Titre: Title:	Stratégies de mappage de tâches sur grappe de calcul hétérogène pour la simulation de transitoires électromagnétiques de réseaux électriques
Auteur: Author:	Julie Durette
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Durette, J. (2025). Stratégies de mappage de tâches sur grappe de calcul hétérogène pour la simulation de transitoires électromagnétiques de réseaux électriques [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/66540/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/66540/
Directeurs de recherche: Advisors:	Antoine Lesage-Landry, & Gunes Karabulut Kurt
Programme: Program:	Génie électrique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Stratégies de mappage de tâches sur grappe de calcul hétérogène pour la simulation de transitoires électromagnétiques de réseaux électriques

JULIE DURETTE
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie électrique

Mai 2025

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Stratégies de mappage de tâches sur grappe de calcul hétérogène pour la simulation de transitoires électromagnétiques de réseaux électriques

présenté par **Julie DURETTE**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Brunilde SANSO, présidente

Antoine LESAGE-LANDRY, membre et directeur de recherche

Gunes KARABULUT KURT, membre et codirectrice de recherche

Quentin CAPPART, membre

REMERCIEMENTS

Ce mémoire de maitrise a été rendu possible grâce au soutien financier de Mitacs et de OPAL-RT Technologies. Je suis reconnaissante envers OPAL-RT Technologies pour leur collaboration et leur appui technique. Je remercie sincèrement le professeur Antoine Lesage-Landry et la professeure Güneș Karabulut Kurt de m'avoir offert l'opportunité de réaliser cette recherche sous leur direction.

RÉSUMÉ

Ce mémoire de maitrise présente une amélioration des stratégies de mappage des tâches pour les simulations de transitoires électromagnétiques (*electromagnetic transient*, EMT) de réseaux électriques en temps réel sur des grappes de calcul hétérogènes. Il aborde l'optimisation de la distribution des calculs parallèles en tenant compte à la fois des liens de communication hétérogènes et des capacités de traitement variables des nœuds de calcul.

Deux modèles complémentaires sont proposés, chacun accompagné de sa formulation mathématique. Le modèle d'assignation optimise le placement des tâches lorsque le nombre de nœuds de calcul est fixe et prédéterminé, en mettant l'accent sur la minimisation des coûts de communication tout en respectant les contraintes de capacité de calcul. Le modèle de *bin packing*, quant à lui, détermine le nombre minimal de nœuds de calcul nécessaires en tenant compte des contraintes de capacité de calcul et de communication, ce qui le rend particulièrement adapté aux scénarios où les ressources de calcul sont limitées.

La méthodologie se concentre sur deux contributions principales. D'abord, la librairie `PuLP`, qui utilise des techniques de propagation d'étiquettes pour le partitionnement multiobjectif, a été adaptée afin de résoudre les problèmes d'assignation et de *bin packing*. Les modifications intègrent les coûts de communication et les contraintes de traitement entre nœuds hétérogènes. Ensuite, les performances de la librairie de partitionnement `SCOTCH`, traditionnellement utilisée pour le partitionnement de maillage de simulation numérique pour des problèmes de mécanique des structures, ont été évaluées pour du mappage de tâches de simulations EMT avec une configuration de grappes hétérogènes. Bien que cette librairie offrait déjà certaines fonctionnalités nécessaires, elles n'avaient pas été testées auparavant dans le mappage de tâches de simulations de réseaux électriques pour des grappes hétérogènes.

La recherche exploite les caractéristiques topologiques propres aux tâches computationnelles des simulations EMT pour optimiser la distribution des tâches sur les grappes de calcul. Elle présente la première application et analyse de performance connue de ces approches de partitionnement pour les tâches de simulation de réseaux électriques.

Les résultats expérimentaux illustrent l'efficacité des deux approches pour répondre à différents cas d'utilisation tout en maintenant les exigences de performance en temps réel. Le modèle d'attribution s'avère efficace pour optimiser les configurations de grappes existantes, tandis que le modèle de *bin packing* permet de déterminer les besoins minimaux en unité de calcul pour la simulation. Ce mémoire fournit des perspectives sur l'implémentation pratique de simulations numériques sur des infrastructures de calcul hétérogènes.

ABSTRACT

This Master's thesis introduces improved task mapping strategies for real-time electromagnetic transient (EMT) simulations on heterogeneous computing clusters. The focus is on optimizing the mapping of parallel computation tasks by taking into account both heterogeneous communication links and varying processing capacities across computing nodes.

The problem is approached through two complementary models, each with its own mathematical programming formulation. The first model, referred to as the bottleneck quadratic semi-assignment problem (BQSAP), focuses on optimizing task configuration when the number of computing nodes is fixed and predetermined, emphasizing the minimization of communication costs while adhering to processing constraints. The second model, formulated as a variable-size bin packing problem with a quadratic constraint for communication costs (Q-VSBPP), primarily aims to minimize the number of partitions, which the assignment problem does not address. By determining the minimal number of computing nodes required and considering both computational capacity and communication constraints, this model is particularly advantageous for resource provisioning scenarios where reducing the number of computers in the cluster is a priority.

Our research contributes two main advancements. First, we adapt the `PuLP` library, which employs label-propagation techniques for multi-objective partitioning, to solve both the BQSAP and Q-VSBPP problems. Our modifications incorporate communication costs and processing constraints across heterogeneous nodes. Second, we evaluate the performance of the partitioning library `SCOTCH`, which is traditionally designed to partition meshes for structural mechanics numerical simulations, on the mapping of EMT parallel simulations on heterogeneous cluster configurations. Although the necessary functionalities existed within this library, to the best of our knowledge, there had been no prior testing of their application for task mapping of EMT simulations on heterogeneous clusters.

The research leverages the topological characteristics inherent to the computational tasks of EMT simulations of electric networks to optimize task configuration on computing clusters. We present the first known application and performance analysis of these partitioning approaches for electric network simulation tasks.

Experimental results illustrate the effectiveness of both approaches in addressing different use cases while maintaining real-time performance requirements. The BQSAP model is effective for optimizing existing cluster configurations, while the Q-VSBPP model allows for determining minimal cluster requirements for the simulation. This thesis provides insights into the practical implementation of numerical simulations on heterogeneous computing infrastructure.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

Abréviation	Définition
BQSAP	*Bottleneck Quadratic Semi-Assignment Problem* – Problème de semi-assignation quadratique avec goulot.
EMT	*Electromagnetic Transient* – Simulation de transitoires électromagnétiques.
FM	*Fiduccia-Mattheyses* – Algorithme de partitionnement de graphes, proposé par Fiduccia et Mattheyses.
HIL	*Hardware-in-the-Loop* – Technique de test intégrant du matériel physique dans une simulation temps réel.
`MPI`	*Message Passing Interface* – Standard pour le calcul parallèle par passage de messages, défini par MPI Forum.
`OpenMP`	*Open Multi-Processing* – Standard pour le calcul parallèle en mémoire partagée, développé par OpenMP Architecture Review Board.
`PuLP`	*Partitioning using Label Propagation* – Librairie de partitionnement basée sur la propagation d'étiquettes, développée par le Sandia National Laboratory.
Q-VSBPP	*Variable-Size Bin Packing Problem with Quadratic Communication Constraints* – Problème de *bin packing* de taille variable avec contraintes quadratiques de communication.
`SCOTCH`	Librairie de mappage statique, de partitionnement de graphes et maillages, développée par le LaBRI/INRIA.

CHAPITRE 1 INTRODUCTION

La gestion, la planification et l'exploitation des réseaux électriques font face à des défis liés à leur complexité et à l'intégration des sources d'énergie renouvelable. Dans ce contexte, la simulation numérique est un outil qui permet de comprendre, d'analyser et d'optimiser le comportement de ces systèmes critiques. Plus spécifiquement, la simulation de transitoires électromagnétiques (EMT, de l'anglais *electromagnetic transient*) permet d'étudier en détail la dynamique rapide du réseau, d'évaluer sa stabilité face à divers événements, tels que les défauts ou les opérations de commutation, et de valider de nouvelles stratégies de contrôle ou l'intégration de nouveaux équipements.

L'utilisation de la simulation EMT est préconisée par des normes internationales, comme IEEE-2800 [1], pour l'étude de l'interconnexion des ressources énergétiques et l'analyse de la stabilité. Ces simulations permettent de réaliser des tests sur les conceptions de systèmes et les procédures d'exploitation, d'identifier des problèmes potentiels et d'appuyer la planification et la prise de décision. Comparée aux essais physiques sur le réseau réel, la simulation numérique offre une alternative nettement moins coûteuse, non disruptive pour l'opération du réseau, et permet souvent des tests plus exhaustifs.

Cependant, la simulation EMT de réseaux électriques à grande échelle soulève un défi computationnel. Ces réseaux modélisent souvent les systèmes de transport d'électricité qui peuvent s'étendre sur de vastes territoires, couvrant parfois plusieurs provinces, états ou pays [2–7]. Les analyses requièrent une puissance de calcul importante, d'autant plus que pour certaines applications, comme l'analyse en temps réel d'incidents ou l'évaluation de systèmes de protection et de contrôle, des contraintes temporelles strictes doivent être respectées. Si les supercalculateurs de grande échelle offrent la puissance de calcul requise pour ces simulations, leur coût élevé et leur disponibilité limitée constituent des freins à leur déploiement généralisé.

Pour surmonter ces contraintes d'accessibilité et de coût, une approche alternative consiste à distribuer les simulations sur des grappes (*clusters*) d'ordinateurs multicœurs modernes. L'utilisation de grappes d'ordinateurs interconnectés offre de la flexibilité et la possibilité d'ajuster la puissance de calcul par l'ajout ou le retrait de machines. Cette méthode peut rendre la simulation numérique plus accessible que l'utilisation de supercalculateurs dédiés.

Cependant, la distribution d'une simulation EMT sur une grappe introduit le problème de mappage des tâches (*task mapping* en anglais). Il s'agit de répartir les différentes parties du calcul entre les nœuds de la grappe. Cette répartition doit être optimisée pour minimiser le temps d'exécution total, ce qui nécessite de considérer la charge de calcul de chaque tâche, la capacité de

traitement des différents ordinateurs et les délais de communication dus aux échanges de données entre tâches sur des machines distinctes. Une assignation sous-optimale peut créer des goulots (*bottleneck* en anglais) où certains nœuds surchargés ou des communications lentes limitent les performances globales et ralentissent l'ensemble de la simulation, affectant le respect des objectifs de performance, notamment pour la simulation en temps réel. Ainsi, nous cherchons à minimiser le goulot, afin d'accélérer autant que possible le temps global requis pour la simulation EMT.

Ce projet de recherche aborde spécifiquement ce problème complexe de mappage pour les simulations EMT parallélisées sur des grappes de calcul hétérogènes. L'objectif principal est de formuler et d'évaluer des stratégies de mappage de tâches qui intègrent ces différentes contraintes – capacités de calcul, coûts de communication – afin d'optimiser la performance globale. L'accent est mis en particulier sur la minimisation du temps de simulation, correspondant à un objectif de type goulot, critique pour les applications en temps réel.

L'amélioration de l'efficacité de ces simulations par une parallélisation optimisée est importante, car elle peut rendre ces outils d'analyse plus accessibles et plus performants. Ainsi, une meilleure assignation des tâches sur des grappes de calcul pourrait ainsi contribuer à une gestion plus efficace des réseaux électriques, faciliter la planification de l'intégration à grande échelle des énergies renouvelables et, ultimement, soutenir la transition vers des systèmes énergétiques plus durables et fiables.

1.1 Définitions et concepts de base

Cette section définit les notions fondamentales pour ce mémoire. Nous présentons la simulation EMT et les principes de la parallélisation de la simulation. Nous définissons ensuite le problème d'optimisation combinatoire qu'est le mappage de tâches et finalement nous décrivons les caractéristiques des tâches de calcul considérées pour ce mémoire.

1.1.1 Simulation de transitoires électromagnétiques (EMT)

Une simulation EMT permet l'analyse des systèmes électriques, offrant une modélisation temporelle à haute résolution des réseaux électriques et de leurs comportements dynamiques [2–8]. Elle capture les événements de commutation rapide, les phénomènes de propagation d'ondes et les réponses dynamiques dans les systèmes électriques pour comprendre la stabilité du système, les exigences de protection et les performances des équipements dans diverses conditions de fonctionnement. La capacité de la simulation EMT à modéliser les phénomènes électromagnétiques avec une résolution de l'ordre de la microseconde à la nanoseconde permet aux ingénieurs d'analyser les interactions entre les dispositifs d'électronique de puissance, les systèmes de contrôle et les

composants du réseau, par exemple.

Les simulations de type EMT peuvent s'exécuter selon deux modalités distinctes, soit en mode hors ligne (*offline*) ou en mode temps réel (*real-time*). La simulation hors ligne, qui est fréquente pour les études d'ingénierie détaillées, se caractérise par un temps de calcul qui est supérieur au temps physique simulé. Cette approche permet d'utiliser des pas de temps (*timestep*, en anglais) très fins, en microsecondes ou moins, ainsi que des modèles complexes pour analyser finement les phénomènes dynamiques. La simulation en temps réel, pour sa part, impose que chaque pas de temps soit calculé en une durée inférieure ou égale à ce pas de temps physique. Atteindre cette performance exige typiquement des plateformes matérielles dédiées et des modèles spécifiquement adaptés ou simplifiés. La simulation en temps réel est indispensable pour des applications interactives telles que les systèmes de test de type *hardware-in-the-loop* (HIL), une technique permettant de valider un composant matériel réel en le connectant à un simulateur qui émule son environnement d'exploitation et ses interactions.

1.1.2 Parallélisation de la simulation EMT

Les simulations EMT engendrent une charge de calcul significative, quel que soit le mode d'exécution. Pour gérer cette demande, le recours à la parallélisation est une stratégie courante. En mode hors ligne, distribuer les calculs sur de multiples cœurs de processeur ou machines permet de réduire considérablement le temps d'exécution global, ce qui permet d'obtenir des résultats de simulations détaillées dans des délais raisonnables et de rendre l'analyse plus efficace. En mode temps réel, la parallélisation est souvent une exigence technique pour parvenir à exécuter les calculs complexes de chaque pas de temps à l'intérieur des contraintes temporelles strictes.

Dans le contexte de la parallélisation des simulations EMT, diverses méthodes de décomposition en tâches de calcul parallèle ont été développées et largement étudiées. De ce fait, ces techniques n'entrent pas dans la définition du projet de recherche présenté dans ce mémoire. Pour une revue exhaustive de ces approches, le lecteur peut se référer aux travaux [9–12].

Parmi les approches existantes, la parallélisation basée sur les délais de propagation physiques des lignes de transmission constitue un exemple représentatif [9]. Cette stratégie exploite le fait que lorsque les délais de propagation excèdent le pas de temps de simulation, ils génèrent des frontières naturelles facilitant le calcul parallèle. Ainsi, différents segments du réseau peuvent être traités indépendamment par une simulation parallèle, préservant la précision de la solution par le biais d'échanges de données synchronisés entre les tâches adjacentes.

La simulation progresse avec des pas de temps discrets, où chaque pas comprend deux phases distinctes : une phase de calcul suivie d'une phase de communication de données. Pendant la

phase de calcul, chaque processeur résout indépendamment les équations pour les segments de réseau qui lui sont assignés en utilisant des données locales. La phase de communication qui suit permet l'échange de valeurs aux frontières entre les segments adjacents, assurant la continuité de la solution à travers l'ensemble du système. Une barrière de synchronisation garantit que cette phase de communication ne débute qu'une fois que tous les nœuds ont achevé leurs calculs pour le pas de temps courant. Le cycle de calcul et de communication se répète pour chaque pas de temps, comme illustré à la figure 1.1, permet de faire évoluer l'état du système simulé au fil du temps.

La durée du pas de temps dans la simulation EMT est donc déterminée par deux goulots : le goulot en temps de calcul, correspondant au nœud ayant la charge computationnelle la plus élevée, et le goulot en temps de communication, défini par le lien le plus lent ou le volume de données le plus important. Minimiser la somme de ces deux goulots en temps permet de réduire la durée du pas de temps et ainsi accélérer la simulation [13, 14].

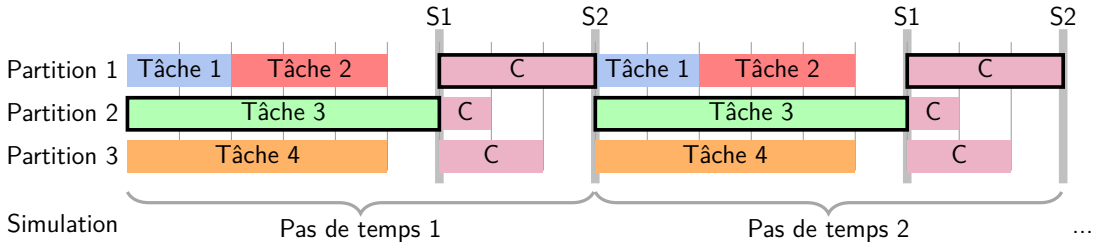


FIGURE 1.1 Représentation d'une assignation de tâches de simulation EMT. Chaque pas de temps (*timestep* en anglais) comprend une phase de calcul, une synchronisation (S1), une phase de communication (C) pour la mise à jour des données entre les tâches, puis une synchronisation finale (S2) qui initie le cycle suivant. La durée du pas de temps est déterminée par les goulots (*bottlenecks* en anglais) sur la phase de calcul à la partition 2 et sur la phase de communication à la partition 1. Les goulots sont encadrés sur la figure.

1.1.3 Problème de mappage de tâches

Le problème de mappage de tâches (*task mapping* en anglais) consiste à attribuer un ensemble de tâches distinctes à un ensemble de ressources disponibles, en respectant certaines contraintes et en visant à optimiser un ou plusieurs objectifs. Dans un mappage dynamique, les décisions d'allocation des tâches aux ressources sont prises en cours d'exécution, en réponse à l'arrivée de nouvelles tâches, aux changements des conditions du système ou à l'évolution des exigences des tâches déjà présentes. À l'inverse, dans le cas d'un mappage statique, l'ensemble des tâches, des ressources et leurs caractéristiques sont connues à l'avance, et l'assignation est déterminée avant le début de l'exécution [15, 16]. C'est le contexte de mappage statique qui prévaut pour le

problème étudié dans ce mémoire. Sur le plan mathématique, le problème de mappage de tâches est un problème d'optimisation combinatoire [17,18]. Selon la structure spécifique du problème et les objectifs, il peut être rattaché à l'une des trois approches fondamentales de sélection de sous-ensembles : le problème de couverture, le problème de *packing* ou le problème de partitionnement. Ces trois problèmes, illustrés à la figure 1.2, se distinguent par les contraintes d'appartenance imposées aux éléments :

- La couverture autorise les chevauchements en exigeant que chaque élément appartienne à au moins un sous-ensemble de la sélection.
- Le *packing* consiste à choisir des sous-ensembles disjoints où chaque élément appartient à au plus un sous-ensemble.
- Le partitionnement impose que chaque élément appartienne à exactement un sous-ensemble, formant ainsi une partition complète.

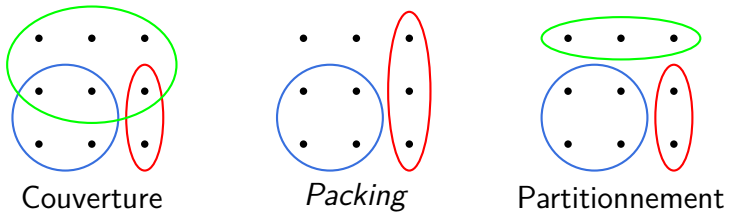


FIGURE 1.2 Illustration de trois problèmes classiques d'optimisation combinatoire : le problème de couverture d'ensemble (à gauche), le problème de *packing* d'ensemble (au centre) et le problème de partition d'ensemble (à droite). La couverture autorise les chevauchements, le *packing* les interdit, et le partitionnement exige une couverture complète sans chevauchements.

Le problème traité dans ce mémoire correspond spécifiquement à un problème de partitionnement : nous cherchons à diviser un ensemble complet de tâches en sous-ensembles disjoints où chaque tâche appartient à exactement un sous-ensemble. La complexité du problème est accrue en raison de plusieurs facteurs : les tâches ont des poids différents, il existe des communications entre les tâches ainsi que des connexions entre les partitions, et les partitions présentent différentes capacités. Le modèle complet sera décrit à la section 3.

Une difficulté inhérente au problème combinatoire traité dans le mémoire réside dans le phénomène d'explosion combinatoire : le nombre total de solutions possibles croît de façon exponentielle avec le nombre de tâches et de ressources. Le problème de mappage de tâches appartient à la classe des problèmes NP-hard [15, Sec. 1.10] [19, p.218, p.226], ce qui signifie qu'aucun algorithme polynomial n'est connu pour trouver la solution optimale dans le cas général. Par conséquent, la recherche d'une solution optimale par énumération exhaustive devient impraticable dès que le problème atteint une taille significative [20].

Comme présenté par [21], des méthodes d'énumération partielle, ne requérant pas une énumération complète du domaine des solutions possibles, peuvent être utilisées pour certains problèmes combinatoires. Certaines variantes du problème de partitionnement peuvent être résolues à l'aide de décompositions Bender, de programmation dynamique, de *branch-and-cut* ou de relaxation lagrangienne [22, 23]. Cependant, la variante du problème traitée dans ce mémoire a une taille considérable et des contraintes supplémentaires, ce qui rend l'utilisation d'heuristiques nécessaire pour l'obtention d'une solution dans les temps requis. Il s'agit d'ailleurs un point soulevé par [23] particulièrement à propos d'une configuration hétérogène de grappes d'ordinateurs. Pour plus d'informations sur les différentes méthodes de résolution, le lecteur est invité à consulter [24–27]. À des fins d'exhaustivité, nous présentons à la section 4.2.1 les résultats des tests préliminaires que nous avons effectués avec le solveur `Gurobi` sur différentes variantes du problème traité.

1.1.4 Caractéristiques du graphe des tâches de calcul

Le problème de mappage de tâches étudié dans ce mémoire se caractérise par l'interaction de trois graphes : (i) le graphe de la grappe d'ordinateurs formé par les nœuds de calcul et les liens de communication, (ii) le graphe représentant le réseau électrique de la simulation EMT et (iii) le graphe des tâches de calcul de la simulation EMT pour lequel chaque tâche représente un sommet et les arêtes expriment les transferts de données entre les tâches voisines à chaque pas de temps de la simulation. Les spécifications des tâches prises comme données d'entrées pour le mappage de tâches sont les suivantes :

- les identifiants uniques de chacune des tâches ;
- les temps de calcul estimés pour un ordinateur de référence, et ;
- une liste d'adjacence avec les volumes de transmission de données entre les tâches à chaque pas de temps pendant la phase de communication, exprimés en nombre de `Doubles` transmis, soit 64 bits par valeur selon la norme IEEE 754 [28].

Comme discuté à la section 1.1.2 précédente, la décomposition de la simulation EMT en tâches de calcul ne fait pas partie du projet actuel. Ainsi, l'étude préliminaire des données d'entrée du problème porte sur la caractérisation des propriétés topologiques du graphe des tâches de calcul à partir des données listées ci-dessus.

Les graphes de tâches de calcul étudiés comprennent jusqu'à 1100 tâches et 1300 arêtes. Il s'agit des tâches de calcul de configuration réelle de simulations EMT de réseaux électriques de grande taille. La figure 1.3 illustre le graphe des tâches de la simulation EMT d'un des réseaux étudiés dans le mémoire et les données numériques sont présentées à la section 4. Afin d'étudier les caractéristiques topologiques des graphes de tâches, nous avons sélectionné des paramètres

pertinents aux graphes de grande taille [29, Ch. 10]. Pour un graphe $G = (\mathcal{V}, \mathcal{E})$ composé d'un ensemble de sommets $\mathcal{V}$ et d'arêtes $\mathcal{E}$:

- la distribution des degrés caractérise la répartition du nombre d'arêtes connectées à chacun des sommets du graphe. Elle permet d'analyser la variabilité de la connectivité entre les différents nœuds ;
- le coefficient de *clustering* $\mathrm{cc}(G)$ représente le niveau de connectivité locale au sein d'un graphe. En d'autres mots, il indique que les voisins d'un nœud ont une forte probabilité d'être également connectés entre eux. Pour chaque sommet $v \in \mathcal{V}$, le coefficient individuel $\mathrm{cc}(v)$ mesure l'interconnexion entre ses voisins directs. Le coefficient de *clustering* global du graphe $\mathrm{cc}(G)$ correspond à la moyenne arithmétique :

$$\mathrm{cc}(G) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \mathrm{cc}(v) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{2\,x}{k(k-1)}\,,$$

  où k est le nombre de voisins du sommet v et x est le nombre d'arêtes reliant les voisins.
- la distance moyenne représente la moyenne de toutes les distances entre les paires de sommets d'un graphe. La distance entre deux sommets représente au nombre d'arêtes sur le plus court chemin les reliant. La distance moyenne constitue un indicateur de la proximité générale entre les sommets du graphe.

Ces paramètres nous permettent de comparer les graphes de tâches avec des modèles canoniques associés aux réseaux à grandes tailles comme les graphes réguliers, aléatoires, de type petit-monde et de type sans-échelle [29, Ch. 10], [30, 31]. Les graphes réguliers possèdent un degré identique pour tous les sommets du graphe. À l'opposé, les réseaux dits aléatoires présentent une distribution de degré exponentielle avec un *clustering* faible et de courtes longueurs de chemin, bornées logarithmiquement du nombre de sommets. Les réseaux de type petit-monde combinent les caractéristiques des deux modèles précédents. D'une part, ils présentent un haut coefficient de *clustering*. D'autre part, et de manière quelque peu contre-intuitive par rapport au fort *clustering*, ils possèdent une faible distance moyenne entre deux sommets quelconques, une propriété habituellement associée aux graphes aléatoires. Cette faible distance se traduit généralement par une distance moyenne proportionnelle au logarithme de la taille du graphe $|\mathcal{V}|$, contrairement à la croissance linéaire observée dans un réseau régulier où chaque nœud possède le même nombre de connexions. Pour les réseaux sans-échelle (*free-scale*), leur caractéristique principale est la distribution de degré qui suit une loi de puissance, créant des structures hétérogènes avec des nœuds centraux hautement connectés. Ces modèles théoriques peuvent être combinés pour capturer des comportements de réseaux plus complexes observés dans les systèmes réels.

La topologie spécifique des graphes de type petit-monde ou sans-échelle peut influencer l'efficacité

de certains algorithmes opérant sur les graphes, notamment ceux liés à la recherche de chemins ou au partitionnement de graphes [32]. Les graphes de tâches étudiés pour ce mémoire démontrent des propriétés de type petit-monde et de type sans-échelle, comme détaillé à la section 4.1.2. L'identification de cette structure dans les graphes de tâches EMT a ainsi guidé le choix et l'adaptation d'algorithmes de partitionnement de ce mémoire.

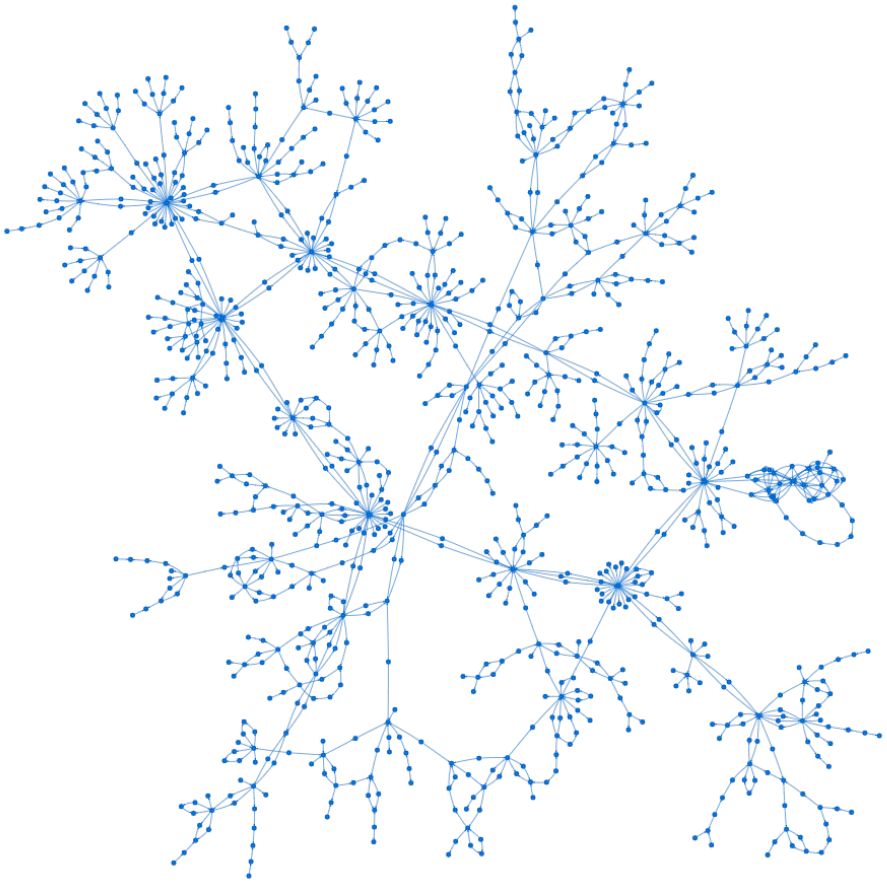


FIGURE 1.3 Visualisation du graphe de tâches de la simulation EMT d'un des réseaux électriques étudiés (le réseau D, décrit à la section 4.1.2), pour lequel chaque tâche est un sommet et les arêtes représentent les transferts de données entre les tâches voisines à chaque pas de temps de la simulation.

1.2 Éléments clés de la problématique

Ce mémoire s'inscrit dans le cadre d'une collaboration de recherche avec l'entreprise OPAL-RT Technologies, un acteur majeur dans le domaine de la simulation numérique temps réel. L'optimisation du processus de mappage de tâches pour HYPERSIM, un logiciel de simulations EMT temps réel, a été ciblée comme un besoin prioritaire, motivant le présent projet de maitrise.

Pour répondre aux besoins de performance, le logiciel HYPERSIM intègre des capacités de calcul parallèle [33]. Cependant, ces stratégies sont actuellement mises en œuvre pour la parallélisation en mémoire partagée, c'est-à-dire uniquement au sein d'un unique serveur de calcul haute performance. La stratégie d'évolution d'OPAL-RT Technologies et les besoins exprimés par ses clients nécessitent de dépasser cette limitation. L'objectif est de permettre la distribution d'une simulation EMT non plus sur un seul serveur, mais sur une flotte de plusieurs simulateurs ou serveurs de calcul interconnectés. Ce paradigme d'exécution vise à permettre la simulation de modèles plus grands et détaillés et faciliter le déploiement des simulations, par exemple sur des grappes de calcul potentiellement hétérogènes ou sur des infrastructures infonuagiques (*cloud*).

Au-delà de la capacité à gérer des grappes hétérogènes, une exigence opérationnelle formulée par OPAL-RT Technologies concerne la rapidité de la résolution du mappage lui-même. Cette étape est préalable au démarrage d'une simulation EMT et retarde le début de celle-ci, obligeant l'utilisateur à attendre le résultat du mappage avant de pouvoir démarrer la simulation. C'est seulement lorsque les calculs de la simulation seront terminés qu'il pourra finalement analyser ou vérifier les résultats de la simulation EMT. Considérant que l'attention et la concentration d'un utilisateur peuvent diminuer rapidement lors de l'attente d'une réponse d'un logiciel [34, p. 153], nous visons un temps de résolution de quelques secondes au maximum pour obtenir une solution de mappage de tâches. Les solutions nécessitant plus de 20 secondes seront considérées comme inadéquates pour un usage opérationnel.

En résumé, la problématique industrielle qui motive ce mémoire est le besoin de disposer d'une nouvelle stratégie d'assignation de tâches pour la simulation EMT en temps réel. Cette stratégie doit non seulement permettre la parallélisation efficace des simulations EMT sur des grappes de serveurs distribués et potentiellement hétérogènes, mais doit également calculer cette assignation rapidement pour s'intégrer de manière fluide dans le travail des utilisateurs du logiciel de simulations.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de développer et d'évaluer un cadre d'optimisation pour l'assignation des tâches de calcul parallèle en vue d'assurer les performances requises pour la simulation EMT en temps réel de réseaux électriques. Le modèle considérera les propriétés des tâches (charge de calcul, dépendances de communication) et les caractéristiques de l'infrastructure de calcul (grappes d'ordinateurs hétérogènes). L'objectif est de minimiser le temps d'exécution global de la simulation, déterminé par le temps de calcul sur les nœuds de la grappe et le temps de communication entre les noeuds.

Pour atteindre cet objectif principal, les sous-objectifs suivants sont définis :

- Développer des modèles d'optimisation pour le mappage des tâches de simulation sur une grappe hétérogène, en considérant explicitement les capacités de traitement variables des nœuds et les coûts de communication non uniformes.
- Adapter et évaluer des algorithmes de partitionnement et des techniques de recherche opérationnelle existantes pour résoudre efficacement les problèmes d'assignation modélisés dans le contexte des simulations EMT de réseaux électriques en temps réel.
- Illustrer, par des tests sur des configurations de grappes hétérogènes, la capacité des approches proposées à améliorer l'équilibrage de la charge de calcul et à réduire les communications internœuds, et ainsi à satisfaire les contraintes de performance en temps réel pour des réseaux de grande taille.

Ce mémoire de maitrise vise à fournir des solutions pratiques pour l'implémentation de simulations numériques EMT de réseaux électriques sur des infrastructures de calcul hétérogènes, en affinant les méthodes de partitionnement de graphes pour mieux correspondre aux exigences de ces simulations et en proposant des alternatives basées sur des modèles de recherche opérationnelle.

1.4 Contributions

Nos travaux présentent deux nouvelles formulations : le problème de semi-assignation quadratique avec goulot (BQSAP, *bottleneck quadratic semi-assignment problem*) et le problème de *bin packing* de taille variable avec contrainte quadratique de goulot pour les coûts de communication (Q-VSBPP, *variable-size bin packing problem with quadratic communication constraints*). Nos modèles améliorent les approches précédentes en intégrant des contraintes de goulot pour la phase de calcul et de communication de la simulation et en considérant les caractéristiques des grappes de calculs hétérogènes, créant ainsi une représentation plus précise des scénarios réels de mappage de tâches où les délais de communication affectent la performance [35].

Nos contributions étendent la littérature actuelle dans deux domaines principaux :

Modélisation

- Nous formulons le BQSAP qui établit un lien explicite entre le partitionnement de graphes et le QSAP avec une structure spécifique d'objectif de type goulot, non documentée précédemment dans la littérature existante consultée.
- Nous introduisons le Q-VSBPP visant à minimiser le délai de communication le plus long, qui impacte directement la performance en temps réel des simulations EMT, contrairement la somme des coûts de communication.
- Nous établissons une correspondance entre les propriétés des graphes de grande taille de type petit-monde et la structure des tâches des simulations EMT de réseaux électriques, et fournissons un cadre pour des stratégies efficaces de partitionnement de tâches.

Algorithmique

- Nous adaptons un algorithme de propagation d'étiquettes, basé sur la librairie `PuLP` [36], pour résoudre la formulation BQSAP en prenant en compte les caractéristiques de communication entre les partitions et leurs capacités.
- Nous proposons une extension de `PuLP` pour la formulation Q-VSBPP avec un goulot sur les coûts de communication en supprimant les contraintes qui fixaient le nombre de partitions, permettant à l'algorithme de minimiser le nombre de partitions.
- Nous évaluons les performances de la librairie `SCOTCH` [37] à la fois pour le BQSAP pour le mappage des tâches de simulation EMT sur des grappes hétérogènes et pour le Q-VSBPP sur des grappes homogènes entièrement connectées.
- Nous illustrons expérimentalement que la librairie `SCOTCH` et `PuLP` permettent d'obtenir des solutions de mappage de tâches dans des temps de résolution comparables et qui respectent les délais demandés. . De plus, la flexibilité de l'algorithme `PuLP` nous permet d'étendre ses fonctionnalités et de résoudre également le problème pour des grappes hétérogènes. La qualité des assignations est comparable pour les problèmes résolus avec les deux solveurs.

1.5 Plan du mémoire

La suite de ce mémoire s'organise de la manière suivante. Le chapitre 2 établit d'abord le contexte de notre mémoire en présentant une revue de la littérature sur le partitionnement de graphes, en particulier pour le mappage des tâches issues des simulations EMT. La section 2.4 détaille notamment des approches algorithmiques existantes pour résoudre ce problème, en analysant leurs principes et leurs limites. Ensuite, le chapitre 3 introduit les modèles mathématiques proposés pour une représentation plus fidèle du problème, intégrant notamment les coûts de communication et les contraintes de capacité. Nous détaillons les modifications spécifiques que nous avons apportées à l'heuristique `PuLP` afin d'implémenter nos modèles et de prendre en compte ces nouvelles considérations à la section 3.5. Finalement, le chapitre 4 présente l'évaluation expérimentale de notre approche avec la configuration des tests effectués, une analyse les résultats numériques obtenus et une comparaison de la performance de notre méthode par rapport aux approches conventionnelles. Une conclusion générale au chapitre 5 synthétise les contributions principales de ce mémoire et propose des pistes pour des recherches futures.

1.6 Diffusion des travaux

Les recherches menées dans le cadre de ce mémoire ont été diffusées à travers les communications suivantes :

- Présentation aux Journées de l'Optimisation, JOPT2025 : Une présentation des travaux a été effectuée le 12 mai 2025, aux Journées de l'Optimisation 2025 aux HEC Montréal, une conférence scientifique organisée annuellement par le GERAD et le CIRRELT.
- Article de conférence accepté : Un article de conférence basé sur ce mémoire a été revu par les pairs et accepté pour la conférence internationale *IEEE International Conference on Smart Grid Communications (SmartGridComm 2025)*.

  Une version préliminaire est accessible via *Les Cahiers du GERAD* :

  — Référence : J. Durette, G. Karabulut Kurt, and A. Lesage-Landry (Mai 2025). Task mapping strategies for electric power system simulations on heterogeneous clusters, Rapport technique, Les Cahiers du GERAD G–2025-32, GERAD, HEC Montréal, Canada.

  Disponible à : `https://www.gerad.ca/fr/papers/G-2025-32`

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de la littérature scientifique et technique sur le mappage de tâches. Nous y examinons les fondements théoriques, les principales approches algorithmiques afin de contextualiser le problème abordé dans ce mémoire.

2.1 Problème de mappage de tâche pour des simulations EMT

Le mappage des tâches, comme décrit à la section 1.1, est une composante centrale de la simulation EMT, en particulier lorsque celles-ci sont exécutées en parallèle pour analyser des systèmes électriques complexes en temps réel ou lorsque les ressources de calcul sont limitées. Cette sous-section examine les travaux antérieurs portant sur le mappage de tâches spécifiquement pour les simulations EMT, notamment en lien avec le simulateur HYPERSIM et d'autres outils similaires.

Des travaux récents, notamment dans le contexte du simulateur HYPERSIM, ont suggéré l'utilisation d'outils de partitionnement de graphes, comme la librairie `SCOTCH` [37], pour améliorer les performances par rapport aux approches antérieures telles que l'algorithme A-étoile [38]. Les études [33,39,40] s'inscrivent dans cette lignée. Plus spécifiquement, [39,40] ont validé l'efficacité des algorithmes de mappage basés sur `SCOTCH` sur des modèles de réseaux réalistes et industriels, incluant la simulation EMT du réseau de transport de France et de systèmes d'électroniques de puissance. Une conclusion notable de ces travaux est que, sur des architectures matérielles homogènes où les latences de communication ont un impact moindre, privilégier l'équilibrage de la charge de calcul peut s'avérer plus bénéfique que la minimisation stricte de la communication. Ces auteurs mentionnent également l'utilisation possible de méthodes de résolution exacte pour évaluer la qualité des solutions heuristiques, bien que ces méthodes exactes soient coûteuses en calcul pour les grands problèmes. Dans le même esprit, [33] détaille l'adaptation du simulateur temps réel HYPERSIM pour des simulations EMT massivement parallèles sur un supercalculateur homogène, soit le SGI UV300, en mettant en œuvre le mappage de tâches basé sur `SCOTCH`. Il est important de noter que ces développements et validations, bien qu'améliorant les temps de résolution, ont été réalisés en considérant des grappes de calcul homogènes et entièrement connectées.

L'étude [35] se concentre spécifiquement sur l'évaluation de la performance de différentes infrastructures de communication, utilisant la librairie de calcul parallèle par passage de messages `MPI` (*Message Passing Interface*) pour la simulation EMT parallèle hors ligne, où les contraintes de temps réel sont moins strictes. Pour ce faire, ils utilisent un grand modèle de réseau partitionné

en 172 tâches. L'accent est mis sur l'impact du coût de communication induit par les différentes infrastructures sur la performance globale de la simulation.

Une simulation temps réel d'un générateur éolien est présentée dans [41]. Dans ce cas, la parallélisation s'effectue toutefois au sein d'un unique serveur sur mémoire partagée : les composants du système de puissance sont simulés sur un processeur, tandis que le système de contrôle associé est simulé sur un autre processeur, le tout synchronisé au même pas de temps.

La comparaison entre une assignation statique et une assignation dynamique des tâches est abordée par [42] pour la parallélisation de simulations EMT sur un serveur multicœur. L'étude compare une assignation statique, déterminée avant exécution selon une charge estimée, à une assignation dynamique, gérée par la librairie pour le calcul parallèle en mémoire partagée `OpenMP` (*Open Multi-Processing*) durant l'exécution pour prendre en compte les tâches dont le temps de calcul varie en cours de simulations. Pour l'application testée, l'assignation statique s'est toutefois avérée plus performante.

2.2 Problèmes d'assignation et de *bin packing*

Le problème de mappage de tâches dans les simulations EMT s'inscrit dans le cadre plus large des problèmes d'optimisation combinatoire, notamment les problèmes d'assignation et de *bin packing*, qui ont fait l'objet de nombreuses recherches théoriques. La formulation fondamentale sur laquelle nous nous appuyons pour ce travail est présentée dans [43]. Le problème d'assignation quadratique (QAP), formulé pour l'optimisation de la localisation d'usines, assigne des installations à des emplacements et minimise à la fois le coût des terrains et la distance entre les installations. Ce modèle s'adapte bien à l'assignation de tâches distribuées car il prend en compte à la fois la partition et les coûts de transfert de données. Quant au problème de *bin packing*, les premiers travaux abordent le problème de la découpe de matières premières (le *cutting stock problem*), en se concentrant sur la minimisation des pertes lors de la découpe de matériaux à partir de rouleaux plus grands [44]. L'aspect important du *bin packing* est que le nombre de bacs, soit les partitions, doit également être minimisé, contrairement au problème d'assignation où le nombre de partitions est fixe.

Greenberg [45] étend le QAP au problème de semi-assignation quadratique (QSAP), permettant d'assigner plusieurs installations à un seul emplacement. Cette formulation est similaire à notre problème d'assignation où plusieurs tâches sont assignées à chaque partition ou ressource de calcul. Cependant, l'objectif dans [45] est la somme des coûts d'assignation, et non les coûts de goulots (*bottleneck*), qui sont plus pertinents dans le cadre de notre problème, comme présenté à la section 1.1.2. Diverses fonctions de coût, contraintes et approches de résolution de varia-

tions existantes sur les problèmes d'assignation sont explorées par [22] et [46], en soulignant les compromis entre optimalité et faisabilité computationnelle. En ce qui concerne le problème de *bin packing* [44] synthétise plusieurs revues présentant un grand nombre de variations du problème.

2.3 Algorithmes de mappage de tâches

L'assignation de tâches et le *bin packing* pour les simulations numériques sur grappes de calcul peuvent être abordés avec plusieurs approches algorithmiques. Les méthodes exactes, telles que celles discutées à la section 1.1.3 et aussi explorées dans [20], peuvent fournir des solutions optimales, mais sont coûteuses en calcul pour les problèmes de grande taille. Pour la taille des réseaux, les méthodes exactes nécessitent un temps de résolution prohibitif, d'où notre focalisation sur les algorithmes heuristiques pour cette étude. Les métaheuristiques offrent un compromis entre la qualité de la solution et le temps d'exécution. Par exemple, une approche comme une adaptation de la recherche tabou [47] a été appliquée à certaines variantes du problème d'assignation quadratique, sans toutefois le tester pour une variante similaire aux problèmes étudiés dans ce mémoire . Les méthodes d'apprentissage profond sont également actuellement explorées dans la littérature, telles que l'apprentissage supervisé des assignations à l'aide de différentes structures de réseaux de neurones [48–50]. Cependant, l'apprentissage profond nécessite des données d'entraînement et entraîne des coûts de calcul tant pour l'entraînement que pour l'inférence.

Les algorithmes dédiés aux graphes constituent une autre option pour résoudre les problèmes de mappage de tâches. Comparés aux métaheuristiques, ils offrent un avantage en termes de vitesse de résolution. Parce qu'ils sont conçus pour fonctionner sur des graphes, ils peuvent exploiter les dépendances entre tâches et les motifs de connectivité [15, 29, 32]. Les revues de la littérature [51–53] identifient des défis persistants dans le partitionnement de graphes, notamment la gestion de taille des poids des nœuds et des partitions très différentes, ce qui se rapproche du problème formulé dans ce mémoire, ainsi que la nécessité d'optimiser pour la performance des goulots plutôt que la minimisation de la coupe d'arêtes (*edge cut*) ou de la somme des coûts de partitionnement. Toutefois, comme l'indique [14, Sec. 2.1], bien que la métrique de la somme des arêtes coupées entre les partitions ne modélise pas exactement les communications dans les calculs numériques parallèles, elle s'avère bien performer en pratique dans les algorithmes de partitionnement pour des calculs parallèles d'équations différentielles.

2.4 Principes généraux des algorithmes sélectionnés

Cette section détaille les principes fondamentaux d'algorithmes de partitionnement de graphes existants qui ont été sélectionnés pour évaluation dans le cadre de ce mémoire, soient ceux des librairies `PuLP` (*Partitioning using Label Propagation*), développée Sandia National Laboratories [36] et `SCOTCH`, développée par le laboratoire Labri de INRIA [37]. Ces deux librairies représentent des exemples de paradigmes distincts dans le domaine du partitionnement de graphes, comme souligné par [51, 52] : `PuLP` s'inscrit dans la catégorie des méthodes basées sur la propagation d'étiquettes, tandis que `SCOTCH` est basé sur les méthodes multiniveaux [54] et de bipartitionnement [55].

L'analyse de ces deux paradigmes distincts d'algorithmes de partitionnement de graphes permet une évaluation comparative de leur adéquation respective aux objectifs d'optimisation et aux contraintes spécifiques de notre problème. Cette évaluation prend en compte les caractéristiques particulières du graphe de tâches issu de la simulation EMT ainsi que l'architecture matérielle de la grappe de calcul ciblée. Pour la librairie `SCOTCH`, nous évaluons ses performances en relation avec les modèles de mappage de tâches du problème étudié. Pour la librairie PuLP, nous adaptons l'algorithme pour les contraintes particulières de la formulation des modèles de notre problème. Cette section sert de complément à la documentation officielle de ces librairies et établit une base nécessaire pour comprendre l'évaluation des algorithmes de mappage effectuée dans ce mémoire ainsi que et les adaptations présentées à la section 3.5.

2.4.1 Présentation de la librairie `SCOTCH`

La librairie `Scotch` est développée par le Laboratoire bordelais de recherche en informatique (LaBRI), qui est aujourd'hui intégré à INRIA Bordeaux [37]. Comme l'explique le professeur Pellegrini dans [56], le projet a été initié en 1992 afin de traiter le problème du placement de tâches communicantes sur les processeurs d'architectures parallèles. L'objectif principal est de minimiser les communications interprocesseurs et la congestion du réseau, en réponse au besoin de méthodes plus rapides et efficaces que les approches antérieures souvent de complexité élevée. À cette époque, l'équipe du LaBRI développait un environnement pour une grappe d'ordinateurs avec des interconnexions à structure pyramidale demandant un outil de placement adéquat pour minimiser les communications interprocesseurs et la congestion.

2.4.1.1 Principes et algorithmes de la librairie SCOTCH

La méthodologie de SCOTCH repose sur la stratégie *diviser pour régner*, appliquée au problème de partitionnement de graphe. Les travaux initiaux qui ont mené à son développement étaient centrés sur l'algorithme de partitionnement dual récursif (*dual recursive bipartitioning*).

Une version parallèle de la librairie SCOTCH, nommée PTSCOTCH, est également développée par le LaBRI. Cette version permet d'effectuer le partitionnement en parallèle, ce qui est particulièrement utile pour traiter des graphes de très grande taille qui excéderaient les limites de la mémoire d'un seul ordinateur. Toutefois, cette version parallèle n'a pas été considérée pour l'application présente, car les temps de résolution observés et les besoins en stockage mémoire pour les graphes étudiés ne le nécessitaient pas.

La librairie SCOTCH permet d'utiliser différentes méthodes de partitionnement de graphes de manière cumulative. Le choix des méthodes les plus appropriées s'effectue en fonction des critères de performance spécifiques requis par l'application, par exemple, qualité de la coupe versus rapidité du partitionnement. Les méthodes de partitionnement comprennent les méthodes multiniveaux et des techniques d'affinage local dites *k-way*. Une seconde famille est celle des méthodes basées sur le partitionnement dual récursif. Celle-ci s'appuie sur un cadre multiniveau utilisant différents algorithmes de bipartitionnement. Parmi ceux-ci figurent une version de l'algorithme de Fiduccia-Mattheyses (FM), un algorithme basé sur la diffusion, une méthode de croissance de graphe gloutonne qui relève de la famille des procédures GRASP (*greedy randomized adaptive search procedure*) [57], ainsi qu'un algorithme d'affinage glouton dit *exactifiant* (*greedy exactifying refinement algorithm*) [37].

Dans la suite de la section, nous décrivons les principes généraux de quelques-unes des méthodes de partitionnement implémentées dans la librairie SCOTCH. Pour plus d'information, le lecteur est invité à consulter [58, 59].

Algorithme Fiduccia-Mattheyses (FM) Son fonctionnement est itératif et se déroule selon les étapes suivantes [55] :

- Initialisation : une partition initiale est générée, qui peut être aléatoire ou venant d'un autre algorithme utilisé préalablement.
- Calcul des gains : Pour chaque nœud libre (non encore déplacée dans la passe courante), son *gain* est calculé, qui représente la réduction du nombre de connexions coupées si ce nœud était déplacé dans l'autre partition.
- Sélection et déplacement : À chaque passe, le nœud non assigné ayant le gain le plus élevé est sélectionné, tout en s'assurant que son déplacement ne viole pas la contrainte d'équilibre

au-delà d'une certaine tolérance. Pour sélectionner rapidement le nœud de meilleur gain et mettre à jour les gains des voisins après un déplacement, FM utilise des listes chaînées triées par gain.

- Verrouillage : Une fois un nœud déplacé, il est verrouillé dans sa nouvelle partition et ne peut plus être déplacé durant cette passe. Cela évite les cycles infinis.
- Mémorisation : L'algorithme continue de déplacer les nœuds un par un, même si les gains deviennent négatifs. Cette étape permet à l'algorithme de Fidduccia-Mattheyses de sortir des minima locaux en poursuivant la recherche locale même lorsque le déplacement dégrade la solution. L'algorithme mémorise l'état de la partition qui a donné le meilleur résultat, soit le minimum de coupures, au cours de la passe et pourra y revenir pour la solution finale.
- Fin de passe : La passe se termine quand tous les nœuds sont verrouillés ou qu'aucun mouvement respectant l'équilibre n'est possible. L'algorithme retourne à la meilleure partition rencontrée durant cette passe.
- Répétition : Plusieurs passes successives sont effectuées jusqu'à ce qu'une passe n'apporte plus d'amélioration du nombre de coupures.

Dans la librairie SCOTCH, pour adapter l'algorithme FM au partitionnement de graphes dans le cadre du mappage de tâches sur des architectures parallèles, des poids sur les arêtes sont introduits pour représenter les volumes de communication et le coût des communications, soit les arêtes coupées, est pris en compte en fonction de la distance entre les processeurs cibles [37]. Cette adaptation engendre une plage de valeurs de gains potentiellement très étendus et variables, invalidant l'hypothèse de gains limités de l'algorithme FM original et rendant impraticable une indexation linéaire directe via une liste chaînée triée par gain. Une indexation logarithmique des gains pour la liste chaînée est utilisée : au lieu d'un seau par valeur de gain, les gains sont regroupés dans des seaux couvrant des intervalles de taille logarithmiquement croissante. Cette technique permet de gérer efficacement la large dynamique des gains avec une structure de taille raisonnable, tout en préservant la complexité temporelle quasi-linéaire par passe de l'algorithme FM et en assurant une qualité de partitionnement comparable à celle d'une indexation linéaire, l'approximation induite étant du même ordre que l'approximation inhérente à FM [60]. Pour plus d'information sur l'algorithme Fidducia-Mattheyses, le lecteur est invité à consulter [61, 62].

Algorithme de *greedy graph growing partitioning* Il s'agit d'un algorithme de bipartitionnement inspiré par [54]. Cet algorithme divise un graphe en deux en partant d'une tâche choisie au hasard. Il construit ensuite progressivement une partition en y agrégeant les tâches voisines sélectionnées de manière à minimiser l'augmentation du coût de communication. Ce processus de croissance continue jusqu'à ce que les deux partitions atteignent un équilibre de taille. Pour améliorer la qualité de la solution, l'algorithme complet est exécuté plusieurs fois depuis des points de départ différent (*restart method*) et seule la meilleure partition trouvée est conservée.

Méthode multiniveau `SCOTCH` utilise aussi l'approche de partitionnement multiniveaux, un concept popularisé par [54]. Cette stratégie consiste d'abord à réduire la taille du graphe en fusionnant des groupes de tâches lors d'une étape dite de *coarsening* ou contraction. Ensuite, un partitionnement initial est effectué sur ce graphe simplifié de plus petite taille. Finalement, le partitionnement est projeté sur le graphe original, et une phase d'ajustement ou de raffinement est appliquée pour améliorer la qualité de la coupe tout en respectant l'équilibre entre les partitions finales. Pour des détails supplémentaires, le lecteur est invité à consulter [51, 52] et [63] qui présente une description détaillée du mécanisme de partitionnement multiniveaux.

2.4.1.2 Performance du code de la librairie `SCOTCH`

La librairie `SCOTCH` est implémentée en `C`. Le document [58, Sec. 5] répertorie les stratégies de programmation, telles que l'usage restreint de branches conditionnelles et la localité des données, expliquant les vitesses d'exécution obtenues avec la librairie.

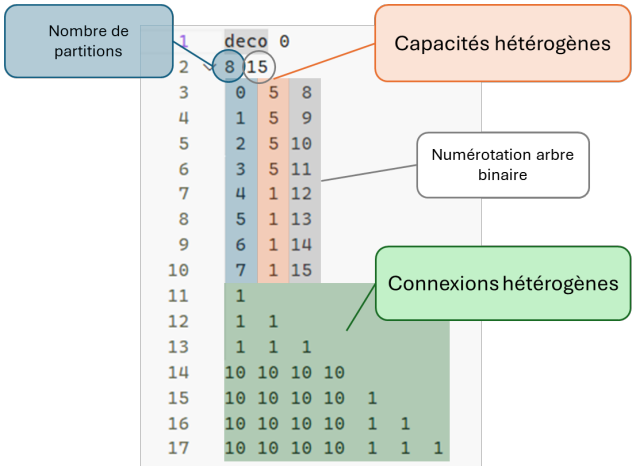
2.4.1.3 Fonctionnalités et usages de la librairie `SCOTCH` pour le problème étudié

La librairie `SCOTCH` fournit deux alternatives pour appeler les fonctions de partitionnement, soit en appelant les routines de la librairie dans un code en `C` grâce au fichier `scotch.h`, soit en appelant les exécutables dans une ligne de commande Linux ou un script `Bash`. Pour les tests présentés dans ce mémoire, l'option des programmes en ligne de commande est plus simple d'utilisation. Comme elle ne demande pas de compilation, elle permet un travail et des itérations plus rapides. Nous listons ci-dessous les fonctionnalités des exécutables en lien avec le mappage de tâches des simulations EMT.

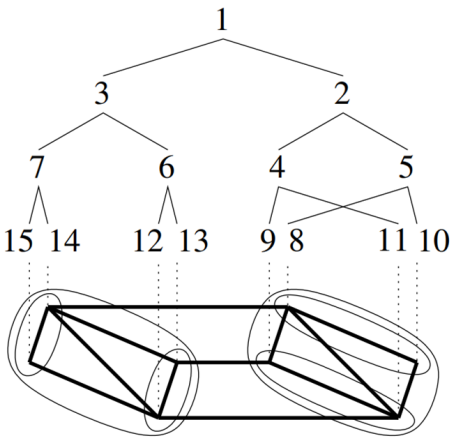
Pour l'assignation des tâches L'exécution est démarrée avec le programme `gmap`. Les données décrivant une grappe de calcul sont inscrites dans un fichier nommé `deco 0`. Le code 2.1 montre un exemple et la figure 2.1 en identifie les parties. Le fichier comprend un en-tête, le nombre de processeurs ou partitions disponibles et la taille de l'arbre binaire associé, c'est-à-dire qui possède autant de feuilles que de processeurs dans la grappe de calcul. L'arbre binaire est utilisé par l'algorithme pour la méthode multiniveau et le fichier `deco 0`, dérivé de l'arbre binaire, montre les temps de communication. À noter que la décomposition multiniveau illustrée et les liens de communication ne sont pas directement reliés, on peut par exemple avoir des temps de communication faible entre des feuilles qui sont éloignées dans l'arbre. . De plus, le fichier liste le numéro de tous les processeurs de la grappe, auxquels est associée la capacité de calcul ainsi que la numérotation associée dans l'arbre binaire. Finalement, les connexions entre les processeurs sont listées à l'aide de la partie triangulaire inférieure de la matrice d'adjacence. Les coefficients représentent la latence de la connexion, un coefficient plus grand signifie que le coût de l'envoi des données est plus élevé [64, p. 7].

CODE 2.1 Exemple du fichier `deco 0`, fichier d'entrée natif de la librairie `SCOTCH`, définisant une architecture d'une grappe de 8 nœuds de capacités et connexions hétérogènes.

```
1  deco 0
2  8 15
3  0 5 8
4  1 5 9
5  2 5 10
6  3 5 11
7  4 1 12
8  5 1 13
9  6 1 14
10 7 1 15
11  1
12  1 1
13  1 1 1
14 10 10 10 10
15 10 10 10 10 1
16 10 10 10 10 1 1
17 10 10 10 10 1 1 1
```



(a) Parties du fichier `deco 0`.



(b) Arbre binaire numérotant les nœuds de calcul, pour la méthode multiniveau (source : Pellegrini [37]).

FIGURE 2.1 Explication du fichier d'entrée natif de la librairie SCOTCH (montré dans le code 2.1), définissant l'architecture d'une grappe de 8 nœuds de capacités et connexions hétérogènes. L'arbre binaire est utilisé par les algorithmes de la librairie SCOTCH. Les connexions entre les processeurs sont indiquées avec la partie triangulaire inférieure de la matrice d'adjacence, pour le graphe représentant les connexions de la grappe de calcul.

Pour le *bin packing* Le programme `gmap` permet aussi de minimiser le nombre de partitions en utilisant les options `gmap -cr -q<pwght>`. Cette fonctionnalité est nommée *clustering for variable-sized architecture* dans la documentation de la librairie [37]. L'utilisation de l'algorithme de double partitionnement récursif avec l'option `-cr` est le seul permettant du *bin packing*. La capacité maximale des partitions est indiquée avec l'option `-q`. Par exemple, `-q50` indique une capacité maximale de 50 *unités* par partition, où l'unité est la même que celle utilisée pour définir la taille des tâches dans le fichier d'entrée du graphe de tâches. Pour l'architecture de la grappe de calcul, seulement les architectures de grappes homogènes complètement connectées ou connectées en hypercube sont disponibles dans SCOTCH au moment de la rédaction de ce mémoire [37]. Par exemple, une architecture entièrement connectée de 18 processeurs homogènes peut être représentée par le fichier d'une ligne, comme montré par le code 2.2.

CODE 2.2 Exemple de fichier d'entrée natif de la librairie SCOTCH, définissant une architecture de grappe de calcul de taille variable avec 18 nœuds disponibles, homogènes et entièrement connectés.

```
1 varcmplt 18
```

2.4.2 Algorithme de partitionnement par propagation d'étiquettes sur un graphe, `PuLP`

L'algorithme `PuLP` a été développé au Sandia National Laboratory [36] dans l'objectif de fournir un algorithme performant dédié au partitionnement de graphes de type petit-monde. L'algorithme permet de résoudre un problème de partitionnement multiobjectif et multicontrainte, ce qui correspond à la forme générale du problème d'optimisation d'assignation quadratique. Le code est sous licence ouverte et est disponible sur Github [36].

Entrées et objectifs de l'algorithme L'algorithme de `PuLP` prend en entrée un graphe non orienté, ou orienté et symétrique. L'objectif principal est de diviser l'ensemble des sommets en un nombre prédéfini de sous-ensembles disjoints appelés partitions. Chaque partition est associée une étiquette unique pour l'identifier.

Dans le contexte applicatif, ce graphe peut modéliser les communications qui existent entre différentes tâches. La qualité d'une partition est évaluée selon plusieurs critères d'équilibre. Premièrement, l'équilibre de la charge en termes de sommets : le nombre de sommets dans chaque partition doit rester proche de la taille moyenne, calculée par le nombre total de sommets divisé par le nombre de partitions. Plus précisément, la taille de chaque partition est contrainte à se situer dans une fourchette définie par des facteurs de tolérance d'équilibrage par rapport à cette taille moyenne.

Deuxièmement, un critère d'équilibre évalue les arêtes internes à chaque partition. Le nombre d'arêtes reliant des sommets appartenant à la même partition, représentant la communication interne, par exemple via mémoire partagée sur un même processeur par exemple, doit rester inférieur à un seuil. Ce seuil est défini par rapport au nombre moyen d'arêtes par partition et un facteur de tolérance supérieur.

Un objectif central du partitionnement est de minimiser la communication entre les partitions. La coupe (*edge cut*) est définie comme l'ensemble des arêtes qui relient des sommets appartenant à des partitions différentes. Ces arêtes représentent les communications qui doivent transiter entre les processeurs. Le balancement vise alors à minimiser les deux indicateurs liés une coupe : d'une part, le nombre total d'arêtes dans la coupe (*total edge cut*) et d'autre part, le nombre maximal d'arêtes de la coupe connectées à une seule partition (*maximal per-partition edge cut*). Minimiser ces valeurs revient à réduire à la fois le volume global de communication interprocesseur et le pic de communication pour la partition la plus sollicitée.

2.4.2.1 Fonctionnement de l'algorithme

L'algorithme `PuLP` est divisé en trois phases : (i) initialisation, (ii) l'équilibrage des sommets et (iii) l'équilibrage des arêtes. Dans chacune des phases, l'algorithme itère sur tous les sommets pour décider si le sommet doit changer de partition.

(i) Phase d'initialisation du partitionnement La phase d'initialisation débute avec une répartition aléatoire des étiquettes. L'étiquette représente la partition à laquelle le sommet est associé. Le poids de chaque partition `part_sizes` est ensuite calculé avec la somme des sommets assignés à chaque partition. Le reste de la phase d'initialisation consiste en une boucle de propagation d'étiquettes qui itère sur tous les sommets du graphe. Les actions suivantes sont effectuées dans la boucle :

- Pour chaque sommet, un calcul est effectué pour évaluer l'influence des sommets voisins sur le sommet actuel `part_counts`. À la fin de la boucle, le sommet sera transféré sur la partition ayant la plus grande *influence*. Pour ce faire, l'algorithme récupère le degré du sommet étudié, la liste des sommets voisins ainsi que leur degré respectif et le poids des arêtes sortantes reliant le sommet à chacun de ses voisins. Pour chacun des voisins, l'algorithme calcule le produit du degré du voisin et du poids de l'arête et ajoute cette valeur à `part_counts` au compte de la partition de ce voisin. Cela permet de mesurer l'influence de la partition du voisin sur le sommet, pondérée par le nombre de connexions et le poids de chacune des connexions, qui peut représenter par exemple la quantité de données transmises sur cette connexion.
- La partition ayant l'influence maximale est ensuite identifiée. Si plusieurs partitions sont d'influence égale, l'une d'elles est choisie au hasard selon une distribution uniforme sur le nombre total de partitions.
- Finalement, le sommet est transféré dans la partition d'influence maximale, sauf si cela diminue la partition en deçà du seuil minimal pour la taille des partitions. Dans la phase d'initialisation, le seuil minimal est fixé à 25% de la taille moyenne des partitions.

Pour expliquer l'algorithme à l'aide d'un exemple illustré à la figure 2.2, prenons un sommet v dans la partition 0 ayant quatre voisins, dans les partitions 1, 2 et 3 ainsi qu'un autre sommet voisin dans sa partition 0, comme à la figure 2.2. Le sommet v communique quatre données avec son voisin de la partition 1 et seulement une donnée les autres voisins des autres partitions. Le voisin de la partition 1 est aussi connecté à 3 autres sommets alors que les deux autres voisins n'ont aucune autre connexion. Il serait préférable que le sommet v soit placé dans la partition 1 afin de minimiser les communications entre les partitions. Et lorsque les deux autres voisins du

sommet v seront considérés à leur tour, ils seront eux aussi transférés dans la partition 1, si cela ne contrevient pas à aucune contrainte.

FIGURE 2.2 Exemple illustrant le concept de propagation d'étiquettes dans un graphe. Le sommet v est ajouté à la partition 1 ce qui réduit les communications entre les partitions 0 et 1.

(ii) Phase équilibrage des sommets La phase d'équilibrage des sommets reprend le fonctionnement de la phase d'initialisation en ajoutant des contraintes.

- Dans la première itération, le poids de la partition est utilisé comme un multiplicateur du compte `part_count` pour rendre le transfert plus attrayant. Toutefois, si la partition dépasse la contrainte de balancement pour le poids des sommets, l'influence de cette partition est mise à zéro afin que le sommet ne puisse pas y être transféré.
- Le sommet est transféré vers la partition ayant la plus grande influence.
- Lorsque tous les sommets du graphe ont été analysés, une deuxième itération de raffinement est effectuée. Cette deuxième itération raffine les critères de transfert et le sommet est transféré à la partition ayant la plus grande influence seulement si le transfert respecte la contrainte de balancement des sommets.

(iii) Phase équilibrage des arêtes Dans la phase d'équilibrage des arêtes, l'algorithme a deux objectifs : balancer les arêtes à l'intérieur de chaque partition et minimiser la coupe maximale entre les partitions. La coupe est la mesure du nombre d'arêtes passant d'une partition à une autre, ou qui ont été coupées par le partitionnement. Similairement avec la phase d'équilibrage des sommets, deux itérations sur les arêtes sont effectuées, avec la deuxième où le transfert du sommet s'effectue seulement si les ratios de balancement pour les sommets et les arêtes s'améliorent avec ce changement.

2.5 Sommaire de la revue de littérature

Cette revue de la littérature établit le contexte du problème du mappage de tâches de simulations EMT parallèles, en situant ce problème dans le cadre plus général des problèmes d'assignation quadratique et de *bin packing*. L'analyse des travaux existants révèle que les algorithmes heuristiques de partitionnement de graphes constituent l'approche privilégiée, compte tenu des limitations computationnelles des méthodes exactes pour les instances de grande taille. Les stratégies multiniveaux, implémentées notamment dans la librairie `SCOTCH`, et les méthodes par propagation d'étiquettes, telles que celle de `PuLP`, représentent des paradigmes pertinents qui ont été examinés. Cette discussion fournit un contexte au travail de modélisation mathématique et aux modifications algorithmiques proposées dans les prochains chapitres du mémoire.

CHAPITRE 3 MODÉLISATION MATHÉMATIQUE ET IMPLÉMENTATION

Dans ce chapitre, nous présentons les modèles mathématiques adaptés pour le mappage de tâches de simulations EMT.

3.1 Notation de modélisation mathématique

Nous utilisons la notation suivante dans nos modèles. Soit $\mathcal{T} = \{1, ..., t\}$ l'ensemble des tâches et $\mathcal{P} = \{1, ..., p\}$ l'ensemble des partitions ou les nœuds de la grappe, avec $t, p \in \mathbb{N}$ étant respectivement le nombre de tâches et le nombre de partitions. Notons que les termes *partition* et *nœud* sont utilisés de façon équivalente, la partition désignant l'aspect logique du regroupement de tâches et le nœud son support physique d'exécution sur la grappe d'ordinateurs.

Caractérisation des tâches : Comme présenté à la section 1.1.4, chaque tâche $i \in \mathcal{T}$ est caractérisée par deux paramètres principaux :

- son temps de calcul estimé $e_i > 0$ sur un nœud de référence, exprimé en microsecondes ;
- le volume de données $a_{ij} > 0$ transmis entre les tâches i et j, exprimé en quantité de `Doubles`, soit 64 bits par valeur selon la norme IEEE 754 [28].

À partir de ces paramètres de base, nous dérivons des mesures d'ensembles pour caractériser l'ensemble des tâches $i \in \mathcal{T}$. Pour les temps de calcul, nous définissons le temps de calcul maximal $e_{\max} = \max_{i \in \mathcal{T}} e_i$ ainsi que le temps moyen $\bar{e} = \frac{1}{t} \sum_{i \in \mathcal{T}} e_i$. Pour les données transmises entre les tâches, le nombre total de paires de tâches communicantes est donné par la pseudonorme ℓ_0, $\|A\|_0$, c'est-à-dire le nombre d'éléments non nuls dans la matrice de communication $A = (a_{ij})_{i,j \in \mathcal{T}}$. Cette métrique représente le nombre d'arêtes dans le graphe de tâches de calcul.

Caractérisation de la grappe : La grappe d'ordinateurs est caractérisée par des paramètres de communication et de capacité de calcul.

Pour la communication entre les nœuds k et l, nous modélisons le temps de transmission par une relation linéaire composée d'un délai $b_{kl} > 0$, exprimé en microsecondes par `Double` transmis, et d'une latence $c_{kl} > 0$, exprimée en microsecondes. Cette modélisation linéaire est validée expérimentalement [65] (section 4.1.1) pour des quantités transmises inférieures à 1000 `Doubles`. Notons aussi que nous considérons un contexte où la grappe est réservée exclusivement pour l'exécution de la simulation parallèle EMT.

Pour le calcul, chaque nœud $k \in \mathcal{P}$ possède un facteur de capacité $\rho_k > 0$ (adimensionnel) qui

modifie proportionnellement le temps de calcul des tâches par rapport au nœud de référence, reflétant les capacités de calcul comme la fréquence de l'horloge du processeur.

Variables de décision : L'assignation d'une tâche $i \in \mathcal{T}$ sur un nœud $k \in \mathcal{P}$ est représentée par une variable binaire $x_{ik} \in \{0, 1\}$.

Dans la section suivante, nous décrivons les modèles proposés en utilisant cette notation.

3.2 Modèle de problème de semi-assignation quadratique avec goulot (BQSAP)

Soit $y > 0$ le goulot sur le temps de calcul, exprimé en microsecondes, et $z > 0$ le goulot sur le temps de communication, également exprimé en microsecondes. La formulation du problème d'optimisation de semi-assignation quadratique avec goulot (*bottleneck quadratic semi-assignment problem*, BQSAP) est :

$$\min_{y,z,x_{ik}} \quad y + z \tag{3.1a}$$

$$\text{s.t.} \quad y \geq \rho_k \sum_{i \in \mathcal{T}} e_i \, x_{ik} \quad k \in \mathcal{P}, \tag{3.1b}$$

$$z \geq b_{kl} \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{T}} a_{ij} \, x_{ik} x_{jl} + c_{kl}, \quad \forall k, l \in \mathcal{P}, \tag{3.1c}$$

$$\sum_{k \in \mathcal{P}} x_{ik} = 1, \quad i \in \mathcal{T}, \tag{3.1d}$$

$$x_{ik} \in \{0, 1\}, \quad i \in \mathcal{T}, k \in \mathcal{P}. \tag{3.1e}$$

où

- l'objectif (3.1a) vise à minimiser le temps total de goulots en combinant les délais de calcul et de communication, exprimés en microsecondes ;
- le goulot en temps de calcul (3.1b) garantit que le temps minimum y représente le temps de calcul total sur chaque nœud k en tenant compte du facteur de capacité du nœud ρ_k ;
- le goulot en temps de communication (3.1c) calcule le délai de communication total en fonction du volume de données transmises et des latences entre les nœuds où les tâches sont assignées. Cette contrainte présente la structure quadratique classique de problème d'assignation [22,46] où le terme $x_{ik}x_{jl}$ modélise l'interaction entre l'assignation de la tâche i au nœud k et l'assignation de la tâche j au nœud l. Cette formulation capture précisément le temps nécessaire pour transmettre les données entre les tâches i et j lorsqu'elles sont respectivement assignées aux nœuds k et l ;
- la contrainte (3.1d) garantit que chaque tâche est assignée à exactement une partition ;

- et finalement, (3.1e) contraint les assignations de tâches à être binaires (assignées ou non assignées), préservant ainsi l'intégrité du mappage.

En somme, cette formulation crée un problème d'optimisation qui tient compte à la fois des goulots en temps de calcul et de communication tout en garantissant des assignations de tâches valides sur l'ensemble du système distribué.

3.3 Modèle de *bin packing* de taille variable et contrainte quadratique de communication (Q-VSBPP)

Soit $Y_{\max} > 0$ le temps de calcul maximal permis sur une partition, exprimé en microsecondes, et $z > 0$ le goulot sur le temps de la phase de communication, également exprimé en microsecondes. La formulation du problème d'optimisation de *bin packing* de taille variable avec contrainte quadratique de goulot pour les coûts de communication (*variable-size bin packing problem with quadratic communication constraints*, Q-VSBPP) est :

$$\min_{z, x_{ik}} \; z \tag{3.2a}$$

$$\text{s.t.} \;\; Y_{\max} \geq \rho_k \sum_{i \in \mathcal{T}} e_i x_{ik}, \quad k \in \mathcal{P}, \tag{3.2b}$$

$$z \geq b_{kl} \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{T}} a_{ij}\, x_{ik} x_{jl} + c_{kl}, \quad \forall k, l \in \mathcal{P}, \tag{3.2c}$$

$$\sum_{k \in \mathcal{P}} x_{ik} = 1, \quad i \in \mathcal{T}, \tag{3.2d}$$

$$x_{ik} \in \{0, 1\}, \quad \forall i \in \mathcal{T}, k \in \mathcal{P}. \tag{3.2e}$$

où

- l'objectif (3.2a) vise à minimiser le goulot en temps de communication z ;
- la contrainte (3.2b) impose que le temps de calcul maximal en tenant compte des facteurs de capacité du nœud ρ_k
- la contrainte (3.2c) conserve la structure quadratique [22] pour modéliser le goulot de communication ;
- et comme dans le modèle précédent, les contraintes (3.2d) et (3.2e) garantissent que chaque tâche est assignée à exactement une partition et que les assignations de tâches sont binaires (assignées ou non assignées), préservant ainsi l'intégrité du mappage.

Ce modèle représente le problème d'assignation des tâches Q-VSBPP où l'objectif est de trouver la répartition optimale des tâches entre les partitions tout en respectant un temps de calcul maximal par partition et en minimisant les coûts de communication. Notons quelques caractéristiques importantes de ce modèle. L'objectif de minimisation du goulot en temps de communication z, sous

une contrainte de temps de calcul maximal $Y_{\max}$ par partition, incite naturellement au regroupement des tâches sur un nombre plus restreint de partitions. En effet, si cette contrainte de temps de calcul était relâchée, $Y_{\max} \to \infty$, la solution optimale consisterait à assigner l'ensemble des tâches à une unique partition, ce qui annulerait le temps de communication, $z = 0$, et minimiserait le nombre de partitions utilisées. Finalement, la modélisation du délai de communication (3.2c) introduit une composante quadratique, car le coût dépend des paires d'assignations de tâches i et j aux partitions k et l, représentées par le terme $x_{ik}x_{jl}$, similairement au modèle précédent. Cette structure est analogue à celle du problème d'assignation quadratique (QAP) tel que formulé originalement par Lawler [43].

3.4 Utilisation de `SCOTCH` pour les problèmes d'assignation et de *bin packing*

Bien que `SCOTCH` contienne déjà les fonctionnalités pour résoudre le problème d'assignation pour les grappes hétérogènes, le paramétrage n'avait pas encore été effectué dans le simulateur EMT HYPERSIM et la revue de littérature ne faisait pas état de l'étude de ce type de configuration de grappes (section 2.1). Nous avons donc configuré les paramètres disponibles pour vérifier que la librairie permettait effectivement d'obtenir des résultats satisfaisants avec un cluster hétérogène pour le problème d'assignation, fournissant une solution de mappage respectant les contraintes de la simulation en temps réel. Les détails de la librairie sont présentés dans la section 2.4.1.3.

Pour le problème de *bin packing*, `SCOTCH` n'offre pas la configuration avec grappe hétérogène, mais nous avons tout de même effectué les tests avec grappe homogène à des fins comparatives.

3.5 Adaptation de la librairie `PuLP` pour le problème du mappage de tâches

Les contributions apportées à l'algorithme original de partitionnement par propagation d'étiquette `PuLP` (*Partitioning using Label Propagation*) [36] sont disponibles sur Github [1] et sont présentées dans cette section.

1. https://github.com/julie9/PuLP-QSAP

3.5.1 Algorithmes de la librairie `PuLP` adaptée

La librairie `PuLP` originale a été adaptée pour traité le problème de mappage de tâches considéré. Les pseudo-codes ci-après décrivent les phases d'initialisation et de balancement des sommets de l'algorithme résultant. L'algorithme 1 détaille la phase d'initialisation à laquelle nous avons ajouté la prise en compte de la capacité hétérogène des partitions, soit le terme ρ_k de (3.1b) et (3.2b). L'algorithme 2 montre la phase de balancement des sommets où nous avons ajouté, en plus des capacités hétérogènes, les coûts de communication entre les partitions, soit le terme b_{kl} de (3.1c) et (3.2c). Nous avons aussi intégré une borne sur la capacité maximale, soit le terme $Y_{\max}$ de (3.2b), ainsi qu'une contrainte permettant de vider une partition dans le cas du problème de *bin packing* de (3.2). Une troisième phase de balancement des arêtes peut aussi être activée, mais n'a pas été requise. Nous avons donc utilisé les deux premières phases uniquement. Les ajouts sont décrits plus en détail dans les sections suivantes.

Algorithme 1 Pseudo-code de la phase d'initialisation de l'algorithme `PuLP` de partitionnement de graphe. *Les modifications apportées à l'algorithme original sont soulignées.*

entrée: Graphe `g`, nombre de partitions `p`, facteur d'équilibrage `vertex_balance`
sortie: Liste `partition[]` indiquant la partition de chaque sommet

```
1:  Calculer capacité maximale par partition selon vertex_balance
2:  Initialiser partition[] ← −1
3:  pour chaque partition p :
4:      Assigner un sommet aléatoire à la partition p
5:  pour chaque sommet i :
6:      p ← partition[i]
7:      pour chaque voisin j non-assigné de i :
8:          si capacité restante de p ≥ poids de j :
9:              partition[j] ← p
10:         sinon si capacité restante de p+1 ≥ poids de j :
11:             partition[j] ← p
12:         sinon
13:             partition[j] ← partition aléatoire disponible
14:         Mettre à jour de la capacité de p et des structures partagées pour OpenMP
15: pour chaque sommet i non-assigné :
16:     Essayer N fois de trouver partition non-pleine
17:     Sinon, assigner à partition aléatoire
```

Algorithme 2 Pseudo-code des phases d'équilibrage et de raffinement de l'algorithme PuLP de partitionnement de graphe. *Les modifications apportées à l'algorithme original sont soulignées.*

entrée: Graphe g, `resultats[]`, `iterations`, `isBinPacking`, `vertex_balance`
sortie: Partitionnement optimisé `resultats[]` du graphe avec équilibrage

```
Calculer capacité max par partition selon vertex_balance
tant que itération externe non atteinte :

    /*** Phase d'équilibrage ***/
    tant que critère d'équilibrage non satisfait :

        /* Calculer les scores de migration pour chaque partition */
        pour chaque sommet i :
            p ← partition[i]
            Initialiser part_counts[] ← 0
            pour chaque voisin j de i :
                poids ← poids de l'arête(i, j)
                part_counts[partition_j] += degre_j × poids

            /* Ajuster selon les poids de communication inter-partitions */
            comm_weights_p[] ← get_comm_weights(p)
            pour chaque partition q :
                part_counts[q] /= comm_weights_p[q]
            meilleure ← partition avec score max

            /* Vérifier les contraintes de migration */
            si migration de i vers meilleure viderait p et mode ≠ BinPacking :
                continue
            sinon si capacité meilleure insuffisante :
                continue

            /* Effectuer la migration */
            partition[i] ← meilleure
            Mettre à jour capacités et structures OpenMP

    /*** Phase de raffinement ***/
    Appliquer optimisations locales
```

3.5.2 Ajout du coût de communication

L'algorithme PuLP original considère uniformément les relations entre toutes les partitions dans le calcul du coût des coupes. Toutefois, le problème d'assignation quadratique prend en compte le coût de communication associé à chaque pair de partitions, qui permet de représenter la vitesse de communication. L'algorithme PuLP original considère une matrice pour les coûts de communication b_{kl} décrit en (3.1) et (3.2). En ajoutant l'utilisation des poids de communication dans l'algorithme, PuLP permet de résoudre le problème d'optimisation BQSAP.

Le code 3.1 ci-dessous illustre comment ces poids de communication sont utilisés dans la fonction de balance des sommets `label_balance_verts`. Le score `part_counts[p]` de chaque partition voisine `p` est multiplié par le coût de communication spécifique `partition_comm_weights[p]` entre `part` et `p`. Cela a pour effet de moduler l'importance des voisins en fonction de la difficulté ou du coût à communiquer avec cette partition. Ce poids sera ensuite utilisé pour décider si un voisin est ajouté à la partition actuelle.

CODE 3.1 Extrait montrant des poids de communication.

```
// Dans pulp.h :
#define out_interpart_weights(g, p, num_part) &g.interpartition_weights[p *
    num_part]

// Dans label_balance_verts.cpp :
// Retrieve communication weights for the current partition 'part'
int* partition_comm_weights = out_interpart_weights(g, part, num_parts);
for (int p = 0; p < num_parts; ++p)
    part_counts[p] *= partition_comm_weights[p];
    /*[...]*/
```

3.5.3 Ajout des capacités hétérogènes des partitions

Les capacités des partitions sont exprimées en référence avec une partition de référence. Chaque partition peut avoir une capacité différente, qui est enregistrée dans la liste `avg_sizes` comme montré dans l'extrait de code 3.2. Cette liste remplace la valeur de taille moyenne des partitions utilisée dans l'algorithme original.

CODE 3.2 Extrait de `label_balance_verts.cpp` montrant l'usage de la capacité des partitions.

```
double unit_avg_size = g.vertex_weights_sum / g.partition_capacities_sum;
double* avg_sizes = new double[num_parts];
for (int i = 0; i < num_parts; ++i)
```

```
4     avg_sizes[i] = unit_avg_size * g.partition_capacities[i];
5     /*[...]*/
```

3.5.4 Condition pour permettre les partitions sans tâche

En ajoutant une condition dans les phases d'équilibrage des sommets et d'équilibrage des arêtes, nous avons permis la réduction, ou non, du nombre de partitions initialement utilisées. Cette caractéristique distingue les modèles de *bin packing* et d'assignation de tâches. Pour le problème d'assignation BQSAP, la contrainte interdit le transfert d'un sommet d'une partition à une autre dans le cas où ce transfert viderait complètement une partition. Cela permet de conserver des tâches sur toutes les partitions. Pour le problème de *bin packing* Q-VSBPP, une tâche est sélectionnée pour être déplacée d'une partition à l'autre même si cela entraine une partition vide. Ce processus rend alors possible la diminution du nombre de partitions utilisées.

3.5.5 Ajout d'une contrainte de capacité maximale des partitions

Dans le Q-VSBPP, aucune partition ne doit dépasser la capacité maximale. Pour ce faire, nous avons ajouté une vérification dans la fonction `init_nonrandom_constrained_capacity()` lors de la phase d'initialisation. Le code 3.3 illustre cette logique : avant d'assigner la tâche voisine `out`, qui n'a pas encore de partition, à la partition courante `part`, une vérification est effectuée pour savoir si l'ajout de son poids `g.vertex_weights[out]` respecte la limite `max_partition_size[part]` de la partition. Un mécanisme de contrôle similaire est également implémenté dans les phases d'équilibrage ultérieures pour maintenir cette contrainte tout au long de l'exécution de l'algorithme.

CODE 3.3 Logique de vérification de capacité (fragment de `init_nonrandom.cpp`).

```
for (long j = 0; j < out_degree; ++j) {
    int out = outs[j];
    if (parts[out] == -1) { // If part not assigned yet
        if ((part_sizes[part] + g.vertex_weights[out])
             < max_partition_size[part])
            parts[out] = part;
            /*[...]*/}}
```

3.6 Sommaire

Ce chapitre détaille notre proposition de modèles pour le problème du mappage de tâches sur des architectures parallèles hétérogènes. Nous avons d'abord introduit la notation nécessaire pour décrire les composantes du problème en vue de présenter le modèle. Par la suite, nous avons présenté deux modèles : le modèle de semi-assignation quadratique avec goulot (BQSAP), visant à minimiser la communication et le temps de calcul les plus coûteux , et le modèle de *bin packing* à taille variable avec contrainte quadratique de communication (Q-VSBPP), qui minimise le nombre de partitions utilisées en considérant des partitions de capacités différentes et des coûts de communication entre les partitions.

De plus, les modèles proposés fournissent un cadre clair pour comprendre les problèmes de mappage des tâches. Les modèles mathématiques, bien qu'implicitement pris en compte par les algorithmes de partitionnement choisis, ont été définis de manière précise, permettant d'évaluer la qualité des solutions obtenues, malgré le fait que les étapes de l'algorithme minimisent la somme des arêtes coupées au lieu du goulot comme tel. La définition précise du modèle a par ailleurs l'avantage de fournir un cadre générique, permettant de généraliser à d'autres problèmes en recherche opérationnelle. L'application de ces algorithmes à des problèmes similaires pourrait donc être envisagée, particulièrement dans les cas où le temps de résolution est critique.

Enfin, nous avons détaillé l'adaptation de la librairie `PuLP` pour intégrer les spécificités de notre problème de mappage. Cela a inclus l'ajout du coût de communication, la prise en compte des capacités hétérogènes et de la capacité maximale des partitions, ainsi que la flexibilité d'autoriser des partitions sans tâche assignée. Ces modèles, ainsi que l'adaptation de la librairie `PuLP`, constituent le cadre mathématique dont l'application et l'évaluation seront présentées dans la section suivante consacrée aux résultats.

CHAPITRE 4 RÉSULTATS NUMÉRIQUES ET DISCUSSION

Nous présentons dans ce chapitre les résultats d'exemples numériques basés sur les modèles du BQSAP et du Q-VSBPP présentés aux sections 3.2 et 3.3 du chapitre précédent.

4.1 Configuration pour les tests

Pour évaluer les stratégies d'assignation des tâches, nous utilisons les paramètres suivants. Afin d'en faciliter la reproductibilité, les modifications que nous avons apportées à la librairie `PuLP` [1]. Le code source de la librairie `SCOTCH` est fourni par le laboratoire INRIA [2].

Pour l'exécution du code, la librairie `PuLP` est paramétrée pour utiliser quatre cœurs en mémoire partagée avec la librairie de calcul parallèle `OpenMP`. Pour l'exécution `SCOTCH`, elle est en séquentiel sur un seul cœur. Une exécution parallèle avec `pthread` est implémentée dans la librairie `SCOTCH`, mais l'utilisation de plusieurs *threads* dégradait la vitesse d'exécution pour les données de notre étude. Les calculs d'assignation sont exécutés sur un ordinateur portable `i7-2.30GHz-16GB`. Cela correspond avec un contexte d'usage réel où la préparation des simulations EMT est effectuée localement sur un ordinateur portable avant que les simulations soient transmises sur une grappe d'ordinateurs de haute performance pour les calculs de la simulation.

4.1.1 Caractéristiques des grappes de calcul

Les grappes considérées pour les exemples numériques présentent les caractéristiques suivantes :

- une grappe hétérogène de 30 nœuds est considérée pour le BQSAP. Elle comprend un serveur à 18 nœuds de calcul et d'un serveur à 12 nœuds, le premier ayant une capacité de calcul deux fois supérieure à celle du second, par exemple en termes de fréquence d'horloge des processeurs ;
- une grappe homogène de 72 nœuds est considérée pour le Q-VSBPP. Il s'agit d'une grappe formée d'un serveur muni d'un processeur de 72 cœurs, entièrement connectés et de capacité de calcul équivalente ;
- une grappe hétérogène de 72 nœuds est considérée pour le Q-VSBPP. La grappe comprend deux serveurs à 18 nœuds et de trois serveurs à 12 nœuds, pour lesquels les deux premiers ont une capacité de calcul deux fois supérieure, par exemple en termes de fréquence d'horloge du processeur, à celle des trois derniers.

1. https://github.com/julie9/PuLP-QSAP
2. https://gitlab.inria.fr/SCOTCH/SCOTCH

Le délai de communication des données entre les serveurs est basé sur les tests expérimentaux effectués sur une grappe de calcul reliée par des liens de type *Dolphin* [66]. Les tests et les résultats expérimentaux sont décrits dans [65]. Nous avons pu établir les coefficients de la relation linéaire de (3.1c) et (3.2c) avec un délai de $b_{kl} = 0{,}0056$ microseconde par quantité de données en `Doubles` transférées entre les partitions k et l et une latence $c_{kl} = 2{,}7$ microsecondes. Cette modélisation linéaire est validée expérimentalement pour des quantités transmises inférieures à 1000 `Doubles` par lien de communication entre les nœuds de la grappe [65], limite respectée par l'ensemble des résultats de mappage obtenus. Pour les partitions situées sur un même serveur, le délai et la latence pour la communication des données sont fixés à un centième de ceux entre les serveurs.

4.1.2 Caractéristiques de tâches de calcul

Les tâches de calcul utilisées pour les exemples numériques sont tirées de configurations réelles de simulations EMT représentant cinq réseaux électriques de grande taille. Les réseaux sont identifiés par A, B, C, D et E pour des raisons de confidentialité. Le tableau 4.1 affiche les principales caractéristiques de ces graphes de tâches, soit leur identifiant, le nombre de tâches $|\mathcal{T}|$, le nombre d'arêtes $\|A\|_0$ qui représentent le nombre de paires de tâches communiquant des données le coefficient de *clustering*, la distance moyenne, comme défini à la section 1.1.4, et le temps de calcul moyen $\bar{e}_i$ et maximal $\max e_i$, en microsecondes. On doit noter ici que toutes les tâches $t \in \mathcal{T}$ sont connectées, c'est-à-dire qu'elle communique des données avec au moins une autre tâche lors de la phase de communication de la simulation EMT. Les tâches sans aucune communication ont été retirées au cours de la phase de prétraitement comme les modèles que nous présentons dans ce mémoire se concentrent sur la prise en compte des communications.

En lien avec les propriétés des graphes de grande taille (section 1.1.4), la figure 4.1 montre une tendance, pour les cinq réseaux considérés dans cette étude, que la moyenne des distances dans le graphe soit bornée par le logarithme de la taille $|\mathcal{T}|$ du graphe, ce qui est une caractéristique des graphes de type *petit-monde*. Bien que les distances ne respectent pas strictement la borne théorique $\ln(|\mathcal{T}|)$ propre aux graphes petit-monde, elles demeurent sous $2\ln(|\mathcal{T}|)$, indiquant une proximité avec cette propriété pour les graphes de tâches de calcul étudiés. La figure 4.2 montre la distribution des degrés des graphes de tâches, c'est-à-dire le nombre de connexions entrantes et sortantes pour chacune des tâches de simulation EMT des cinq réseaux, présentant un profil exponentiel, propre aux graphes de type *sans-échelle*.

TABLEAU 4.1 Caractéristiques des tâches de calcul, pour les simulations EMT de cinq réseaux électriques de grande taille (A, B, C, D et E) : le nombre de tâches de calcul $|\mathcal{T}|$, le nombre d'arêtes $\|A\|_0$ représentant les paires de tâches communicantes, le coefficient de *clustering* global, la distance moyenne, comme défini à la section 1.1.4, et le temps de calcul moyen $\bar{e}_i$ et maximal $\max e_i$ des tâches de calcul $i \in \mathcal{T} = \{1, ..., t\}$, en microsecondes.

	Graphe des tâches de calcul				Tâches de calcul	
Réseau électrique	Nombre de tâches, $\|\mathcal{T}\|$	Nombre d'arêtes (paires de tâches reliées), $\|A\|_0$	Coefficient de *clustering*	Distance moyenne	Temps de calcul moyen, $\bar{e}_i$ [µs]	Temps de calcul maximal, $\max e_i$ [µs]
A	553	747	0,00	7,6	0,50	17,3
B	435	545	0,24	8,6	2,90	27,4
C	46	53	0,00	3,8	18,17	140,0
D	1106	1295	0,51	14,1	0,88	17,0
E	457	539	0,00	13,5	7,95	102,3

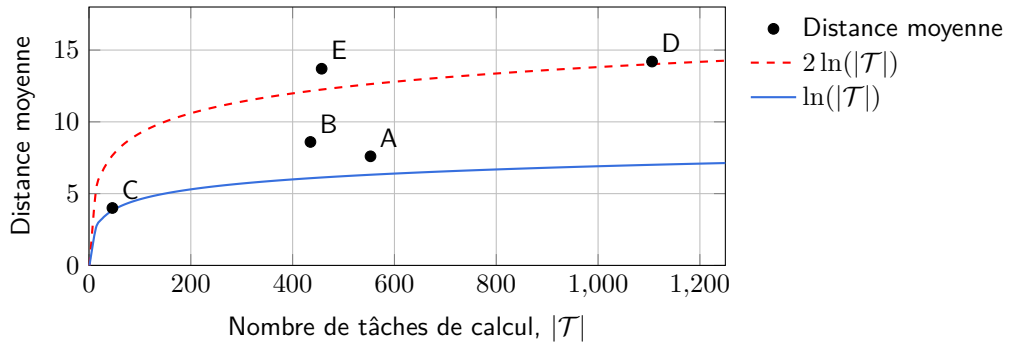
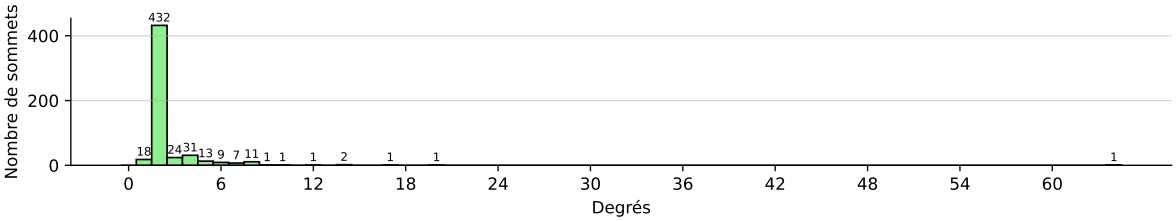


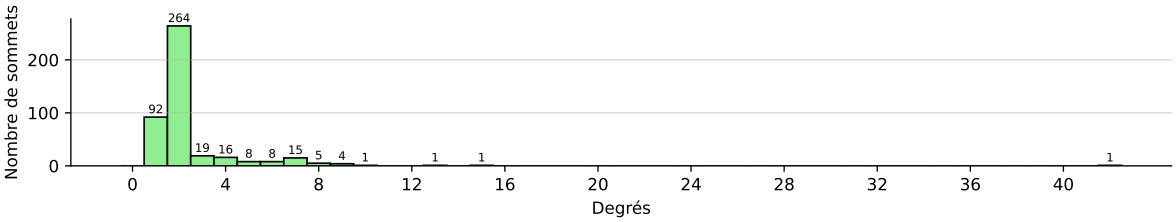
FIGURE 4.1 Distance moyenne dans les graphes de tâches de calcul, pour les simulations EMT de cinq réseaux électriques de grande taille (A, B, C, D et E), comme défini à la section 1.1.4. Bien que les distances ne respectent pas strictement la borne théorique $\ln(|\mathcal{T}|)$ propre aux graphes petit-monde, elles demeurent sous $2\ln(|\mathcal{T}|)$, indiquant une proximité avec cette propriété pour les graphes des tâches étudiés.

4.2 Résultats numériques

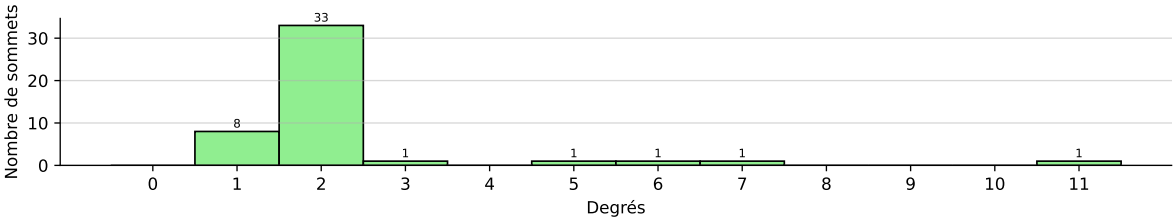
Cette section détaille les résultats expérimentaux visant à évaluer les performances des méthodes proposées pour les modèles BQSAP et Q-VSBPP, en incluant une comparaison à une méthode exacte et des tests sur différentes configurations de grappes de calcul.



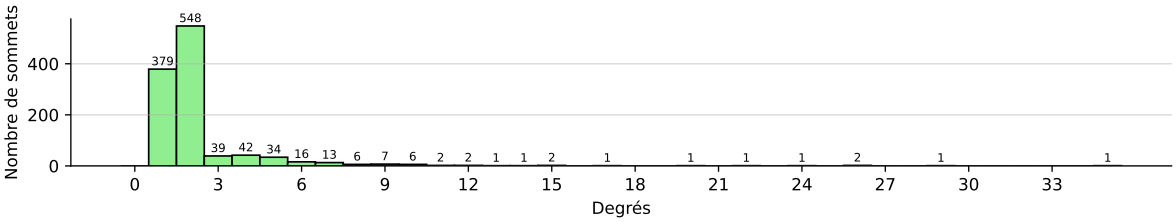
(a) Réseau A



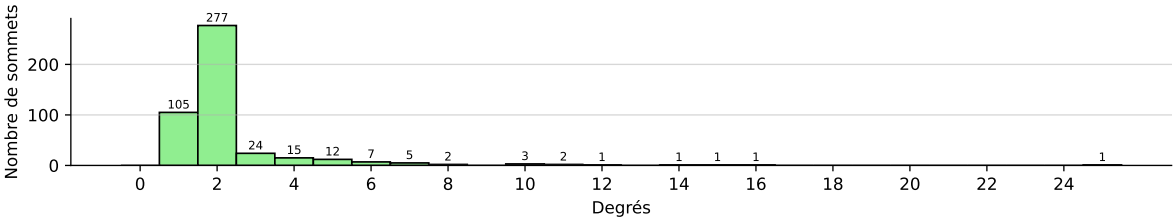
(b) Réseau B



(c) Réseau C



(d) Réseau D



(e) Réseau E

FIGURE 4.2 Répartition des degrés des graphes de tâches de calcul, pour les simulations EMT de cinq réseaux électriques de grande taille (A, B, C, D et E), comme défini à la section 1.1.4. La majorité des tâches (les sommets du graphe) sont de degré 2, c'est-à-dire qu'elles communiquent des données avec deux autres tâches voisines, alors qu'une minorité est de degrés plus élevés et sert de point de connexion avec un plus grand nombre de tâches .

4.2.1 Comparaison avec une méthode exacte

Pour évaluer la qualité des solutions issues des algorithmes heuristiques proposés, soit ceux de la librairie SCOTCH et ceux de la librairie de PuLP modifiée, nous avons effectué une comparaison en utilisant Gurobi Optimizer, un solveur mathématique commercial basé sur la méthode de résolution *branch-and-cut* [67]. Cette approche est un algorithme exact qui recherche la solution optimale ; s'il est interrompu, il fournit la meilleure solution identifiée et une estimation sur une borne inférieure de la valeur optimale, permettant le calcul d'un écart d'optimalité, désigné par le terme *gap*. L'objectif est d'établir une référence pour les résultats des heuristiques. Ces tests se concentrent sur le modèle BQSAP appliqué à une configuration de grappe de calcul comptant jusqu'à 20 nœuds et modélisant deux serveurs hébergeant chacun la moitié des nœuds. Ces nœuds présentent une capacité de calcul homogène, mais ont des délais de communication hétérogènes.

Dans les premières expérimentations, nous avons constaté que l'obtention de solutions optimales pour ce problème requiert un temps de résolution considérable. Nous avons donc conduit deux séries de tests avec des limites de temps pour le solveur Gurobi : de 60 secondes et de 2 heures. Avec une limite de 60 secondes, Gurobi n'obtient pas l'optimalité pour le problème ayant plus 14 partitions. Le *gap* rapporté par Gurobi par rapport à sa borne inférieure est entre 30% et 85%. Les solutions obtenues par les heuristiques de PuLP et SCOTCH présentent des assignations plus performantes, autant pour le goulot sur le temps de calcul et que celui de la phase de communication, en comparaison avec les assignations fournies par Gurobi après 60 s, et ce particulièrement pour les instances où le *gap* estimé par Gurobi est non négligeable. Le temps d'exécution des heuristiques est inférieur : moins de 10 secondes pour l'approche PuLP et moins de 0.1 seconde pour SCOTCH, comme détaillé dans les tableaux des sections suivantes.

L'augmentation de la limite de temps de résolution à 2 heures permet d'améliorer les solutions et de réduire les *gaps* d'optimalité rapportés par les exécutions du solveur Gurobi. Toutefois, la condition d'optimalité n'est pas atteinte dans tous les cas testés, spécifiquement pour les instances du réseau D pour un partitionnement sur des grappes homogènes de plus de 16 nœuds, qui mènent à un *gap* entre 10% et 30% à l'arrêt du solveur. Dans l'ensemble, la qualité des solutions fournies par les heuristiques est comparable à celle des résultats obtenus avec Gurobi après 2 heures d'exécution, comme illustré à la figure 4.3.

Ces résultats comparatifs soulignent la complexité calculatoire significative du problème d'assignation pour les méthodes exactes, même pour des configurations de moins de 20 partitions et avec un temps de résolution étendu jusqu'à 2 heures. Pour une configuration de grappe plus complexe, comme la grappe hétérogène de 30 nœuds, les résultats de Gurobi après 2 heures de résolution montrent une solution sous-optimale (figure 4.4). Ces tests appuient ainsi l'intérêt des approches heuristiques développées dans cette étude, lesquelles offrent un excellent compro-

mis entre la qualité des solutions obtenues et la rapidité d'exécution, permettant d'obtenir des assignations performantes dans des délais très courts et compatibles avec une utilisation pratique.

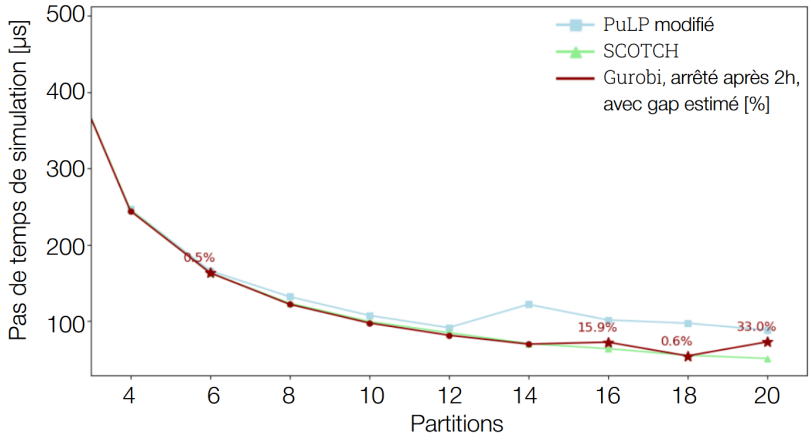
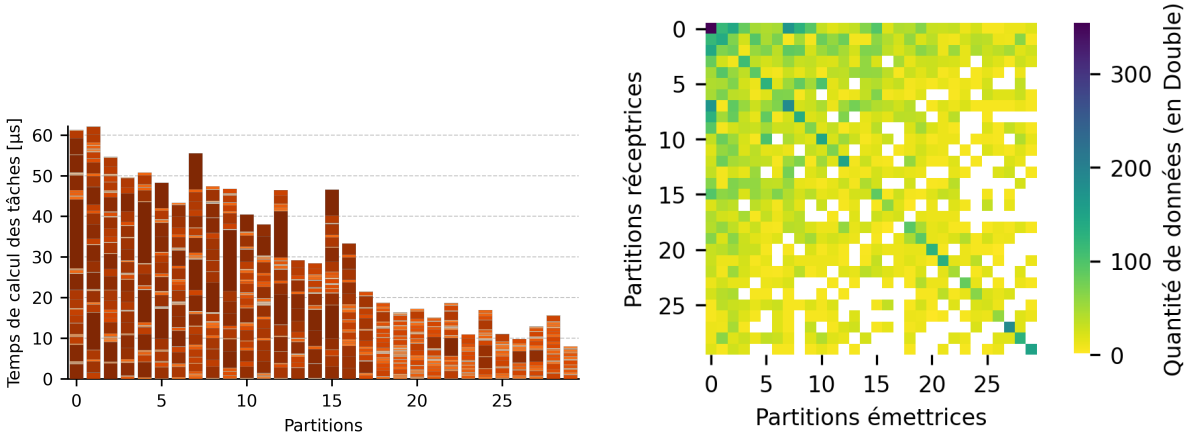


FIGURE 4.3 Tests préliminaires simplifiés sur 4 à 20 partitions d'une grappe homogène, montrant la durée d'un pas de temps de simulation (phases calcul et communication), en microsecondes, pour des résultats de mappage des tâches de calcul de la simulation EMT du réseau D. On remarque que les trois méthodes obtiennent des résultats similaires pour ce cas simple. Le temps de résolution pour `Gurobi` est toutefois arrêté manuellement à 2h, alors que ceux des algorithmes `PuLP` et `SCOTCH` sont de l'ordre de la seconde.



(a) Temps de la phase de calcul par partition

(b) Quantité de données transmises entre les partitions

FIGURE 4.4 Solution de mappage pour le BQSAP avec les tâches de calcul de simulation EMT du réseau D sur une grappe hétérogène de 30 nœuds. L'exécution de `Gurobi` a été arrêtée avant optimalité après 2h de résolution.

4.2.2 BQSAP sur une grappe hétérogène de 30 nœuds

Nous présentons maintenant les résultats obtenus pour le BQSAP sur une grappe hétérogène de 30 nœuds, en utilisant les solveurs PuLP et SCOTCH. Le tableau 4.2 montre les temps de simulation prévus pour les phases de calcul et de communication de la simulation EMT des cinq réseaux pour les assignations qui ont été déterminées par les solveurs. Le temps maximal de la phase de calcul des tâches dépasse le temps de la tâche la plus longue pour les réseaux B, D et E. Ce résultat s'explique par une assignation de plusieurs tâches par processeurs, conformément à la définition du problème comme étant un problème de semi-assignation. Les solutions de PuLP présentent des pas de temps de simulation au plus 77% supérieur par rapport à ceux obtenus avec l'assignation par SCOTCH. En termes de performances du solveur, PuLP nécessite entre 0,2 à 3,8 secondes pour calculer une solution, tandis que SCOTCH résout systématiquement le problème en moins de 0,1 seconde.

La figure 4.5 compare le comportement du solveur sur le réseau D en termes de schémas de communication. SCOTCH montre une tendance à regrouper les communications de manière plus efficace sur des partitions adjacentes. Cela permet de réduire le délai de communication global, comme le montre le tableau 4.2. Bien que les résultats de PuLP présentent des communications moins étroitement groupées sur les nœuds voisins, le regroupement des communications sur des liens plus rapides est tout de même observé. Comme la grappe de calcul est composée de deux serveurs de 18 et 12 nœuds, les communications entre les paires de tâches assignées sur les nœuds 0-17 et 18-29 engendrent des temps de communications plus faibles comme il s'agit de communications entre les nœuds sur un même serveur (section 4.1.1).

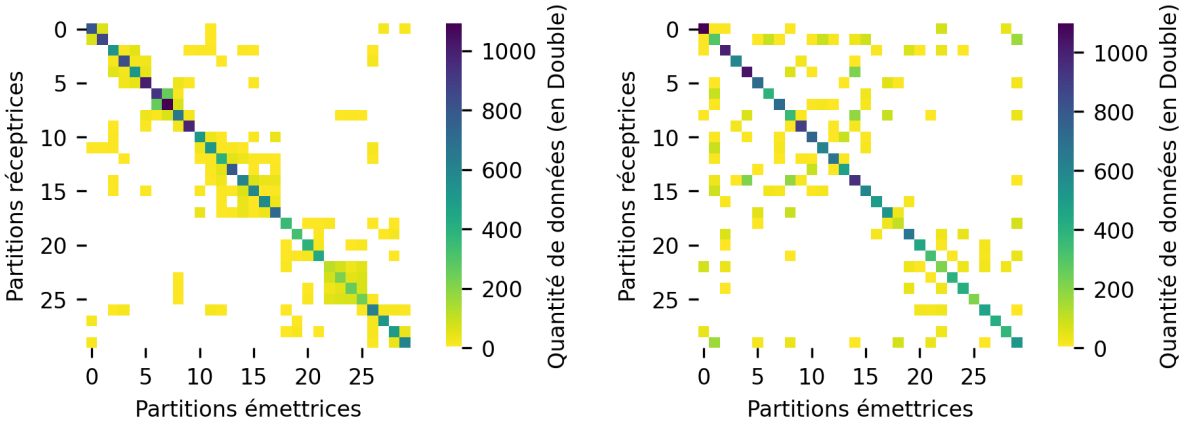
4.2.3 Q-VSBPP sur une grappe homogène de 72 nœuds

Ensuite, nous comparons les résultats obtenus pour le problème Q-VSBPP sur une grappe homogène en utilisant SCOTCH et PuLP. Le Q-VSBPP permet de minimiser le nombre de partitions requises. Le temps maximal pour la phase de calcul $Y_{\max}$ par partition est déterminé par le temps de calcul de la tâche la plus longue, comme défini par (3.2b) du modèle Q-VSBPP et présenté dans le tableau 4.1 pour chacun des cinq réseaux.

Le tableau 4.3 montre des résultats similaires pour les deux solveurs pour ce qui est des pas de temps de simulation, avec des différences est d'au plus de 2%, sauf pour le mappage du réseau E pour lequel le temps de simulation donné par PuLP est supérieur de 44%. La répartition des tâches du réseau D sur les partitions est présentée à la figure 4.6 avec leur temps de calcul respectif. Pour le nombre de partitions trouvées, les résultats pour chaque solveur sont comparables, dans une plage de $\pm$13%, à l'exception du réseau A, où PuLP identifie près deux fois plus de partitions

TABLEAU 4.2 BQSAP sur une grappe hétérogène de 30 nœuds.

Réseau	Solveur	Temps de résolution [s]	Goulot		Pas de temps de la simulation [µs]
			Temps de la phase de calcul [µs]	Temps de la phase de communication [µs]	
A	SCOTCH	0.050	8.66	19.80	28.46
	PuLP	0.228	9.71	25.31	35.02
B	SCOTCH	0.023	26.34	47.34	73.68
	PuLP	0.240	42.16	16.70	58.86
C	SCOTCH	0.001	70.00	14.00	84.00
	PuLP	3.847	140.00	8.36	148.36
D	SCOTCH	0.103	20.51	19.42	39.93
	PuLP	0.282	32.65	12.18	44.83
E	SCOTCH	0.022	75.80	19.77	95.57
	PuLP	0.204	143.48	11.67	155.15



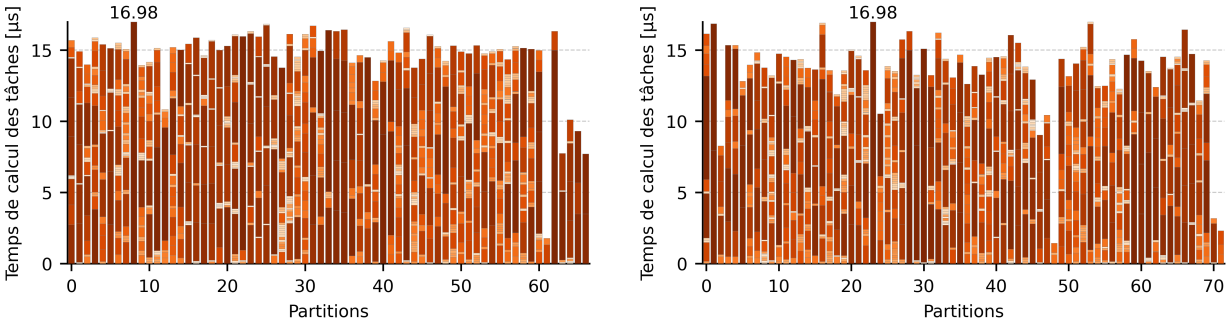
(a) SCOTCH (b) PuLP

FIGURE 4.5 Quantité de données transmises entre les partitions durant la phase communication, pour le BQSAP, avec les tâches de calcul de simulation EMT du réseau D sur une grappe hétérogène de 30 nœuds.

à utiliser. Ceci est potentiellement dû à la limitation de l'algorithme de PuLP, identifiée dans la section 2.4.2.1, soit son incapacité à échanger des étiquettes, ou partitions, qui ne sont pas voisines, ce qui pourrait limiter la minimisation du nombre de partitions. En termes de temps de résolution du solveur, PuLP demande entre 0,1 et 0,3 seconde et SCOTCH résout le problème en moins de 0,15 seconde.

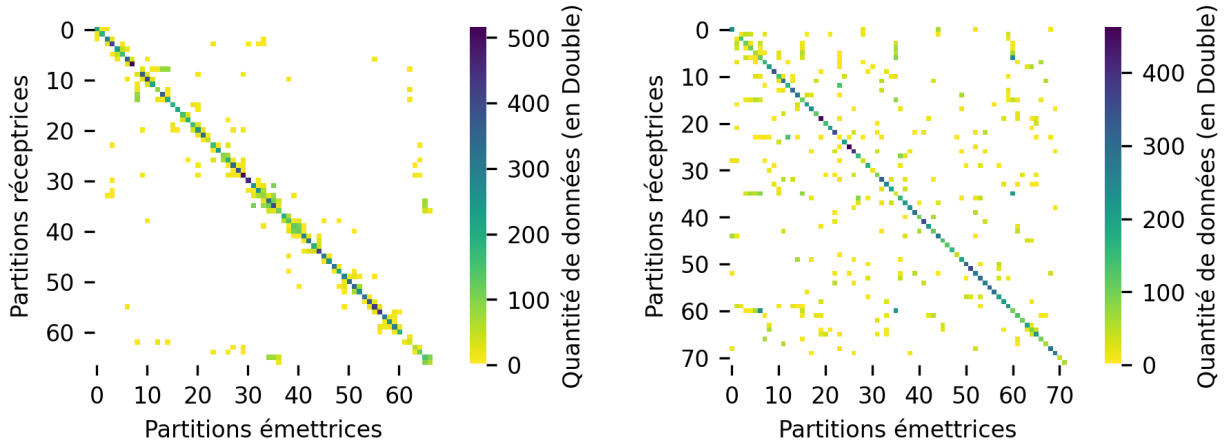
Les figures 4.6 et 4.7 illustrent les communications entre les partitions dans les assignations

obtenues avec les deux solveurs sur le réseau D. Étant donné qu'une grappe homogène constituée d'un seul serveur de 72 nœuds est utilisée pour ces assignations, une distribution relativement uniforme de la communication entre les partitions est observée. Conformément aux résultats de la section précédente, PuLP présente des motifs de communication moins compacts en comparaison avec SCOTCH qui regroupe les paires de tâches communicantes sur les partitions voisines.



(a) SCOTCH (b) PuLP

FIGURE 4.6 Temps de la phase de calcul par partition, pour le Q-VSBBP avec les tâches de calcul de simulation EMT du réseau D sur une grappe homogène.



(a) SCOTCH (b) PuLP

FIGURE 4.7 Quantité de données transmises entre les partitions, pour le Q-VSBBP avec les tâches de calcul de simulation EMT du réseau D sur une grappe homogène de 72 nœuds.

TABLEAU 4.3 Q-VSBPP sur une grappe homogène de 72 nœuds.

Réseau	Solveur	Temps de résolution [s]	Nombre de partitions utilisées	Goulot		Pas de temps de la simulation [µs]
				Temps de la phase de calcul [µs]	Temps de la phase de communication [µs]	
A	SCOTCH	0.010	19	17.4	0.38	17.8
	PuLP	0.274	57	17.3	0.87	18.2
B	SCOTCH	0.010	72	27.4	0.74	28.1
	PuLP	0.278	64	27.4	0.66	28.1
C	SCOTCH	0.001	10	140.0	0.17	140.2
	PuLP	0.140	14	140.0	0.22	140.2
D	SCOTCH	0.030	67	17.0	0.20	17.2
	PuLP	0.298	72	17.0	0.36	17.3
E	SCOTCH	0.012	57	102.3	0.20	102.5
	PuLP	0.235	58	144.0	0.31	144.3

4.2.4 Q-VSBPP sur une grappe hétérogène de 72 nœuds

Enfin, nous présentons les résultats obtenus avec le solveur PuLP pour le Q-VSBPP sur une grappe hétérogène de 72 nœuds répartis sur trois serveurs. Au moment de la rédaction de ce mémoire, le support de SCOTCH pour le *bin packing* est limité aux grappes homogènes entièrement connectées ou connectées en hypercube et n'a donc pas pu être utilisé pour ces tests. Comme pour les résultats de Q-VSBPP homogènes à la section précédente, le temps maximal pour la phase de calcul $Y_{\max}$ est fixé au temps de calcul de la tâche la plus longue, présenté dans le tableau 4.1 pour les cinq réseaux.

Les résultats présentés dans le tableau 4.4 montrent que les temps de résolution du solveur pour ce problème se situent entre 0,05 et 0,1 seconde. Les solutions de *bin packing* utilisent entre 10 et 69 nœuds sur les 72 disponibles de la grappe. Le temps maximal de la phase de communication constitue une part importante du pas de temps total de simulation, jusqu'à 73% pour le réseau A. Cela peut s'expliquer en partie par les caractéristiques de communication hétérogènes de la grappe comme la communication de données entre des nœuds plus distants sur différents serveurs entraîne des délais plus longs.

La figure 4.8 montre l'assignation des tâches pour le réseau D sur 69 partitions. Les temps de la phase de calcul de chacune des partitions sont illustrés par la figure 4.8a où les partitions 0-35, associés aux serveurs plus performants, se voient attribuer davantage de temps de calcul, comparativement aux partitions 36-68, associées aux serveurs moins performants. Concernant le volume de communication, bien que moins concentré que dans les résultats précédents obtenus

avec SCOTCH, nous observons que la répartition regroupe principalement les tâches communicantes sur la même partition, comme le montre la diagonale de la figure 4.8b, réduisant ainsi une partie de la charge de communication dans la grappe.

TABLEAU 4.4 Q-VSBPP sur une grappe hétérogène de 72 nœuds avec le solveur PuLP.

Réseau	Temps de résolution [s]	Nombre de partitions utilisées	Goulot		Pas de temps de la simulation [µs]
			Temps de la phase de calcul [µs]	Temps de la phase de communication [µs]	
A	0.073	52	16.4	44.4	60.8
B	0.086	58	27.2	35.9	63.2
C	0.050	10	144.1	8.3	152.4
D	0.096	69	17.0	22.3	39.3
E	0.067	57	100.4	16.8	117.2

4.3 Discussion

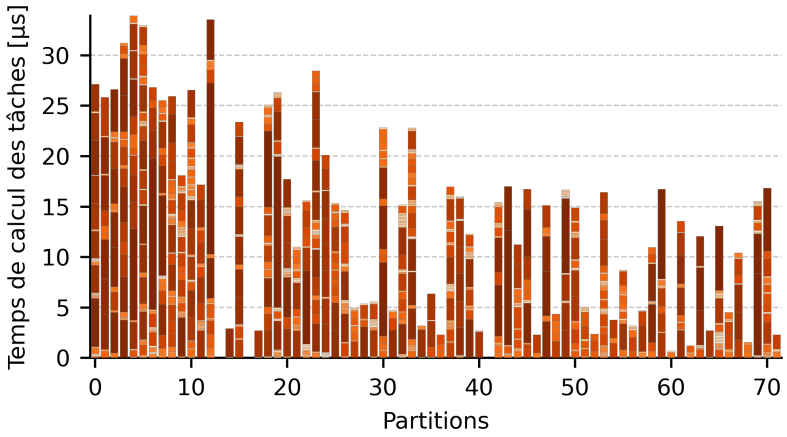
Ce mémoire offre de nouveaux éclairages sur le mappage des tâches en environnements de calcul hétérogènes. Notamment, en établissant un pont entre les algorithmes de partitionnement de graphes et les modèles de recherche opérationnelle, nous avons montré empiriquement que les outils existants peuvent être adaptés efficacement pour relever des défis spécifiques aux simulations de réseaux électriques.

Notre adaptation de l'algorithme PuLP pour résoudre les problèmes BQSAP et Q-VSBPP ainsi que les tests effectués avec la librairie SCOTCH mettent en évidence la polyvalence de ces approches pour le partitionnement de graphes et l'assignation de tâches. Les améliorations de performance observées dans nos expérimentations suggèrent que les méthodes de partitionnement basées sur les graphes, lorsqu'elles sont combinées à des modèles précis, peuvent offrir des avantages pratiques par rapport aux techniques traditionnelles de recherche opérationnelle telles que les algorithmes exacts et les métaheuristiques. Ceci est particulièrement manifeste pour les simulations à grande échelle nécessitant de larges grappes de calcul ; le temps de résolution rapide pour le mappage des tâches est crucial pendant l'étape préparatoire pour faciliter la mise en œuvre des simulations.

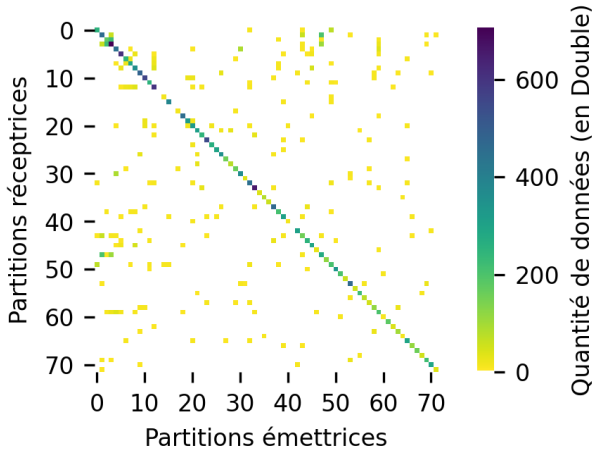
L'introduction de la contrainte de goulot dans BQSAP et Q-VSBPP répond à une caractéristique pratique importante dans les environnements de calcul hétérogènes. Le temps de la simulation est directement relié au temps de calcul plus long sur une des partitions et le délai maximal de communication entre deux partitions. Nos résultats indiquent que cette contrainte capture efficacement les limitations de performance inhérentes aux grappes réelles, conduisant à des mappages

de tâches qui prennent en compte les goulots en matière de capacité et de coûts de communication. La prise en compte du délai de communication maximal permet d'assurer une performance constante des simulations en temps réel.

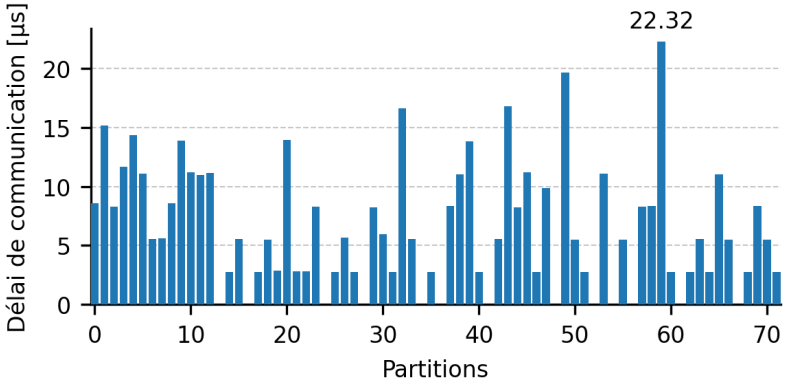
L'efficacité computationnelle obtenue grâce à notre approche a des implications pratiques pour les simulations EMT. Les temps de résolution plus rapides permettent des reconfigurations plus fréquentes des mappages de tâches, ce qui est particulièrement significatif lors de la gestion des modifications du réseau électrique au cours de la phase de conception. Cette capacité permet de tester efficacement plusieurs configurations du réseau électrique.



(a) Temps de la phase de calcul



(b) Quantité de données transmises entre les partitions durant la phase communication



(c) Temps de la phase communication par partition

FIGURE 4.8 Q-VSBPP avec les tâches de calcul de simulation EMT du réseau D sur une grappe hétérogène de 72 nœuds avec le solveur PuLP.

CHAPITRE 5 CONCLUSION

En conclusion de ce mémoire, ce chapitre synthétise les travaux réalisés sur les stratégies d'assignation de tâche pour des simulations de transitoires électromagnétiques (EMT) sur des grappes de calcul hétérogènes, en discute les limitations et propose les voies potentielles pour des recherches ultérieures.

5.1 Synthèse des travaux

Dans ce mémoire, nous illustrons expérimentalement l'efficacité des algorithmes de partitionnement de graphes, en particulier pour le problème de la semi-assignation quadratique (BQSAP) et le problème de *bin packing* de taille variable avec des contraintes de communication quadratiques (Q-VSBBP), pour le mappage de tâches de simulations EMT. Les résultats obtenus avec l'extension de la librairie `PuLP` mettent en évidence sa flexibilité vis-à-vis de l'architecture des grappes ainsi que la formulation du problème, ce qui permet son adaptation aux problèmes d'assignation et de *bin packing*. Cependant, le partitionnement des tâches généré par `PuLP` entraîne des pas de temps de simulation légèrement plus élevés par rapport aux résultats obtenus avec la librairie de partitionnement `SCOTCH`.

5.2 Limitations de la solution proposée

La solution proposée, bien que fonctionnelle, comporte certaines limitations inhérentes aux outils et stratégies employés. D'une part, au niveau des algorithmes de base, l'approche de *bin packing* utilisée via `SCOTCH` ne permet pas de prendre en compte des capacités différentes pour chaque partition et des coûts de communications hétérogènes, limitant potentiellement son déploiement dans un contexte réel où les ressources computationnelles seraient limitées. De plus, l'optimisation effectuée avec l'algorithme modifié de `PuLP` est restreinte aux échanges entre des partitions des nœuds voisins immédiats, ce qui peut freiner l'exploration globale de l'espace des solutions. Bien que l'utilisation de redémarrages multiples atténue ce problème en diversifiant les points de départ, la portée de l'exploration reste locale lors de chaque itération de l'algorithme.

D'autre part, certaines stratégies spécifiques à l'initialisation du partitionnement pourraient être affinées. L'initialisation actuelle repose sur une sélection aléatoire de nœuds, alors qu'une approche basée sur des métriques topologiques, comme le degré de connectivité, pourrait constituer un point de départ pertinent pour l'optimisation. Par ailleurs, des techniques de prétraitement visant à agréger des structures spécifiques présentées à la section 4.1.2, telles que les groupes de nœuds

fortement connectés, n'ont pas été implémentées. L'exploration de telles approches, qui rappellent les stratégies multiniveaux éprouvées en partitionnement de graphes et utilisées dans SCOTCH, pourrait potentiellement améliorer la qualité de la partition initiale.

5.3 Améliorations futures

Des pistes d'amélioration peuvent être explorées pour enrichir les travaux présentés dans ce mémoire. Un premier ensemble d'améliorations concerne directement les outils de partitionnement et leurs applications. Il serait pertinent d'étendre les capacités de la librairie SCOTCH pour intégrer la gestion du *bin packing* pour des grappes hétérogènes, afin de mieux supporter différents environnements de calcul. Bien que la vitesse actuelle des algorithmes soit adéquate pour les réseaux étudiés, l'application à des systèmes de plus grande taille pourrait bénéficier des fonctionnalités de *repartitionnement*, comme celles implémentées par SCOTCH. Celles-ci visent à minimiser les changements sur une partition existante, ce qui serait avantageux pour l'adaptation dynamique dans des simulations EMT de très grande taille.

Un second axe majeur de recherche consisterait en une analyse plus fondamentale du problème de partitionnement lui-même et de la qualité des solutions obtenues. Bien que les résultats expérimentaux soient concluants pour notre cas d'usage, une étude plus poussée de l'écart d'optimalité est nécessaire pour les applications exigeant une précision accrue. Cette analyse pourrait être complétée par une étude de la complexité intrinsèque du problème, incluant l'analyse de la robustesse des solutions face aux variations des données d'entrée, en s'inspirant de [68], l'application de méthodologies d'analyse de sensibilité [69], l'étude des transitions de phase [70], et l'analyse du paysage de la fonction objectif (*fitness landscape*) [71]. Ces études permettraient de mieux comprendre les limites et les forces de l'approche actuelle.

En lien avec les applications spécifiques des simulations EMT, il serait pertinent d'explorer des méthodes alternatives agissant en amont du partitionnement lui-même pour la séparation du réseau en tâches de calcul. Des techniques de séparation de tâches basées sur des méthodes de compensation, telles que décrites dans [10, 72], pourraient potentiellement réduire davantage le temps de calcul des tâches individuelles et la quantité de données transmises entre les tâches, offrant une alternative ou un complément à l'approche actuelle focalisée sur le délai des lignes.

Enfin, une perspective intéressante est le développement de simulations EMT multirésolution. Dans ce schéma, différentes tâches seraient simulées avec des pas de temps distincts, n'impliquant des échanges de données que tous les n pas. Une suggestion de prétraitement consisterait à ajuster l'estimation du temps de calcul e_i/n tout en conservant l'estimation du volume de communication a_{ij} pour anticiper les pics de communication lors des synchronisations.

RÉFÉRENCES

[1] *IEEE Standard for Interconnection and Interoperability of Inverter-Based Resources (IBRs) Interconnecting with Associated Transmission Electric Power Systems*, Norme IEEE-2800.

[2] J. Machowski, J. W. Bialek et J. R. Bumby, *Power System Dynamics : Stability and Control*, 2e éd. Chichester, U.K. : Wiley, 2008.

[3] P. Kundur, N. J. Balu et M. G. Lauby, *Power System Stability and Control*, ser. The EPRI Power System Engineering Series. New York : McGraw-Hill, 1994.

[4] A. Ametani, "Electromagnetic Transients Program : History and Future," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 16, n°. 9, p. 1150–1158, 2021.

[5] G. Gharehpetian, A. Yazdani et B. Zaker, *Power System Transients : Modelling Simulation and Applications*, 1er éd. Boca Raton : CRC Press, nov. 2022.

[6] J. C. Goulart De Siqueira et B. D. Bonatto, *Introduction to Transients in Electrical Circuits : Analytical and Digital Solution Using an EMTP-based Software*, ser. Power Systems. Cham : Springer International Publishing, 2021.

[7] J. H. Chow et J. J. Sanchez-Gasca, *Power System Modeling, Computation, and Control*, 1er éd. Wiley, déc. 2019.

[8] J. Mahseredjian, "Régimes transitoires électromagnétiques : Simulation," *Techniques de l'ingénieur Réseaux électriques et applications*, vol. base documentaire : TIP302WEB., n°. ref. article : d4130, 2008.

[9] S. Dennetiere, B. Bruned, H. Saad et E. Lemieux, "Task Separation for Real-Time Simulation of the CIGRE DC Grid Benchmark," dans *2018 Power Systems Computation Conference (PSCC)*, juin 2018, p. 1–7.

[10] B. Bruned, J. Mahseredjian, S. Dennetière, J. Michel, M. Schudel et N. Bracikowski, "Compensation Method for Parallel and Iterative Real-Time Simulation of Electromagnetic Transients," *IEEE Transactions on Power Delivery*, p. 1–8, 2023.

[11] F. M. Uriarte, *Multicore Simulation of Power System Transients*. IET Digital Library, juin 2013.

[12] M. Xiong, B. Wang, D. Vaidhynathan, J. Maack, M. J. Reynolds, A. Hoke, K. Sun, D. Ramasubramanian, V. Verma et J. Tan, "An open-source parallel EMT simulation framework," *Electric Power Systems Research*, vol. 235, p. 110734, oct. 2024.

[13] K. Schloegel, G. Karypis et V. Kumar, "Graph Partitioning for High Performance Scientific Simulations," Report, mars 2000.

[14] B. Hendrickson et T. G. Kolda, "Graph partitioning models for parallel computing," *Parallel Computing*, vol. 26, n°. 12, p. 1519–1534, nov. 2000.

[15] C.-E. Bichot et P. Siarry, édit., *Graph Partitioning*, 1er éd. Wiley, févr. 2013.

[16] A. Boulmier, N. Abdennadher et B. Chopard, "Optimal load balancing and assessment of existing load balancing criteria," *Journal of Parallel and Distributed Computing*, vol. 169, p. 211–225, nov. 2022.

[17] M. Conforti, G. Cornuéjols et G. Zambelli, *Integer Programming*, ser. Graduate Texts in Mathematics. Cham : Springer International Publishing, 2014, vol. 271.

[18] D. Bertsimas et R. Weismantel, *Optimization over Integers*. Belmont : Dynamic Ideas, 2005.

[19] M. R. Garey et D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-completeness*, ser. A Series of Books in the Mathematical Sciences. New York : W. H. Freeman, 1979.

[20] T. Nowatzki, M. C. Ferris, K. Sankaralingam, C. Estan, N. Vaish et D. A. Wood, *Optimization and Mathematical Modeling in Computer Architecture*. Cham, Switzerland : Springer, 2014.

[21] M. J. Brusco et S. Stahl, *Branch-and-Bound Applications in Combinatorial Data Analysis*, ser. Statistics and Computing. New York : Springer, 2005.

[22] R. E. Burkard, "Quadratic assignment problems," dans *Handbook of Combinatorial Optimization*, P. M. Pardalos, D.-Z. Du et R. L. Graham, édit. New York, NY : Springer New York, 2013, p. 2741–2814.

[23] F. Pellegrini, "Static Mapping of Process Graphs," dans *Graph Partitioning*. John Wiley & Sons, Ltd, 2013, ch. 5, p. 115–136.

[24] A. Billionnet et S. Elloumi, "Best reduction of the quadratic semi-assignment problem," *Discrete Applied Mathematics*, vol. 109, n°. 3, p. 197–213, mai 2001.

[25] S. H. Bokhari, *Assignment Problems in Parallel and Distributed Computing*, ser. The Kluwer International Series in Engineering and Computer Science. Boston, MA : Springer US, 1987, vol. 32.

[26] F. Malucelli et D. Pretolani, "Lower bounds for the quadratic semi-assignment problem," *European Journal of Operational Research*, vol. 83, n°. 2, p. 365–375, juin 1995.

[27] S. Voss, "Heuristics for Nonlinear Assignment Problems," dans *Nonlinear Assignment Problems : Algorithms and Applications*, P. M. Pardalos et L. S. Pitsoulis, édit. Boston, MA : Springer US, 2000, p. 175–215.

[28] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, p. 1–84, 2019.

[29] K. Erciyes, *Algebraic Graph Algorithms : A Practical Guide Using Python*, ser. Undergraduate Topics in Computer Science. Cham, Switzerland : Springer, 2021.

[30] S. Bornholdt, *Handbook of Graphs and Networks : From the Genome to the Internet*, 1er éd. Hoboken : John Wiley & Sons, Incorporated, 2006.

[31] D. J. Watts et S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, n°. 6684, p. 440–442, juin 1998.

[32] D. A. Bader, H. Meyerhenke, P. Sanders et D. Wagner, *Graph Partitioning and Graph Clustering*, ser. Contemporary Mathematics. Providence (R.I.) : American mathematical society, 2012, n°. v. 588.

[33] P. Le-Huy, M. Woodacre, S. Guérette et É. Lemieux, "Massively Parallel Real-Time Simulation of Very- Large-Scale Power Systems," dans *International Conference on Power Systems Transients*, Seoul, South Korea, juin 2017.

[34] H. Plattner, *A Course in In-Memory Data Management : The Inner Mechanics of In-Memory Databases*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2014.

[35] P. Le-Huy, S. Guérette et F. Guay, "Performance evaluation of communication fabrics for offline parallel electromagnetic transient simulation based on MPI," *Electric Power Systems Research*, vol. 223, p. 109629, oct. 2023.

[36] G. M. Slota, K. Madduri et S. Rajamanickam, "PuLP : Scalable multi-objective multi-constraint partitioning for small-world networks," dans *2014 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA, oct. 2014, p. 481–490.

[37] F. Pellegrini, "Scotch and libScotch 7.0 User's Guide," https ://gitlab.inria.fr/scotch/scotch/-/blob/master/doc/scotch_user7.0.pdf, août 2024.

[38] T. Wong, "Répartition automatique des tâches parallèles : Application dans la simulation de réseaux électriques en temps réel," Thèse de doctorat, Ecole Polytechnique, Montreal (Canada), Canada – Quebec, CA, 1999.

[39] B. Bruned, P. Rault, S. Dennetière et I. M. Martins, "Use of efficient task allocation algorithm for parallel real-time EMT simulation," *Electric Power Systems Research*, vol. 189, p. 106604, déc. 2020.

[40] B. Bruned, "Amélioration de la vitesse de calcul et de la précision des outils de simulation des phénomènes transitoires électromagnétiques sur les réseaux électriques," Thèse de doctorat, Nantes Université, janv. 2023.

[41] O. Tremblay, R. Gagnon et M. Fecteau, "Real-Time Simulation of a Fully Detailed Type-IV Wind Turbine," dans *International Conference on Power Systems Transients (IPST2013)*, Vancouver, Canada, juill. 2013.

[42] R. Yonezawa et T. Noda, "A Study of Solution Process Parallelization for an EMT Analysis Program Using OpenMP," dans *International Conference on Power Systems Transients*, Seoul, 2017.

[43] E. L. Lawler, "The Quadratic Assignment Problem," *Management Science*, vol. 9, n°. 4, p. 586–599, juill. 1963.

[44] M. Delorme, M. Iori et S. Martello, "Bin packing and cutting stock problems : Mathematical models and exact algorithms," *European Journal of Operational Research*, vol. 255, n°. 1, p. 1–20, nov. 2016.

[45] H. Greenberg, "A quadratic assignment problem without column constraints," *Naval Research Logistics Quarterly*, vol. 16, n°. 3, p. 417–421, 1969.

[46] E. Çela, *The Quadratic Assignment Problem*, ser. Combinatorial Optimization. Boston, MA : Springer US, 1998, vol. 1.

[47] A. Silva, L. C. Coelho et M. Darvish, "Quadratic assignment problem variants : A survey and an effective parallel memetic iterated tabu search," *European Journal of Operational Research*, vol. 292, n°. 3, p. 1066–1084, août 2021.

[48] A. Gatti, Z. Hu, T. Smidt, E. Ng et P. Ghysels, "Deep learning and spectral embedding for graph partitioning," dans *Proceedings of the 2022 SIAM Conference on Parallel Processing for Scientific Computing*, févr. 2022, p. 25–36.

[49] Z. Tan et Y. Mu, "Learning solution-aware transformers for efficiently solving quadratic assignment problem," dans *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, vol. 235, Vienna, Austria, juill. 2024, p. 47 627–47 648.

[50] A. Nowak, S. Villar, A. S. Bandeira et J. Bruna, "Revised Note on Learning Quadratic Assignment with Graph Neural Networks," dans *2018 IEEE Data Science Workshop (DSW)*, Lausanne, Switzerland, juin 2018, p. 1–5.

[51] T. A. Ayall, H. Liu, C. Zhou, A. M. Seid, F. B. Gereme, H. N. Abishu et Y. H. Yacob, "Graph Computing Systems and Partitioning Techniques : A Survey," *IEEE Access*, vol. 10, p. 118 523–118 550, 2022.

[52] Ü. Çatalyürek, K. Devine, M. Faraj, L. Gottesbüren, T. Heuer, H. Meyerhenke, P. Sanders, S. Schlag, C. Schulz, D. Seemaier et D. Wagner, "More Recent Advances in (Hyper)Graph Partitioning," *ACM Comput. Surv.*, vol. 55, n°. 12, p. 253 :1–253 :38, mars 2023.

[53] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders et C. Schulz, "Recent Advances in Graph Partitioning," dans *Algorithm Engineering : Selected Results and Surveys*. Cham : Springer International Publishing, 2016, p. 117–158.

[54] G. Karypis et V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, vol. 20, n°. 1, p. 359–392, janv. 1998.

[55] C. Fiduccia et R. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions," dans *19th Design Automation Conference*. Las Vegas, NV, USA : IEEE, 1982, p. 175–181.

[56] F. Pellegrini, "Contributions to parallel multilevel graph partitioning," Thèse de doctorat, Université de Bordeaux I, 2009.

[57] M. G. Resende et C. C. Ribeiro, *Optimization by GRASP : Greedy Randomized Adaptive Search Procedures*. New York, NY : Springer New York, 2016.

[58] F. Pellegrini, "Distillating knowledge about SCOTCH," dans *Combinatorial Scientific Computing*, ser. Dagstuhl Seminar Proceedings (DagSemProc), U. Naumann, O. Schenk, H. D. Simon et S. Toledo, édit., vol. 9061. Dagstuhl, Germany : Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2009, p. 1–12.

[59] ——, "Scotch 7.0 Maintainer's Guide," https ://gitlab.inria.fr/scotch/scotch/-/blob/master/doc/scotch_maint7.0.pdf, 2024.

[60] F. Pellegrini et J. Roman, "Experimental Analysis of the Dual Recursive Bipartitioning Algorithm for Static Mapping," Université Bordeaux I, France, Research Report 1038-96, 1996.

[61] P. H. Madden, "Partitioning with the Kernighan-Lin heuristic," Binghamton, New York State, avr. 2020.

[62] B. W. Kernighan et S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, n°. 2, p. 291–307, févr. 1970.

[63] M. F. Faraj et C. Schulz, "Buffered Streaming Graph Partitioning," *ACM J. Exp. Algorithmics*, vol. 27, p. 1.10 :1–1.10 :26, oct. 2022.

[64] F. Pellegrini et C. Lachat, "Process Mapping onto Complex Architectures and Partitions Thereof," Inria Bordeaux Sud-Ouest, Report, déc. 2017.

[65] OPAL-RT TECHNOLOGIES, "Dolphin with Orchestra Performance Testing," Montreal, Data and Report PXH830_Dolphin_Bench_Tool.

[66] V. Krishnan, T. Miller et H. Paraison, "Dolphin express : A transparent approach to enhancing PCI Express," dans *2007 IEEE International Conference on Cluster Computing*, sept. 2007, p. 464–467.

[67] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2025.

[68] P. Kouvelis et G. Yu, *Robust Discrete Optimization and Its Applications*, ser. Nonconvex Optimization and Its Applications, P. Pardalos et R. Horst, édit. Boston, MA : Springer US, 1997, vol. 14.

[69] N. Ploskas et N. Samaras, *Linear Programming Using MATLAB®*, ser. Springer Optimization and Its Applications. Cham : Springer International Publishing, 2017, vol. 127.

[70] S. Verel, S. L. Thomson et O. Rifki, "Where the Really Hard Quadratic Assignment Problems Are : The QAP-SAT Instances," dans *Evolutionary Computation in Combinatorial Optimi-*

zation, T. Stützle et M. Wagner, édit., vol. 14632. Cham : Springer Nature Switzerland, 2024, p. 129–145.

[71] P. Merz et B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, n°. 4, p. 337–352, nov. 2000.

[72] W. F. Tinney, "Compensation Methods for Network Solutions by Optimally Ordered Triangular Factorization," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-91, n°. 1, p. 123–127, janv. 1972.