabilisation le l'algorithme de forêt couvrante. Le graphe extrait du simulateur *Madhoc* est un réseau mobile de 170 stations et d'environ 1500 liens. L'algorithme de forêt couvrante est exécuté jusqu'à stabilisation. L'algorithme commence avec 10 arbres dans une seule composante connexe. Le nombre d'arbres décroît rapidement (moins de 10 mouvements). À partir de 7 arbres le nombre de mouvements de jetons nécessaires pour faire diminuer le nombre d'arbres augmente de manière exponentielle. Notons que l'échelle de l'axe des ordonnées est logarithmique. Il a fallu près de 60000 mouvements au deux derniers jetons pour se rencontrer.

Les résultats de la simulation reportés dans la figure 8.2 mettent en évidence le problème de la stabilisation de l'algorithme. Il semble très efficace pour réduire rapidement le nombre d'arbres quand ceux-ci sont nombreux mais également petits. Son efficacité semble s'effondrer quand la taille des arbres augmente. En effet, le nombre de stations dans la simulation est constant. Il y a autant de stations au début de la simulation avec 10 arbres qu'à la fin avec 2 arbres. Le nombre d'itérations nécessaires à la stabilisation de l'algorithme semble donc être directement lié à la taille des arbres. Cette hypothèse sera confirmée par l'analyse probabiliste.

8.3.2 Comportement de l'algorithme dans un contexte dynamique

Considérons maintenant le problème dans un véritable contexte dynamique où le temps dont dispose l'algorithme ne lui suffit pas à se stabiliser. Nous reprenons les graphes dynamiques de connexions de *Madhoc* et contrôlons le nombre d'itérations possibles pour l'algorithme, c'est-à-dire que nous considérons une vitesse relative limitée.

La figure 8.3 présente la variation du ratio entre le cardinal de la forêt couvrante et le nombre de composantes connexes en fonction de la vitesse relative. Étant donné que le nombre de composantes connexes n'est pas maîtrisé, on s'attache d'avantage au

ratio entre nombre d'arbres et nombre de composantes connexes qu'au nombre absolu d'arbres produits.

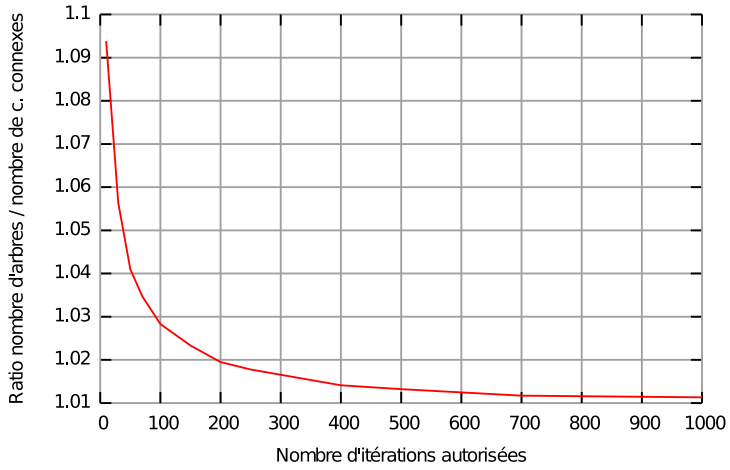


FIGURE 8.3: Ratio entre le nombre d'arbres que l'algorithme parvient à maintenir quand le nombre d'itérations est contraint par rapport au nombre de composantes connexes dans le graphe. Dans la simulation le nombre moyen de composantes connexes est supérieur à 1. On s'intéresse donc au ratio entre nombre moyen d'arbres et nombre moyen de composantes connexes. La même simulation est exécutée pour chaque valeur du « nombre d'itérations autorisé ». Le réseau est composée de 200 stations d'une portée de 50 mètres se déplaçant selon le modèle *Random Waypoint* dans un espace d'un kilomètre carré. Il y a 100 étapes de simulation.

8.3.3 Modification de l'algorithme

Les résultats observés pour l'algorithme original nous ont incité à rechercher des optimisations afin d'améliorer ses performances. L'algorithme original prévoit à chaque itération un mouvement purement aléatoire du jeton, le choix d'un prochain sommet se fait donc au hasard parmi les voisins faisant partie de l'arbre. Nous avons choisi de modifier le fonctionnement de l'algorithme en intervenant sur la mobilité du jeton. Cette mobilité peut être modifiée de deux manières simples. Une première manière consiste à autoriser le jeton à temporiser sur les stations, une autre manière, à lui interdire, lorsque cela est possible les « allers-retours ».

Les simulations du scénario dans lequel la temporisation est autorisée montrent que les résultats se dégradent de façon notable. Nous avons donc abandonné cette option au profit de la suivante.

Nous proposons d'utiliser un mécanisme inspiré d'une heuristique bien connue dans les problématiques d'optimisation qu'est la recherche tabou [GL93]. L'idée est que le je-

ton se souvient des dernières stations qu'il a visité de façon à les éviter dans ses choix futurs. Il construit une liste de taille fixée de façon à ne pas surcharger la structure de données qui transite sur les sommets. Le but de ce mécanisme est de forcer les jetons à préférer les stations n'ayant pas été visitées depuis un certain temps et donc de le forcer à visiter un maximum de stations différentes. Une station ne peut être insérée deux fois dans la liste tabou, elle peut en revanche être réordonnée.

La figure 8.4 montre l'impact de l'utilisation de listes tabous sur le nombre de mouvements nécessaires à la stabilisation de l'algorithme. Une liste tabou de taille 0 correspond à l'algorithme original de forêt couvrante. Nous observons que le temps de stabilisation de l'algorithme avec l'utilisation d'une liste tabou, est divisé par une valeur proche de 3 et ce dès l'utilisation d'une liste tabou de taille 1. En revanche l'allongement de la taille de la liste n'apporte pas d'amélioration significative.

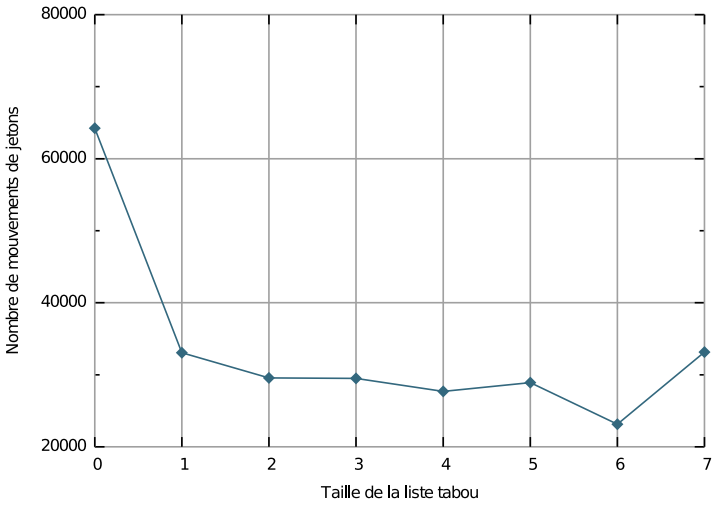


FIGURE 8.4: Impact de la taille de la liste tabou sur le nombre d'itérations. Le nombre d'itérations compté est celui nécessaire à l'algorithme de forêt couvrante pour se stabiliser (obtenir un arbre par composante connexe) dans la simulation de la figure 8.2. Notons qu'une taille de liste égale à 0 correspond au comportement de l'algorithme original.

Observons maintenant les performances de l'algorithme lorsque la vitesse de l'algorithme est limitée. Les performances de l'algorithme original et de la version tabou sont représentées sur la figure 8.5. On peut constater que l'algorithme tabou est meilleur indépendamment de la valeur de la vitesse relative. Il peut donc remplacer avantageusement l'algorithme original y compris pour des réseaux fortement dynamiques.

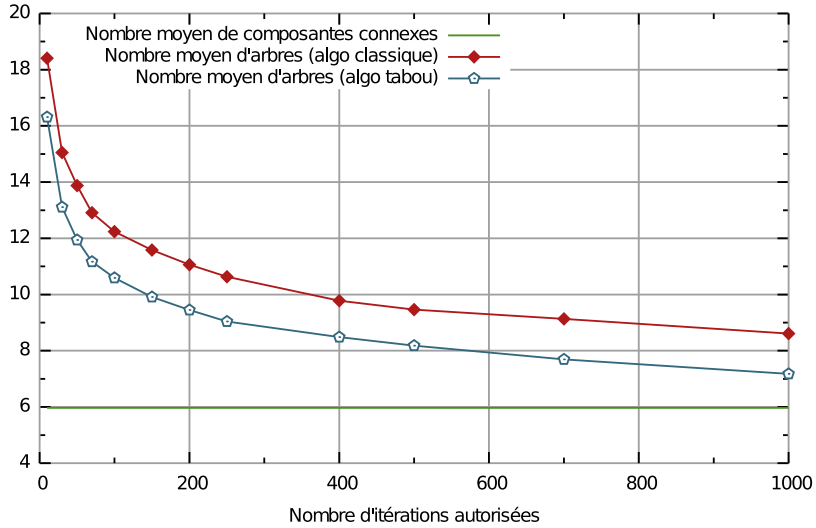


FIGURE 8.5: Comparaison entre l'approche originale et l'approche tabou. Le même graphe dynamique est utilisé. La simulation est répétée plusieurs fois en faisant varier la vitesse relative de l'algorithme. Le réseaux composé de 200 stations avec un rayon de couverture de 100 mètres dans un espace ouvert de un kilomètre carré. Les stations se déplacent selon le modèle *Random Waypoint*. Les performances de l'algorithme tabou sont meilleures quelle que soit la valeur de la vitesse relative (nombre d'itérations autorisées).

8.3.4 Conclusion sur les simulations

Notre objectif dans ce chapitre est de produire et de maintenir une forêt couvrante dans un graphe dynamique tout en minimisant le nombre d'arbres de la forêt. L'algorithme original, fidèle au modèle *DAGRS*, se montre efficace lorsque les arbres sont de petite taille. En revanche ses performances se dégradent fortement avec l'augmentation de la taille des arbres.

Afin d'évaluer la qualité de l'algorithme, nous utilisons différentes métriques. Nous mesurons dans un premier temps le nombre de mouvements de jetons nécessaires pour faire diminuer le nombre d'arbres dans le graphe considéré. Ensuite nous nous intéressons au comportement de l'algorithme en en fonction de sa vitesse relative. Nous mesurons alors le ratio « nombre d'arbres » sur « nombre de composantes connexes ».

Les observation que nous pouvons faire sont les suivantes. Pour une composante connexe donnée, l'effort pour passer de n arbres à $n-1$ est maximal pour $n=2$. La seconde observation est que la vitesse relative de l'algorithme joue un rôle dans l'amélioration du ratio « nombre d'arbres » sur « nombre de composantes connexes ». Cependant

l'amélioration ne varie pas linéairement avec l'augmentation de la vitesse. Autrement dit, en terme de réseaux, si on supposait un gain en vitesse de transmission, l'obtention d'un unique arbre par composante connexe ne serait pas garantie. L'une des voies possibles d'amélioration consiste à concevoir de nouvelles stratégies de déplacement des jetons.

C'est ce que nous avons réalisé en contraignant le déplacement des jetons à l'aide d'une liste tabou. Les résultats de simulation semblent indiquer une amélioration à la fois dans le ratio « nombre d'arbres » sur « nombre de composantes connexes » et dans le nombre de mouvements nécessaires à l'algorithme. Dans la section suivante nous effectuons une analyse probabiliste dans l'objectif d'expliquer partiellement ces observations.

8.4 Analyses

Dans les simulation précédentes nous avons remarqué que l'effort pour passer de n arbres à $n-1$ est maximal pour $n = 2$. En conséquence les analyses présentées dans cette section ne considèrent que le cas de la fusion de deux arbres en prenant l'hypothèse que le graphe est connexe.

8.4.1 Analyse de l'algorithme opérant sur un graphe statique

Selon l'algorithme chaque jeton se déplace à l'intérieur de son arbre en empruntant seulement les arêtes qui appartiennent à cet arbre. Le graphe étant connexe, il existe dans ce graphe des arêtes dont les deux extrémités appartiennent à deux arbres différents. Lorsque deux jetons sont positionnés sur des sommets adjacents, les deux arbres fusionnent. Dans l'algorithme les jetons se déplacent de manière aléatoire. Ainsi, la position suivante de chaque jeton ne dépend que de sa position courante. En conséquence le processus de rencontre des jetons est un processus Markovien. Les matrices stochastiques sont obtenues à partir des matrices d'adjacence des arbres.

Considérons deux arbres A_1 et A_2 de cardinalité respective n_1 et n_2 dans un graphe liés par un ensemble L de k liens. Les deux jetons de ces arbres se rencontreront s'ils sont présents simultanément sur les extrémités de l'un de ces liens. La figure 8.6 illustre notre propos.

Nous nous intéressons donc ici à la « probabilité de rencontre ». Cette probabilité peut être exprimée en fonction de la « probabilité de présence » d'un jeton sur un sommet. En effet cette probabilité est égale à la somme des produits des probabilités de présence des jetons sur les extrémités d'un même lien appartenant à L, ce qu'exprime l'équation suivante :

$$P_{rencontre}(j_1, j_2) = \sum_{e=(v_1,v_2)\in L} P_{présence}(j_1, v_1) \times P_{présence}(j_2, v_2) \tag{8.1}$$

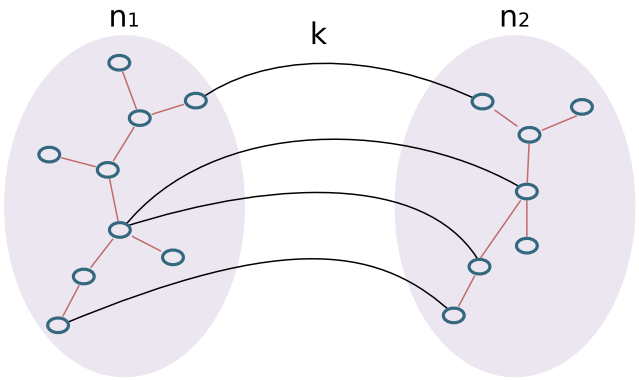


FIGURE 8.6: Schéma d'un graphe couvert par deux arbres A_1 et A_2 partageant k liens (ici $k = 4$).

$P_{présence}(j_1, v_1)$ est la probabilité de présence du jeton de l'arbre A_1 sur l'extrémité v_1 ($v_1 \in A_1$) du lien e ($e \in L$). Cette probabilité de présence d'un jeton sur un sommet particulier de l'arbre dépend du déplacement du jeton dans l'arbre. En effet, la position d'un jeton à la date t dépend entièrement de sa position à la date $t-1$.

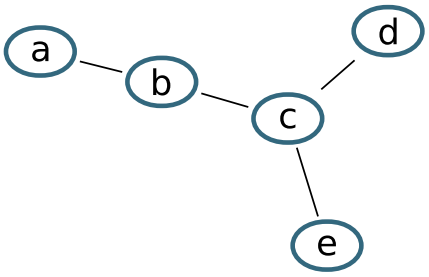
Probabilité de présence

À partir de la matrice d'adjacence des sommets de l'arbre considéré, il est possible de définir la matrice des « probabilités de passage » M_p de dimension $n \times n$ (avec n le nombre de sommets de l'arbre). À chaque élément (i, j) de la matrice est associée la probabilité pour un jeton situé sur le sommet i de se rendre sur le sommet j. La somme des valeurs de chaque ligne de la matrice vaut 1. Définissons maintenant le vecteur de « probabilité de présence » initial v_0 qui définit la position initiale du jeton. Le produit ordinaire matriciel $v_0 \times M_p$ donne le vecteur v_1 de probabilité de présence du jeton après un mouvement. De manière itérative il est possible de définir la probabilité de présence du jeton en fonction du vecteur initial v_0 au bout de m mouvements. Ainsi le vecteur de probabilité de présence v_m au bout de m mouvements vaut :

$$v_m = v_{m-1} \times M_p \tag{8.2}$$

Considérons l'exemple de l'arbre A dans la figure 8.7. La matrice de probabilités de passage de A est M_p. La première colonne correspond au sommet a, la seconde à b, *etc.*, de même pour les lignes.

Si par exemple le jeton est initialement sur le sommet a, le vecteur de probabilité de présence initial v_0 vaut $(1,0,0,0,0)$. Les vecteurs de probabilités de présences aux étapes



$M_p =$

	a	b	c	d	e
a	0	1	0	0	0
b	$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
c	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
d	0	0	1	0	0
e	0	0	1	0	0

FIGURE 8.7: Un arbre A et sa matrice de « probabilité de passage » M_p.

suivantes vaudront alors :

$$\begin{aligned}
v_0 &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
v_1 &= \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} \\
v_2 &= \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \end{pmatrix} \\
v_3 &= \begin{pmatrix} 0 & \frac{2}{3} & 0 & \frac{1}{6} & \frac{1}{6} \end{pmatrix} \\
v_4 &= \begin{pmatrix} \frac{1}{3} & 0 & \frac{2}{3} & 0 & 0 \end{pmatrix} \\
v_5 &= \begin{pmatrix} 0 & \frac{5}{9} & 0 & \frac{2}{9} & \frac{2}{9} \end{pmatrix}
\end{aligned}$$

Cette technique permet de donner pour n'importe quel arbre la probabilité de présence d'un jeton en chacun de ses sommets. À partir des vecteurs calculés précédemment, un phénomène propre au mode de déplacement dans un arbre est observé. En effet, quand un jeton se trouve sur un sommet donné de l'arbre, la probabilité à l'étape suivante qu'il se trouve sur ce même sommet est nulle car il n'est pas autorisé à attendre sur ce sommet. La probabilité de présence de ce jeton sur les sommets voisins est en revanche non nulle. Il faut attendre deux itérations pour que la probabilité de présence du jeton sur le sommet de départ redevienne non nulle. Après plusieurs itérations ce phénomène que l'on nomme « respiration » continue d'être observé et explique les valeurs obtenues. Pour donner une idée plus juste de la probabilité de présence, les valeurs de probabilités de présence sont moyennées d'une étape à l'autre.

En observant l'évolution moyennée des probabilités de présence on observe une convergence des valeurs sur chaque sommet. Nous observons que ces valeurs limites pour chaque sommet tendent vers le ratio entre le degré du sommet et la somme des degrés de tous les sommets de l'arbre.

Proposition. La probabilité de présence d'un jeton J qui se déplace selon une marche aléatoire dans un arbre $T = (S, A)$, est, sur un sommet donné v :

$$P(J, v) = \frac{degré(v)}{\sum_{v' \in S} degré(v')}$$

Probabilité de rencontre

Après avoir calculé la probabilité de présence, la probabilité de rencontre de deux jetons peut être à son tour calculée. Nous observons l'évolution de cette probabilité en fonction de plusieurs grandeurs. Ainsi la figure 8.8 illustre les résultats obtenus pour un ensemble de paires d'arbres quelconques mais identiques. Les différentes courbes correspondent à des tailles différentes. On constate que la probabilité de rencontre s'effondre lorsque la taille des arbres augmente. Ce résultat rejoint les observations faites en simulation.

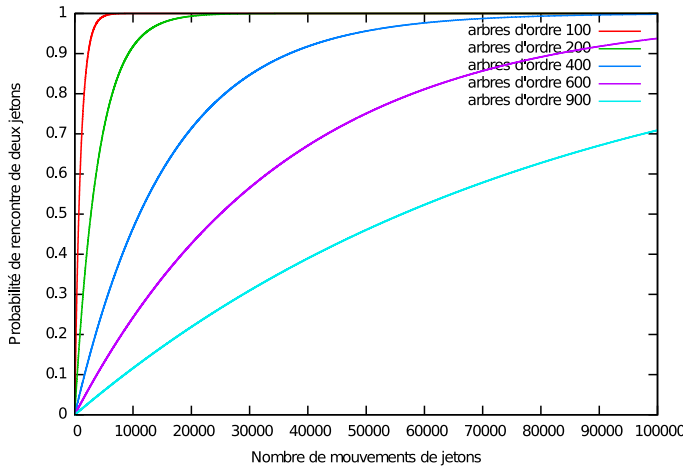


FIGURE 8.8: Évolution de la probabilité de rencontre de deux jetons en fonction de la taille des arbres et du nombre de mouvements des jetons.

La taille des arbres n'est pas le seul paramètre qui influe sur la probabilité de rencontre, le nombre de liens entre les deux arbres est également important. La figure 8.9 montre l'évolution de la probabilité de rencontre entre deux jetons quand leurs arbres sont de taille fixe et que le nombre de liens qu'ils partagent change. On observe une nette variation de la probabilité de rencontre qui augmente avec le nombre de liens inter-arbres. Cette augmentation n'est pas linéaire.

8.4.2 Analyse de l'algorithme à liste tabou

La motivation première pour l'introduction d'une liste tabou dans l'algorithme était d'intervenir sur la manière dont le jeton se déplace dans l'arbre. Intuitivement nous pensons que l'introduction de cette liste améliore la mobilité du jeton dans l'arbre, et donc augmente la probabilité de rencontre des deux jetons. Pour commencer nous montrons que l'amélioration de la mobilité du jeton augmente la probabilité de rencontre. Nous nous positionnons dans le cas limite où le jeton se déplace dans un graphe complet.

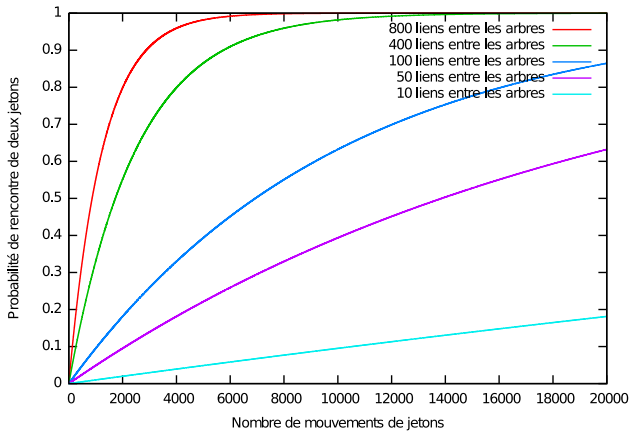


FIGURE 8.9: Évolution de la probabilité de rencontre en fonction du nombre de liens inter-arbres. Les deux arbres sont quelconques d'ordre 1000. Le nombre de liens entre les deux arbres varie de 10 à 800.

Nous proposons de calculer la probabilité de rencontre de deux jetons appartenant à deux sous-graphes complets de taille respective n_1 et n_2 liés par k arêtes dont les deux extrémités appartiennent chacune à un sous-graphe complet différent. La probabilité de rencontre des jetons est donc égale à $P(rencontre) = \frac{k}{n_1 \times n_2}$. La probabilité que les jetons ne se rencontrent pas est donc $1 - P(rencontre)$. Après p mouvements la probabilité de rencontre des deux jetons est égale à :

$$1 - \left(1 - \frac{k}{n_1 \times n_2}\right)^p$$

Les résultats présentés dans la figure 8.10 confirment que l'amélioration de la mobilité des jetons augmente leur probabilité de rencontre.

L'ajout d'une liste tabou dans l'algorithme introduit un mécanisme de mémoire dans la marche aléatoire des jetons dans les arbres. Le processus n'est donc plus Markovien, il n'est plus possible de produire des matrices stochastiques directement à partir des matrices d'adjacence. Bien que nous n'ayons pas prouvé mathématiquement que l'introduction d'une liste tabou permettait d'améliorer les performances de l'algorithme, les simulations semblent indiquer que l'approche tabou améliore la mobilité du jeton. Comme nous l'avons montré par ailleurs (voir figure 8.10) l'amélioration de la mobilité du jeton améliore les résultats de l'algorithme.

8.5 Conclusion

Dans ce chapitre nous avons étudié le comportement d'un algorithme de construction et de maintien d'une forêt couvrante dans un réseau mobile *ad hoc*. Les différentes

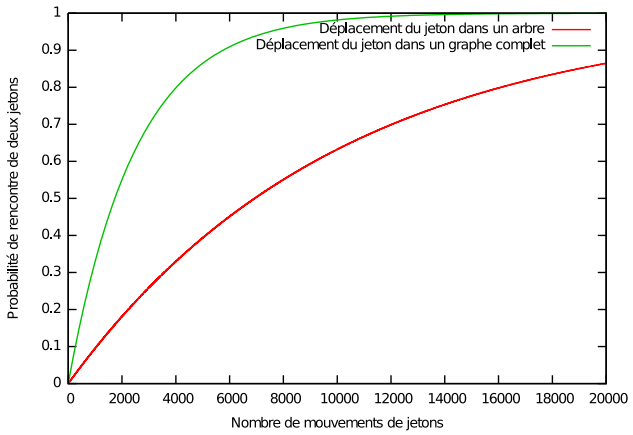


FIGURE 8.10: Comparaison de l'évolution de la probabilité de rencontre des jetons lorsque leur déplacement est contraint par une topologie d'arbre et lorsque leur déplacement est non contraint (graphe complet).

simulations nous ont permis de comprendre le fonctionnement de l'algorithme et nous ont poussé à proposer une amélioration du fonctionnement divisant par trois en moyenne le nombre de mouvements de jetons pour atteindre la stabilisation de l'algorithme. Cette amélioration repose sur l'introduction d'une modification de la politique de déplacement du jeton. Une analyse probabiliste nous a permis de conforter les observations faites en simulation à propos du comportement de l'algorithme original. Nous suivons actuellement une piste prometteuse pour valider mathématiquement l'efficacité de l'amélioration proposée.

CHAPITRE 9

CONCLUSION

Les contributions de cette thèse relèvent du domaine des graphes, de l'algorithmique distribuée, des métaheuristiques et des réseaux mobiles *ad hoc*. En effet, la contrainte présente dans tous les problèmes considérés est celle d'environnements dynamiques et distribués et le soucis de modélisation et de résolution décentralisée est omniprésent dans ce travail.

Le cadre applicatif et les principaux résultats appartiennent aux réseaux mobiles *ad hoc* et nous espérons avoir apporté une contribution utile et pertinente pour les futures investigations dans le domaine.

Plus en amont du cadre applicatif, la source des questionnements qui a conduit à ce travail prend corps dans l'étude et la modélisation des systèmes complexes. Les réseaux d'interactions et l'intelligence collective en constituent les points de départ. Notre espoir est que le formalisme proposé pour les graphes dynamiques et les méthodes d'optimisation distribuées soient utiles pour l'étude des *MANETs* mais trouvent également des applications dans d'autres contextes.

9.1 Autour des graphes dynamiques

L'étude des différents modèles abordant la notion de graphes dynamiques (chapitre 2) nous a mené à deux constats. Le premier est que les modèles étudiés sont formalisés dans un contexte applicatif particulier et qu'il est parfois difficile de les formuler hors de ce contexte sans les dénaturer. Les auteurs de ces modèles ne formulent pas toujours explicitement leur modèle de graphe. Celui-ci est parfois implicite dans le problème considéré ou dans le processus de résolution proposé. La seconde est que même si tous

les modèles étudiés contiennent des fonctionnalités importantes et intéressantes, aucun d'eux n'est assez général pour inclure tous les autres. Même si le modèle de graphe évolutif de A. FERREIRA et ses collègues permet d'inclure, lors d'une analyse différée, la plupart des évènements de modification des graphes des autres modèles. il ne permet pas de considérer le processus qui est à l'origine de la création des évènements, comme le font les modèles de génération de graphes aléatoires, tels que, par exemple, le modèle d'attachement préférentiel de A. L. BARABÁSI et R. ALBERT. La notion d'« équation dynamique » de H. BERSINI nous a également semblé très pertinente, mais le manque de formalisme du modèle nous a limité dans son utilisation.

Cette étude nous a poussé à proposer dans le chapitre 3 un modèle de graphe dynamique incluant toute forme de dynamique possible, en incluant la notion de « processus d'évolution », qui, à l'aide d'une base temporelle choisie, permet de définir, dans le modèle de graphe, le comportement dynamique de celui-ci. Nous proposons également dans ce modèle des métriques adaptées.

9.2 Autour des méthodes d'intelligence collectives

Le chapitre 4 nous a permis d'effectuer un tour d'horizon des différentes implémentations autour de la métaphore des fourmis. Après nos investigations, notre constat fut que même si l'intelligence collective est une notion très utilisée dans le contexte de l'optimisation, il n'existe pas, à notre connaissance, de modèle général, dédié aux environnements dynamiques et distribués.

Le chapitre 5 fut l'occasion de définir un modèle général de résolution de problèmes modélisés par des graphes dynamiques.

9.3 Autour des problématiques de réseaux mobiles *ad hoc*

La dernière partie de cette thèse fut consacrée aux réseaux mobiles *ad hoc* pour deux raisons. Les réseaux mobiles *ad hoc* sont naturellement modélisables à l'aide de graphes dynamiques et pourraient justifier à eux seuls la formalisation du modèle de graphe dynamique du chapitre 3. Ensuite les *MANETs* présentent des problèmes variés dans un contexte dynamique et décentralisé. Ils nous donnent ainsi la possibilité d'expérimenter différents algorithmes adaptés au contexte.

Le chapitre 6 a proposé de présenter les *MANETs* et de faire le lien avec les graphe dynamiques et en particulier avec notre définition.

Le chapitre 7 a proposé d'utiliser l'approche générale d'optimisation de problèmes en environnement dynamique explicité dans le chapitre 5, pour mettre en œvre un algorithme de construction et de maintien de chemins dans un *MANETs*, en considérant deux objectifs.

Enfin le chapitre 8 a présenté l'analyse faite autour d'un algorithme de construction et de maintien d'une forêt couvrante, dans un *MANET*.

9.4 Perspectives de ce travail

Les perspectives de ce travail sont multiples tant au niveau théorique des graphes dynamiques qu'au niveau applicatif des problématiques liées aux *MANETs*. La multiplicité des perspectives tient aussi au fait que ce travail est une déclinaison du verrou scientifique de l'équipe RI_2C. De ce fait, plusieurs travaux de thèse nouvellement engagés sont en étroit rapport avec ce travail.

Les perspectives à court terme sont les suivantes. Nous souhaitons achever le travail engagé avec l'analyse de l'algorithme « à liste tabou » de forêt couvrante du chapitre 8. Nous possédons une bonne intuition et un début de preuve intéressante.

Concernant l'algorithme fourmis de construction et de maintien de chemins courts et robustes dans un *MANETs*, notre intuition est qu'il est adaptable à un réseau mobile *ad hoc* tolérant les délais. Nos prochains travaux porteront donc sur la simulation d'une adaptation de l'algorithme dans un graphe peu dense, où il n'existe pas de chemin de bout en bout entre une source et une destination. Plus tard il sera envisageable d'implémenter cet algorithme sur un véritable réseau mobile *ad hoc* de machines.

Une bibliothèque logicielle de graphes dynamiques *GraphStream*, développée durant cette thèse est toujours en évolution. L'augmentation des fonctionnalités offertes, notamment en terme de génération de graphes et de métriques, reste une priorité.

À une échelle de temps plus grande, d'autres perspectives issues de ce travail sont envisagées. Suite à la formulation des graphes dynamiques et à leur lien avec les *MANETs*, nous avons entrepris d'analyser plus en détail les graphes dynamiques issus de simulations d'entités spatialisées utilisant un modèle de déplacement défini. Ce travail engagé avec G. C. ABOUE-NZE se focalise sur l'évolution de métriques dédiées à la dynamique des graphes observés. De l'observation de différents modèles de mobilité et différentes vitesses de déplacement, nous espérons déduire différentes familles de graphes dynamiques respectant des propriétés données. L'observation porte sur des métriques ayant un intérêt dans un contexte dynamique comme l'évolution des distributions des degrés, le nombre de composantes connexes, le diamètre ou encore le nombre total de rencontres.

Nous souhaitons également entreprendre une analyse comparative des graphes évolutifs obtenus à partir de graphes formulés avec notre modèle. Ce projet, encore à un stade embryonnaire, aurait pour objectif de découvrir, s'il existe, le lien entre le « processus d'évolution » d'un graphe dynamique, tel que nous l'avons formulé, et une classe de graphes évolutifs.

Enfin, dans l'équipe RI_2C, l'implémentation de véritables réseaux mobiles *ad hoc* est assurée par J. FRANZOLINI. Les expérimentations qu'il mène à bien durant sa thèse, ouvrent de nouvelles questions. Nous cherchons en particulier à déterminer si les algo-

rithmes que nous développons en simulation (comme ceux présentés dans cette thèse) peuvent s'appliquer à de véritables *MANETs*. Ensuite nous espérons pouvoir évaluer concrètement l'impact de contraintes techniques telles que les durées de synchronisation ou les effets de « stations cachées ».

BIBLIOGRAPHIE

[AB06] A. Alaoui and C. Bertelle, editors. *Emergent Properties in Natural and Artificial Dynamical Systems*. Understanding Complex Systems. Springer, Berlin/Heidelberg, 2006.

[ACKN04] P. Amar, J.-P. Cornet, F. Képès, and V. Norris, editors. *Proceedings of the Evry Spring School on "Modelling and Simulation of Biological Processes in the Context of Genomics"*, 2004.

[Ada99] Lada A. Adamic. The small world web. In S. Abiteboul and A. M. Vercoustre, editors, *Proc. 3rd European Conf. Research and Advanced Technology for Digital Libraries, ECDL*, number 1696, pages 443–452. Springer-Verlag, 1999.

[BA99] Albert-László Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286 :509, 1999.

[BABM05] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11 b mesh network. *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 31–42, 2005.

[Bak96] P. Bak. *How Nature works : the science of self-organized criticality*. Springer-Verlag, 1996.

[BDGO07] Cyrille Bertelle, Antoine Dutot, Frédéric Guinand, and Damien Olivier. Organization detection for dynamic load balancing in individual-based simulations. *Multi-Agent and Grid Systems*, 3(1) :42, 2007.

[BDT99] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence : from natural to artificial systems*. Oxford University Press, 1999.

[Ber02] H. Bersini. And the winner is...not the fittest [coevolution]. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1510–1515, 12-17 May 2002.

[Ber05] Hugues Bersini. *des réseaux et des sciences - Biologie, informatique, sociologie : l'omniprésence des réseaux*. vuibert informatique. Vuibert, 2005.

[Bet01a] C. Bettstetter. Smooth is better than sharp : a random mobility model for simulation of wireless networks. *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 19–27, 2001.

[Bet01b] Christian Bettstetter. Mobility modeling in wireless networks : Categorisation, smooth movment, and border effects. *ACM Mobile Computing and Communications Review*, 5(3) :11, 2001.

[BG03] Jacob Beal and Seth Gilbert. Rambonodes for the metropolitan ad hoc network. Technical Report MIT Technical Report MIT-AIM-2003-27, MIT, dec 2003.

[BH04] F. Bai and A. Helmy. *Wireless Ad Hoc and Sensor Networks*, chapter A Survey of Mobility Modeling and Analysis in Wireless Adhoc Networks. Kluwer Academic Publishers, 2004.

[BHS99] B. Bullnheimer, R.F. Hartl, and C. Strauss. A new rank-based version of the ant system : a computational study. *Central European Journal for Operations Research and Economics*, 7(1) :25–38, 1999.

[BKM+00] Andrei Z. Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet L. Wiener. Graph structure in the web. *Computer Networks*, 33(1-6) :309–320, 2000.

[BLM+06] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks : Structure and dynamics. *Physics Reports*, 424 :175–308, 2006.

[BMJ+98] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, 1998.

[BRS03] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Trans. Mobile Computing*, 2(3) :257–269, July–September 2003.

[BS03] Benjamín Barán and Matilde Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In *Proceedings of the 21st IASTED International Conference*, Innsbruck, Austria, February 2003.

[BXFJ03] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2) :267–285, April 2003.

[Cas06] Arnaud Casteigts. Model driven capabilities of the da-grs model. In *ICAS '06 : Proceedings of the International Conference on Autonomic and Autonomous Systems*, page 24, Washington, DC, USA, 2006. IEEE Computer Society.

[Cas07] Arnaud Casteigts. *Contribution à l'algorithmique distribuée dans les réseaux mobiles ad hoc - Calculs locaux et réétiquetages de graphes dynamiques*. PhD thesis, LaBRI - université de Bordeaux 1, 2007.

[CC06] A. Casteigts and S. Chaumette. Dynamicity aware graph relabeling systems and the constraint based synchronization : a unifying approach to deal with dynamic networks. In *International Conference on Wireless Algorithms, Systems and Applications (WASA'06)*, volume 4138, pages 688–697, Xi'An, China, 2006. LNCS.

[CCL03] I. Chlamtac, M. Conti, and J. J.-N. Liu. Mobile ad hoc networking : imperatives and challenges. *Ad Hoc Networks*, 1 :13–64, 2003.

[CDGO06] Alain Cardon, Antoine Dutot, Frédéric Guinand, and Damien Olivier. *Emergent Properties in Natural and Artificial Dynamical Systems*, chapter Competing Ants for Organization Detection : application to Dynamic Distribution, pages 25–52. Springer complexity : Understanding Complex Systems. Springer Verlag, 2006.

[CdVHM00] O. Cordon, I.F. de Viana, F. Herrera, and L. Moreno. A new aco model integrating evolutionary computation concepts : the best-worst ant system. In Marco Dorigo, Luca Maria Gambardella, Martin Middendorf, and Thomas Stützle, editors, *Abstract proceedings of ANTS 2000 – From Ant Colonies to Artificial Ants : Second International Workshops on Ant Algorithms*, pages 22–29. IEEE Transaction on Evolutionary Computation, 2000.

[CGMT03] Marco Conti, Silvia Giordano, Gaia Maselli, and Giovanni Turi. Mobileman : Mobile metropolitan ad hoc networks. In *Proc. Eight International IFIP-TC6 Conference, Venice, Italy, Lecture Notes in Computer Science LNCS 2775, pp. 173-178*, pages 194–205, September 2003.

[CH97] D. Costa and A. Hertz. Ants can colour graphs. *The Journal of the Operational Research Society*, 48(3) :295–305, 1997.

[CJL+01] T. Clausen, P. Jacquet, A. Laouiti, et al. Optimized link state routing protocol. *IEEE INMIC Pakistan*, 2001, 2001.

[Com99] IEEE LAN/MAN Committee. Higher speed phy extension in the 2.4 ghz band, 1999.

[CPV03] Corina Cortes, Daryl Pregibon, and Chris Volinsky. Computational methods for dynamic graphs. *Journal of Computational & Graphical Statistics*,

12(4) :950–970(21), December 2003. ISSN 1061-8600, Online ISSN : 1537-2715.

[CRA] CRAWDAD. http://crawdad.cs.dartmouth.edu/.

[dBSD00] M. den Besten, T. Stutzle, and M. Dorigo. Ant colony optimization for the total weighted tardiness problem. *Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature*, 1917 :611–620, 2000.

[DC97] M. Dorigo and G. Di Caro. *New Ideas in Optimization. D. Corne and M. Dorigo and F. Glover eds*, chapter The Ant Colony Optimization Meta-Heuristic, pages 11–32. McGraw-Hill, 1997.

[DC07] Narsingh Deo and Aurel Cami. Preferential deletion in dynamic models of web-like networks. *Inf. Process. Lett.*, 102(4) :156–162, 2007.

[DCDG04] G. Di Caro, F. Ducatelle, and L. M. Gambardella. Anthocnet : an ant-based hybrid routing algorithm for mobile ad hoc networks. *Proceedings of Parallel Problem Solving from Nature (PPSN VIII)*, 3242 :461–470, 2004.

[DD98] Gianni DiCaro and Marco Dorigo. Antnet : Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9 :317–365, 1998.

[DG97a] M. Dorigo and L. M. Gambardella. Ant colony system : a cooperative learning approach to the travelingsalesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1) :53–66, 1997.

[DG97b] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2) :73–81, July 1997.

[DGH+01] K. Doerner, W. Gutjahr, R. Hartl, C. Strauss, and C. Stummer. Ant colony optimization in multiobjective portfolio selection. In *Proceedings of the 4th Metaheuristics Intl. Conf., MIC'2001, July 16–20, Porto, Portugal*, pages 243–248, 2001.

[DGH+04] Karl Doerner, Walter J. Gutjahr, Richard F. Hartl, Christine Strauss, and Christian Stummer. Pareto ant colony optimization : A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131(1-4) :79–99, october 2004.

[DI06] Camil Demetrescu and Giuseppe F. Italiano. Fully dynamic all pairs shortest paths with real edge weights. *J. Comput. Syst. Sci.*, 72(5) :813–837, 2006.

[DI07] Camil Demetrescu and Giuseppe F. Italiano. Algorithmic techniques for maintaining shortest routes in dynamic networks. *Electronic Notes in Theoretical Computer Science*, 171(1) :3–15, April 2007.

[Dij59] Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[DMC91] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical report, Dipartimento di Elettronica e Informatica, Politecnico di Milano, 1991.

[DMC96] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system : optimization by a colony of cooperating agents. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 26(1) :29–41, 1996.

[Dor92] M. Dorigo. *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[Dut05] Antoine Dutot. *Distribution dynamique adaptative par des mécanismes d'intelligence collective, détection d'organisations par des techniques de collaboration et de compétition*. PhD thesis, LITIS, Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes, FST, Université du Havre, 2005.

[EBJ04] H. Levine E. Ben-Jacob. Des fleurs de bactéries. *Pour la Science - Les formes de la vie*, HS-44 :78–83, 2004.

[EK95] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43, 1995.

[ER61] P. Erdös and A. Rényi. On the evolution of random graphs. *Bulletin of the Institute of International Statistics*, 38 :343–347, 1961.

[ES02] Casper Joost Eyckelhof and Marko Snoek. Ant systems for a dynamic tsp. In *ANTS '02 : Proceedings of the Third International Workshop on Ant Algorithms*, pages 88–99, London, UK, 2002. Springer-Verlag.

[Fal03] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03 : Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[Fer02] Afonso Ferreira. On models and algorithms for dynamic communication networks : The case for evolving graphs. In *4e rencontres francophones sur les Aspects Algorithmiques des Telecommunications (ALGOTEL'2002)*, Mèze, France, May 2002.

[FMSN96] Daniele Frigioni, Alberto Marchetti-Spaccamela, and Umberto Nanni. Fully dynamic output bounded single source shortest path problem (extended abstract). In *SODA : ACM-SIAM Symposium on Discrete Algorithms*

(A Conference on Theoretical and Experimental Analysis of Discrete Algorithms), 1996.

[FS03] S. Fenet and C. Solnon. Searching for maximum cliques with ant colony optimization. *Applications of Evolutionary Computing, Proceedings of EvoWorkshops*, 2611 :236–245, 2003.

[GD95] Luca Maria Gambardella and Marco Dorigo. Ant-q : A reinforcement learning approach to the traveling salesman problem. In *International Conference on Machine Learning*, pages 252–260, 1995.

[GETA99] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. MACS-VRPTW : A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.

[GL93] Fred Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

[Glo] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/.

[GLR05] Mario Gerla, Christoph Lindemann, and Ant Rowstron. P2p manet's - new research issues. In Mario Gerla, Christoph Lindemann, and Antony Rowstron, editors, *Perspectives Workshop : Peer-to-Peer Mobile Ad Hoc Networks - New Research Issues*, number 05152 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005. <http ://drops.dagstuhl.de/opus/volltexte/2005/213> [date of citation : 2005-01-01].

[GM02a] M. Guntsch and M. Middendorf. Applying population based aco to dynamic optimization problems. In *M. Dorigo et al., editor, ANTS 2002.*, 2002.

[GM02b] Michael Guntsch and Martin Middendorf. A population based approach for ACO. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and G"unther Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002 : EvoCOP, EvoIASP, EvoSTim*, volume 2279, pages 71–80, Kinsale, Ireland, 3-4 2002. Springer-Verlag.

[GPK02] T. Giraud, J. Pedersen, and L. Keller. Evolution of Supercolonies : the Argentine ants of southern Europe. *PNAS*, 99(9) :6075–6079, 2002.

[GR05] Frédéric Guidec and Hervé Roussain. *Parallel and Distributed Processing and Applications*, volume 3358/2005 of *Lecture Notes in Computer Science*, chapter Asynchronous Document Dissemination in Dynamic Ad Hoc Networks, pages 44–48. 2005.

[Gra59] P.-P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez belicositermes natalensis et cubitermes s.p. la théorie de la stigmergie : essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux*, 6 :41–80, 1959.

[GTD99] LM Gambardella, ED Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2) :167–176, 1999.

[Gär03] Felix C. Gärtner. A survey of self-stabilizing spanning-tree construction algorithms. Technical report, 2003.

[Haa97] Zygmunt Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the 6th IEEE International Conference on Universal Person Communications Record, ICUPC '97*, volume 2, pages 562–566, October 1997.

[HC04] G. Grégoire H. Chaté. La forme des groupements animaux. *Pour la Science - Les formes de la vie*, HS-44 :57–61, 2004.

[Heu98] J. C Heudin. *L'Evolution au bord du chaos*. Hermes, 1998.

[HGPC99] X. Hong, M. Gerla, G. Pei, and C.C. Chiang. A group mobility model for ad hoc wireless networks. *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60, 1999.

[Hog07] Luc Hogie. *Mobile Ad Hoc Networks : Modelling, Simulation and Broadcast-based Applications*. PhD thesis, University of Le Havre, University of Luxembourg, April 2007.

[Hog08] Luc Hogie. Madhoc, a mobile ad hoc network simuator `http://agamemnon.uni.lu/~lhogie/madhoc/`, 2003-2008.

[IEE] IEEE802.11. `http://www.ieee802.org/11/`.

[Jac01] Optimized link state routing protocol. In *IEEE INMIC Pakistan*, 2001. Best paper award.

[JBRAS05] Amit P. Jardosh, Elisabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. Real-world environment models for mobile network evaluation. *IEEE Journal on Selected Areas in Communications*, 23(3) :622–632, 2005.

[JK01] R. C. Eberhart J. Kennedy. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[KRS00] Ravi Kumar, Prabhakar Raghavan, and Sridhar Rajagopalan D. Sivakumar. Stochastic models for the web graph. In *FOCS : IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 57–65. 41th IEEE Symposium On Foundations of Computer Science (FOCS), 2000.

[KWG99] M.I. Kazantzidis, L. Wang, and M. Gerla. On fairness and efficiency of adaptive audio application layers for multihop wireless networks. In *Mobile Multimedia Communications, 1999. (MoMuC '99) 1999 IEEE International Workshop on*, pages 357–362, 15-17 Nov. 1999.

[LH99] B. Liang and ZJ Haas. Predictive distance-based mobility management for pcs networks. *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3, 1999.

[LM99] G. Leguizamon and Z. Michalewicz. A new version of ant system for subset problems. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, 6-9 July 1999.

[LM01] G. Leguizamon and Z. Michalewicz. An ant system for the maximum independent set problem. In *Proceedings of the 2001 Argentinian Congress on Computer Science*, pages 1027–1040, 2001.

[LM06] Matthieu Latapy and Clemence Magnien. Measuring fundamental properties of real-world complex networks, Sep 2006.

[Man99] V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4) :358–369, 1999.

[MC99] V. Maniezzo and A. Colorni. The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering, IEEE Transactions on*, 11(5) :769–778, 1999.

[MGS09] Nicolas Monmarché, Frédéric Guinand, and Patrick Siarry, editors. *Fourmis artificielles Des bases algorithmiques aux concepts et réalisations avancés*. Traité IC2. Hermès-Lavoisier, 2009.

[MLPW03] Shiwen Mao, Shunan Lin, S.S. Panwar, and Yao Wang. A multipath video streaming testbed for ad hoc networks. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 5, pages 2961–2965Vol.5, 6-9 Oct. 2003.

[NS-] NS-2. http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[OPN] OPNet. http://www.opnet.com/.

[PAR] PARSEC. http://pcl.cs.ucla.edu/projects/parsec/.

[Pat05] Pubudu N. Pathirana. Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Transactions on Mobile Computing*, 4(3) :285–296, 2005. Member-Nirupama Bulusu and Senior Member-Andrey V. Savkin and Member-Sanjay Jha.

[PBGK08] Apivadee Piyatumrong, Pascal Bouvry, Frédéric Guinand, and Kittichai. Trusted spanning tree for delay tolerant manets. In *to appear in the Proceedings of the 2008 IEEE/IFIP International Symposium on Trust, Security and Privacy for Pervasive Applications (TSP-08)*, Shanghai, China, December 2008.

[PPC06] Luciana Pelusi, Andrea Passarella, and Marco Conti. Beyond manets : dissertation on opportunistic networking. Technical report, IIT-CNR - COMPUTER NETWORKS DEPARTMENT, 01 2006.

[PR99a] CE Perkins and EM Royer. Ad-hoc on-demand distance vector routing. *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pages 90–100, 1999.

[PR99b] Charles Perkins and Elizabeth Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99 : Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90, Washington, DC, USA, 1999. IEEE Computer Society.

[PS97] S. Pallottino and M. G. Scutellà. Shortest path algorithms in transportation models : classical and innovative aspects. Technical Report TR-97-06, Università di Pisa - Dipartimentao di Informatica, 1997.

[Rey87] C. W. Reynolds. Flocks, Herds, and Schools : A Distributed Behavioral Model. *Computer Graphics*, 21(4) :25–34, 1987. SIGGRAPH '87 Conference.

[RR96] G. Ramalingam and Thomas W. Reps. An incremental algorithm for a generalization of the shortest-path problem. *J.Algorithms*, 21(2) :267–305, 1996.

[SAR] SARAH. http ://www-valoria.univ-ubs.fr/sarah/presentation.shtml.

[SH+00] T. Stutzle, H.H. Hoos, et al. Max-min ant system. *Future Generation Computer Systems*, 16(8) :889–914, 2000.

[SHBR97] R. Schoonderwoerd, O.E. Holland, J.L. Bruten, and L.J.M. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2) :169, 1997.

[SM01] M. Sánchez and P. Manzoni. Anejos : a java based simulator for ad hoc networks. *Future Generation Computer Systems*, 17(5) :573–583, 2001.

[STZ01] J. Shawe-Taylor and J. Zerovnik. Ants and graph coloring. In *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, ICANNGA'01*, pages 276–279, 2001.

[THB+02] Jing Tian, Joerg Haehner, Christian Becker, Illya Stepanov, and Kurt Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *SS '02 : Proceedings of the 35th Annual Simulation Symposium*, page 337, Washington, DC, USA, 2002. IEEE Computer Society.

[The97] G. Theraulaz. *Auto-organisation et comportement*. Collection Systèmes Complexes. Hermès, 1997.

[vdZM99] S. van der Zwaan and C. Marques. Ant colony optimisation for job shop scheduling. In *Proceedings of the Third Workshop on Genetic Algorithms and Artificial Life (GAAL 99)*, 1999.

[Wal94] H. Walter. *L'aventure des langues en occident*. Robert Laffont, 1994.

[WDV+97] M. Werner, C. Delucchi, H.J. Vogel, G. Maral, and J.J. De Ridder. Atm-based routing in leo/meo satellite networks with intersatellitelinks. *Selected Areas in Communications, IEEE Journal on*, 15(1) :69–82, 1997.

[Wol84] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10 :1–35, 1984.

[XC07] Toby Xu and Ymg Cai. Streaming in manet : Proactive link protection and receiver-oriented adaptation. In *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, pages 178–185, 11-13 April 2007.

[Zwi03] H. Zwirn. La complexité, science du xxi^{e} siècle ? *Pour la Science - La complexité*, 314 :28–29, 2003.

PUBLICATIONS

[Aboue-Nze et al.(2009)Aboue-Nze, Guinand, and Pigné] C. G. Aboue-Nze, F. Guinand, and Y. Pigné. Impact of obstacles on the degree of mobile ad hoc connection graphs. In *proceedings of the 5th International Conference on Networking and Services (ICNS 2009)*, Valencia, Spain, April 2009. IEEE.

[Balev et al.(2007)Balev, Guinand, and Pigné] S. Balev, F. Guinand, and Y. Pigné. Maintaining shortest paths in dynamic graphs. In *NCP07 Noncomvex Programming Local & Global Approaches*, pages 117–118, National Institute of Applied Sciences, Rouen, France, December 2007.

[Balev et al.(à paraître)Balev, Gaci, and Pigné] S. Balev, O. Gaci, and Y. Pigné. *Fourmis artificielles Des bases algorithmiques aux concepts et réalisations avancés*, chapter Fourmis artificielles et bioinformatique (repliement de protéine, alignement multiple et sequençage par hybridation). Hermès-Lavoisier, à paraître.

[Dutot and Pigné(à paraître)] A. Dutot and Y. Pigné. *Fourmis artificielles Des bases algorithmiques aux concepts et réalisations avancés*, chapter Tour d'horizon des problèmes combinatoires traités par les fourmis artificielles. Hermès-Lavoisier, à paraître.

[Dutot et al.(2007a)Dutot, Guinand, Olivier, and Pigné] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné. Graphstream : A tool for bridging the gap between complex systems and dynamic graphs. In *EPNACS : Emergent Properties in Natural and Artificial Complex Systems*, 2007a.

[Dutot et al.(2007b)Dutot, Guinand, Olivier, and Pigné] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné. On the decentralized dynamic graph-coloring problem. In *Workshop on Complex Systems and Self-Organization Modelling (Cossom 2007), satellite workshop of European Simulation and Modelling Conference (ESM'2007). October 22-24. St Julian's Malta.*, 2007b.

[Dutot et al.(à paraître)Dutot, Guinand, Olivier, and Pigné] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné. *Fourmis artificielles Des bases algorithmiques aux concepts et réalisations avancés*, chapter Principes généraux de résolution de problèmes combinatoires par colonie de fourmis. Hermès-Lavoisier, à paraître.

[Guinand and Pigné(2006)] F. Guinand and Y. Pigné. *Emergent Properties in Natural and Artificial Dynamical Systems*, chapter Problem Solving and Complex Systems, pages 53–86. Springer : Complexity. Understanding Complex Systems. Springer Verlag, 2006.

[Guinand and Pigné(2008)] F. Guinand and Y. Pigné. An ant-based model for multiple sequence alignment. In I. Lirkov, S. Margenov, and J. Wasniewski, editors, *Large-Scale Scientific Computing*, volume 4818 of *Lecture Notes in Computer Science*, pages 553–560. Springer Verlag, 2008.

[Pigné(2006)] Y. Pigné. Artificial ant-based systems for the resolution of optimization problems. In P. Amar, F. Képès, V. Norris, M. Beurton-Aimar, and J.-P. Mazat, editors, *Proceedings of the Bordeaux spring school on Modelling Complex Biological Systems in the context of genomics*, page 102, 2006.

INDEX