Curriculum Vitae : Gabriel Le Bouder

Adresse professionnelle :
LaBRI, Université de Bordeaux – Campus Talence
Domaine Universitaire
351 cours de la Libération, 33405 Talence
e-mail : prenom.n-om A u-bordeaux.fr

Date de naissance : 22/04/1994
Nationalité : Française

Situation Actuelle

— ATER depuis le 1er janvier 2025
— Université de rattachement : Université de Bordeaux
— Laboratoire de rattachement : LaBRI

Doctorat

"**Optimisation de la Mémoire pour Algorithmes Distribués Auto-Stabilisants**"
Soutenue le 6 Janvier 2023 à Sorbonne Université, devant un jury composé de :
— M. Stéphane Devismes, Professeur, MIS Université de Picardie, Rapporteur.
— M. Christian Scheideler, Professeur, Université de Paderborn (Allemagne), Rapporteur.
— M. Nicolas Hanusse, Directeur de Recherche, LaBRI, CNRS, Examinateur.
— Mme Alessia Milani, Professeure, LIS Aix-Marseille Université, Examinatrice.
— M. Sébastien Tixeuil, Professeur, LIP6 Sorbonne Université, Examinateur.
— Mme Lélia Blin, Professeure, LIP6 Univ Evry, Directrice de thèse.
— M. Franck Petit, Professeur, LIP6 Sorbonne Université, Directeur de thèse.

Thématiques de Recherche

Algorithmique distribuée, minimisation de la mémoire, optimisation de ressources, agents mobiles, tolérance aux pannes, graphes dynamiques.

Formation et Diplômes

2015 - 2019 ENS Cachan (ENS Paris-Saclay), Département Informatique, Cachan, France.
2019 MPRI, Master Parisien de Recherche en Informatique, Université Paris Diderot, Paris.
2018 Agrégation de Mathématiques option informatique.
2016 Licence Informatique, Université Paris Diderot, Paris, France.

Activités professionnelles

01-01-2025 - 31-08-2025 : ATER, UF d'informatique, Université de Bordeaux, LaBRI.

01-09-2024 - 31-12-2024 : Postdoc en algorithmique pour graphes dynamiques et géométriques, LaBRI, Université de Bordeaux.

01-09-2023 - 31-08-2024 : Postdoc sur les systèmes biologiques distribués, LMF, Université Paris-Saclay.

01-09-2022 - 31-08-2023 : ATER, UFR d'Ingénierie, Sorbonne Université, LIP6.

01-10-2019 - 06-01-2023 : Doctorant en algorithmique distribuée, LIP6 équipe DELYS (DistributEd aLgorithms and sYstems), Sorbonne Université.

01-03-2019 - 31-08-2019 : Stage de Master 2 en algorithmique distribuée encadré par Lélia Blin : *Minimisation de la mémoire dans les algorithmes auto-stabilisants bavards*, LIP6 équipe NPA (Network Performances Analysis), Sorbonne Université.

01-06-2017 - 15-08-2017 : Stage de Master 1 en réseaux (contrôle de congestion) encadré par Olivier Bonaventure : *A deeper understanding of BBR*, UC Louvain.

01-07-2016 - 15-08-2016 : Stage de Licence 3 en cryptographie encadré par Pierre-Alain Fouque : *A study over Goldreich's hash function for SPHINCS*, Université Rennes 1 - IRISA.

1 Activités de Recherche

1.1 Articles publiés

Journal

[1] Lélia Blin, Colette Johnen, Gabriel Le Bouder, and Franck Petit. Silent anonymous snap-stabilizing termination detection. In *Distributed Computing, DC*.

[2] Lélia Blin, Laurent Feuilloley, and Gabriel Le Bouder. Optimal space lower bound for deterministic self-stabilizing leader election algorithms. *Discrete Mathematics & Theoretical Computer Science, DMTCS*, vol. 25 :1, March 2023.

Conférences Internationales avec Comité de Sélection

Articles

[3] Cyril Gavoille, Nicolas Hanusse, Gabriel Le Bouder, and Taïssir Marcé. Distributed Freeze Tag : a Sustainable Solution to Discover and Wake-up a Robot Swarm. In *Symposium on Principles of Distributed Computing, PODC '25*, 2025. arXiv :2503.22521.

[4] Lélia Blin, Gabriel Le Bouder, and Franck Petit. Optimal memory requirement for self-stabilizing token circulation. In *31st International Colloquium On Structural Information and Communication Complexity, SIROCCO 2024, may 27-29, 2024, Vietri sul Mare, Salerno, Italy*, 2024. Rang : B, Taux de sélection : 50%.

[5] Lélia Blin, Colette Johnen, Gabriel Le Bouder, and Franck Petit. Silent anonymous snap-stabilizing termination detection. In *41st International Symposium on Reliable Distributed Systems, SRDS 2022, Vienna, Austria, September 19-22, 2022*, pages 156–165. IEEE, 2022. Rang : B.

[6] Lélia Blin, Laurent Feuilloley, and Gabriel Le Bouder. Optimal space lower bound for deterministic self-stabilizing leader election algorithms. In Quentin Bramas, Vincent Gramoli, and Alessia Milani, editors, *25th International Conference on Principles of Distributed Systems, OPODIS 2021, December 13-15, 2021, Strasbourg, France*, volume 217 of *LIPIcs*, pages 24 :1–24 :12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. Rang : B, Taux de sélection : 40%.

Brief Announcement

[7] Lélia Blin, Laurent Feuilloley, and Gabriel Le Bouder. Brief announcement : Memory lower bounds for self-stabilization. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, volume 146 of *LIPIcs*, pages 37 :1–37 :3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. Rang : A.

Conférences Nationales avec Comité de Sélection

[8] Cyril Gavoille, Nicolas Hanusse, Gabriel Le Bouder, and Taïssir Marcé. Réveil de robots distribué optimal. In *16e rencontres francophones sur les aspects algorithmiques des télécommunications, Algotel 2025, June 2 - 6, Saint Valery-sur-Somme, France*, 2025.

[9] Lélia Blin, Laurent Feuilloley, and Gabriel Le Bouder. Borne inférieure optimale pour la complexité spatiale des algorithmes déterministes auto-stabilisants d'élection. In *13e rencontres francophones sur les aspects algorithmiques des télécommunications, Algotel 2022, May 30 - June 3, Université Paris-Saclay, France*, 2022.

[10] Lélia Blin, Colette Johnen, Gabriel Le Bouder, and Franck Petit. Détection de terminaison autostabilisante silencieuse, anonyme et instantanée. In *13e rencontres francophones sur les aspects algorithmiques des télécommunications, Algotel 2022, May 30 - June 3, Université Paris-Saclay, France*, 2022.

1.2 Récompenses

[6] a été désigné *Best Student Paper* en 2021 à OPODIS, 25th International Conference on Principles of Distributed Systems (référence en page 10).

1.3 Travaux en cours

— Des résultats sur l'auto-stabilisation et la simulation dans les graphes dynamiques sont en cours de rédaction, et seront soumis très prochainement.
— Un projet portant sur la détection de symétries dans les réseaux à homonymes, en collaboration avec une chercheuse du LIS, est en cours et fera rapidement l'objet d'une soumission.
— Un projet portant sur la construction de MDS robuste, en collaboration avec des chercheurs du LIP6 et de l'Université de Genève, est en cours.
— Des travaux réalisés durant mon postdoc au LMF, dans le contexte de la modélisation de systèmes biologiques, ont vocation à être prochainement soumis à *PLOS Computational Biology*.

1.4 Autres activités de recherche

1.4.1 Relecture d'articles

— MFCS 2025
— SIROCCO 2025
— PODC 2024
— DISC 2021

1.4.2 Interventions en séminaires et workshop

— 2nd Workshop Complexity and Algorithms (CoA 2022), 26-28 Octobre 2022, Institut Henri Poincaré (IHP), Paris, France (page du workshop). Présentation : Optimal Space Lower Bound for Deterministic Self-Stabilizing Leader Election Algorithms
— Séminaire auprès de l'équipe SDMA du MIS, le 2 mars 2023, Amiens, France.
— Séminaire "Algorithmes Distribués" du LaBRI, le 6 mars 2023, Bordeaux, France.
— Séminaire organisé par l'équipe DALGO du LIS le 29 mars 2023, Aix-Marseille, France.
— Séminaire organisé par l'équipe Réseaux de l'ICube le 30 mars 2023, Strasbourg, France.
— Séminaire organisé par l'équipe GALaC du LISN le 31 mars 2023, Orsay, France.
— Séminaire organisé par l'IRIF le 3 avril 2023, Paris, France.
— Séminaire "Algo" organisé par le GREYC le 5 avril 2023, Caen, France.
— Séminaire organisé par l'équipe ACES, Telecom Paris le 27 février 2024, Palaiseau, France.
— Séminaire organisé par l'équipe LCP du LACL le 25 mars 2024, Créteil, France.
— Séminaire organisé par l'équipe ParSys du LISN le 3 avril 2024, Orsay, France.
— Meeting des ANR DUCAT et SkyData, 3-7 juin 2024, Grenoble, France (page de l'évenement).
— Séminaire "Algorithmes Distribués" du LaBRI, le 25 septembre 2024, Bordeaux, France.
— Meeting de l'ANR ENEDISC, 25-26 février 2025, Paris, France (page de l'évènement).

1.4.3 Encadrement

— mai-juin 2025 : encadrement d'un stage de recherche de L3 sur l'étude de propriétés de certains graphes géométriques.

1.4.4 Animation de la vie scientifique

— Animation de la table ronde "Après le doctorat" aux Journées de l'École Doctorale Mathématiques et Informatique, 3 avril 2025, Bordeaux, France.

2 Enseignement

2.1 Résumé des Enseignements dispensés : 559,25h eq-TD.

Mon service en tant que Moniteur puis ATER en 2019-2023 a été effectué à Sorbonne Université dans l'UFR d'Ingénierie (UFR 919) en filière Informatique. Mon service en tant que Postdoc a été effectué auprès d'une classe de M1 du MPRI à l'ENS Paris-Saclay. Mon service d'ATER en 2024-2025 s'est déroulé dans l'UF d'informatique de l'Université de Bordeaux.

Statut	Année	Comp.	Niveaux	nb. étu.	Volume	Type	Intitulés des cours
ATER	24-25	UFI UB	M2	1	14	Stage	Encadrement de Stage en Entreprise
ATER	24-25	UFI UB	M1	6	34	Projet	Projet de Programmation
ATER	24-25	UFI UB	L2	30	80	CM/TD/TP	Algorithmique des structures de données arborescentes
Postdoc	23-24	ENS PS	M1	15	14	TD/TP	Aspects Probabilistes de l'Informatique
Postdoc	23-24	ENS PS	M1	3	14	Projet	Initiation à la Recherche
ATER	22-23	UFR SU	M1	30	14	CM/TD/TP	Algorithmique Répartie
ATER	22-23	UFR SU	L3	34	38,5	TD/TP	Fondements des systèmes d'exploitation
ATER	22-23	UFR SU	L2	25	77	TD/TP	Structures de données
ATER	22-23	UFR SU	L2	34	38,5	TD/TP	Programmation Shell et initiation au système
ATER	22-23	UFR SU	L2	25	28	TD/TP	C avancé
Moniteur	21-22	UFR SU	L2	25	38,5	TD/TP	Structures de données
Moniteur	21-22	UFR SU	L2	25	22,75	TD/TP	Programmation fonctionnelle
Moniteur	20-21	UFR SU	L2	25	31,5	TD/TP	C avancé
Moniteur	20-21	UFR SU	L1	25	40	TP	Éléments de programmation 2
Moniteur	19-20	UFR SU	L1	25	60	TP	Éléments de programmation 2
Moniteur	19-20	UFR SU	L2/L3	25	14,5	TP	Initiation aux systèmes d'exploitation

Enseignements par type

Type	Cours	Niveau	Volume Cours	Volume tot.
Recherche	Encadrement de Stage en Entreprise	M2	14	28
	Initiation à la Recherche	M1	14	
Algorithmique	Algorithmique des Structures de Données Arborescentes	L2	80	223,5
	Aspects Probabilistes de l'Informatique	M1	14	
	Algorithmique Répartie	M1	14	
	Structures de données	L2	115,5	
Programmation	Projet de Programmation	M1	34	216,25
	C avancé	L2	59,5	
	Programmation fonctionnelle	L2	22,75	
	Éléments de programmation 2	L1	100	
Système	Fondements des systèmes d'exploitation	L3	38,5	91,5
	Programmation Shell et initiation au système	L2	38,5	
	Initiation aux systèmes d'exploitation	L2/L3	14,5	

2.2 Descriptif des enseignements dispensés

Mon service d'enseignement a été effectué trois années en tant que moniteur, une année en tant qu'ATER, une année en tant que postdoc par des vacations, et une autre année en tant qu'ATER. J'ai enseigné les bases de la programmation en C en Licence 1 et en Licence 2. En Licence 1 au second semestre, les étudiants ont une connaissance basique de la programmation apprise à travers le langage de script Python, mais de nombreuses compétences de base sont encore à acquérir ou à approfondir, la notion de boucle, les sauts conditionnels, les pointeurs et plus généralement la gestion de la mémoire, mais également la compilation et l'exécution de code.

En Licence 2 au premier semestre, nous reprenons et approfondissons ces notions, et en abordons de nouvelles plus complexes, la compilation séparée, la lecture et écriture dans des fichiers, les problèmes liés à l'utilisation de la mémoire dans le tas, l'analyse des erreurs du compilateur, et l'utilisation d'outils comme ddd ou valgrind pour les réparer, la notions de structure, de liste chaînées, d'arbres binaires. Nous introduisons également des notions particulièrement avancées, comme les pointeurs de fonctions et les pointeurs vides, en faisant réaliser aux étudiants une bibliothèque générique de gestion de listes. À

mi-semestre, un projet simulant l'évolution d'un écosystème proie-prédateur est également demandé aux étudiants, sur une durée de 5 semaines.

Au second semestre de la Licence 2, l'UE de structure de données permet de mettre en application les compétences en langage C acquises précédemment, à l'occasion de la découverte de structures de données plus complexes que les tableaux ou les listes, et les problèmes algorithmiques qui y sont liés. Nous étudions notamment les tables de hachage, les tas, les graphes, certains arbres binaires particuliers (ABR, arbres rouge-noir), introduisons différents algorithmes de tris de liste (tri rapide, tri fusion, tri par tas notamment).

D'autre part, j'ai également enseigné la programmation fonctionnelle au premier semestre de Licence 2, à travers le langage de programmation Ocaml. Nous y approfondissons la programmation récursive, et introduisons la notion de récursion terminale. Les étudiants codent des fonctions prenant comme paramètre des fonctions, typiquement des itérateurs, et progressivement les itérateurs classiques d'Ocaml sur les listes (fold_left, map, iter) sont découverts, utilisés, et également combinés entre eux. Les étudiants apprennent également à définir des types propres, utilisés pour définir différents types d'arbres sur lesquels ils sont amenés à travailler, à définir des fonctions de recherche, de calcul de chemin.

Sur un tout autre aspect, j'ai également enseigné des éléments plus accès *système*. En Licence 2, au premier semestre, j'ai enseigné la Programmation Shell et Initiation au Système. Le langage utilisé est le bash, qui permet relativement simplement de mettre en applications les notions introduites durant cette UE. Après quelques semaines dédiées essentiellement à maîtriser le langage bash, les redirections de flux, et les différents utilitaires classiques fournis par la distribution Linux ou par le noyau, tels que wc, grep, sed, kill, nous abordons en profondeurs des notions plus fondamentales du système. En particulier, nous introduisons les notions de cycle de vie d'un processus, d'exécution concurrente et d'ordonnanceur, de synchronisation entre processus (à l'aide de wait, de signaux, et de verrous Linux).

Au second semestre de Licence 2, nous approfondissons ces notions systèmes dans l'UE d'Initiation aux Systèmes d'exploitation, en utilisant cette fois ci le langage C. Une nouveauté importante de cette UE est la découverte des tubes permettant la communications entre processus, tubes anonymes et tubes nommés.

Au second semestre de Licence 3, nous allons encore plus loin dans l'enseignement système à travers l'UE de fondements des systèmes d'exploitation. Nous y introduisons différentes nouveaux concepts tels que les interruptions matérielles, les différents types d'ordonnancements, les sémaphores, la gestion de la mémoire, ou encore les systèmes de fichier.

En Algorithmique Répartie, en Master 1, j'ai présenté les principes des horloges logiques et des algorithmes à vagues en cours et en TD. En TP nous implémentons certains algorithmes à l'aide de la bibliothèque MPI.

Pendant mon postdoc, j'ai enseigné le cours d'Aspects Probabilistes de l'Informatique à une classe de M1 composée essentiellement d'étudiants de l'ENS Paris Saclay. À l'occasion de ce cours, nous introduisons et utilisons à la fois des notions mathématiques, en particulier les probabilités discrètes et continues, mais aussi des questions algorithmiques, notamment les questions traitant des algorithmes probabilistes, leur complexité, leur validité.

Cette année là, j'ai également été impliqué dans le cours d'Introduction à la Recherche pour une classe d'étudiants de M1. Pendant huit semaines, j'ai encadré un groupe de trois étudiants travaillant sur deux sujets de recherche différents. Je leur ai transmis différents aspects du métier de chercheur, comme la recherche bibliographique, le fait de présenter ses travaux, la façon de lire correctement un article de recherche.

En tant qu'ATER à l'Université de Bordeaux, j'ai pris en charge des enseignements d'algorithmiques des structures de données arborescentes, avec le langage de programmation Ocaml comme support. Ce

cours de niveau L2 est dédié à la présentation des structures d'arbres binaires usuels, ABR, AVL, Arbres rouge-noir, et à l'introduction d'algorithmes sur ces arbres, recherche, insertion, suppression.

J'ai également pris en charge quatre groupes de 5 à 6 étudiants de M1 dans l'UE projet de programmation. Chaque groupe doit développer une application de jeu de plateau comme le scrabble ou les échecs, dans un langage parmi C, Python, Java.

Durant ces cinq dernières années, j'ai démontré ma capacité à enseigner à la fois des cours d'introduction à l'informatique, au niveau licence, et ce dans une large palette de langages de programmation et selon différents thèmes, et d'autre part des cours plus avancés, au niveau Master, que ce soit dans mon domaine de recherche ou non. J'ai également découvert d'autres facettes du métier d'enseignant, la préparation de sujets (en tant que relecteur), la réalisation de barèmes, et bien entendu la correction de copies. De plus, j'ai mené ces activités d'enseignement régulièrement au cours des années, ce qui démontre que je conçois les activités d'enseignement comme faisant partie intégrante de mon activité de chercheur.

3 Travaux de Recherche

3.1 Optimisation de la mémoire pour algorithmes distribués auto-stabilisants

Période Doctorat & ATER 2019-2023

Mots-Clefs Algorithmes Distribués, Tolérance aux Pannes, Auto-Stabilisation, Minimisation de la Mémoire, Anonymat

Contexte

L'auto-stabilisation est un paradigme adapté aux systèmes distribués, particulièrement susceptibles de subir des fautes transitoires. Ces fautes peuvent être des erreurs de corruption de mémoire, de messages, la rupture d'un lien de communication, et peuvent plonger le système dans un état incohérent. Un protocole est auto-stabilisant si, quel que soit l'état initial du système, il garantit un retour à un fonctionnement normal en temps fini.

Deux mesures de qualité principales s'appliquent aux algorithmes conçus pour les systèmes distribués, la complexité temporelle et la complexité spatiale. On appelle complexité temporelle d'un algorithme le temps de calcul nécessaire à sa convergence. Cette complexité peut être impactée par la nature asynchrone inhérente aux systèmes distribués, qui peut se traduire par un ralentissement de certaines unités de calcul par rapport à d'autre. La complexité spatiale d'un algorithme est le nombre de bits d'information contenu dans chaque message échangé dans le réseau.

Avec le développement de réseaux d'objets connectés, censés être autonomes, il devient central de concevoir des algorithmes ayant un faible coût en termes de consommation énergétique et peu exigeants en termes de ressources. Une des manières d'appréhender ces problèmes est de chercher à réduire la taille des messages échangés entre les différents nœuds du réseau. Mon travail se concentre sur l'optimisation de la mémoire nécessaire à la communication pour les algorithmes distribués auto-stabilisants. Nous nous intéressons à l'étude de borne inférieure pour la complexité spatiale de certains problèmes, tout comme à la conception d'algorithmes efficaces en mémoire pour la résolution de certains problèmes.

Résultats

Dans le cadre de ma thèse, j'ai obtenu avec mes encadrants et d'autres chercheurs des résultats des deux types. Notre première contribution [6] est un résultat d'impossibilité, qui se traduit également comme une borne inférieure pour la mémoire nécessaire à la résolution de certains problèmes. Nous montrons que, quel que soit le type de réseau considéré (filaire, asynchrone, anonyme...), il n'est pas possible d'utiliser la puissance algorithmique fournie par l'existence d'identifiants uniques en utilisant une mémoire inférieure à $O(\log \log n)$ bits par nœud n étant le nombre de nœuds du réseau. Nous montrons qu'avec une mémoire inférieure à cette borne, un algorithme se comporte de la même manière sur un réseau avec identifiants que sur un réseau anonyme (sans identifiants). Nous obtenons notre résultat par un argument de dénombrement de fonctions, en montrant qu'une mémoire trop petite ne laisse pas assez de possibilité pour un algorithme de se comporter différemment en fonction de l'identifiant fourni. Étant donné que certains problèmes sont connus comme impossibles à résoudre dans les réseaux anonymes, ce résultat d'équivalence constitue bien une borne inférieure pour la complexité de ces problèmes. En particulier, des problèmes fondamentaux comme l'élection ou la construction d'arbre ne sont pas résolubles dans les réseaux anonymes, et ont donc tous deux une complexité en $\Omega(\log \log n)$ bits par nœud. La seule borne connue précédemment pour ces problèmes dans le cas général était en $\omega(1)$, c'est-à-dire qu'il avait été démontré impossible de les résoudre avec une mémoire constante en la taille du réseau. Notre borne améliore significativement l'état de l'art.

Nous proposons également deux algorithmes pour deux problèmes différents, tous deux peu exigeant en mémoire et l'un étant en réalité optimal. Ce dernier [4] est un algorithme auto-stabilisant de circulation de jeton dans un type particulier de réseaux, que sont les DODAGs, les graphes orientés acycliques avec racine unique. La circulation de jeton consiste en la circulation perpétuelle d'un unique jeton, qui doit visiter de façon équitable tous les nœuds du réseau. Notre algorithme fonctionne sous un ordonnanceur non-équitable (le plus contraignant), et avec une mémoire en $O(\log \log n)$ bits par nœud. Notre résultat sur la borne inférieure s'applique dans le cadre de ce travail, ce qui nous permet d'affirmer l'optimalité de notre

algorithme. Dans le modèle étudié, la principale difficulté dans la conception d'un tel algorithme avec une mémoire en $O(\log \log n)$ bits par nœud vient du fait que les différents nœuds du réseau communiquent systématiquement avec l'ensemble de leurs voisins, et non avec un unique voisin. Or pour faire passer le jeton d'un nœud à l'autre sans dupliquer ce jeton, il est nécessaire d'implémenter un mécanisme simulant l'envoi d'un message d'un nœud à un unique de ses voisins. Les nœuds communiquant de façon identique à tous leurs voisins, la distinction d'un unique destinataire d'un message est généralement réalisée en indiquant dans le message l'identifiant de son destinataire, ce qui a un coût en mémoire de $O(\log n)$. De la même manière, il est impossible pour un nœud de stocker un pointeur vers l'un de ses voisins, ceci ayant un coût en $O(\log \Delta)$, Δ étant le degré du réseau. Notre algorithme est, à notre connaissance, le premier algorithme utilisant $O(\log \log n)$ bits par nœud sur des réseaux de degré non-borné.

Enfin, notre dernier résultat [5] est double. Le résultat principal est un algorithme de détection de terminaison, instantanément stabilisant, pour les réseaux anonymes. Cette détection repose sur l'observation continue d'un algorithme tiers fonctionnant indépendamment sur le même réseau. Chaque nœud peut recevoir localement une requête de détection de terminaison de cet algorithme. Après une telle requête, le nœud doit fournir en temps fini une réponse à la question de savoir si l'algorithme observé a globalement stabilisé ou non. Notre algorithme utilise $O(\log D)$ bits par nœud, et fournit une réponse en $O(D)$ pas de calcul, D étant le diamètre du réseau. Cet algorithme est le premier algorithme de détection de terminaison qui soit instantanément stabilisant et utilisable dans les réseaux anonymes. Il a de plus une faible complexité spatiale. Notre algorithme se base sur l'utilisation de deux compteurs, synchronisés l'un avec l'autre de différentes manières. Le premier mode de synchronisation est que l'un des compteurs n'est activé que sous la condition que l'autre compteur a été décrémenté jusqu'à 0. Le second mode de synchronisation est que ces deux compteurs ne sont décrémentés qu'au moment où un troisième compteur, compteur d'unison, est incrémenté. L'unison est un problème classique des systèmes distribués, et de nombreux algorithmes d'unison auto-stabilisants pour les réseaux anonymes existent dans la littérature. Notre algorithme de détection de terminaison peut s'appuyer sur n'importe lequel de ces algorithmes, vu comme une boîte noire. Ceci n'est possible que parce que nous avons démontré certaines propriétés globales que respectent nécessairement tous les algorithmes auto-stabilisants, ce qui constitue la seconde partie de notre contribution. Il s'agit d'un travail théorique d'analyse des propriétés globales qui sont nécessairement vérifiées par tout algorithmes qui respecte la spécification du problème de l'unison.

Questions ouvertes

Notre algorithme de détection de terminaison ne retourne que des réponses positives : aucune réponse n'est fournie tant que le l'algorithme observé n'a pas réellement stabilisé. Adapter notre solution pour pouvoir retourner des réponses négatives, tout en maintenant la propriété de stabilisation instantanée, sans faux négatif, n'est pas simple à réaliser. Deux autres questions restent en suspens : la première concerne la nécessité pour les nœuds de connaitre une borne sur le diamètre du réseau pour résoudre ce problème, dans le cas des réseaux anonymes, et la seconde, liée à la première, concerne l'optimalité en mémoire de notre algorithme instantanément stabilisant pour les réseaux anonymes. Plus généralement se pose la question de la nécessité d'utiliser $\Omega(\log D)$ bits par nœud pour résoudre des problèmes d'observation globale dans les réseaux anonymes.

D'autre part, notre borne inférieure en $\Omega(\log \log n)$ bits par nœud s'applique à de nombreux problèmes pour lesquels des algorithmes plus ou moins efficaces existent. La question qui subsiste concerne l'optimalité de cette borne. En ce qui concerne les problèmes de l'élection et de la construction d'arbre, les algorithmes les plus efficaces nécessitent $O(\log \log n + \log \Delta)$ bits par nœud. La question de savoir s'il est possible de résoudre ces deux problèmes en utilisant exactement $\Theta(\log \log n)$ bits par nœud reste ouverte.

D'autre part, notre algorithme de circulation de jeton nous permet d'être optimistes quant à la possibilité de concevoir un algorithme auto-stabilisant de circulation de jeton pour les réseaux quelconques, et non plus seulement les DODAGs, avec une complexité spatiale de $\Theta(\log \log n)$ bits par nœud. Trois problèmes fondamentaux doivent être résolus pour obtenir un tel résultat dans le cadre de l'auto-stabilisation. Le premier est l'existence de plusieurs racines. La présence de multiples racines n'est pas la situation la plus compliquée, puisqu'il nous semble possible d'implémenter une négociation à distance, sur la base des identifiants, entre les différentes racines. Le deuxième problème est l'absence de racine. L'absence de racine est elle un peu plus compliquée à gérer, et notamment à détecter. En effet en l'absence de

racine, on peut tout à fait envisager un réseau dans lequel aucun token ne circule, et aucun nœud pour détecter l'anomalie. Le troisième problème fondamental, qui n'est pas sans lien avec l'absence de racine, est la présence de cycles dans le graphe ces cycles étant par définition proscrits dans les DODAGs. La difficulté majeure consiste en la détection de ces cycles, avec une mémoire en $O(\log \log n)$. Les symétries causées par la présence de cycles rendent également très compliquée l'adaptation de notre algorithme aux graphes généraux.

L'existence d'un tel algorithme, résolvant la circulation de jeton dans les graphes généraux en $\Theta(\log \log n)$ bits par nœuds, constituerait un outil particulièrement intéressant pour la conception d'algorithmes efficaces en mémoire pour d'autres problèmes, et notamment pour l'élection et la construction d'arbre.

3.2 Systèmes Biologiques

Période Postdoc 2023-2024

Mots-Clefs Systèmes Biologiques, Processus Cellulaires Stochastiques, Algorithmique Distribuée

Contexte

Mon premier postdoc s'intégrait dans le cadre du projet ANR Dreamy qui vise à développer des solutions innovantes au problème de construire des circuits distribués à l'aide de bactérie, en s'appuyant sur des méthodes algorithmiques et théoriques.

Le développement de la génétiques ces dernières décennies rendent possible d'utiliser des colonies de bactéries à des fins de calcul. Chaque bactérie est considérée comme une unité de calcul, qui peut de diverses manières communiquer ou influencer les bactéries autour d'elle. Étant donné le grand nombre de tels agents, et leur capacités de calcul relativement faibles, l'utilisation de méthodes issues du calcul distribué est une approche adaptée à l'étude des capacités de calcul de tels systèmes.

Résultats et Suites

Le premier problème sur lequel je me suis penché est l'introduction de nouvelles méthodes formelles pour estimer la vitesse de croissance d'une population de bactéries. Cette vitesse de croissance est un outil permettant notamment de mesurer la performance d'un algorithme distribué appliqué à une colonie de bactérie selon une métrique spécifique à ces systèmes : le coût en énergie nécessaire à l'implémentation des règles de l'algorithme par ces systèmes biologiques. Ce travail a abouti à un article de recherche en cours de rédaction.

Une fois que ce travail sera terminé, d'autres questions ouvertes qu'il serait intéressant d'étudier sont entre autres la résolution de problèmes dans ce modèle de calcul, tels que le consensus, les propriétés des systèmes oscillants, ou bien le contrôle de population.

Plus généralement, la recherche de liens existants entre les diverses méthodes formelles considérées en algorithmique distribuées (l'auto-stabilisation, les agents Byzantins, les protocoles de population, etc.) et l'étude de systèmes biologiques tels que les colonies de bactéries est un domaine extrêmement large dans lequel de nombreuses questions sont toujours ouvertes.

3.3 Algorithmique des Graphes Dynamiques et Géométriques

Période Postdoc/ATER 2024-2025

Mots-Clefs Graphes Dynamiques, Agents Mobiles, Algorithmes Distribués, Synchronisation, Exploration, Robustesse.

Contexte

Certain réseaux tels que les réseaux cellulaires, les réseaux Wi-Fi, ou les réseaux d'agents mobiles, ne peuvent pas être considérés statiques au cours du temps, étant donné que les agents considérés se déplacent dans l'espace. Les effets sur la topologie du réseau ne peuvent pas être modélisés comme de simples fautes transitoires, et doivent au contraire être vus comme faisant partie intégrante du réseau.

Différents modèles existent pour représenter de tels réseaux, et différentes stratégies coexistent pour résoudre des problèmes dans ces réseaux. Un premier modèle est celui des agents mobiles, entités se déplaçant dans le plan en suivant un algorithme et en fonction des interactions avec leur environnement, en vue de réaliser une tâche particulière. Un autre modèle est celui des graphes dynamiques, graphes dont les arêtes ne sont pas stables au cours du temps. Ce modèle permet de capturer l'asynchronie dans les communications entre les processus. Un troisième modèle est celui de la robustesse, paradigme qui consiste à étudier l'existence et le calcul de solutions à des problèmes de graphes valides dans tout sous graphe connecté du graphe initial. Ce paradigme vise à trouver des solutions qui permettent de ne pas avoir à tenir compte des aléas dans les communications du réseau.

Résultats

Durant mon postdoc je me suis intéressé à ces trois paradigmes. Je travaille en collaboration avec Colette Johnen sur l'élaboration de solutions auto- ou pseudo-stabilisantes pour des problèmes de synchronisation dans certaines classes de graphes dynamiques, dans l'optique de construire des transformeurs, permettant d'utiliser des algorithmes distribués statiques dans certaines classes de graphes dynamiques. J'ai également travaillé autour du problème d'existence de solutions robustes pour des problèmes de spanners, en particulier autour du MDS, en collaboration avec Arnaud Casteigts et Swan Dubois. Enfin, j'ai travaillé avec Cyril Gavoille, Nicolas Hanusse, et Taïssir Marcé sur la version distribuée d'un problème d'agents mobiles, initialement posé comme un problème centralisé : le Freeze Tag, ou réveil de robots. Nous avons apporté des algorithmes performants, ainsi que des bornes inférieures pour la résolution de ce problème dans différentes variantes distribuées de ce problème. Dans plusieurs cas, nos solutions sont donc démontrées optimales. Ce dernier travail a été soumis à PODC. Une continuation naturelle de ces travaux est celle de l'étude du Freeze Tag distribué dans un contexte asynchrone, dans lequel de nouveaux problèmes algorithmiques émergent.