Titre: Title:	Accélération d'une méthode d'agrégation dynamique de contraintes par apprentissage automatique pour le problème de construction d'horaires de conducteurs d'autobus
Auteur: Author:	Jonathan Brasseur
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Brasseur, J. (2022). Accélération d'une méthode d'agrégation dynamique de contraintes par apprentissage automatique pour le problème de construction d'horaires de conducteurs d'autobus [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10534/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10534/
Directeurs de recherche: Advisors:	François Soumis, & Guy Desaulniers
Programme: Program:	Maîtrise recherche en mathématiques appliquées

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Accélération d'une méthode d'agrégation dynamique de contraintes par apprentissage automatique pour le problème de construction d'horaires de conducteurs d'autobus

JONATHAN BRASSEUR
Département de mathématiques et de génie industriel

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

Août 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Accélération d'une méthode d'agrégation dynamique de contraintes par apprentissage automatique pour le problème de construction d'horaires de conducteurs d'autobus

présenté par **Jonathan BRASSEUR**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Issmaïl EL HALLAOUI, président
François SOUMIS, membre et directeur de recherche
Guy DESAULNIERS, membre et codirecteur de recherche
Louis-Martin ROUSSEAU, membre

REMERCIEMENTS

J'aimerais remercier ma famille m'ayant supporté tout au long de ma maîtrise.

Je remercie mon directeur de recherche François Soumis et mon codirecteur de recherche Guy Desaulniers pour le support et l'aide qu'ils m'ont apportés au long de ce projet. Ils étaient toujours disponibles et positifs, et leurs conseils m'ont grandement aidé à passer à travers les défis que j'ai rencontrés.

J'aimerais aussi remercier GIRO pour le financement de ce projet et pour les discussions productives avec les membres de l'équipe.

Et un grand merci à Issmaïl El Hallaoui et Louis-Martin Rousseau pour avoir accepté de faire partie du jury de ma maîtrise.

RÉSUMÉ

Les compagnies de transport en commun ont la difficile tâche de créer des quarts de travail pour l'ensemble de leurs conducteurs d'autobus. Chaque segment de voyage doit se faire attribuer un conducteur. Ce problème de construction d'horaire de conducteurs d'autobus (DSP) est d'une taille importante tant par rapport à la quantité de bus que par rapport au nombre de lignes faisant partie du problème. C'est aussi un problème engendrant de grands coûts économiques associés à la rémunération des conducteurs.

Dans ce mémoire, on présente une modification de la méthode actuelle de résolution de DSP qui permettrait d'accélérer la résolution de ce problème tout en arrivant à une solution optimale. Cette modification repose sur la méthode d'agrégation dynamique de contraintes (DCA). Nous présentons une nouvelle méthode qui utilise l'apprentissage automatique afin de créer les agrégations initiales à fournir à DCA. Habituellement, dans le cadre de résolution de DSP, les agrégations initiales sont créées en regroupant tous les segments faisant partie d'un même horaire de bus sous une seule agrégation dans l'ordre auquel le bus les parcourt. Ces agrégations permettent au solveur de traiter ces voyages comme en étant un seul au point de vue des contraintes de partitionnement, ce qui réduit le nombre de contraintes et ainsi accélère la résolution. Ces agrégations basées sur les sorties de bus accélèrent fortement la résolution du problème. Elles ne sont donc pas optimales, car il arrive souvent, dans la solution optimale, que les segments faisant partie d'une même sortie de bus ne soient pas tous couverts par le même conducteur.

La modification présentée dans ce mémoire consiste à intégrer un modèle d'apprentissage automatique dans le processus de composition des agrégations initiales de DCA, ce qui permettrait de générer des agrégations qui ressembleraient mieux aux quarts de travail de la solution de la relaxation linéaire au nœud 0 du *branch and price* et ainsi nécessiter moins de désagrégations afin d'atteindre la solution optimale. Cette approche se cadre dans le nouveau paradigme en optimisation combinatoire qui consiste à utiliser les outils de l'intelligence artificielle en concert avec les outils utilisés dans le domaine de l'optimisation.

Le rôle de l'algorithme d'apprentissage automatique est de prédire, au préalable de la résolution, les endroits où on devrait avoir des échanges de conducteurs dans la solution de la relaxation linéaire du nœud 0 du *branch and price*. Un échange de conducteurs se produit lorsqu'un conducteur quitte un autobus et est remplacé par un nouveau conducteur à un point de relève. Les agrégations de départ basées sur les sorties de bus sont brisées aux endroits où un échange de conducteurs a été prédit. Ainsi, des agrégations nécessitant moins

de désagrégation afin d'atteindre la solution optimale sont créées.

Cette tâche de prédiction est formulée comme étant une tâche de classification binaire. Les données utilisées pour l'entraînement des modèles d'apprentissage automatique sont obtenues à partir de problèmes générés par un nouveau générateur permettant de créer des problèmes inspirés de données réelles. La position des arrêts et les fréquences relatives entre les lignes sont similaires à celle de la Société de Transport de Montréal. Deux ensembles de 12 problèmes ont été créés ayant en moyenne 893 et 1075 voyages. Nous comparons les performances d'un modèle de forêt d'arbres décisionnels, d'un réseau de neurones convolutifs et d'un réseau de neurones graphiques. Nous montrons que ces nouvelles agrégations créées à l'aide de l'apprentissage automatique permettent d'accélérer la résolution des DSP d'en moyenne 20.1% à 32,6% comparativement aux agrégations créées à partir des sorties de bus.

ABSTRACT

Transit companies have the difficult task of creating shifts for all their bus drivers. Each trip segment must be assigned a driver. This driver scheduling problem (DSP) is large in terms of both the number of buses and the number of routes involved. It is also a problem with large economic costs associated with the salaries of the different drivers.

In this paper, we present a modification to the current solution method of the DSP that would speed up the resolution of this problem while arriving at an optimal solution. This modification is based on the dynamic constraint aggregation (DCA) method. We present a new method that uses machine learning to create the initial aggregations to provide to DCA. Typically, for solving DSP, initial aggregations are created by grouping all segments that are part of the same bus route under a single aggregation in the order in which the bus traverses them. These aggregations allow the solver to treat these segments as one from the perspective of the partitioning constraints, which reduces the number of constraints and thus speeds up the resolution. These aggregations based on bus routes strongly accelerate the solution of the problem. However, they are not optimal, because in the optimal solution, it often happens that the segments belonging to the same bus route are not all covered by the same driver.

The modification presented in this dissertation consists in integrating a machine learning model in the process of composing the initial aggregations of DCA, which would allow generating aggregations that would better resemble the drivers shifts of the solution of the linear relaxation at node 0 of the branch-and-bound algorithm and thus require fewer disaggregations in order to reach the optimal solution. This approach is in line with the new paradigm in combinatorial optimization, which consists in using the tools of artificial intelligence in concert with the tools used in the optimization domain.

The role of the machine learning algorithm is to predict, prior to the solution, the places where we should have an exchange of drivers in the solution of the linear relaxation at node 0 of the branch-and-price algorithm. An exchange of drivers occurs when a driver leaves a bus and is replaced by a new driver at a relief point. Aggregations based on bus routes are broken at locations where an exchange of drivers has been predicted. Thus, aggregations requiring less disaggregation in order to reach the optimal solution are created.

This prediction task is formulated as a binary classification task. The data used to train the machine learning models are obtained from problems generated by a new generator to create problems inspired by real data. The location of the stops and the relative frequencies between

the lines are similar to that of the *Société de Transport de Montréal*. Two sets of 12 problems were created with an average of 893 and 1075 trips. We compare the performance of a random forest classifier, a convolutional neural network and a graphical neural network. We show that these new aggregations created using machine learning speed up the DSP resolution by an average of 20.1% to 32.6% compared to aggregations created from bus routes.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

DSP	*Duty Scheduling Problem*
DCA	*Dynamic Constraint Aggregation*
STM	Société de Transport de Montréal
VSP	*Vehicle Scheduling Problem*
SDVSP	*Single-Depot Vehicle Scheduling Problem*
MDVSP	*Multiple-Depot Vehicle Scheduling Problem*
GPS	*Global Positioning System*
GNN	*Graph Neural Network*
RNN	*Recurrent Neural Network*
Bi-RNN	*Bidirectional Recurrent Neural Network*
TP	*True Positive*
FP	*False Positive*
TN	*True Negative*
FN	*False Negative*
TN	*True Negative Rate*
LSTM	*Long Short-Term Memory*
ReLU	*Rectified Linear Unit*
Adam	*Adaptive Moment Estimation*
GCN	*Graph Convolutional Network*

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contexte du projet

Le projet abordé dans ce mémoire se cadre dans un contexte d'optimisation des transports en commun. Diminuer les coûts d'opération tout en assurant la qualité du service est un défi de taille pour les différentes sociétés de transport à travers le monde. La dépense de loin la plus importante de leur budget est la rémunération des employés. Par exemple, pour la Société de Transport de Montréal (STM), la rémunération des employés constitue $68,3\%$ de leur budget de dépenses qui totalise $1,6$ milliard de dollars en 2022 [1]. Il est donc d'une importance capitale d'optimiser les horaires du personnel afin de limiter ces coûts. Afin de créer des horaires de travail engendrant le moins de coûts possible tout en maintenant la qualité du service, les sociétés de transport utilisent les outils de la recherche opérationnelle. Ces outils sont nécessaires dus à la très grande taille et complexité des problèmes de construction d'horaires. En effet, la STM possède 1974 bus circulants sur 224 lignes [1]. Déterminer le nombre de conducteurs nécessaires afin de couvrir l'entièreté de ces lignes d'une façon optimale, tout en respectant les contraintes sur le temps de travail des employés, est une tâche particulièrement difficile. De plus, les contraintes sur le temps de travail des employés peuvent être complexes et devoir respecter ces contraintes complique le problème. Dans ce mémoire, il sera question d'explorer la possibilité d'utiliser des outils de l'intelligence artificielle afin d'accélérer la vitesse de création d'horaires de chauffeurs d'autobus.

Comme décrit dans Desaulniers et Hickman [2], les sociétés de transport passent à travers 3 étapes différentes afin de créer les quarts de travail de chaque conducteur.

1. L'étape de planification stratégique. Celle-ci consiste à déterminer l'itinéraire de chaque ligne de bus afin de répondre à la demande des passagers.
2. L'étape de planification tactique. Cette étape consiste à déterminer la fréquence nécessaire sur chaque ligne afin de répondre à la demande des passagers sans dépasser la capacité des bus, et de convertir ces fréquences en horaire désignant une heure de départ pour chaque voyage nécessaire.
3. La dernière étape, qui est au cœur de ce mémoire, est la planification opérationnelle.
   (a) La première tâche à réaliser lors de la planification opérationnelle est de déterminer un horaire pour chaque autobus tout en minimisant le nombre de véhicules nécessaires et la somme des distances parcourues. Ces problèmes s'appellent des problèmes de création d'horaires d'autobus abrégés en VSP pour *Vehicle Schedu-*

ling Problem en anglais.

(b) La deuxième tâche est d'attribuer des conducteurs aux différents segments constituant les horaires de bus déterminés lors de l'étape précédente tout en minimisant le nombre de conducteurs nécessaires. Ces problèmes s'appellent des problèmes de création d'horaires de conducteurs abrégés en DSP pour *Duty Scheduling Problem* en anglais.

1.2 Définitions

Il est important de définir certains termes utilisés dans l'industrie du transport en commun et de la recherche opérationnelle qui seront utilisés tout au long de ce document.

Du point de vue des passagers, il y a :

— Les arrêts sont les endroits où les passagers peuvent embarquer ou débarquer d'un autobus.

— Un voyage est un départ sur une ligne étant planifié à un certain temps précis au courant de la journée.

— Les lignes sont les différents trajets disponibles aux passagers se déplaçant à l'intérieur du réseau. Elles sont composées de plusieurs arrêts. Pour chaque ligne, une certaine quantité de voyages lui est associée au courant de la journée.

Du point de vue des autobus, il y a :

— Un dépôt constitue un endroit où les véhicules et conducteurs commencent et terminent leur quart de travail.

— Une sortie de bus est l'horaire associé à un bus déterminé lors de la résolution d'un VSP et brisé au passage de l'autobus au dépôt.

— Les déplacements à vide sont tous les déplacements d'un autobus ne se trouvant pas sur une ligne. Ces déplacements peuvent ainsi être des déplacements vers un dépôt ou provenant d'un dépôt. Ils peuvent aussi être le déplacement entre la fin d'une ligne et le départ d'une autre.

Du point de vue des conducteurs, il y a :

— De nombreux points de relève sont situés sur une ligne. Ils sont des endroits où il est possible pour un conducteur d'embarquer ou débarquer d'un autobus.

— Les segments de voyage sont les différentes parties d'un voyage découpé par les points de relève composant le voyage. Chaque segment doit être couvert par un conducteur et un autobus.

— Une pièce de travail est un ensemble de segments couverts par un même conducteur conduisant le même bus tout au long de la pièce sans prendre de pause.

— Un quart de travail est l'ensemble de segments attribués à un même conducteur.

— On dit qu'il y a échange de conducteurs sur un point de relève lorsque selon un ensemble de quarts de travail créés, un chauffeur quitte l'autobus et un nouveau conducteur prend la relève de la conduite sur un même point de relève.

1.3 Objectifs de recherche

Plusieurs méthodes de résolution existent pour résoudre des DSP et un grand nombre d'entre elles sont présentées dans Desaulniers et Hickman [2]. Une méthode grandement utilisée permettant de trouver une bonne solution rapidement tout en respectant un ensemble de contraintes associées aux conducteurs tel que des temps de travail maximums et temps de pause minimums est la méthode de génération de colonnes. Celle-ci repose sur la décomposition de Dantzig–Wolfe. De nombreuses techniques d'accélération ont été créées afin d'accélérer la méthode de génération de colonnes. Une qui nous intéresse particulièrement est la méthode d'agrégation dynamique de contraintes.

Cette méthode agrège certains segments selon une liste d'agrégations (appelée partition initiale) ayant été fournie au préalable de la résolution. Ces agrégations diminuent la quantité de contraintes de partitionnement et ainsi accélèrent la résolution du DSP. Elles peuvent toutefois être remises en cause en cours de résolution en modifiant l'agrégation courante. Ainsi, l'accélération dépend de la qualité de la partition créée au préalable de la résolution qui est fournie au solveur. Plus que cette partition est compatible avec les quarts de travail des conducteurs dans la solution optimale, plus que l'accélération apportée par la méthode est importante. Dans ce mémoire, il sera question d'essayer de créer la partition initiale à l'aide d'algorithmes d'apprentissage automatique afin de déterminer ces partitions initiales le plus précisément possible et exploiter au maximum le potentiel d'accélération de la méthode d'agrégation dynamique de contraintes. L'algorithme d'apprentissage automatique a comme objectif d'identifier les échanges de conducteurs faisant partie de la solution de la relaxation linéaire au nœud 0. Les agrégations basées sur les sorties de bus seront ainsi brisées aux endroits où l'algorithme d'apprentissage prédit un échange de conducteurs.

1.4 Plan du mémoire

La revue de littérature au chapitre 2 abordera différentes méthodes de modélisation et de résolution associées aux DSP. Il sera aussi question de l'utilité de l'intelligence artificielle dans un contexte de recherche opérationnelle. Le chapitre 3 décrira un générateur de problèmes permettant la création d'une quantité importante de données inspirées de problèmes réels à utiliser par l'algorithme d'apprentissage automatique. Le chapitre 4 présentera la modélisation du DSP et les méthodes de résolution ayant été utilisées. Le chapitre 5 évaluera le potentiel maximal d'amélioration apporté par la méthode de création des agrégations initiales en brisant les agrégations basées sur les sorties de bus aux endroits où un échange de conducteurs a lieu dans la solution optimale de la relaxation linéaire au nœud 0. Le chapitre 6 décrira les outils d'apprentissage automatique utilisés, l'architecture des modèles appliqués, les performances obtenues dans le cadre de l'identification des échanges de conducteurs et l'accélération apportée par les partitions créées par apprentissage automatique. Le chapitre 7 apportera une conclusion au mémoire, énoncera certaines limitations du projet et discutera de potentielles suites.

CHAPITRE 2 REVUE DE LITTÉRATURE

Cette section abordera en premier lieu les modélisations utilisées pour les VSP et les DSP. Il sera ensuite question d'une méthode de résolution étant utilisée pour résoudre des DSP qui est la méthode de génération de colonnes. La méthode de *branch and price* sera aussi abordée. Finalement, nous présenterons des problèmes de recherche opérationnelle pour lesquels des outils de l'intelligence artificielle ont été utilisés afin d'aider leur résolution.

2.1 Modélisation de VSP

Soit une liste de voyages pour lesquelles les informations suivantes sont connues :

- Une heure de départ et d'arrivée
- Un lieu de départ et de fin
- La durée des trajets entre toutes combinaisons de lieux de départ et d'arrivée de tous les voyages

La résolution du VSP consiste à attribuer les différents voyages à différents bus de façon à minimiser les coûts. Ces coûts sont constitués de deux composantes, une composante fixe engendrée par l'utilisation de chaque autobus et une composante variable représentant la distance totale parcourue par les bus. Chaque bus doit commencer et terminer son assignation à un dépôt. Bunte et al. [3] présentent une revue exhaustive des différentes manières de modéliser un VSP. Le VSP à un seul dépôt (SDVSP) est un problème assez facile à résoudre dans le sens qu'il existe des modélisations permettant de le résoudre en un temps polynomial [4].

Bodin et al. [5] présentent une approche basée sur un réseau de flot. Cette approche est représentée à la figure 2.1 où deux voyages indexés par i sont représentés et où $*$ représente plusieurs arcs. Chaque voyage est représenté comme étant deux nœuds, un nœud x_i' et un nœud x_i'' reliés par un arc. Des contraintes limitant le flot sur ces arcs assurent qu'un seul autobus soit attribué à chacun des voyages. Les nœuds d' et d'' représentent le dépôt. Des arcs représentés en tiretés sortants du nœud d' sont reliés à tout nœud x_i' représentant les voyages. Des arcs représentés en tireté entrants au nœud d'' sont reliés à tous nœuds x_i'' représentant les voyages. Un arc relie les deux nœuds dépôts d' et d''. Sur cet arc se trouve un coût représentant le coût fixe associé à l'utilisation de chaque autobus. Sur les autres arcs se trouve un coût représentant le déplacement associé à l'arc. Chaque nœud x_i'' est relié par un arc au nœud x_i' de tous autres voyages possibles à desservir suite au premier voyage. Cette

modélisation peut être résolue en résolvant un problème de flot à coût minimum. Daduna et Paixão [6] présentent une version de cette formulation où une contrainte sur l'arc reliant les deux nœuds dépôts est utilisée afin de contrôler le nombre d'autobus faisant partie de la solution.

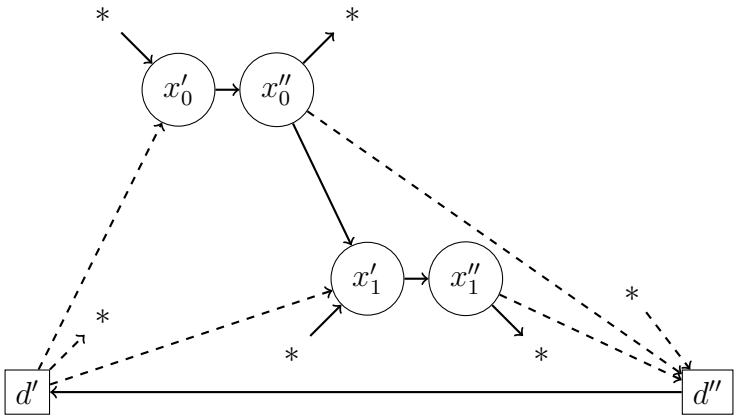


FIGURE 2.1 Représentation du réseau d'un VSP

Le VSP à plusieurs dépôts (MDVSP) est un problème plus complexe où les véhicules peuvent commencer et terminer leurs sorties de bus à différents dépôts possibles. Chacun des dépôts a une quantité de véhicules disponibles lui étant associés. Ce problème est NP-hard tel que prouvé dans Bertossi et al. [4].

Le premier modèle ayant pu résoudre le MDVSP de façon exacte est le modèle présenté par Carpaneto et al. [7]. Celui-ci représente le problème sous la forme d'un réseau où chaque véhicule disponible à chaque dépôt et chaque voyage sont représentés comme étant un nœud. Pour chaque nœud véhicule, un arc sortant et un arc entrant le relient à chacun des nœuds voyage. Cette modélisation est résolue par *branch and bound*. La problématique principale avec cette modélisation est que le nombre de contraintes croît exponentiellement selon le nombre de bus disponibles.

Une autre approche de modélisation des MDVSP est celle où le problème est plutôt représenté comme un problème de flot multicommodités où il y a donc plusieurs nœuds sources et puits. Ainsi, nous avons un sous-réseau par dépôt modélisé de la même façon que le réseau présenté à la figure 2.1. Des contraintes de couvertures sont utilisées sur chaque arc représentant un voyage de chaque sous-réseau afin d'assurer qu'un seul bus soit associé à chacun des voyages. La contrainte sur le nombre de bus associés à chaque réseau peut être représentée par une contrainte de flot sur l'arc reliant la source et le puits. Une méthode exacte qui utilise ce modèle est présentée dans Forbes et al. [8] où l'algorithme du simplexe dual et le *branch and bound* sont utilisés afin d'obtenir une solution entière optimale. Une approche plus

récente permettant de résoudre ce problème utilisant le même réseau est celle présentée dans Hadjar et al. [9]. Celle-ci consiste à utiliser la décomposition de Dantzig-Wolfe au modèle de multicommodités, et donc utilise le même réseau. Le modèle résultant est donc un problème de partitionnement d'ensemble résolu par génération de colonnes. La solution en nombres entiers de ce problème est déterminée par *branch and cut*. Ce problème étant fortement dégénéré, Oukil et al. [10] présente une méthode de stabilisation qui permet de résoudre des problèmes de grande taille fortement dégénérés.

2.2 Modélisation de DSP

Les DSP sont résolus après avoir trouvé la solution d'un VSP au préalable. Le DSP consiste à attribuer les différents segments composant les sorties de bus, ayant été déterminées lors de la résolution d'un VSP, à différents conducteurs afin de minimiser les coûts totaux. Ces coûts sont composés d'une partie fixe engendrée par chaque conducteur nécessaire et une composante variable associée au temps passé à l'extérieur du dépôt pour chaque conducteur. Ce problème peut être représenté par un problème de partitionnement d'ensemble. Les contraintes de partitionnement du DSP contraignent chaque segment à être couvert par exactement un conducteur. Les DSP sont beaucoup plus difficiles à résoudre que les MDVSP à cause de la complexité des règles encadrant les quarts de travail des conducteurs qui parfois ne peuvent seulement être modélisées par des relations non linéaires. Dans certains cas, l'ensemble des quarts de travail possibles peut être déterminé au préalable par un algorithme d'énumération considérant les règles de chaque conducteur tel que présenté dans Smith et Wren [11]. Toutefois, en pratique, les DSP contiennent un très grand nombre de quarts de travail possibles, ce qui crée des problèmes de très grande taille pour lesquels il est difficile d'atteindre l'optimalité.

Pour résoudre les problèmes de plus grande taille, la majorité des études ayant été faites sur ces problèmes peuvent être classées en deux catégories différentes. Les méthodes basées sur la génération de colonnes et les méthodes heuristiques.

La méthode de génération de colonnes pour le DSP a été introduite par Desrochers et Soumis [12] où le problème maître est un problème de partitionnement et le sous-problème est un problème de plus court chemin avec contraintes de ressources. La solution en nombres entiers du DSP est déterminée par *branch and price*. Un aspect négatif de la résolution par génération de colonnes utilisant un problème de plus court chemin contraint comme sous-problème est que cette approche ne permet pas de modéliser certaines contraintes plus complexes sur les quarts de travail des conducteurs. Certaines méthodes ont été présentées afin de limiter cet aspect négatif. Une première est de simplement rejeter toutes colonnes générées

ne respectant pas les contraintes complexes. Une deuxième présentée par de Silva [13] est de formuler les sous-problèmes comme étant des modèles de programmation par contraintes et utiliser les outils de la programmation par contraintes pour les résoudre. Cette méthode permet de résoudre des problèmes ayant jusqu'à 495 segments. Une troisième est présentée par Grötschel et al. [14]. Celle-ci consiste à définir des contraintes de chemins irréalisables pour chaque colonne ne respectant pas les contraintes complexes et ajouter ces contraintes aux sous-problèmes. Une autre faiblesse de la méthode de génération de colonnes appliquée au DSP est le besoin de faire résoudre un très grand nombre de problèmes de plus courts chemins contraints afin de générer les colonnes à ajouter au problème maître restreint. La résolution de ces sous-problèmes peut prendre en moyenne 90% du temps de résolution du problème [15]. C'est pour cela qu'un grand nombre de méthodes ont été développées afin de diminuer le nombre de résolutions de sous-problèmes nécessaires afin de trouver la solution optimale. Une telle méthode est présentée par Fores et al. [16] où il est proposé de créer un ensemble de quarts de travail valides selon la convention collective des employés au préalable de la résolution et ainsi fournir cet ensemble à l'algorithme de génération de colonnes afin de sélectionner les meilleurs quarts de travail de l'ensemble. Cette approche ne permet toutefois pas de garantir que la solution trouvée est la solution optimale. Cette approche est améliorée par Shen et Chen [17] qui définissent l'ensemble de quarts de travail potentiels à l'aide d'un algorithme prenant en compte les caractéristiques du problème à résoudre. Cette approche est démontrée comme étant plus rapide que l'approche de génération de colonnes traditionnelle.

Pour les méthodes heuristiques, une méthode présentée par Chen et Niu [18] attribut des quarts de travails dépendamment de la période de la journée associée au quart. Ces périodes sont le matin, le jour et le soir. Les auteurs formulent ce problème comme étant un problème en nombres entiers avec variables binaires ayant comme objectif de minimiser les coûts associés au temps non productif des conducteurs. Un algorithme de recherche tabou est utilisé pour résoudre ces problèmes. Une méthode permettant d'obtenir des résultats intéressants est celle présentée dans Kecskeméti et Bilics [19] où les performances d'un algorithme évolutionniste, inspiré de celui présenté dans Marchiori et Steenbeek [20], utilisé pour générer les colonnes du problème de partitionnement sont comparées aux performances obtenues à l'aide d'un algorithme de génération de colonnes. La qualité des solutions trouvées par génération de colonnes est en moyenne meilleure. Toutefois, le temps de résolution est plus rapide à l'aide de l'algorithme évolutionniste. Une approche hybride combinant la génération de colonnes à l'algorithme évolutionniste est présentée. Celle-ci est moins rapide que la méthode utilisant seulement l'algorithme évolutionniste. Par contre, elle est plus rapide que la méthode par génération de colonnes et permet d'obtenir des solutions de qualité similaire.

Portugal et al. [21] critiquent l'approche traditionnelle reposant sur une modélisation sous la

forme d'un problème de partitionnement où le seul objectif est de minimiser les coûts. Les auteurs avancent que cette approche simplifie trop fortement quelques aspects des problèmes réels résolus en industrie, ce qui empêche les entreprises d'utiliser les résultats donnés par cette approche, car ceux-ci doivent être modifiés afin de pouvoir être implantés dans une situation réelle. Les auteurs concluent donc que les entreprises préfèrent avoir des solutions partielles pouvant facilement être ajustées contrairement à une solution complète n'étant pas implantable. Tóth et Krész [22] présentent une méthode à deux étapes qui permettrait d'obtenir un grand nombre de solutions partielles. Ces solutions sont déterminées par l'utilisation d'un algorithme glouton. La première étape trouve un grand nombre de solutions partielles et la deuxième crée des quarts de travail complets prenant en considération toutes contraintes supplémentaires afin que la solution soit prête à être implantée.

2.3 Génération de colonnes et *branch and price*

L'algorithme de génération de colonnes appliqué au DSP est décrit pour la première fois dans Desrochers et Soumis [12]. Cette méthode, qui repose sur la décomposition de Dantzig-Wolfe introduite dans Dantzig et Wolfe [23], permet d'éviter d'énumérer l'entièreté des quarts de travail possibles lors de la résolution. La génération de colonnes est appliquée pour résoudre le problème maître qui est la relaxation linéaire d'un problème de partitionnement. Elle démarre avec un petit sous-ensemble de variables. On parle alors du problème maître restreint. À une itération donnée, sa résolution fournit une solution primale et une solution duale. Pour savoir si la solution primale constitue aussi une solution optimale pour le problème maître, on résout les sous-problèmes pour vérifier s'il existe des variables avec coût réduit négatif. S'il y en a, on les ajoute au problème maître restreint et on commence une nouvelle itération. Sinon, on arrête la génération de colonnes avec une solution optimale de la relaxation linéaire. Ainsi, la génération de colonnes permet de résoudre des problèmes d'optimisation linéaire de grande taille de façon exacte. De nombreuses méthodes d'accélération pour la méthode de générations de colonnes ont été présentées telles que la méthode d'agrégation dynamique de contrainte [24]. Celle-ci permet de fortement réduire la quantité de contraintes de partitionnement se trouvant dans le problème maître restreint. Elle permet de fournir une partition de départ au préalable de la résolution du DSP. Cette partition contient une liste d'agrégations. Chaque agrégation est constituée d'une liste de tâches. Les tâches faisant partie d'une même agrégation seront traitées à l'aide d'une seule contrainte de partitionnement au niveau du problème maître. Il est ainsi possible de grandement diminuer le nombre de contraintes de partitionnement qui constitue plus de 90% des contraintes du problème [24].

La méthode du *branch and price* a été introduite dans un contexte de résolution de DSP

par Desrochers et Soumis [12]. Celle-ci est une généralisation de la méthode de *branch and bound* introduite par Land et Doing [25]. De nombreuses variantes de cet algorithme existent, différenciées par les règles de branchement utilisées. La méthode de *branch and price* combine la méthode de *branch and bound* avec la méthode de génération de colonnes. La différence étant qu'à chaque nœud un problème maître pour lequel on a relâché les contraintes d'intégrité est résolu par génération de colonnes.

2.4 Apprentissage automatique en recherche opérationnelle

L'article Bengio et al. [26] discute du potentiel de l'utilisation de l'apprentissage automatique dans un contexte de résolution de problèmes d'optimisation combinatoire. Deux motivations pour l'utilisation d'apprentissage automatique dans ce contexte sont présentées. Tout d'abord, dans les cas où on possède une connaissance empirique ou théorique des décisions devant être prises par l'algorithme d'optimisation, l'apprentissage automatique peut être utilisé pour sélectionner ces décisions et ainsi diminuer le temps de calcul requis associé à la prise de décision. Cette approche est intitulée l'apprentissage par imitation. Un exemple d'utilisation de cette approche est présenté dans Alvarez et al. [27] où les décisions de branchement prises lors du *branch and bound* sont prises par un algorithme d'apprentissage automatique. Ces algorithmes peuvent être entrainés par apprentissage supervisé. La deuxième motivation pour l'utilisation d'apprentissage automatique dans un contexte de résolution de problème d'optimisation combinatoire est celle où au lieu de prendre une décision parmi un ensemble de décisions préexistantes, l'algorithme d'apprentissage est utilisé afin d'établir une nouvelle décision qui pourrait potentiellement être plus performante. Celle-ci peut être apprise par apprentissage par renforcement. Cette approche est intitulée l'apprentissage d'une politique. Un exemple utilisant cette approche est présenté dans Khalil et al. [28] où dans un contexte de problème du voyageur de commerce, la décision déterminant le prochain nœud du trajet est apprise par apprentissage automatique. Ainsi, une nouvelle politique de sélection est créée. Dans cet exemple, la politique est apprise en utilisant un réseau de neurones graphiques (GNN) entrainé par apprentissage par renforcement.

2.5 Lien avec le projet

Ce chapitre présente différents ouvrages décrivant des méthodes de modélisation et résolution de VSP et DSP. Ces méthodes seront utilisées afin de modéliser notre problème. La décomposition de Dantzig-Wolfe, la méthode de génération de colonnes et le *branch and price* sont les méthodes qui vont être utilisées afin de résoudre nos problèmes de DSP. Deux approches

d'intégration de l'apprentissage automatique dans la résolution de problèmes d'optimisation combinatoire ont été présentées. La méthode de sélection de politique utilisant l'apprentissage supervisé est celle qui sera utilisée afin de créer les partitions initiales de la méthode d'agrégation dynamique de contraintes en identifiant les échanges de conducteurs sur les points de relève dans un contexte de résolution de DSP.

CHAPITRE 3 CRÉATION DE DONNÉES

Dans ce chapitre, il sera question de l'outil de génération de données qui a été créé. Celui-ci permet la création d'un grand nombre de problèmes de manière efficace, ressemblant à ceux résolus en industrie. Tout d'abord, le processus de création des problèmes par le générateur ayant servi de base à l'outil de génération sera décrit ainsi que les problèmes créés par le générateur. Les modifications apportées au générateur seront présentées et justifiées et les données de la STM sur lesquelles reposent les modifications du générateur seront présentées.

3.1 Le générateur existant

Pour évaluer le potentiel de généralisation de la nouvelle méthode de génération d'agrégation sur les problèmes en industrie, avoir accès à un grand nombre de données est essentiel. Un grand nombre d'instances est nécessaire afin d'obtenir une performance adéquate de la part de l'algorithme d'apprentissage automatique. Dû à la difficulté de trouver un jeu de données réelles contenant plusieurs problèmes, il a été décidé que les données utilisées dans le cadre de ce projet soit généré à partir d'un générateur. Le générateur de problèmes VCS présenté dans l'article Knut et al. [29] a été utilisé comme base. Ce dernier prend en entrant une liste de paramètres et un fichier entrant, et permet de créer à partir de ceux-ci des VCS.

3.1.1 Les problèmes créés

Le problème généré par le générateur de base est de type VCS à un seul dépôt. À partir de celui-ci, un VSP et un DSP ont été créés. Le SDVSP est représenté sous une forme de problème de flot à coût minimum pouvant être résolu en temps polynomial contrairement au problème à multiples dépôts qui est NP-difficile [4]. Le DSP est pour sa part un problème NP-difficile, et ce même à un seul dépôt. Ceci s'explique par le fait que les conducteurs sont contraints par une convention collective contrairement aux autobus. Un bus pourrait rouler toute la journée sans pause contrairement aux conducteurs qui nécessitent des règles complexes encadrant leurs pauses et durée de quart de travail. Ces contraintes supplémentaires rendent le problème NP-difficile [30].

3.1.2 Fichier d'entrée et paramètres

Le générateur entame le processus de génération d'un problème en lisant un fichier contenant les données géographiques du réseau. Ce fichier est composé de trois parties distinctes.

Arrêts de bus. Cette partie est composée des différents arrêts de bus composant le réseau de bus devant être représenté. Pour chacun de ces arrêts, une paire de coordonnées X et Y lui est associée afin que le générateur puisse situer cet arrêt. De plus, les arrêts de bus où un échange de conducteur peut être effectué sont identifiés.

Lignes de bus. Les différents arrêts composant les lignes de bus sont identifiés dans l'ordre qui détermine l'itinéraire de la ligne.

Dépôt. Le lieu qui jouera le rôle de dépôt dans le réseau est déterminé.

À l'aide de ce fichier entrant, le générateur a toute l'information lui étant nécessaire afin de positionner le dépôt et les arrêts de bus composant le réseau.

3.1.3 Paramètres du générateur

En plus du fichier entrant, d'autres paramètres sont disponibles permettant de modifier les caractéristiques des problèmes créés. Ces paramètres définissent les caractéristiques des quarts de travail des conducteurs afin de créer des contraintes similaires à celles retrouvées en industrie. Les valeurs des paramètres utilisées sont celles suggérées dans Freling [31]. Les valeurs de ceux-ci se trouvent en annexe. Ces paramètres sont décrits ci-dessous :

WalkExtraTime. Temps supplémentaire donné à un déplacement à pied comparativement à un déplacement en bus pour un déplacement ayant la même distance.

SignOnTime et SignOfTime. Donnent un délai à chaque conducteur respectivement en début et fin de quart de travail.

NbOfPieces. Donne le nombre de pièces de travail minimum et maximum qu'un conducteur peut réaliser au courant d'un quart de travail.

MinDutyLength et MaxDutyLength. Déterminent la durée maximale et minimale d'un quart de travail d'un conducteur.

MinPieceLength et MaxPieceLength. Déterminent la durée maximale et minimale d'une pièce de travail.

MinBreakLength et MaxBreakLength. Délimitent le temps que peut prendre une pause pour un conducteur. Ces pauses sont prises entre deux pièces de travail.

MinWorkTime et MaxWorkTime. Déterminent le temps minimum et maximum qu'un employé peut travailler sans pause.

BusFixedCost et DriverFixedCost. Déterminent respectivement le coût associé à chaque autobus et conducteur nécessaire afin de couvrir tous les voyages.

BusTimeAwayCost. Détermine un prix associé au temps d'utilisation en minutes des bus.

Deux ensembles de paramètres sont utilisés afin de permettre deux types de quarts de travail différents. Le premier type représente le quart d'un conducteur ne prenant aucune pause et le deuxième représente le quart où un conducteur travaille assez longtemps afin de nécessiter une pause selon la convention collective. Le problème créé par le générateur est donc un problème composé de deux réseaux distincts, un réseau pour chaque type de quart de travail décrit par les paramètres ci-haut.

3.1.4 Attribution des lignes et heures de départ de chaque voyage

En plus de l'information détenue dans le fichier entrant, le générateur a besoin de déterminer une heure de départ pour chaque voyage. Le nombre de voyages devant être généré par le générateur est fourni en paramètre. Pour chaque voyage, une pige au hasard pondérée par les poids d'un paramètre nommé $TripFrequence$ est effectuée. Ce paramètre contient une valeur pour chaque heure de la journée. Pour déterminer la minute de départ du voyage, un chiffre entre 0 et 59 est tiré au hasard. Chaque minute a le même poids, donc chaque minute a la même chance d'être tirée. Ainsi, un temps de départ est déterminé pour chaque voyage demandé au générateur.

La prochaine étape du générateur est d'associer les différents voyages à des lignes. Le générateur réalise ceci en associant de façon aléatoire une ligne du problème à chacun des voyages. Chaque ligne a le même poids, contrairement aux heures de départ. Ainsi, il y a à peu près autant de voyages assignés à chacune des lignes.

3.1.5 Problématiques associées au générateur existant

Les problèmes générés par le générateur de base n'arrivent pas à répondre aux besoins de ce projet. En effet, une qualité désirée est que ces problèmes ressemblent le plus possible aux problèmes résolus en industrie afin d'évaluer le potentiel d'accélération réel de la méthode. La méthode de détermination des temps de départ des voyages et l'attribution de ceux-ci aux différentes lignes peuplant le réseau engendre de nombreuses disparités entre les problèmes créés par le générateur et ceux résolus en industrie.

TABLEAU 3.1 Nombre de voyages par heure trouvé à partir des données de la STM de deux lignes différentes

Heures	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Voyages	0	0	0	0	0	0	5	7	5	5	5	5	4	4	5	6	5	5	5	4	4	4	4	0

Heures	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Voyages	0	0	0	0	0	0	4	6	2	0	0	0	0	0	0	2	5	3	2	0	0	0	0	0

Dans le tableau 3.1, on peut voir les fréquences de deux lignes réelles tirées de la STM pour chaque heure de la journée. On peut clairement voir que les deux lignes ont des fréquences très différentes les unes des autres. Une ligne étant en fonction toute la journée contrairement à l'autre étant en fonction seulement lors des heures de pointe. Il serait impossible de représenter ces fréquences réelles à l'aide du générateur existant.

La fréquence des départs associée à chaque ligne d'un réseau est déterminée par la demande pour le service sur le territoire que la ligne dessert [2]. De plus, les minutes de départ des voyages ne sont pas distribuées de façon aléatoire comme il est le cas avec le générateur. En effet, les départs sont espacés afin d'assurer une certaine distribution de l'offre de service. Les départs sont en général distribués de façon assez constante à l'intérieur d'une même heure avec, par exemple, un départ à toutes les 20 minutes.

3.2 Le générateur modifié

Pour créer des données ressemblant le plus possible à celles utilisées en industrie, il a été nécessaire de modifier le générateur. Celui-ci a été changé afin qu'il soit possible de lui fournir une liste contenant l'heure de départ et la ligne exacte de chaque voyage. Il est donc maintenant possible de fournir une liste de départs ressemblant à une liste de départs réelle.

Un ensemble de données provenant de la STM a été utilisé afin d'extraire des informations statistiques à propos des lignes et des horaires associés. Les données ont été utilisées pour déterminer des heures de départ pour chacun des réseaux et ainsi créer des données artificielles ressemblant le plus possible aux lignes réelles. Les coordonnées des arrêts de bus formant les différents réseaux générés ont aussi été déterminées à l'aide des données de la STM afin de créer des réseaux similaires aux réseaux réels en industrie. Le nouveau processus de création des problèmes sera expliqué plus en détail dans les sous-sections suivantes.

3.2.1 Présentation des données utilisées

Les données nous ayant été fournies sont divisées en deux groupes distincts. Le premier contient tous les trajets de bus planifiés pour les dates du 26 août 2016 au 28 octobre 2016. Pour chacun de ces trajets, ses points de départ et d'arrivée sont fournis tout comme ses heures de début et de fin à la minute près. Avec ces données, il a été possible de déterminer les horaires de départ prévus pour chaque ligne constituant le réseau de Montréal, et ce pour chaque heure. Ces données nous ont aussi permis de situer le dépôt associé à chacune des lignes. Le deuxième ensemble de données fournies a été celui constitué des données réelles déterminées à l'aide de GPS positionnés à l'intérieur des véhicules. Ces données ont permis de déterminer le trajet de chaque ligne. Toutefois, pour l'ensemble de données réelles, seul un sous-ensemble de bus était équipé de GPS, ce qui a restreint la quantité de lignes pour lesquelles il a été possible de déterminer le trajet. La figure 3.1 représente un ensemble de lignes recréées à partir des données de la STM. Les lignes représentées sont les lignes associées à un même dépôt et pour lesquelles il a été possible de recréer l'itinéraire. Les unités sur les axes sont les coordonnées fournies par la STM associées à chaque arrêt de bus modifiées afin que la valeur minimale soit 0 pour les deux axes. Le dépôt associé aux lignes est représenté par un point rouge.

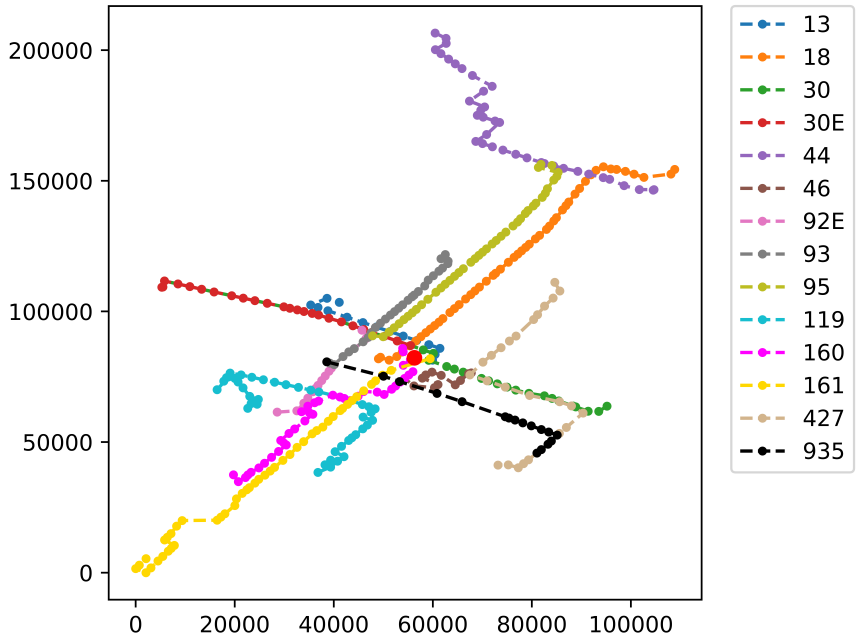


FIGURE 3.1 Ensemble de lignes de la STM associées à un même dépôt

Des réseaux ont pu être créés à partir des lignes pour lesquelles il a été possible d'identifier leurs itinéraires. Pour chaque réseau, un dépôt a été sélectionné afin que chaque ligne faisant

partie du réseau soit associée au même dépôt. Quatre lignes ont été sélectionnées à partir desquelles un fichier entrant a été créé. En ajoutant un équivalent dans la direction inverse de chaque ligne, chacun des réseaux créés est constitué de huit lignes distinctes. Ces lignes ont été sélectionnées afin d'engendrer le plus d'échanges de conducteurs possible. Il a été supposé que plus il y a d'intersections entre les lignes dans le réseau, plus il y allait avoir d'échange de conducteurs. Cette hypothèse se justifie par le fait que plus la distance entre deux points de relève est petite, plus il est facile d'y faire des échanges de conducteurs. Pour chaque ligne, deux à trois arrêts ont été déterminés comme points de relève. Les arrêts situés aux intersections de plusieurs lignes ou à une station de métro ont été priorisés lors de la sélection des points de relève. Un réseau artificiel créé à partir des données de la STM, pour lequel un fichier entrant a été créé, est représenté à la figure 3.2. La figure représente la position de chaque arrêt de bus et le trajet des différentes lignes. Le dépôt associé aux lignes est représenté par un point rouge.

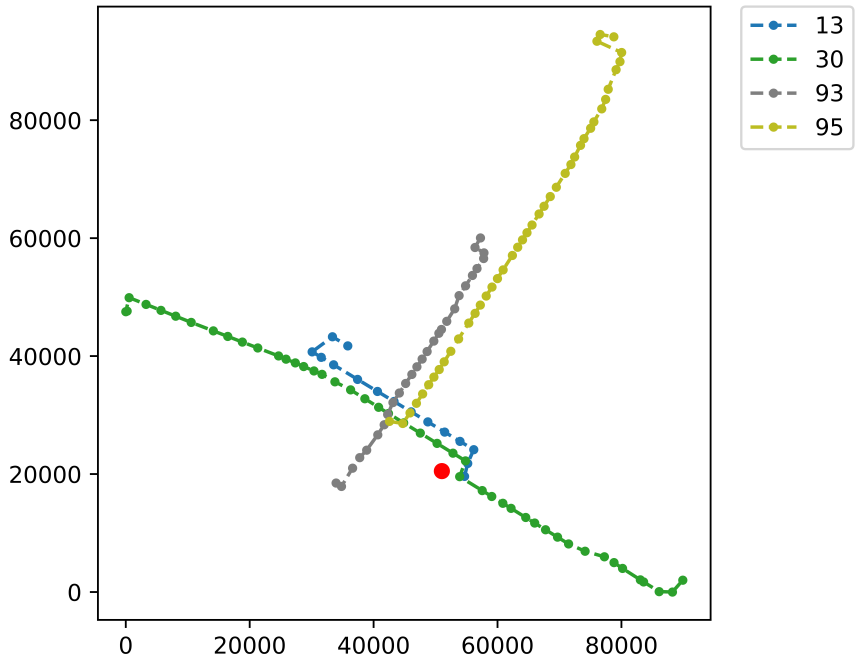


FIGURE 3.2 Sous-ensemble de lignes de la STM associées à un même dépôt

3.2.2 Attribution des heures de départ pour chaque ligne

À l'aide du générateur modifié acceptant une liste de temps de départ pour chaque ligne et un fichier entrant contenant les coordonnées d'un réseau de la STM, il est possible de créer des problèmes ressemblant à des problèmes réels résolus en entreprise. Une procédure permettant

de créer un grand nombre de problèmes basés sur le même réseau de transport a été élaborée. Les problèmes résultants ont à la fois une distribution des voyages sur les différentes lignes et heures de départ similaires aux vrais problèmes résolus par les sociétés de transport et sont différents les uns par rapport aux autres. Cette procédure est composée des quatre étapes énoncées ci-dessous.

1. Pour chaque ligne composant le réseau pour lequel un grand nombre d'instances doit être créé, leur nombre de départs par heure est déterminé en lisant les données de la STM. Un tableau tel que le tableau 3.1 est créé.
2. Le nombre de départs le plus élevé est identifié comme étant la valeur *valeur_max*. Celle-ci est encerclée dans le tableau 3.2.

TABLEAU 3.2 Nombre de voyages par heure trouvé à partir des données de la STM de deux lignes différentes

Heures	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Voyages	0	0	0	0	0	0	5	(7)	5	5	5	5	4	4	5	6	5	5	5	4	4	4	4	0

Heures	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Voyages	0	0	0	0	0	0	4	6	2	0	0	0	0	0	0	2	5	3	2	0	0	0	0	0

3. Un paramètre *délai_min* est fourni au générateur. Cette valeur dicte le délai minimal entre les départs sur une même ligne lors de la création des départs par le générateur. Cette valeur donne donc le délai entre les départs pour la ligne à l'heure associée à la *valeur_max*. Les autres délais entre les départs sont déterminés par la formule $délai = \frac{valeur\_max}{nbr\ voyages} * délai\_min$, où *nbr voyages* est le nombre de voyages sur une ligne à l'heure pour laquelle on veut obtenir le délai entre les voyages. Le tableau 3.3 représente les délais trouvés à partir des valeur dans le tableau 3.2 pour chaque heure et ligne si le paramètre *délai_min* prend la valeur de 10.
4. Ainsi à l'aide des délais déterminés à l'étape 3, le générateur crée une liste de départs pour chaque ligne. Le délai entre chaque départ est la valeur associée à la ligne et à l'heure déterminée lors de l'étape précédente.

Cette nouvelle méthode reflète mieux les horaires réels. En effet, les minutes associées à chaque départ ne sont pas tirées au hasard. Elles sont réparties uniformément au long de la même période. Le délai entre chaque départ est modulé par la demande de la période. Aussi, chaque ligne a des fréquences différentes les unes par rapport aux autres, et ce pour

TABLEAU 3.3 Délai en minutes entre les départs sur deux lignes différentes

Heures	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Délai	0	0	0	0	0	0	14	(10)	14	14	14	14	18	18	14	12	14	14	14	18	18	18	18	0

Heures	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Délai	0	0	0	0	0	0	18	12	35	0	0	0	0	0	0	35	14	23	35	0	0	0	0	0

chaque heure de la journée. De plus, les fréquences relatives pour chaque heure et ligne ont été extraites des données de la STM. Toutefois, une faiblesse de ce nouveau générateur est qu'il n'est plus possible de déterminer au préalable combien de voyages seront créés. Par contre, il est facile de trouver une valeur appropriée au paramètre *délai_min* générant un nombre de voyages assez proche du nombre de voyages désirés. Afin de créer un grand nombre de données, un paramètre a été ajouté permettant de modifier légèrement le tableau déterminé à l'étape 1 contenant les nombres de départ tirés des données de la STM. Une nouvelle valeur est déterminée pour chaque valeur du tableau en y additionnant le chiffre retourné par une loi uniforme discrète retournant la valeur de -1, 0 ou 1. La valeur retournée est additionnée à la valeur représentant le nombre de voyages dans le tableau. Ceci permet de légèrement modifier les fréquences déterminant les temps de départ du problème tout en gardant les fréquences relatives entre les différentes lignes et heures proches de celles tirées de la STM.

3.2.3 Création de deux jeux de données

À l'aide du générateur modifié, deux jeux de données ont été créés. Sur ces deux jeux de données, le potentiel d'accélération de l'utilisation d'agrégations basées sur les sorties de bus brisées aux endroits où il y a échange de conducteurs dans la solution de la relaxation linéaire au nœud 0 sera évalué au chapitre 5. Sur ces deux mêmes jeux de données, le potentiel d'accélération des agrégations basées sur les sorties de bus brisées aux endroits où un échange de conducteurs est prédit par apprentissage automatique sera évalué au chapitre 6.

Chacun des deux jeux de données est composé de 12 problèmes pour un total de 24 problèmes différents. Le premier ensemble A contient des problèmes légèrement plus grands en termes de voyages que ceux de l'ensemble B. Les tailles maximales, moyennes et minimales en nombre de voyages des problèmes composant les jeux de données sont représentées au tableau 3.4.

Tableau 3.4 Taille des problèmes en voyages composant les deux jeux de données

	A	B
Maximum	1136	932
Moyenne	1075	893
Minimum	957	822

3.3 Conclusion sur la création de données

En conclusion, un outil permettant de générer des réseaux artificiels de tailles variables ayant des caractéristiques inspirées du réseau de la STM a été créé. Celui-ci permet d'obtenir un nombre important de données en vue de l'étape d'apprentissage automatique. Les problèmes résultants sont différents, mais ils ont des fréquences relatives entre les heures et les différentes lignes les composants semblables à celle du réseau de la STM. Deux jeux de données composés de 12 problèmes chacun ont été créés.

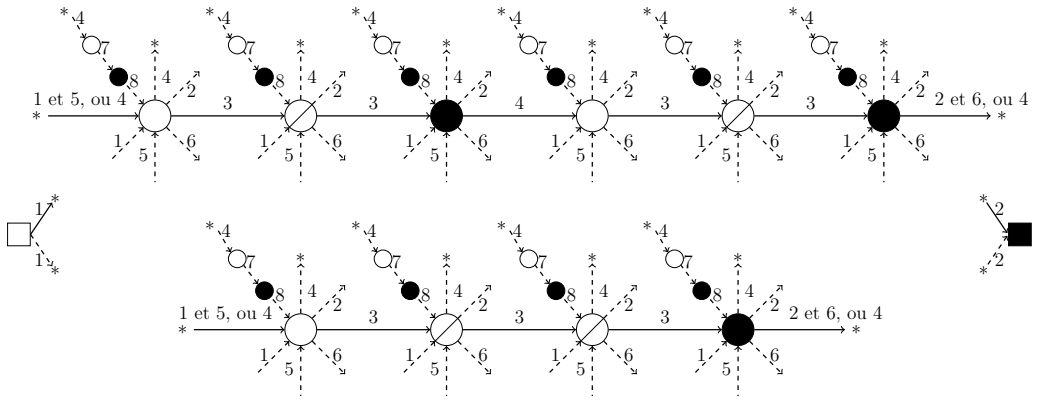
CHAPITRE 4 MODÉLISATION ET RÉSOLUTION DES PROBLÈMES

Ce chapitre présentera les réseaux modélisant les DSP. La méthode de génération de colonnes ayant été utilisée pour résoudre les DSP sera présentée tout comme le problème maître et les sous-problèmes associés au problème. La méthode d'agrégation dynamique de contraintes servant à accélérer la résolution des DSP sera aussi décrite tout comme la méthode d'agrégation dynamique de contraintes à multiphases.

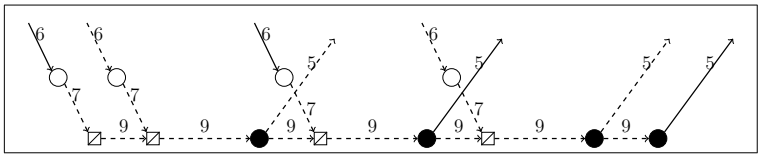
4.1 Description des réseaux

Les VSP sont modélisés sous la forme de simples problèmes du flot à coût minimum. La résolution de ceux-ci nous permet d'obtenir les trajets de bus qui serviront à créer les DSP leur étant associés. La structure du réseau du DSP est représentée à la figure 4.1.

L'architecture du réseau utilisée est fortement inspirée de celle décrite dans l'article Haase et al. [29]. Chaque nœud du graphe représente un endroit précis à un temps donné. Les lieux associés aux nœuds peuvent être des points de relève ou le dépôt, l'endroit où les conducteurs commencent et terminent leur quart de travail. Les arcs représentent un mouvement entre deux nœuds. Les tâches sont situées sur les arcs. Il y a une tâche par segment. Ce mouvement peut être physique et temporel lorsque l'arc représente le déplacement entre les deux lieux différents associés à chacun de ses nœuds, ou seulement temporel lorsque les deux nœuds associés à l'arc sont au même endroit physique. Dans ce cas, l'arc représente une attente ou une pause. Pour chaque voyage, un ensemble de nœuds est créé. Cet ensemble contient trois types de nœuds différents : les nœuds $start\_of\_trip$ sont les nœuds de départ et les nœuds end_of_trip sont les nœuds finaux pour chaque voyage. Chaque voyage n'a qu'un seul nœud de chaque type. Les nœuds $relief$ sont les nœuds intermédiaires du voyage. Leur quantité peut varier selon la ligne associée au voyage. Dans l'exemple représenté par la figure 4.1, nous avons trois voyages différents. Les deux voyages du haut ont chacun un nœud $relief$ et le voyage du bas en a deux. Un ensemble d'arcs est associé à chacun de ces nœuds. Pour les nœuds $start\_of\_trip$, nous avons plusieurs arcs entrants représentant les différentes possibilités pour un conducteur de se rendre au départ d'un voyage. Le conducteur peut se rendre au nœud $start\_of\_trip$ soit au volant d'un autobus ou non. On dit qu'il conduit lorsqu'il arrive au volant d'un autobus et qu'il est à pied sinon. Tous les arcs où le conducteur conduit ont été déterminés à partir de la solution du VSP résolu au préalable. Pour les nœuds $start\_of\_trip$, les arcs entrants de conduite sont soit un arc $start\_of\_duty$ et un autre $from\_break$ ou un seul arc $inter\_trip$. Ces arcs représentent des déplacements à vide. Un arc $start\_of\_duty$ et



Arcs associés au dépôt



Légende des arcs

1	*start_of_duty*	4	*inter − trip*	7	*break*	⟶	conduite d'un autobus
2	*end_of_duty*	5	*from_break*	8	*new_p_of_w*	- - - - →	déplacements à pied
3	*d − trip*	6	*to_break*	9	*wait*	∗	arcs multiples

Légende des nœuds

source	*start_of_trip*	*relief*	*start_of_break*
sink	*end_of_trip*	*min_break*	*end_of_break*

FIGURE 4.1 Représentation du réseau du DSP

un autre *from_break* indiquent que selon la solution du VSP, l'autobus arrivant à ce nœud provient du dépôt et donc que le conducteur couvrant cet arc doit provenir de la source du réseau ou du dépôt. Un seul arc *inter_trip* indique que le bus arrivant à ce nœud provient du nœud final d'un autre voyage. Sur le réseau représenté par la figure 4.1, nous n'avons qu'un seul arc *inter_trip* entre les deux voyages du haut, ce qui signifie que selon la solution du VSP, ces deux voyages doivent être couverts un à la suite de l'autre par le même bus. Un seul arc de conduite sortant est associé au nœud *start_of_trip* qui est un arc de type *d_trip*. Les arcs *d_trip* sont les arcs reliant les différents nœuds d'un même voyage entre eux. Les arcs associés au nœud *end_of_trip* sont assez similaires. Ils ont un seul arc entrant de type *d_trip*. Pour les arcs sortants, soit deux arcs, un de type *end_of_duty* et un de type *to_break* indiquant que l'autobus ayant terminé le voyage associé au nœud retourne

au dépôt, ou un seul arc de type *inter_trip* indiquant que l'autobus complétera un autre voyage. Ces arcs représentent des déplacements à vide. Pour les nœuds $relief$, nous avons seulement un arc entrant et un arc sortant, tous les deux de type *d_trip*.

Pour les arcs tiretés associés à des déplacements à pied par les conducteurs, tous les types de nœuds associés aux voyages ont les mêmes arcs entrants et sortants. Les arcs entrants sont de types $start\_of\_duty$, $from\_break$ et $new\_p\_of\_w$ représentant respectivement trois possibilités de provenance du conducteur : la source du réseau, le dépôt et un autre point de relève. Nous avons l'équivalent pour les arcs sortants où nous avons trois arcs de type end_of_duty, to_break et $inter\_trip$ représentant les mêmes trois destinations possibles pour le conducteur. Nous avons aussi un ensemble de nœuds et arcs représentant le dépôt. On réfère à Knut et al. [29] pour une description en détail de cette partie du réseau.

4.2 Description mathématique du problème maître et des sous-problèmes

Le problème résolu est celui du DSP à un dépôt ayant une flotte de véhicules homogènes. Ce problème peut être efficacement résolu par génération de colonnes. Dans notre cas, cette approche de résolution fait appel à deux sous-problèmes différents ayant comme but de générer des quarts de travail respectant les contraintes sur les quarts de travail des employés. En industrie, ces contraintes proviennent de la convention collective déterminant les temps de travail maximaux et minimaux et les pauses obligatoires des travailleurs. Nous utilisons deux types de quarts de travail possibles ayant des contraintes distinctes telles que présentés dans Freling [31], les quarts de travail composés d'une seule pièce de travail et ceux composés de deux. Leurs caractéristiques sont décrites au tableau 4.1. Comprenant les quarts de travail générés par les sous-problèmes respectant les contraintes des conducteurs, le problème maître s'assure que les contraintes de partitionnement soient respectées afin que toutes tâches soient exactement couvertes par un conducteur.

Posons K comme étant l'ensemble des différents types de quarts de travail et W l'ensemble des trajets à couvrir par des conducteurs. L'ensemble des différents quarts de travail possibles de type k est dénoté par P^k. Pour chaque quart de travail $p \in P^k$, nous lui associons un coût c_p^k et un paramètre a_{wp}^k prenant la valeur 1 si le quart de travail couvre la tâche $w \in W$. Le problème maître fait appel aux variables binaires θ_p^k, $k \in K$, $p \in P^k$. Une telle variable prend la valeur 1 si le quart de travail p fait partie de la solution.

$$\min_{\theta_p^k} \quad \sum_{k \in K} \sum_{p \in P} c_p^k \theta_p^k \tag{4.1}$$

$$\text{s.c.} \quad \sum_{k \in K} \sum_{p \in P} a_{wp}^k \theta_p^k = 1 \quad \forall w \in W, \tag{4.2}$$

$$\theta_p^k \in \{0, 1\} \quad \forall p \in P^k, k \in K \tag{4.3}$$

La fonction objectif (4.1) consiste à minimiser les coûts engendrés par les quarts de travail faisant partie de la solution, tout en s'assurant que chaque tâche soit couverte par exactement un seul conducteur selon les contraintes (4.2). Les coûts associés aux quarts de travail générés sont composés de deux types de coûts différents : Une composante fixe et une variable. Tel que présenté dans Freling [31], ces valeurs ont été choisies de façon à ce que le nombre de conducteurs faisant partie de la solution soit minimisé, et qu'à nombre de conducteurs égal, le nombre de minutes passées à l'extérieur des dépôts par les conducteurs soit minimisé.

Les sous-problèmes sont des problèmes de plus courts chemins ayant des contraintes de ressources sur les nœuds du réseau présenté à la section 4.1. Ces problèmes sont résolus par un algorithme d'étiquetage. Les contraintes sur les ressources sont affichées dans le tableau 4.1 et décrites ci-dessous. Les valeurs utilisées sont les mêmes que celles utilisées dans Knut et al. [29].

Tableau 4.1 Caractéristiques des deux types de quarts de travail

Type 1	Minimum	Maximum
Nbr. de pièces	1	1
Durée totale (min)	15	300

Type 2	Minimum	Maximum
Nbr. de pièces	2	2
Durée totale (min)	45	600
Durée pièce (min)	15	300
Durée pause (min)	15	90
Durée travaillée (min)	30	480

Nbr. de pièces. Le nombre de pièces de travail qu'un conducteur peut faire durant son quart de travail.

Durée totale. La durée totale du quart de travail du conducteur.

Durée pièce. La durée de chaque pièce de travail faite par un conducteur.

Durée pause. Le temps de pause accordé à un conducteur. Une pause est obligatoire entre deux pièces de travail.

Durée travaillée. La durée pour un conducteur passée à conduire un autobus.

Afin de contraindre les quarts de travails à prendre ces valeurs, sept ressources différentes ont été utilisées.

min_p_of_w_nb et max_p_of_w_nb. Ces ressources permettent de fixer le nombre de pièces de travail effectuées par le conducteur.

max_duty_length. Contraint par le haut la durée totale travaillée par le conducteur.

max_break_length. Contraint par le haut la durée totale de la pause du conducteur.

max_work_time. Contraint par le haut la durée totale travaillée du conducteur.

min_p_of_w_length et max_p_of_w_length. Contraint la durée de chaque pièce de travail

Chaque arc composant le réseau a donc une consommation de ressource lui étant associée. Chaque nœud a un intervalle de valeurs permises pour chaque ressource de façon à ce que tous les quarts de travail générés par le sous-problème respectent les contraintes présentées au tableau 4.1.

Il est intéressant de noter qu'il n'est pas nécessaire d'avoir une ressource pour la durée minimale de pause, car celle-ci est directement prise en compte par la structure du réseau. De plus, un quart de travail faisant partie de la solution optimale ne peut contenir qu'un trajet contenant seulement des arcs représentant de la marche, car celui-ci ne couvrait aucune tâche et aurait un coût positif. Ainsi, chaque quart de travail de type 1 contient une pièce de travail et chaque quart de travail de type 2 contient deux pièces de travail et une pause. De cette façon, les contraintes sur les durées minimales de la durée totale et du temps travaillé sont redondantes avec les contraintes sur le nombre de pièces de travail minimum et de pause minimale [29]. De plus, on voit que pour les quarts de travail de type 1, la seule ressource nécessaire au sous-problème est la ressource sur la durée totale et que tous arcs et nœuds représentant des pauses peuvent être retirés, ce qui simplifie fortement le sous-problème.

On peut facilement constater que ce problème est particulièrement propice à la dégénérescence. En effet, le sous-problème a tendance à générer des quarts de travail couvrant de nombreuses tâches. Ceci est dû au fait que la valeur de la fonction objectif qu'essaie de minimiser le sous-problème est causée en majorité par le coût fixe associé à chaque employé

supplémentaire. Ainsi, le sous-problème a tendance à générer des quarts de travail où les conducteurs travaillent longtemps et couvrent de nombreuses tâches. Les colonnes générées par le sous-problème contiennent donc un grand nombre d'éléments non nuls. Nous avons ainsi une base contenant un grand nombre de variables dégénérées et donc l'algorithme exécute plusieurs pivots dégénérés, ce qui ralentit fortement la résolution du problème.

4.3 Description de la méthode d'agrégation dynamique de contraintes

Il existe une quantité importante de méthode d'accélération pour la génération de colonnes. La méthode d'agrégation dynamique de contraintes est particulièrement efficace lorsqu'elle est utilisée dans le cadre de la résolution de DSP. Elle est décrite en détail dans El Hallaoui et al. [24]. Les auteurs de cet article ont démontré que son application peut réduire la durée de la résolution des problèmes par un facteur de 8 pour des instances comportant jusqu'à 1600 contraintes de partitionnement. La méthode est itérative et consiste à fortement réduire la quantité de contraintes du partitionnement du modèle (4.1) tout en ajustant d'itération en itération les contraintes considérées. Il est toutefois important de mentionner que le potentiel d'accélération accordé par la méthode d'agrégation dynamique de contraintes dépend fortement de la qualité des agrégations fournies au modèle. En effet, si les agrégations fournies sont de mauvaise qualité, ce qui veut dire que les tâches faisant partie d'une même agrégation ne devraient pas être couvertes par un même quart de travail dans la solution optimale, il sera nécessaire de passer par un processus de désagrégation lors de la résolution, ce qui va ralentir la vitesse de résolution.

Une amélioration à la méthode d'agrégation dynamique de contraintes a été présentée dans El Hallaoui et al. [32] s'intitulant la méthode à multiphases d'agrégation dynamique de contraintes. Celle-ci pousse le potentiel d'accélération de la méthode d'agrégation originale encore plus loin et permet d'accélérer celle-ci par un facteur de 4,5. Dans le cadre de la méthode originale, la partition initiale du problème est désagrégée soit lorsque le coût réduit des variables incompatibles générées par les sous-problèmes est nettement meilleur que celle respectant la partition, ou lorsqu'aucune variable compatible n'est générée par le sous-problème. Le critère pour déterminer le moment quand désagréger est défini par les équations (4.4) et (4.5). La notation utilisée représente :

P''_Q : L'ensemble des colonnes respectant la partition Q généré à une itération de génération de colonnes.

$\bar{P'_Q}$: L'ensemble des colonnes ne respectant pas la partition Q généré depuis le début de la résolution du problème.

$\bar{c_p}$: Le coût réduit négatif de la colonne p.

λ : Un paramètre prenant une valeur entre 0 et 1.

$$\text{si } \{p \in \bar{P'_Q} : \bar{c_p} < 0\} \neq \emptyset \text{ et} \tag{4.4}$$

$$r = \frac{\min_{p \in P''_Q} \bar{c_p}}{\min_{p \in \bar{P'_Q}} \bar{c_p}} < \lambda \tag{4.5}$$

Ainsi, il y aura désagrégation lorsque la valeur de r devient plus petite que λ après une itération de génération de colonnes. La désagrégation va se faire afin de rendre compatibles les colonnes incompatibles ayant le coût réduit minimum. Un désavantage important de cette méthodologie est que la désagrégation ne prend pas en compte à quel point les colonnes ne sont pas compatibles. Une colonne ayant un coût réduit très négatif pourrait toutefois être très incompatible avec la partition et donc devrait nécessiter de fortement la désagréger. Ainsi, on augmente fortement le nombre d'agrégations dans la partition, ce qui augmente directement le nombre de contraintes de partitionnement et diminue donc la vitesse de résolution du problème.

C'est ici que la méthode à multiphases offre une alternative. La méthode utilise la notion de colonnes incompatibles. Pour une partition Q donnée contenant un ensemble L d'agrégations, les colonnes $s \in S$ sont dites compatibles avec l'agrégation $l \in L$ si ces colonnes couvrent l'ensemble des tâches faisant partie de l'agrégation l ou aucune d'entre elles. Ainsi, si une colonne s est compatible avec l'entièreté des agrégations $l \in L$, on dit que cette colonne est compatible avec la partition Q. La méthode à multiphases offre une nouvelle définition qui est le nombre d'incompatibilités d'une colonne. Cette valeur pour chaque colonne est déterminée selon une nouvelle ressource introduite aux sous-problèmes, la ressource d'incompatibilité. Ainsi, le nombre d'incompatibilité est comptabilisé tout au long du chemin généré par le sous-problème. Pour chaque colonne générée, nous avons un nombre d'incompatibilités k lui étant associé.

La méthode sépare la résolution du problème en plusieurs phases, en commençant par la phase 0. On augmente progressivement le nombre d'incompatibilités dans les colonnes introduites au problème. Ainsi à la phase 0, seules les colonnes ayant 0 incompatibilité peuvent être générées et donc introduites au problème maître restreint. À la phase 1, les colonnes ayant une seule incompatibilité peuvent être introduites et ainsi de suite jusqu'à la phase $\lfloor \frac{|W|}{2} \rfloor$ où W est l'ensemble des tâches à couvrir. De cette manière, les colonnes introduites sont prioritairement peu incompatibles, ce qui diminue moins rapidement la quantité d'agrégations

et donc permet de mieux maintenir le potentiel d'accélération de la méthode d'agrégation dynamique de contraintes.

Dans le cadre de résolution de DSP, il est assez facile de déterminer de bonnes agrégations de départ qui ne nécessiteraient pas trop de désagrégation. En effet, on remarque que les conducteurs dans la solution n'ont pas tendance à souvent changer de véhicules. Ainsi, utiliser les sorties de bus trouvées lors de la résolution du VSP comme agrégations initiales donne déjà un très bon potentiel d'accélération. Ces agrégations basées sur les sorties de bus trouvées au préalable de la résolution du DSP sont les agrégations auxquelles les temps de résolution obtenus avec la méthode que nous proposons seront comparés.

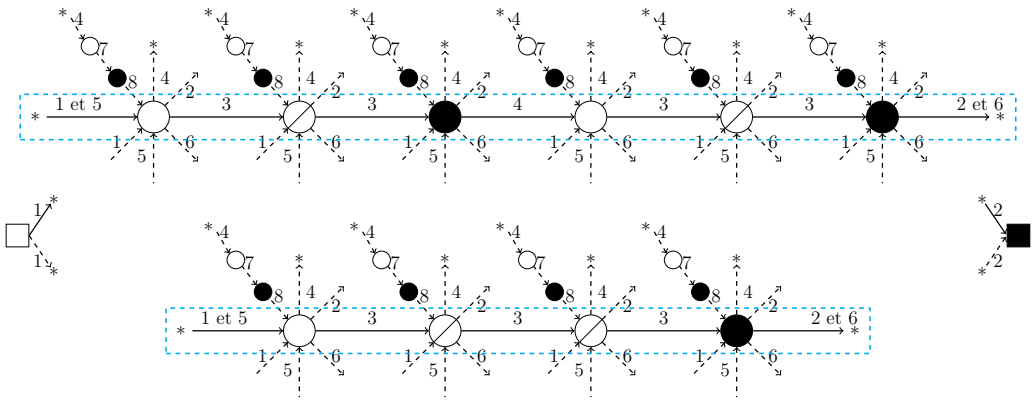


FIGURE 4.2 Méthode d'agrégation selon la solution du VSP

Nous voyons un exemple d'une agrégation basée sur la solution du VSP à la figure 4.2 où les encadrés en cyan représentent deux agrégations faisant partie de la partition initiale. La figure représente les agrégations dans la situation où selon la solution du VSP, nous avons une sortie de bus couvrant les 7 arcs du haut représentant deux voyages et une sortie de bus effectuée par un bus différent qui couvre les 5 tâches faisant partie du voyage du bas. On voit qu'on passe de 12 tâches, et donc 12 contraintes de partitionnement, à seulement deux à l'aide de la méthode d'agrégation dynamique de contraintes.

CHAPITRE 5 TESTER LE POTENTIEL EN OPTIMISATION

Dans ce chapitre, les performances de la méthode d'agrégation dynamique de contraintes seront évaluées dans un contexte de résolution de DSP. Particulièrement lorsque celle-ci est utilisée pour agréger les tâches selon les sorties de bus déterminées au préalable par la résolution d'un VSP. Les performances de cette méthode telle que décrite dans El Hallaoui et al. [24] seront comparées au scénario où les agrégations sont créées de façon parfaite. La création et la définition des agrégations parfaites seront décrites et les résultats obtenus à l'aide de celles-ci seront présentés. D'autres méthodes ayant été utilisées afin d'accélérer la résolution seront décrites et évaluées telles que la méthode avec arcs restreints, la méthode utilisant de la dominance exacte sur la ressource d'incompatibilité et la méthode brisant les agrégations à d'autres endroits que les échanges de conducteurs de la solution de la relaxation linéaire au nœud 0. L'accélération apportée à la résolution de DSP dans son ensemble incluant le *branch and price* sera aussi évaluée. Ce chapitre nous permettra donc de déterminer le potentiel d'accélération d'une méthode créant une partition initiale pour les problèmes de DSP à l'aide d'apprentissage automatique dans le cas où les prédictions seraient parfaites. Les différentes stratégies testées nous permettront de déterminer les meilleures à utiliser lors de l'utilisation d'algorithme d'apprentissage automatique.

5.1 Agrégations parfaites

5.1.1 Modifications apportées aux agrégations basées sur les sorties de bus

Avant de tester la création des agrégations par algorithme d'apprentissage automatique, nous avons testé le potentiel de la méthode en créant des agrégations dites parfaites. L'accélération trouvée avec ces agrégations parfaites est une borne supérieure pour l'accélération qui pourra être obtenue avec l'apprentissage automatique. Ces agrégations parfaites ont été déterminées en résolvant les problèmes créés par le générateur décrit au chapitre 3 en utilisant des agrégations basées sur les sorties de bus. À partir des solutions de la relaxation linéaire au nœud 0 de ces problèmes, nous avons identifié pour chacun des problèmes sur quels points de relève il y avait la présence d'un échange de conducteurs. À l'aide de cette information, des agrégations parfaites ont été créées et les temps de résolution de ces problèmes ont été comparés à ceux obtenus en utilisant les agrégations basées sur les sorties de bus.

Nous voyons un exemple de telles agrégations à la figure 5.1 où les encadrés en cyan représentent trois agrégations faisant partie de la partition initiale. Les arcs en bleu foncé sont ceux

faisant partie de la solution. On peut voir que selon la solution, il y a échange de conducteurs au troisième nœud du voyage du haut. Ainsi, la partition initiale telle que représentée à la figure 4.2 basée sur les sorties de bus sera coupée à cet endroit. Par conséquent, la nouvelle partition représente mieux la solution optimale au problème et nécessitera donc moins de désagrégations afin d'atteindre la solution optimale.

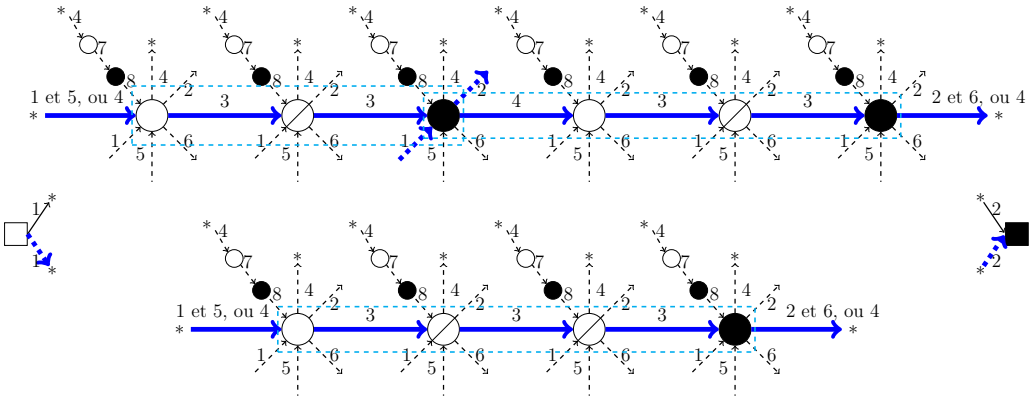


FIGURE 5.1 Agrégations brisées où il y a échange de conducteurs

5.1.2 Résultats

Le tableau 5.1 présente les temps de résolutions obtenus en utilisant des agrégations parfaites comparés aux temps obtenus à l'aide d'agrégations basées sur les sorties de bus afin d'atteindre la solution de la relaxation linéaire du nœud 0. Les pourcentages d'accélération maximums, moyens et minimums sont présentés.

TABLEAU 5.1 Pourcentages d'accélération obtenus à l'aide d'agrégations parfaites

	A	B
Maximum	83.5	78.0
Moyenne	63.3	60.0
Minimum	49.7	35.4

Nous constatons une accélération significative lorsque des partitions dites parfaites sont utilisées. Le temps de résolution est diminué de moitié pour la très grande majorité des instances, et ce pour les deux ensembles de problèmes de tailles différentes. Ces résultats nous informent d'une borne supérieure sur le pourcentage d'accélération pouvant être atteint à l'aide de l'apprentissage automatique. À l'aide d'un algorithme permettant de parfaitement prédire s'il y a échange de conducteurs sur tous les points de relève des problèmes, l'accélération obtenue

par rapport à la méthode d'agrégation de base serait celle présentée dans ce tableau. Le coût associé à la solution trouvée en utilisant les nouvelles agrégations est le même que celui trouvé à l'aide des agrégations de base.

5.2 Méthode avec arcs restreints

5.2.1 Description de la méthode

Une autre méthode d'accélération a été testée. Celle-ci consiste à restreindre le solveur à seulement utiliser les arcs respectant les partitions lors de la phase 0 de la résolution du problème avec agrégation dynamique de contrainte à multiphases. Ceci a comme effet de nettement diminuer le nombre d'arcs devant être considéré par le solveur lors de la résolution du sous-problème. En effet, seuls les chemins respectant la partition peuvent faire partie d'une solution. L'accélération trouvée avec la restriction d'arcs sur des agrégations parfaites est une borne supérieure pour l'accélération qui pourra être obtenue avec l'apprentissage automatique en utilisant la restriction d'arcs.

On voit par exemple à la figure 5.2 que tous les arcs identifiés en rouge ne respectent pas la partition formée de trois agrégations. Ainsi, ces arcs ne feraient pas partie des sous-problèmes en phase 0. On voit que cette méthode diminue significativement la quantité d'arcs faisant partie du réseau.

Cette méthode n'a pas d'impact sur la résolution du problème maître restreint en phase 0. En effet, celui-ci considère déjà seulement les chemins compatibles avec la partition due à l'agrégation de contraintes. Cette méthode d'accélération a plutôt un impact sur les sous-problèmes. En effet, dans la méthode d'agrégation de contraintes originale, les sous-problèmes génèrent des colonnes potentielles ayant un coût réduit négatif sans prendre en compte la partition. Après avoir généré un ensemble de colonnes, seules les colonnes compatibles avec la partition sont incorporées au problème maître. Avec la méthode de restriction d'arcs, le sous-problème n'est composé que d'arcs compatibles avec la partition. Ceci accélère la résolution du problème de deux façons. Tout d'abord, les sous-problèmes sont plus petits, ce qui accélère sa résolution. De plus, seules des colonnes compatibles avec la partition sont générées. Ainsi, le temps passé à générer des colonnes incompatibles est éliminé.

Cette méthode a toutefois le potentiel de ralentir la résolution. Dans la méthode d'agrégation dynamique de contraintes, les colonnes incompatibles sont utilisées afin de déterminer s'il est temps de modifier la partition. N'ayant pas accès à cette information, il n'est pas possible de déterminer quand désagréger de cette façon. La désagrégation va plutôt s'effectuer lorsque toutes les colonnes compatibles auront été générées en phase 0 et qu'on passe en phase 1.

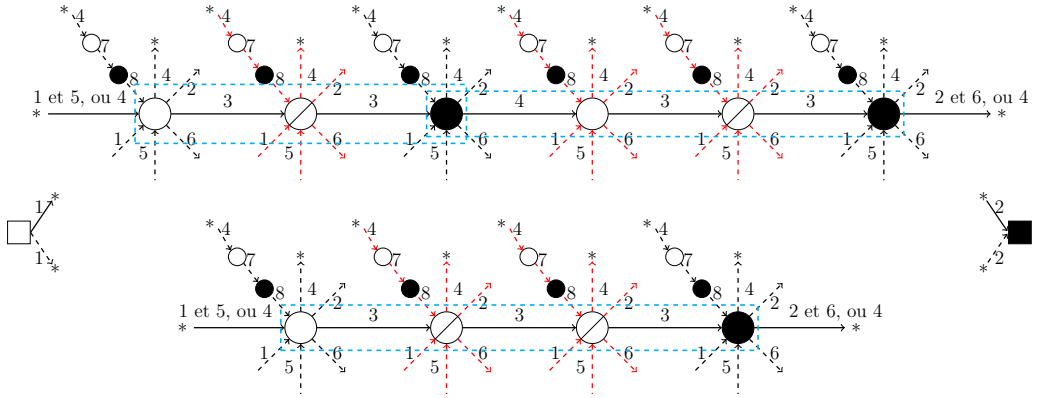


FIGURE 5.2 Méthode d'agrégation avec arcs restreints

Ceci n'est toutefois pas problématique s'il est possible d'atteindre la solution optimale en phase 0. C'est le cas lorsqu'on brise les agrégations selon les échanges de conducteurs dans la solution optimale.

5.2.2 Résultats

Le tableau 5.2 présente les temps de résolutions obtenus en utilisant des agrégations parfaites et en restrictant le solveur à seulement utiliser les arcs respectant les partitions lors de la phase 0 comparés aux temps obtenues à l'aide d'agrégations basées sur les sorties de bus. Les pourcentages d'accélération maximums, moyens et minimums sont présentés.

TABLEAU 5.2 Pourcentages d'accélération obtenus à l'aide d'agrégations parfaites et de la restriction d'arcs

	A	B
Maximum	90.2	91.6
Moyenne	77.9	79.9
Minimum	68.7	61.5

On constate que cette méthode offre un potentiel d'accélération important sur les deux jeux de données. Celui-ci est plus important que lorsque seules les agrégations parfaites sont utilisées. Les accélérations minimums à l'aide de cette méthode offrent une plus grande accélération que l'accélération moyenne lorsque seules les agrégations parfaites sont utilisées. Il faut toutefois rappeler que la partition initiale est compatible avec la solution de la relaxation linéaire au nœud 0, car la solution est utilisée pour la création des agrégations de base. Ceci ne va pas être le cas lors de la création des agrégations à l'aide d'algorithme d'apprentissage automatique où

il ne va pas nécessairement être possible d'atteindre la solution optimale en phase 0. Dans ce cas, cette méthode pourrait donc engendrer un ralentissement de la résolution, car il ne sera pas possible de changer de phase avant d'avoir engendré l'entièreté des colonnes possibles en phase 0. C'est pour cette raison que la résolution avec et la résolution sans l'utilisation de cette méthode seront testées lors de l'utilisation d'algorithme d'apprentissage automatique.

5.3 Heuristique de dominance sur la ressource d'incompatibilité

5.3.1 Description de la méthode

Lors de l'utilisation d'agrégations parfaites, il a été remarqué que pour certains problèmes, le solveur passait un certain temps dans le processus de désagrégation, et ce même si la partition initiale fournie avait été créée à partir de la solution de la relaxation linéaire au nœud 0. Ce résultat est contraire à ce qui était attendu lors de l'utilisation d'agrégations parfaites. La source de cette désagrégation a été déterminée comme étant l'utilisation d'un paramètre du solveur utilisé par défaut. Ce paramètre force l'utilisation d'une méthode de dominance heuristique dans le sous-problème sur la ressource d'incompatibilité. Lors de l'utilisation de la méthode d'agrégation dynamique à multiphases, une ressource d'incompatibilité est déterminée indiquant pour chaque quart de travail généré par le sous-problème son degré d'incompatibilité par rapport aux agrégations. Le sous-problème domine sur cette ressource seulement dans le cas où pour deux chemins différents passant par un même nœud, la dernière tâche complétée par les deux chemins est la même. Toutefois, lors de l'utilisation d'une heuristique pour la dominance, la dernière tâche complétée n'est pas mémorisée. Ainsi, deux chemins venant de terminer deux tâches différentes pourraient se dominer sur les nœuds représentant le dépôt. Comme en phase 0, seuls les chemins respectant les partitions peuvent faire partie de la solution, il est possible que pour une tâche, tous chemins couvrant celle-ci se fassent dominer par des chemins ne couvrant pas cette tâche. Il serait donc impossible de couvrir celle-ci en phase 0. Nous avons donc testé l'utilisation d'une méthode exacte de dominance où la tâche précédente est retenue. Ceci permettrait de générer des chemins couvrant toutes les tâches du problème et donc d'atteindre la solution optimale en phase 0 en utilisant une partition initiale parfaite. Toutefois, utiliser une telle méthode de dominance exacte diminuerait fortement le nombre de chemins dominés et pourrait donc ralentir la résolution du sous-problème.

Afin de déterminer si le temps perdu à désagréger est supérieur au temps gagné à utiliser une heuristique pour la dominance sur la ressource d'incompatibilité, nous avons comparé les temps obtenus sur les problèmes générés en utilisant l'heuristique à ceux obtenus par une

méthode exacte.

5.3.2 Résultats

Le tableau 5.3 présente les temps de résolutions obtenus en utilisant des agrégations parfaites et une méthode de dominance exacte sur la ressource d'incompatibilité comparés aux temps obtenus à l'aide d'agrégations basées sur les sorties de bus. Les pourcentages d'accélération maximums, moyens et minimums sont présentés.

Tableau 5.3 Pourcentages d'accélération obtenus à l'aide d'agrégations parfaites et d'une méthode de dominance exacte sur la ressource d'incompatibilité

	A	B
Maximum	57.6	72.8
Moyenne	36.5	49.4
Minimum	5.9	15.7

On constate qu'en utilisant une méthode de dominance exacte sur la ressource d'incompatibilité, l'accélération est moindre que lorsqu'on utilise une méthode de dominance heuristique. La solution optimale de la relaxation linéaire est toutefois toujours trouvée en phase 0 de la résolution. Comme le temps de résolution est toutefois plus grand, on peut conclure que le temps sauvé à ne pas avoir besoin de désagréger pour atteindre la solution optimale est moins grand que le temps perdu lors de la résolution du sous-problème à utiliser une méthode exacte.

5.4 Potentiel de la méthode en brisant les agrégations à d'autres endroits

5.4.1 Description de la méthode

Briser les agrégations basées sur les sorties de bus aux endroits où il y a un échange de conducteurs dans la solution de la relaxation linéaire au nœud 0 présente un potentiel d'accélération important. Toutefois, prédire les échanges de conducteurs peut être une tâche difficile pour un algorithme d'apprentissage automatique. En effet, pour environ 7% des points de relève, il y a échange de conducteur. Réussir à prédire un tel pourcentage faible en occurrences dans le cadre d'une tâche de classification binaire peut poser problème aux algorithmes d'apprentissage automatique.

Nous avons testé quelques idées permettant d'augmenter ce pourcentage et ainsi faciliter la tâche de l'algorithme de classification. Tout d'abord, nous avons évalué le potentiel de

briser les agrégations aux endroits où un échange de conducteurs a eu lieu dans une colonne générée par le sous-problème au lieu de briser les agrégations aux endroits où un échange de conducteurs a eu lieu dans la solution de la relaxation linéaire au nœud 0. La cardinalité de l'ensemble des colonnes générées lors de la résolution étant beaucoup plus élevée que celle de l'ensemble de colonnes faisant partie de la solution de la relaxation linéaire au nœud 0, nous pouvons nous attendre à ce que le nombre d'échanges de conducteurs soit plus élevé. De plus, comme les quarts de travail contenant ces échanges de conducteurs sont générés par le sous-problème afin de faire partie de la solution du problème maître, nous pouvons penser que les échanges faisant partie des colonnes générées offrent un certain potentiel de diminution de la valeur objectif si un échange de conducteurs a lieu à cet endroit. Il serait ainsi potentiellement avantageux de briser les agrégations initiales à ces endroits.

Nous avons aussi testé la possibilité de briser les agrégations aux endroits où un échange de conducteurs était présent dans une solution obtenue à une itération de génération de colonnes. La quantité d'agrégations brisées serait moindre que lorsqu'on brise où il y a échange dans une colonne générée. Toutefois, les échanges de conducteurs auraient potentiellement plus de chance de faire partie de la solution de la relaxation linéaire au nœud 0, ce qui pourrait apporter une meilleure accélération de la résolution.

5.4.2 Résultats

Au tableau 5.4, nous pouvons voir les pourcentages maximums, moyens et minimums de nœuds étant considérés comme ayant un échange de conducteurs sur l'ensemble de problèmes A et B. Le pourcentage d'échange de conducteurs est montré sur 3 ensembles de nœuds différents. L'ensemble #1 montre le pourcentage d'échange dans la solution optimale de la relaxation linéaire au nœud 0. L'ensemble #2 montre le pourcentage dans une solution obtenue à une itération de génération de colonnes et l'ensemble #3 montre le pourcentage dans toutes colonnes générées par le sous-problème. Comme prévu, les pourcentages varient selon l'ensemble de nœuds pour lequel les échanges de conducteurs ont été observés.

Tableau 5.4 Pourcentages de points de relève où il y a échange de conducteurs

	#1		#2		#3	
	A	B	A	B	A	B
Maximum	8.18	8.97	35.8	33.6	54.6	50.6
Moyenne	6.29	7.32	27.8	30.4	36.8	41.4
Minimum	4.06	2.90	24.5	25.9	31.9	33.4

Au tableau 5.5, nous pouvons voir l'accélération obtenue selon l'endroit où les agrégations ont été brisées. On voit que pour les groupes #2 et #3, aucune accélération n'est observée. Nous avons même un fort ralentissement de la résolution des problèmes.

Tableau 5.5 Pourcentages d'accélération dépendamment d'où sont brisées les agrégations

	#1		#2		#3	
	A	B	A	B	A	B
Maximum	83.5	78.0	−0.3	−9.6	−83.3	−41.5
Moyenne	63.3	60.0	−42.6	−54.5	−130.5	−233.7
Minimum	49.7	35.4	−77.8	−115.7	−212.8	−782.2

Ce fort ralentissement peut s'expliquer par le fait que le pourcentage de nœuds ayant des échanges de conducteurs est beaucoup trop élevé pour les ensembles #2 et #3. Ainsi, le nombre d'agrégations augmente énormément ce qui ralentit fortement le potentiel d'accélération offert par la méthode d'agrégation dynamique de contraintes. En effet, on voit au tableau 5.6 que pour le groupe #2, le nombre d'agrégations initiales augmente en moyenne de 221.7% pour les problèmes de l'ensemble A et augmente en moyenne de 224.8% pour les problèmes de l'ensemble B. Pour le groupe #3, le nombre d'agrégations initiales augmente en moyenne de 313.6% pour les problèmes de l'ensemble A et augmente en moyenne de 330.4% pour les problèmes de l'ensemble B. Le nombre de contraintes de partitionnement dans le problème maître augmente donc du même facteur, ce qui ralentit fortement sa résolution. Lors de la prédiction avec algorithmes d'apprentissage automatique, il est donc nécessaire de maintenir un pourcentage de bris assez proche de celui obtenu en brisant selon la solution optimale de la relaxation linéaire au nœud 0 afin de ne pas trop fortement diminuer le potentiel d'accélération offert par l'agrégation dynamique de contraintes.

Tableau 5.6 Augmentations en pourcentages du nombres d'agrégations dans la partition initiale par rapport au groupe # 1

	#2		#3	
	A	B	A	B
Maximum	325.5	458.7	488.3	653.7
Moyenne	221.7	224.8	313.6	330.4
Minimum	149.5	144.3	230.9	206.5

5.5 Potentiel d'accélération dans l'atteinte de la solution en nombres entiers

Nous avons démontré qu'un potentiel d'accélération existe lorsque les agrégations basées sur les sorties de bus sont brisées aux endroits où il y a échange de conducteurs dans la solution de la relaxation linéaire au nœud 0. Le temps de résolution afin d'atteindre la solution de la relaxation linéaire au nœud 0 diminue fortement. Il serait toutefois intéressant de constater une accélération dans l'atteinte de la solution en nombres entiers du problème. Nous avons donc comparé les temps de résolution afin d'atteindre la solution entière du problème en utilisant les agrégations basées sur les sorties de bus comparativement aux agrégations trouvées en brisant les agrégations aux endroits où il y a échange de conducteurs dans la solution de la relaxation linéaire au nœud 0.

Le tableau 5.7 représente les pourcentage d'accélération maximums, moyens et minimums dans l'atteinte de la solution en nombres entiers sur les différents problèmes des jeux de données A et B. Le temps de résolution en utilisant les agrégations basées sur les sorties de bus est comparé à celui obtenu avec les agrégations déterminées en brisant les agrégations aux endroits où il y a échange de conducteurs dans la solution de la relaxation linéaire au nœud 0.

Tableau 5.7 Pourcentages d'accélération dans l'atteinte de la solution en nombres entiers

	A	B
Maximum	82.1	74.0
Moyenne	60.9	50.2
Minimum	38.3	27.0

Nous constatons que le potentiel d'accélération n'existe pas seulement dans le contexte d'atteinte de la solution de la relaxation linéaire du nœud 0. Le potentiel d'accélération se maintient jusqu'à l'atteinte d'une solution optimale entière.

5.6 Conclusion

Il existe un potentiel d'accélération de la méthode d'agrégation dynamique de contraintes en brisant les agrégations basées sur les sorties de bus trouvées au préalable en résolvant un VSP, aux endroits où il y a échange de conducteurs dans la solution de la relaxation linéaire au nœud 0. Cette méthode apporte une accélération jusqu'à l'atteinte de la solution optimale en nombres entiers. La méthode offre un plus grand potentiel lorsque les arcs ne respectant pas les agrégations initiales sont retirés du problème en phase 0. Il est toutefois

possible que ce potentiel disparaisse s'il n'est pas possible d'atteindre la solution optimale en phase 0. Il est pertinent de conserver l'algorithme de dominance heuristique sur la ressource d'incompatibilité de la méthode d'agrégation dynamique à multiphases et ce même s'il n'est pas possible d'atteindre la solution optimale en phase 0. Afin de conserver un potentiel d'accélération, il est important de ne pas trop briser les agrégations basées sur les sorties de bus pour éviter de faire disparaitre le potentiel d'accélération de la méthode d'agrégation dynamique de contraintes.

CHAPITRE 6 CRÉATION DE LA PARTITION INITIALE À L'AIDE D'APPRENTISSAGE AUTOMATIQUE

Briser les agrégations basées sur les sorties de bus aux endroits où il y a échange de conducteurs permet d'accélérer la résolution de DSP tel que présenté au chapitre 5. Toutefois, afin d'exploiter ce potentiel d'accélération, il faut arriver à faire une prédiction de qualité sur les différents points de relève composant le réseau. Ne pas réussir à identifier les points de relève faisant partie de la solution de la relaxation linéaire au nœud 0 causerait l'algorithme d'agrégation dynamique de contraintes à devoir passer par le processus de désagrégation ce qui ralentirait la résolution. Prédire un échange de conducteurs sur des points de relève où il ne devrait pas en avoir briserait les agrégations de façon inutile, ce qui augmenterait le nombre d'agrégations et donc diminuerait le potentiel d'accélération de DCA. Un tel problème de prédiction sur chaque point de relève peut être exprimé selon une tâche d'apprentissage supervisée.

Dans ce chapitre, la tâche de prédiction des échanges de conducteurs sera décrite. Les différentes caractéristiques créées sur lesquelles repose la prédiction sur chaque nœud seront décrites. L'algorithme de la forêt d'arbres décisionnels et le modèle ayant servi à évaluer la pertinence des différentes caractéristiques seront présentés, tout comme le score utilisé pour qualifier les performances d'un modèle. Le chapitre terminera avec la présentation de deux algorithmes d'apprentissage profond et leurs modèles associés, le réseau de neurones récurrents (RNN) et le réseau de neurones graphique (GNN). Les accélérations obtenues à l'aide des partitions créées par ces modèles seront évaluées. Les modèles de forêt d'arbres de décision ont été construits à l'aide de la librairie Scikit-learn [33] et les modèles de RNN et GNN ont été créés à l'aide de la librairie TensorFlow [34].

6.1 Description de la tâche d'apprentissage supervisée

Une tâche d'apprentissage supervisée consiste à apprendre une fonction de prédiction à partir d'exemples étiquetés. Ayant 2 jeux de données de 12 problèmes créés par notre générateur présenté au chapitre 3 et la solution de la relaxation linéaire au nœud 0 de chaque problème, nous avons un nombre important d'exemples annotés. Notre ensemble de problèmes contient 78282 points de relève au total. Tout point de relève pour lequel un échange de conducteurs se trouve dans la solution de la relaxation linéaire au nœud 0 est étiqueté comme ayant la valeur 1 et tous les autres comme ayant la valeur 0. Nous sommes donc dans un contexte de classification binaire. Il est toutefois nécessaire d'associer des caractéristiques pour chaque

point de relève sur lesquels va se reposer l'algorithme d'apprentissage automatique pour faire sa prédiction.

Il a été possible de créer des caractéristiques associées à chaque point de relève à partir des réseaux créés par le générateur en lisant le fichier de problèmes compatibles avec le solveur et les fichiers de paramètres fournis au générateur. À partir de ceux-ci, les caractéristiques que nous avons testées sont les suivantes. Celles-ci ont été testées afin d'évaluer lesquelles ont le plus grand potentiel de prédiction des échanges de conducteurs.

Heure de passage. L'heure exacte représentée en minutes à laquelle un conducteur et un bus doivent se trouver au point de relève selon l'horaire créé par le générateur.

Coordonnées. Les coordonnées représentant l'endroit où se trouve le point de relève dans le réseau selon le plan en deux dimensions utilisé par le générateur. Ces coordonnées sont composées d'une valeur x et d'une valeur y.

Ligne. La ligne à laquelle est associée le point de relève, étant une valeur catégorielle, la ligne est représentée sous un encodage *one-hot*.

Distance parcourue et restante sur la ligne. Ces deux valeurs représentent la distance qui reste à être parcourue et la distance du trajet déjà parcourue sur la ligne associée au point de relève en question.

Distance du dépôt. Cette valeur représente la distance entre le dépôt et le point de relève.

Retours sur le point de relève dans le même bus. Les sorties de bus étant déterminées au préalable de la résolution du DSP, il est possible de déterminer pour chaque point de relève l'autobus et la sortie de bus lui étant associés. À partir de ces informations, il est possible de déterminer le nombre de fois que le bus va passer sur le point de relève tout au long de la sortie de bus.

Retours faits sur le point de relève et restants dans le même bus. Indique le nombre de retours sur le point de relève ayant déjà été faits ou restants par le bus associé au point de relève. Ainsi, la somme de la valeur retours faits sur le point de relève dans le même bus et retours restants sur le point de relève dans le même bus est égale à la valeur retours sur le point de relève dans le même bus.

Points de relève parcourus et restants sur la ligne. Représente le nombre de points de relève déjà parcourus et restants sur la ligne associée au point de relève.

Durée du trajet parcouru et restant par le bus. Indique la durée du trajet parcouru ou restant à être parcouru par le bus lorsque celui-ci se retrouve sur le point de relève en question.

Nombre de points de relève accessibles. À l'aide du réseau représentant le problème, il est possible d'identifier tous les autres points de relève pour lesquels un conducteur peut provenir où se diriger lors d'un échange de conducteurs. Un point de relève est considéré comme accessible pour un conducteur lorsqu'il coûte moins cher de s'y rendre directement au lieu de s'y rendre en passant par le dépôt. Cette valeur représente le nombre de points de relève étant accessibles.

Durée totale de la sortie de bus. Indique la durée totale de la sortie de bus associée au point de relève.

6.2 Forêt d'arbres décisionnels

Pour l'application d'algorithmes d'apprentissage automatique, nous avons utilisé la méthodologie présentée dans le livre *Deep Learning* [35]. Nous avons donc commencé par créer un modèle simple permettant d'obtenir des prédictions sur chaque point de relève et à partir de celui-ci identifier les problèmes de la méthode et le potentiel d'amélioration possible. Nous avons commencé par tester un algorithme de forêt d'arbres décisionnels. Celui-ci va servir à nous fournir de l'information par rapport à la pertinence des différentes caractéristiques créées. De plus, il demande moins de données que les méthodes d'apprentissage profond avant d'être performant [36].

Les arbres décisionnels sont au cœur de la méthode de forêt d'arbres décisionnels. Ces arbres ont une structure composée d'embranchements et de nœuds. La classe de la donnée introduite à l'arbre est obtenue dépendamment de la feuille sur laquelle la donnée se trouve à la fin du parcours à travers l'arbre. À chaque nœud, une caractéristique de la donnée est sélectionnée. Chaque nœud parent a deux embranchements allant vers deux nœuds fils. Ces deux embranchements séparent les données selon la valeur d'une caractéristique afin que les deux embranchements couvrent l'entièreté des valeurs possibles de la caractéristique. Dans un contexte d'apprentissage automatique, les caractéristiques sur les nœuds et les valeurs sur les embranchements sont déterminées selon l'algorithme d'apprentissage automatique. Celles-ci sont déterminées de façon à ce que le meilleur branchement soit sélectionné afin de départager les classes du problème. Ceci est fait en optimisant un certain critère à chaque embranchement. Le processus est recommencé jusqu'à ce que chaque feuille de l'arbre ne contienne plus qu'une classe. Cette classe sera la prédiction du modèle pour une donnée qui finirait à cet endroit, lorsque passée dans la structure de l'arbre.

Nous avons donc les composantes suivantes :

— $x \in R^n$ où n est le nombre de caractéristiques et x_j la valeur de la caractéristique j de

la donnée.

— y l'étiquette associée à x.

— m un nœud de l'arbre décisionnel.

— p_m le nœud père du nœud m.

— Q_m l'ensemble de données au nœud m.

— n_m le nombre de données au nœud m.

— $\theta = (j, t_m)$ où θ est un embranchement candidat composé de la caractéristique j et d'un seuil t_m séparant les deux embranchements vers les nœuds fils.

Chaque embranchement sépare les données Q_m de façon à ce que l'embranchement fils de gauche soit composé des données Q_m^{gauche} et celui de droite par les données Q_m^{droite} :

$$Q_m^{gauche}(\theta) = \{(x, y) | x_j \leq t_m\} \tag{6.1}$$

$$Q_m^{droite}(\theta) = Q_m \setminus Q_m^{gauche}(\theta) \tag{6.2}$$

Les embranchements sont sélectionnés selon un algorithme glouton où le meilleur embranchement est sélectionné. Ce processus se fait de façon récursive jusqu'à ce que chaque feuille ne contienne qu'une classe de données. Le gain d'information des différents embranchements candidats est déterminé comme étant $IG(Q_m, \theta)$. Le nombre de données au nœud fils de gauche du noeud m est défini comme étant n_m^{gauche} et le nombre de données au nœud fils de droite du noeud m est défini comme étant n_m^{droite}.

$$IG(Q_m, \theta) = H(Q_m) - \frac{n_m^{gauche}(\theta)}{n_m} H(Q_m^{gauche}(\theta)) - \frac{n_m^{droite}(\theta)}{n_m} H(Q_m^{droite}(\theta)) \tag{6.3}$$

Le meilleur embranchement θ^* au nœud m est donc celui qui apporte le plus grand gain d'information.

$$\theta^* = \arg\max_{\theta} IG(Q_m, \theta) \tag{6.4}$$

Le meilleur embranchement est donc dépendant de la fonction $H(Q)$ qui est une fonction d'impureté. Les deux fonctions les plus souvent utilisées dans un contexte d'arbre de classification et qui sont celles utilisées dans le cadre de ce projet sont l'indice de diversité de Gini

et le gain d'information définis ci-dessous dans un contexte de classification binaire où f_1 et f_0 sont les proportions d'étiquettes positives et négatives au nœud m :

— Indice de diversité de Gini = $f_1(1 - f_1) + f_0(1 - f_0)$

— Gain d'information = $-f_1 \log(f_1) - f_0 \log(f_0)$

La forêt d'arbres décisionnels est une méthode d'apprentissage automatique qui est un ensemble d'arbres décisionnels. Chaque arbre faisant partie de la forêt a été entrainé sur un sous-ensemble de données. Pour chaque valeur x, une prédiction est effectuée par chacun des différents arbres décisionnels. La prédiction retournée par le plus grand nombre d'arbres sera celle retournée par la forêt d'arbres décisionnels. Comparativement à la méthode d'arbres décisionnels, la forêt d'arbres décisionnels permet d'obtenir une meilleure exactitude tout en diminuant le surapprentissage [37]. Il existe toutefois certains risques avec cette méthode qui doivent être adressés, particulièrement dans notre contexte de prédiction des échanges de conducteurs.

Surapprentissage La méthode crée des arbres n'ayant aucune limite de profondeur ou limite sur le nombre de données par feuille. Les arbres créés avec ces paramètres auront tendance à fortement surapprendre sur leur ensemble de données. Il est donc nécessaire de changer ces valeurs. La profondeur de l'arbre et la quantité minimale de données par feuille ont donc été considérées comme étant des hyperparamètres. La classe retournée par une feuille est donc la classe s'y trouvant en majorité.

Classes déséquilibrées Pour les arbres décisionnels, lorsque certaines classes dominent, l'algorithme a tendance à créer des arbres qui sont fortement biaisés envers les classes dominantes [38]. Dans un contexte de prédiction des échanges de conducteurs, la classe négative domine fortement la classe positive telle que présentée au tableau 5.4. Afin d'atténuer ce problème, nous avons utilisé des poids sur les différentes classes afin de leur donner une importance supplémentaire et ainsi pénaliser plus fortement les erreurs sur la classe positive. Ces poids sont utilisés à deux endroits dans l'algorithme. Premièrement, ils sont utilisés pour pondérer la fonction d'impureté. Deuxièmement, ils sont utilisés aux feuilles de l'arbre. La classe est maintenant déterminée selon un vote pondéré par les poids fournis à l'algorithme [39].

6.2.1 Processus d'hyperparamétrage

Afin de minimiser le surapprentissage et le biais envers les classes dominantes, le processus d'hyperparamétrage est particulièrement important. L'hyperparamétrage c'est fait en utilisant la validation croisée à 5 blocs. Cette méthode consiste à séparer notre ensemble de

données d'entrainement en 5 blocs distincts. Ensuite, l'entrainement du modèle se fait sur 4 blocs de l'ensemble et l'évaluation de la performance du modèle se fait sur le 5e bloc de l'ensemble. Cette procédure est répétée 5 fois. La performance du modèle retournée est la moyenne obtenue sur les 5 modèles entrainés. Comparée à la méthode de réserver une partie de l'ensemble de données pour l'hyperparamétrage, la validation croisée nous permet de garder un ensemble de données plus grand sur lequel faire l'entrainement. Elle demande toutefois plus de calcul, ce qui se traduit par un hyperparamétrage plus long.

Un problème de l'ensemble A et un problème de l'ensemble B ont été sélectionnés comme étant l'ensemble de données test et un modèle a été entrainé sur le reste des données. Ce processus a été recommencé 12 fois afin que tous les problèmes des deux ensembles A et B aient fait partie de l'ensemble de données test et ainsi avoir une prédiction sur l'ensemble de nos données. 500 itérations de recherches aléatoires ont été faites afin de trouver les meilleurs hyperparamètres. Les hyperparamètres suivants ont été optimisés et les valeurs testées sont présentées dans le tableau 6.1.

— Nbr d'arbres : Le nombre d'arbres décisionnels faisant partie de la forêt.

— Profondeur maximale : La profondeur maximale que peuvent avoir les arbres décisionnels formant la forêt.

— Fonction d'impureté : La fonction d'impureté $H(Q)$ utilisée pour déterminer le gain d'information apporté par un embranchement potentiel.

— Poids sur les classe 1 et 0 : Les poids utilisés sur les deux classes afin de contrer le biais envers la classe dominante.

— Données par feuille minimales : Le nombre de données minimales dans chaque feuille de l'arbre.

Tableau 6.1 Espace des hyperparamètres testés

Hyperparamètres	Espace
Nbr d'arbres	50,51,. . .,150
Profondeur maximale	1,2,. . .,50
Fonction d'impureté	Indice de diversité de Gini, Gain d'information
Poids sur les classes 1 et 0	1 :1, 2 :1,. . ., 10 :1
Données par feuille minimales	1,2,. . .,50

L'espace formé par les différentes valeurs d'hyperparamètres a été exploré à l'aide de la recherche aléatoire. La recherche aléatoire est considérée comme étant plus performante par rapport à la recherche par grille lorsque la dimensionnalité de l'ensemble est grande. En effet,

ajouter des paramètres affectant peu la performance du modèle ne ralentit pas le processus dans le cadre de l'hyperparamétrage aléatoire. La recherche par grille devient plus chère, en termes de calcul, de façon exponentielle, peu importe la qualité des hyperparamètres. Contrairement à la recherche aléatoire où la qualité des hyperparamètres n'impacte pas la vitesse d'hyperparamétrage [40]. 500 ensembles d'hyperparamètres ont été testés.

Un score est nécessaire afin de quantifier la qualité d'un modèle obtenu lors de l'hyperparamétrage. Le score traditionnellement utilisé en apprentissage automatique est l'exactitude. Dans un contexte de classification binaire, cette valeur est la suivante.

$$Exactitude = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{6.5}$$

Où TP est le nombre de vrais positifs, FP le nombre de faux positifs, TN le nombre de vrais négatifs et FN le nombre de faux négatifs. Ce score n'est pas approprié dans un contexte de prédiction des points de relève. En effet, nous avons un ensemble de données fortement déséquilibrées où autour de seulement 7% des données sont étiquetées comme étant positives tel que présenté dans le tableau 5.5. Ainsi, notre modèle pourrait faire une prédiction négative sur l'ensemble de nos données et atteindre une exactitude de 93%. Afin de limiter ce comportement, un autre score a été utilisé afin d'évaluer la performance des différentes combinaisons d'hyperparamètres.

Lors de la phase d'hyperparamétrage, le score F_β a été utilisé. Ce score est exprimé ci-dessous.

$$F_\beta = \frac{(1 + \beta^2) * TP}{(1 + \beta^2) * TP + \beta^2 * FN + FP} \tag{6.6}$$

Ou exprimé sous la forme de précision et rappel où $précision = \frac{TP}{TP+FP}$ et $rappel = \frac{TP}{TP+FN}$.

$$F_\beta = (1 + \beta^2) * \frac{précision * rappel}{(\beta^2 * précision) + rappel} \tag{6.7}$$

Le F_β est une généralisation du F_1 souvent utilisé dans un contexte de problème déséquilibré. Le F_β permet de pondérer l'importance du rappel comparativement à la précision contrairement au F_1 qui les considère comme étant égaux. Dans un contexte de prédiction d'échanges de conducteurs, il n'est pas clair que la précision a la même valeur que le rappel. Une moins bonne précision impliquerait une plus grande quantité d'agrégations brisées, ce qui ralentirait le potentiel d'accélération engendré par la méthode d'agrégation dynamique de contraintes. En contrepartie, un moins bon rappel impliquerait qu'une certaine quantité d'échanges de conducteurs n'ont pas été identifiés par l'algorithme. Cela implique que pendant le processus

de résolution, il sera nécessaire de passer par le processus de désagrégation de DCA. Il n'est pas clair que ces deux impacts négatifs sur la vitesse de résolution des problèmes de DSP soient équivalents.

6.2.2 Sélection des caractéristiques

Un premier modèle a été créé ayant comme but d'évaluer les caractéristiques étant les plus prometteuses dans le cadre de la tâche de prédiction des échanges de conducteurs. Seules les caractéristiques les plus importantes seront conservées lors de la création des modèles subséquents. Comme nous sommes dans un contexte où la quantité de données est limitée, il est important de s'assurer que les caractéristiques utilisées soient significatives afin de limiter les risques du fléau de la dimension, ou *curse of dimensionality* en anglais. Ce terme a été présenté par Richard Bellman dans *Dynamic Programming* [41]. L'idée derrière ce terme utilisé dans un contexte d'apprentissage automatique est que plus l'espace de dimension représentant les données augmente, autrement dit, plus que nous avons un grand nombre de caractéristiques, plus le volume représentant l'espace de données augmente. Ainsi, la distance entre les différentes données augmente pour le même nombre de données. Cette plus grande distance pourrait rendre difficile pour un algorithme d'apprentissage automatique d'identifier des zones associées à une même classe. Pour limiter cet effet, il est pertinent d'effectuer une sélection des caractéristiques afin de ne pas utiliser des caractéristiques inutiles à la tâche de classification. La méthode de l'importance de la caractéristique de permutation a été utilisée afin de déterminer les caractéristiques les plus importantes au modèle. Cette méthode consiste à mélanger les valeurs d'une caractéristique de façon aléatoire et ensuite tester les performances du modèle, tel que présenté dans Breiman [37]. Introduisons l'information suivante afin de présenter la méthode de l'importance de la caractéristique de permutation.

— m un modèle de prédiction entrainé sur l'ensemble des données originales.

— D un ensemble de données originales où les colonnes sont les caractéristiques j et chaque rangée représente une donnée individuelle.

— K étant le nombre de répétitions.

— k étant une répétition dans l'ensemble $\{1, \ldots, K\}$.

Les étapes de la méthode sont les suivants.

1. Calculons un score de référence s obtenu par le modèle m sur l'ensemble de données D.
2. Pour chaque $k \in K$.
    — Mélanger de façon aléatoire les entrées de la colonne j de l'ensemble de données D afin de générer une nouvelle version de l'ensemble de données $\tilde{D}_{k,j}$

— Calculer le score s_k obtenu par le modèle m sur l'ensemble de données $\tilde{D}_{k,j}$.

3. L'importance i_j pour chaque caractéristique j est définie selon :

$$i_j = s - \frac{1}{|K|} \sum_{k=1}^{K} s_{k,j} \tag{6.8}$$

Plus i_j est élevée, plus la caractéristique j contribue de façon importante à la performance du modèle d'apprentissage. Il est toutefois important de mentionner que cette méthode ne transmet pas la valeur intrinsèque d'une caractéristique. En effet, cette méthode donne l'importance de la caractéristique pour le modèle s entraîné. Si le modèle s est un mauvais modèle, nous ne pouvons pas conclure qu'une caractéristique j n'est pas pertinente seulement par le fait que la valeur i_j est faible. Il est possible que pour un bon modèle, la valeur i_j soit élevée. Il est ainsi nécessaire d'évaluer si la performance du modèle s est supérieure à la performance obtenue par chance afin de tirer des conclusions sur l'importance des caractéristiques.

L'exactitude balancée a été utilisée afin de comparer la performance du modèle à une prédiction aléatoire. Sa définition est la suivante où le taux de vrai négatif (TNR) = $\frac{TN}{TN+FP}$.

$$\text{Exactitude Balancée} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{6.9}$$

Ou de façon équivalente

$$\text{Exactitude Balancée} = \frac{1}{2}\left(rappel + TNR\right) \tag{6.10}$$

Un modèle comprenant l'ensemble des caractéristiques a été entrainé et hyperparamétré tel que décrit à la section 6.2.1. Le F_β avec une valeur de $\beta = 1$ a été utilisé pour donner un score de performance aux modèles testés lors de l'hyperparamétrage. Les performances de ceux-ci obtenues sur les deux jeux de données A et B sont présentées au tableau 6.2.

Tableau 6.2 Performances du modèle entrainé sur l'ensemble des caractéristiques

	A					B				
	Rappel	Précision	TNR	Exact. Bal.	F_1	Rappel	Précision	TNR	Exact. Bal.	F_1
Maximum	62.4	42.2	93.3	77.0	50.1	63.1	39.1	92.4	76.8	48.1
Moyenne	54.1	29.4	91.3	72.7	37.8	53.0	30.5	90.5	71.7	38.8
Minimum	45.0	18.4	89.6	67.5	27.0	40.5	12.8	87.7	66.4	21.2

Nous constatons au tableau 6.2 que les performances du modèle sont nettement supérieures aux performances obtenues de façon aléatoire avec une exactitude balancée moyenne de 72.7

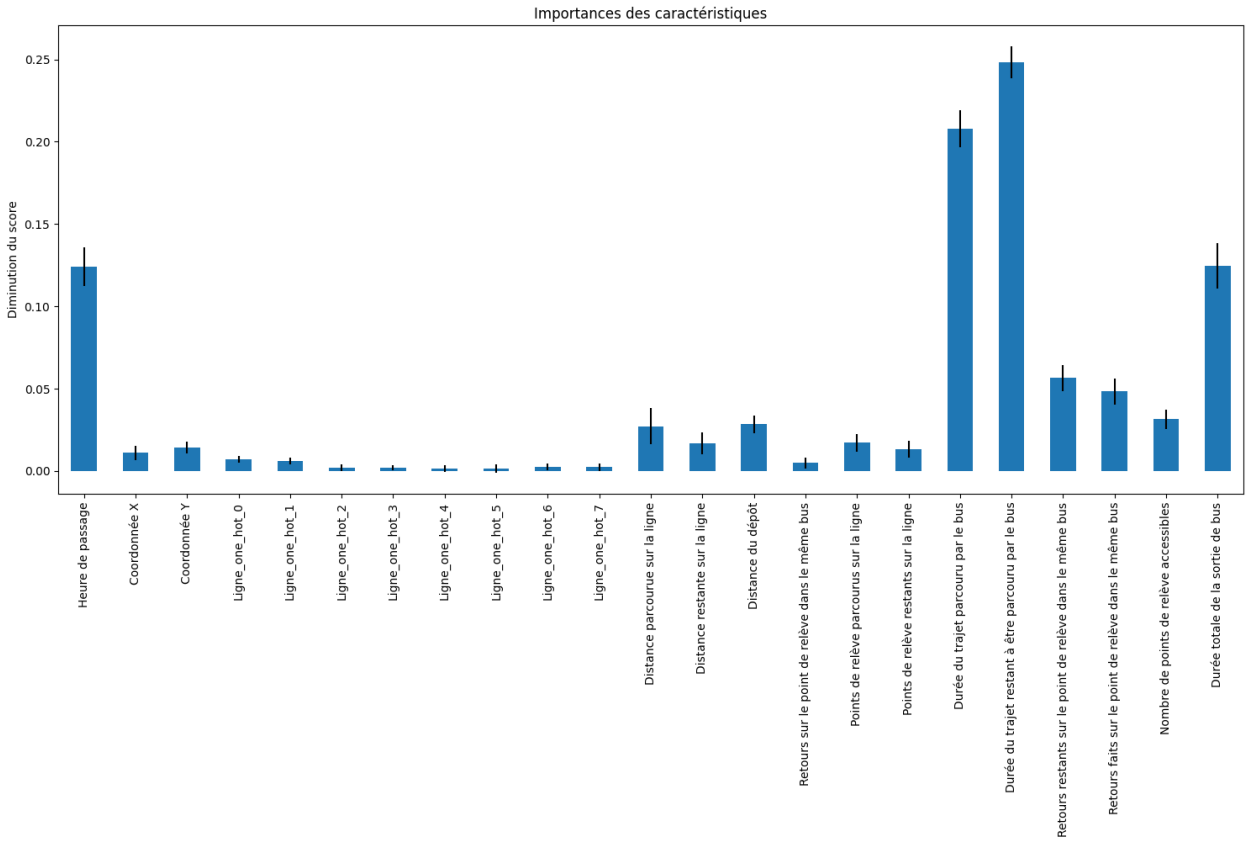


FIGURE 6.1 Importance des caractéristiques

pour l'ensemble A et de 71.7 pour l'ensemble B, soit des scores nettement supérieurs à 50 obtenus avec des prédictions aléatoires. Il est donc possible de juger de la pertinence des différentes caractéristiques à l'aide de la méthode de l'importance de la caractéristique de permutation.

À la figure 6.1, nous voyons les valeurs moyennes de i_j trouvées pour chaque caractéristique j en abscisse sur 50 itérations de mélanges aléatoires des valeurs des caractéristiques. Les barres d'erreur représentent l'écart type obtenu sur les 50 itérations.

Il est intéressant de constater que ce sont les caractéristiques décrivant la sortie de bus qui sont plus significatives. Les caractéristiques fournissant de l'information par rapport à la ligne associée au point de relève sont beaucoup moins pertinentes.

Pour les autres modèles entraînés, seules les caractéristiques ayant été jugées significatives ont été conservées. Ainsi, seules les caractéristiques ayant un indice i_j supérieur à 0.03 ont été conservées qui sont donc les suivantes.

— Heure de passage

— Durée du trajet parcouru par le bus

— Durée du trajet restant à être parcouru par le bus

— Retours restants sur le point de relève dans le même bus

— Retours faits sur le point de relève dans le même bus

— Nombre de points de relève dans le même bus

— Durée totale de la sortie de bus

Utilisant la même méthodologie d'hyperparamétrage présentée à la section 6.2.1 et utilisant une valeur de $\beta = 1$, 12 nouveaux modèles ont été créés en n'utilisant cette fois-ci que les 7 caractéristiques ayant été jugées significatives. Le tableau 6.3 présente les performances des modèles obtenues avec les 7 caractéristiques significatives.

Tableau 6.3 Performances du modèle entrainé sur les caractéristiques significatives

	A					B				
	Rappel	Précision	TNR	Exact. Bal.	F_1	Rappel	Précision	TNR	Exact. Bal.	F_1
Maximum	71.6	40.2	91.4	79.8	49.7	62.5	37.4	90.6	76.1	46.6
Moyenne	59.8	27.9	89.6	74.7	37.7	57.4	29.8	89.4	73.4	38.8
Minimum	54.0	17.2	87.9	71.7	26.7	49.6	12.4	87.0	70.0	20.7

Nous constatons que les performances du modèle obtenu avec 7 caractéristiques significatives sont similaires à celles obtenues avec l'ensemble des caractéristiques (voir tableau 6.2). Ceci confirme que les caractéristiques n'ayant pas été sélectionnées ne sont pas significatives. Seules les 7 caractéristiques significatives seront donc conservées pour les prochains modèles présentés.

6.2.3 Performance des modèles et accélérations obtenues

Trois groupes de modèles de forêts d'arbres décisionnels ont été entrainés en utilisant les 7 caractéristiques significatives. La différence entre ces trois groupes étant la valeur de β utilisée pour évaluer les modèles lors du processus d'hyperparamétrage. Les valeurs ayant été testées sont 2, 1 et 0.5. Pour 2, le score de rappel vaut 2 fois plus que celui de la précision. Pour 0,5 la précision vaut 2 fois plus que le rappel et pour une valeur de 1 les deux ont la même pondération. Les performances de ces modèles sont présentées dans le tableau 6.4. Comme prévu, les valeurs de rappel et de précision changent fortement dépendamment de la valeur de β utilisée.

TABLEAU 6.4 Performances du modèle obtenues en utilisant différentes valeurs de β

		A					B				
β		Rappel	Précision	TNR	Exact. Bal.	F_β	Rappel	Précision	TNR	Exact. Bal.	F_β
2	Maximum	90.2	29.9	81.9	85.1	38.2	88.5	28.8	83.4	83.4	36.2
	Moyenne	83.0	21.5	79.5	81.3	31.3	81.0	23.9	79.7	80.4	32.4
	Minimum	74.6	12.3	75.6	77.8	22.9	69.1	11.0	75.9	75.3	21.9
1	Maximum	71.6	40.2	91.4	79.8	49.7	62.5	37.4	90.6	76.1	46.6
	Moyenne	59.8	27.9	89.6	74.7	37.7	57.4	29.8	89.4	73.4	38.8
	Minimum	54.0	17.2	87.9	71.7	26.7	49.6	12.4	87.0	70.0	20.7
0.5	Maximum	46.8	48.5	96.1	71.2	57.8	49.7	48.1	97.1	71.5	56.8
	Moyenne	35.6	32.7	95.1	65.4	39.8	35.8	36.0	95.0	65.4	42.3
	Minimum	25.8	19.2	94.0	60.6	24.3	24.6	16.0	93.3	60.2	21.9

Des agrégations initiales ont été construites à l'aide des modèles créés à partir de différentes valeurs de β. Les agrégations basées sur les sorties de bus ont été brisées aux endroits où les modèles d'apprentissage automatique prédisaient un échange de conducteurs. Les temps de résolution obtenus avec des partitions créées à partir des prédictions sont présentés dans le tableau 6.5 selon la valeur de β utilisée lors de l'hyperparamétrage. Deux résolutions ont été faites pour chaque modèle à chaque problème. Une première résolution n'utilisant pas la méthode avec arcs restreints présentée à la section 5.2 et une l'utilisant. Nous pouvons voir que la valeur $\beta = 2$ apporte moins d'accélération que les deux autres valeurs. Les agrégations créées par le modèle utilisant $\beta = 2$ favorise le rappel lors de l'hyperparamétrage des modèles, ils ont donc tendance à plus briser les agrégations ce qui se traduit par un plus grand nombre d'agrégations et donc une réduction importante du potentiel d'accélération offert par DCA. Les deux autres valeurs de β donnent des résultats assez similaires. On préfère toutefois $\beta = 1$, car pour la majorité des résolutions, l'accélération moyenne, minimale et maximale sont meilleures que celles obtenues avec $\beta = 0.5$. Pour les modèles d'apprentissage automatique suivants, une valeur de $\beta = 1$ sera utilisée. Nous constatons aussi au tableau 6.6 que la méthode de restriction d'arcs apporte en moyenne une accélération supplémentaire lorsqu'elle est utilisée. Elle permet aussi à un plus grand nombre de problèmes d'être résolus plus rapidement qu'avec les agrégations basées sur les sorties de bus.

Tableau 6.5 Accélérations obtenues avec différentes valeurs de β

β	2		1		0.5	
	A	B	A	B	A	B
Maximum	49.7	56.8	45.7	53.5	40.1	39.5
Moyenne	−0.3	0.2	19.5	20.1	20.2	16.4
Minimum	−60.1	−81.5	−6.6	−1.9	−33.1	−8.2
Résolutions accélérés	6/12	6/12	11/12	9/12	11/12	9/12

Tableau 6.6 Accélérations obtenues avec différentes valeurs de β en utilisant la restriction d'arcs

β	2		1		0.5	
	A	B	A	B	A	B
Maximum	68.5	79.4	53.8	57.5	49.8	55.3
Moyenne	20.5	25.3	28.0	30.9	27.3	29.5
Minimum	−44.1	−38.9	−17.6	−6.1	−8.7	−13.6
Résolutions accélérés	7/12	8/12	10/12	11/12	10/12	11/12

6.3 Réseaux de neurones récurrents

Une faiblesse de l'algorithme de forêt d'arbres décisionnels est que la prédiction sur un nœud repose seulement sur les caractéristiques du nœud pour lequel on fait une prédiction. Toutefois, nous savons qu'une prédiction sur un nœud dépend des caractéristiques et des prédictions ayant été faites sur les autres nœuds d'une même sortie de bus. À l'aide du processus de sélection de caractéristiques, nous avons identifié que les caractéristiques associées aux pièces de travail sont les plus pertinentes à la détermination des échanges de conducteurs, ce qui suggère que la prédiction sur un point de relève dépend des autres points de relève faisant partie de la sortie de bus. Une difficulté supplémentaire nous empêchant d'exploiter ces caractéristiques à l'aide d'une architecture de forêt d'arbres décisionnels est que le nombre de points de relève à l'intérieur d'une sortie de bus n'est pas le même d'une sortie de bus à un autre. Afin d'avoir accès à cette information lors de la prédiction, nous avons testé l'utilisation d'un réseau de neurones récurrents (RNN). Ces réseaux adressent ce problème en utilisant des boucles à l'intérieur de leur architecture.

Nous avons à la figure 6.2 une représentation d'un RNN à gauche et un RNN sous sa forme déroulée dans un contexte de tâches plusieurs à plusieurs où pour chaque donnée composée d'un ensemble de caractéristiques à chaque temps (t), une prédiction est faite. La particularité

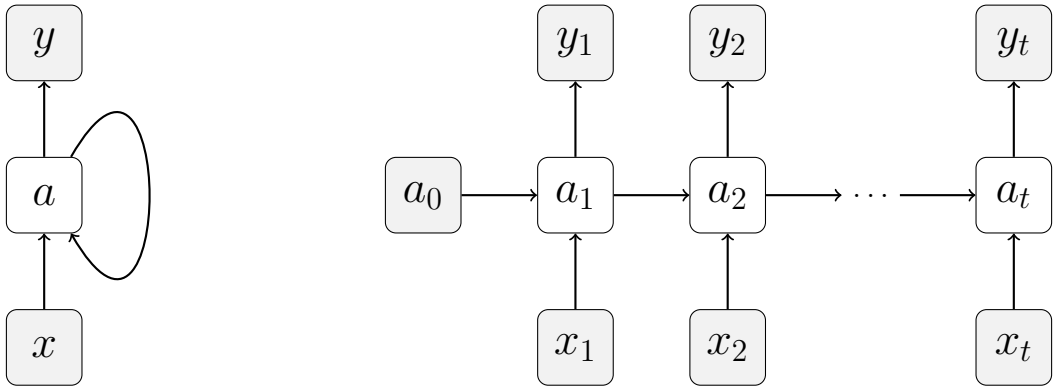


FIGURE 6.2 Architecture RNN

du RNN est que les poids sont partagés à chaque temps (t) du réseau. Il n'existe ainsi que 3 ensembles de poids à apprendre. De plus, une valeur a_{t-1} provenant du temps précédent est utilisée lors de la prédiction. Cette valeur est mise à jour à chaque temps (t) du réseau selon l'équation (6.11). Ainsi nous avons a_t étant la valeur de l'état caché au temps t. Les poids W_{ax} et W_{aa} sont les mêmes pour tout temps (t) du réseau. Les prédictions sont faites par l'équation (6.12) où un biais b_y est additionné à a_{t-1} multiplié par un autre ensemble de poids W_{ya}. Encore une fois, les poids W_{ya} et le biais b_y sont partagés à tout temps (t) du réseau. Les fonctions g_1 et g_2 sont des fonctions d'activation.

$$a_t = g_1\left(W_{aa}a_{(t-1)} + W_{ax}x_{(t)}\right) \tag{6.11}$$

$$y_t = g_2\left(W_{ya}a_{(t)} + b_y\right) \tag{6.12}$$

Nous avons donc un réseau où l'information à chaque temps (t) est distribuée à travers le réseau de façon à ce que toutes prédictions aux temps ultérieurs influencent la prédiction faite au temps (t). Toutefois, cette forme simple de RNN comporte un certain désavantage. La prédiction y_t se base que sur de l'information provenant des temps précédents et des caractéristiques du point de relève. Dans le contexte de notre tâche de prédiction des échanges de conducteurs, il serait intéressant que les prédictions sur les points de relève ne reposent pas seulement sur les caractéristiques des points de relèves précédents de la sortie de bus, mais plutôt sur les caractéristiques de l'entièreté des points de relève de la sortie de bus. Les réseaux de neurones bidirectionnels (Bi-RNN) permettent d'accomplir cette tâche.

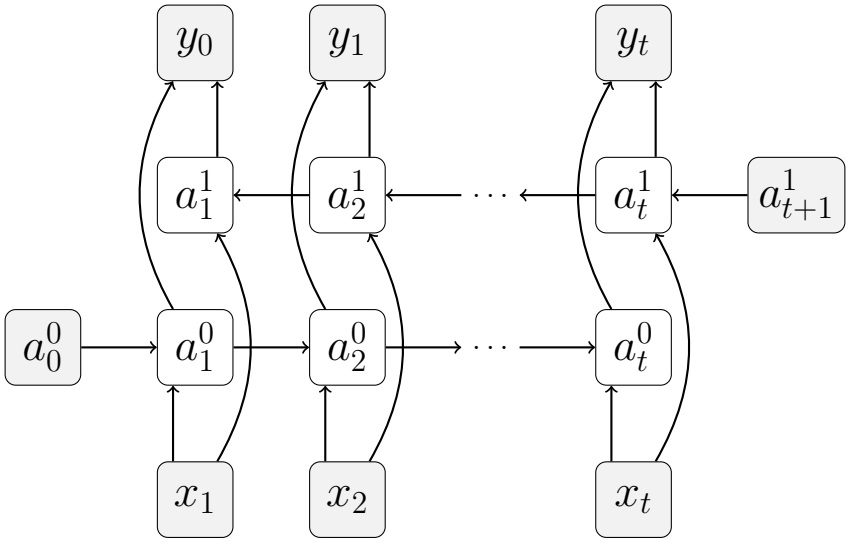


FIGURE 6.3 Architecture Bi-RNN

Les réseaux de neurones bidirectionnels sont une combinaison de deux couches de réseaux de neurones récurrents allant dans des sens opposés. Une commençant au temps 0 en allant en direction du temps t étant la couche 0 et une dans le sens inverse partant de $t+1$ et allant vers le temps 1 étant la couche 1, toutes les deux présentées à la figure 6.3. La prédiction de l'étiquette y_t se fait à partir des sorties des unités cachées au temps t des deux couches. L'information des deux couches est combinée à l'aide de la fonction *combin*. Cette fonction peut entre autres être la somme, la multiplication, la moyenne ou la concaténation de vecteurs.

$$a_t^1 = g_1\left(W_{aa}^1 a_{(t+1)}^1 + W_{ax}^1 x_{(t)}\right) \tag{6.13}$$

$$a_t^0 = g_1\left(W_{aa}^0 a_{(t-1)}^0 + W_{ax}^0 x_{(t)}\right) \tag{6.14}$$

$$y_t = g_2\left(combin\left(W_{ya}^0 a_{(t)} + b_y^0, W_{ya}^1 a_{(t)} + b_y^1\right)\right) \tag{6.15}$$

6.3.1 Notre modèle

Un réseau de neurones récurrents bidirectionnels a été utilisé dans notre tâche de prédiction d'échange de conducteurs sur les points de relève. Les données générées à partir du générateur présenté au chapitre 3 ont été formatées de façon à ce que chaque donnée entrante au modèle soit une sortie de bus x déterminée lors de la résolution du VSP et qu'à chaque temps t de la sortie de bus nous ayons un point de relève x_t composés des caractéristiques présentées à la

section 6.1. Comme présenté à la figure 6.3, nous avons une prédiction y_t pour chaque point de relève. Le réseau est composé de deux couches cachées formant une couche bidirectionnelle. Chaque unité formant le réseau récurrent est de type *long short-term memory* (LSTM) afin de diminuer les risques d'explosion du gradient. Nous référons à Hochreiter er al. [42] pour une description de ce type d'unité. La dernière couche du modèle est une couche à un seul neurone avec une fonction d'activation de type sigmoïde. La fonction *combin* utilisée est la concaténation de vecteurs.

Une étape d'hyperparamétrage a été faite, les hyperparamètres suivants ont été optimisés et les valeurs testées sont présentées dans le tableau 6.7 :

— Taux d'apprentissage : La valeur utilisée par l'optimiseur comme taux d'apprentissage.

— Poids sur les classes 1 et 0 : Les poids utilisés sur les deux classes afin de contrer le biais envers la classe dominante.

— *Dropout* : Le taux de dropout utilisé sur les couches cachées du modèle, le taux utilisé est celui suggéré dans Srivastava et al. [43].

— Fonctions d'activation : La fonction d'activation utilisée par la couche bidirectionnelle.

— Neurones couches RNN : Le nombre de neurones dans chacune des deux couches formant la couche bidirectionnelle.

Tableau 6.7 Hyperparamètres du modèle Bi-RNN

Hyperparamètres	Espace
Taux d'apprentissage	0.0001, 0.001, 0.01
Poids sur les classes 1 et 0	5 :1, 6 :1,. . .,10 :1
Dropout(%) sur les couches cachées	0, 50
Fonctions d'activation	ReLU, sigmoïde
Neurones couches RNN	4,8,12,. . .,100

Un problème de l'ensemble A et un problème de l'ensemble B ont été sélectionnés comme étant l'ensemble de données test sur lesquels une prédiction a été faite. 75% du reste des points de relève ont été utilisés comme ensemble d'entrainement et le 25% restant comme étant l'ensemble de validation. Ce processus a été fait 12 fois afin que tous les problèmes des deux ensembles A et B aient été utilisés comme ensemble de données test et que des prédictions aient été faites sur tous les points de relève. L'optimiseur utilisé est le Adam présenté dans Kingma et al. [44]. La fonction de perte utilisée est l'entropie croisée binaire pondérée par les poids en hyperparamètre. Pour chaque ensemble test différent, 100 itérations de recherches aléatoires ont été faites afin de trouver les meilleurs hyperparamètres. Chaque

modèle créé à chaque itération de la recherche aléatoire a été entrainé sur 100 époques ou jusqu'à ce que la perte sur l'ensemble de validation augmente pendant 5 époques consécutives.

6.3.2 Performance des modèles et accélérations obtenues

Tableau 6.8 Performances du modèle Bi-RNN

	A					B				
	Rappel	Précision	TNR	Exact. Bal.	F_1	Rappel	Précision	TNR	Exact. Bal.	F_1
Maximum	66.3	42.3	95.6	75.4	46.0	75.6	37.0	94.7	79.0	42.7
Moyenne	50.5	26.6	90.2	70.4	33.9	49.7	28.4	89.9	69.8	35.2
Minimum	34.0	14.8	84.4	64.7	22.7	37.4	11.2	82.4	64.8	18.6

Les résultats obtenus sont présentés au tableau 6.8. Nous constatons que nous obtenons des performances assez similaires à celles obtenues à l'aide des forêts d'arbres décisionnels. On remarque toutefois que les écarts entre la valeur maximale et minimale du rappel sont plus élevés que celles obtenues par la forêt d'arbres décisionnels. Ceci peut s'expliquer par la difficulté d'hyperparamétrer les modèles d'apprentissage profond, pour lesquels moins d'itérations de recherche aléatoire ont été faites comparativement aux modèles de forêt d'arbres décisionnels. Plus d'itérations de recherche aléatoire pourraient être nécessaires afin de trouver les meilleurs hyperparamètres.

Tableau 6.9 Accélérations obtenues par le modèle Bi-RNN

	Avec restriction d'arcs		Sans restriction d'arcs	
	A	B	A	B
Maximum	57.1	60.6	46.8	36.3
Moyenne	32.6	22.1	20.4	9.7
Minimum	−13.7	−26.6	−8.3	−43.9
Résolutions accélérés	11/12	10/12	10/12	9/12

Le tableau 6.9 rapporte les gains en temps de calcul obtenus. Nous constatons que les Bi-RNN ont obtenu de meilleurs résultats sur l'ensemble de problèmes A sur lesquels l'accélération moyenne atteint 32,6% et 11 des 12 problèmes de l'ensemble sont accélérés. L'algorithme de forêt d'arbres décisionnels obtient toutefois de meilleurs résultats sur l'ensemble B.

6.4 Réseaux de neurones graphiques

Les réseaux de neurones graphiques (GNN) sont des algorithmes d'apprentissage automatique pouvant accomplir des tâches variées telles que la classification de nœuds, la prédiction d'arcs et la classification de graphe. Ces réseaux sont en quelque sorte une généralisation des réseaux de neurones convolutifs classiques. Une revue exhaustive des différentes applications et algorithmes utilisant ce type de réseau a été présentée dans Wu et al. [45]. Des structures de données en 2 dimensions dans le cas des images en noir et blanc ou en 3 dimensions dans le cas des images en couleurs sont des cas typiques de l'application de réseaux de neurones convolutifs. Dans le cas des GNN, nous avons une plus grande flexibilité au niveau de la structure des données qui sont plutôt représentées sous la forme d'un graphe composé de nœuds et arcs ou arêtes. Les réseaux de neurones graphiques sont donc des algorithmes particulièrement appropriés aux tâches de classification qui peuvent se représenter sous une forme de tâche de classification de nœuds. Dans le cadre de notre problème, nous sommes dans un contexte de résolution de DSP. Comme décrit à la section 4.1, le DSP est formulé sous la forme d'un réseau. Nous pouvons exploiter cette structure à l'aide des réseaux de neurones graphiques.

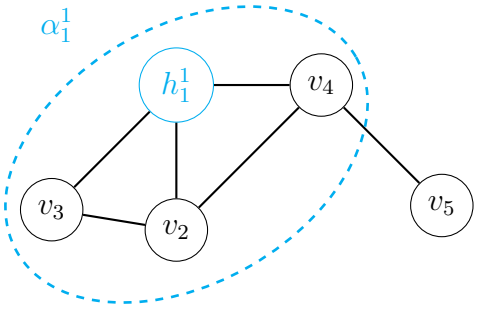


FIGURE 6.4 Architecture GNN

Les GNN utilisent les caractéristiques des nœuds et la structure du graphe afin de faire une prédiction. L'intuition derrière les GNN est que ces algorithmes utilisent une méthode d'agrégation pour chaque nœud permettant d'utiliser les caractéristiques des nœuds voisins en plus des caractéristiques du nœud pour lequel nous sommes en train de faire une prédiction. Ainsi, en répétant ce processus d'agrégation K fois, il est possible pour l'algorithme d'utiliser les caractéristiques des nœuds situés à une distance de K nœuds du nœud pour lequel l'algorithme fait une prédiction. Ce processus est séparé en deux étapes distinctes ; l'étape d'agrégation et l'étape de combinaison. L'étape d'agrégation consiste à agréger l'information contenue dans l'ensemble des nœuds voisins. L'étape de combinaison consiste à combiner l'information

obtenue à l'étape d'agrégation aux caractéristiques du nœud sur lequel on fait une prédiction.

$$\alpha_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{k-1} : u \in N(v)\}) \tag{6.16}$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{k-1}, \alpha_v^k) \tag{6.17}$$

Ici le graphe présenté à la figure 6.4 est représenté comme étant $G = (V, E)$ où V est l'ensemble des nœuds et E est l'ensemble des arêtes formant le graphe. Nous avons aussi $N(v)$ qui est l'ensemble des nœuds voisins au nœud v. Pour tout $v \in V$, nous avons un vecteur x_v contenant les caractéristiques du nœud. Pour chaque nœud, nous voulons obtenir une prédiction z_v indiquant la classe du nœud en question. Il y a K itérations du processus d'agrégation indicées par k. Nous avons $h_v^0 = x_v$ et à la dernière étape $h_v^k = z_v$. La fonction (6.16) agrège l'information contenue dans les nœuds voisins. À la figure 6.4, les caractéristiques des nœuds voisins du nœud en cyan sont agrégées afin de former α_1^1. α_1^1 est ensuite combiné aux caractéristiques du nœud en cyan afin de créer un nouveau vecteur h_1^1. Ces deux étapes permettent de mettre à jour l'état d'un nœud en lui incorporant les informations contenues sur les nœuds voisins. De nombreuses architectures différentes de GNN existent. Elles sont principalement différenciées par leurs fonctions AGGREGATE (6.16) et COMBINE (6.17).

6.4.1 Notre modèle

Une faiblesse de l'algorithme de forêt d'arbres décisionnels est que chaque prédiction repose seulement sur les caractéristiques d'un seul nœud. L'utilisation d'un GNN permet de corriger cette faiblesse en incorporant les caractéristiques des nœuds voisins lors de la prédiction. Une certaine modification de notre problème a toutefois été nécessaire afin d'adapter la structure du réseau à l'application d'un GNN. Tout d'abord, tous les nœuds n'étant pas des points de relèves ont été retirés. En effet, c'est seulement sur ces nœuds qu'une prédiction doit être faite par rapport à s'il doit avoir un échange de conducteurs sur ce nœud dans la solution de la relaxation linéaire au nœud 0. Donc, dans le réseau présenté au chapitre 4, seuls les nœuds $start\_of\_trip$, end_of_trip et $relief$ sont conservés. Pour les arcs, seuls les arcs reliant les différents points de relève entre eux ont été conservés. Donc, seulement les arcs de type $d-trip$ et ceux de types $inter-trip$ ont été conservés. Ces derniers relient maintenant directement les points de relève entre eux, les nœuds $start\_of\_break$ et end_of_break n'étant plus nécessaires. Ainsi, les arcs dans ce nouveau graphe représentent tous les déplacements potentiels pour un conducteur se trouvant à un point de relève en direction d'un autre sans

passer par le dépôt. De cette façon, le GNN aura accès aux caractéristiques de tous les points de relève auquel un conducteur pourra se diriger ou provenir, après ou avant un potentiel échange de conducteurs. Il est raisonnable de penser que d'avoir un échange de conducteurs ou pas sur un point de relève est influencée par les caractéristiques des autres points de relève étant accessibles à un chauffeur si celui-ci effectue un échange de conducteurs. Par exemple, un point de relève où les seuls autres points de relève accessibles à un conducteur effectuant un échange de conducteurs se trouvent très loin de celui-ci aura sûrement moins de chance d'effectuer un échange de conducteurs comparativement à un autre point de relève où les points de relève accessibles aux conducteurs se trouvent très proches. Les échanges de conducteurs nécessitant moins de temps de transport étant plus efficaces.

Le réseau de neurones graphiques ayant été utilisé est le réseau de neurones convolutifs graphique (GCN) tel que décrit dans Kipf et Welling [46]. Celui-ci combine les étapes d'agrégation et de combinaison en utilisant la formule suivante à chaque itération où W sont les poids appris.

$$\alpha_v^{(k)} = \phi\left(W \cdot \text{MOYENNE}\{h_u^{(k-1)}, \forall u \in N(v) \cup \{v\}\}\right) \tag{6.18}$$

À chaque couche de convolutions graphiques, les données entrantes sur lesquelles sont appliqués les poids appris lors de l'entraînement sont la moyenne des caractéristiques des nœuds $N(v) \cup \{v\}$. Une seule couche de convolution graphique a été utilisée, ce qui permet à la prédiction sur chaque nœud d'accéder aux caractéristiques de tous ses nœuds voisins. Une dernière couche cachée dense a aussi été utilisée. L'utilisation de cette dernière couche a été considérée comme étant un hyperparamètre. La dernière couche sortie est composée d'un seul neurone et d'une fonction d'activation sigmoïde.

Le processus d'hyperparamétrage est très similaire à celui ayant été décrit à la section 6.3.1 pour le modèle de réseau de neurones récurrents. Un problème de l'ensemble A et un problème de l'ensemble B ont été sélectionnés comme étant l'ensemble de données test sur lesquelles une prédiction a été faite. 75% du reste des points de relève ont été utilisées comme ensemble d'entraînement et le 25% restant comme étant l'ensemble de validation. Ce processus a été fait 12 fois afin que tous problèmes des deux ensembles A et B aient été utilisés comme ensemble de données test et que des prédictions aient été faites sur tous les points de relève. L'optimiseur utilisé est le Adam présenté dans Kingma et al [44]. La fonction de perte utilisée est l'entropie croisée binaire pondérée par les poids en hyperparamètres. Pour chaque ensemble test différent, 200 itérations de recherches aléatoires ont été faites afin de trouver

les meilleurs hyperparamètres. Chaque modèle aux itérations de la recherche aléatoire a été entrainé sur 100 époques ou jusqu'à ce que la perte sur l'ensemble de validation augmente pendant 5 époques consécutives.

Les hyperparamètres testés sont ceux présentés au tableau 6.10.

Tableau 6.10 Hyperparamètres du modèle GCN

Hyperparamètres	Espace
Taux d'apprentissage	0.0001, 0.001, 0.01
Poids sur les classe 1 et 0	5 :1, 6 :1, . . .,10 :1
Dropout(%) sur les couches cachées	0, 50
Fonctions d'activation	ReLU, sigmoïde
Nombre de neurones dans la couche cachée GCN	4, 8, 12, . . .,100
Utilisation de la couche cachée dense	oui, non
Nombre de neurones dans la couche cachée dense	4, 8, 12, . . .,100

Une représentation de l'architecture du réseau utilisé par l'algorithme de GCN, en utilisant les mêmes nœuds du réseau montré à la figure 4.1, est représentée à la figure 6.5. Deux sorties de bus sont représentées où tous les nœuds sont reliés par des arêtes à tous les autres nœuds étant accessibles à un conducteur effectuant un échange de conducteurs. L'ensemble des arêtes du nœud en cyan ont été représentées. Celui-ci est donc relié aux nœuds précédents et suivants de la sortie de bus lui étant associé et à trois autres nœuds d'une autre sortie de bus. Lors de la prédiction déterminant s'il devrait avoir un échange de conducteurs sur le nœud en cyan, les caractéristiques de ces cinq nœuds et du nœud en cyan seront utilisées contrairement aux modèles de forêt d'arbres décisionnels où seulement les caractéristiques du nœud pour lequel on fait une prédiction sont accessibles ou pour le modèle de réseau de neurones récurrents où seulement les nœuds faisant partie de la même sortie de bus sont accessibles.

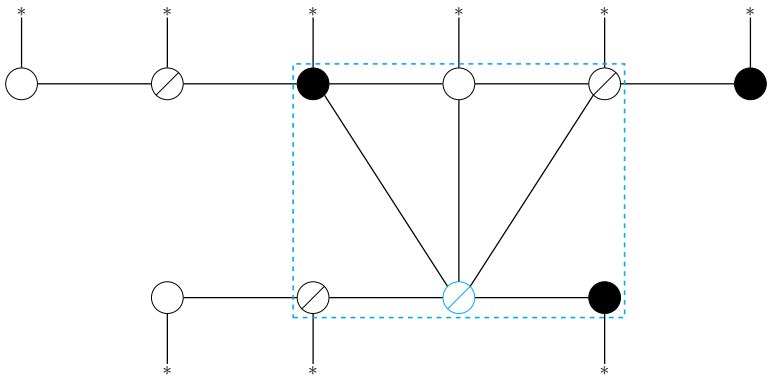


FIGURE 6.5 Architecture GCN

6.4.2 Performance des modèles et accélérations obtenues

Le tableau 6.11 présente les résultats du GCN obtenus pour différents critères de performance. Le tableau 6.12 présente l'accélération obtenue en utilisant les agrégations déterminées par le GCN. On constate que c'est le modèle obtenant les moins bonnes performances tant pour la qualité de la prédiction du modèle que pour les accélérations obtenues.

TABLEAU 6.11 Performances du modèle GCN

	A					B				
	Rappel	Précision	TNR	Exact. Bal.	F_1	Rappel	Précision	TNR	Exact. Bal.	F_1
Maximum	49.3	16.6	90.8	63.6	24.9	61.5	20.3	91.5	67.7	26.4
Moyenne	38.1	12.9	82.4	60.3	19.0	46.7	14.7	78.3	62.5	21.8
Minimum	30.6	8.8	78.0	55.9	14.3	27.5	6.6	73.0	58.1	11.9

TABLEAU 6.12 Accélérations obtenues par le modèle GCN

	Avec restriction d'arcs		Sans restriction d'arcs	
	A	B	A	B
Maximum	29.5	44.2	5.0	34.8
Moyenne	−40.9	−27.7	−57.6	−65.4
Minimum	−119.6	−152.3	−123.8	−209.1
Résolutions accélérés	1/12	4/12	1/12	1/12

Une raison possible pourquoi les GCN ne performent pas bien est la grande quantité de voisins qu'a chaque nœud dans le réseau. On voit dans le tableau 6.13 que la quantité de nœuds voisins à chaque nœud est très élevée. Comme la fonction COMBINE (6.17) utilisée par le GCN est la moyenne, les valeurs des caractéristiques des nœuds sont modifiées par les

caractéristiques de nœuds voisins où il ne doit pas avoir d'échange de conducteurs dans la solution de la relaxation linéaire au nœud 0. Les caractéristiques uniques de chaque nœud étiqueté positivement qui pourraient être utilisées lors de la prédiction sont donc diluées par les caractéristiques des nœuds voisins étant majoritairement étiquetés de façon négative. Ainsi, pour obtenir de meilleures prédictions à l'aide d'un GNN, il faudrait revoir la structure du réseau ou les formules utilisées pour la fonction AGGREGATE (6.16) et COMBINE (6.17).

Tableau 6.13 Nombres de voisins par nœud

	A	B
Maximum	37	32
Moyenne	10.5	9.3
Minimum	1	1
Médiane	9	8

6.5 Conclusion

Le tableau 6.14 présente les accélérations moyennes obtenues par les différents modèles présentés dans ce chapitre et celles obtenues à l'aide d'agrégations parfaites.

Tableau 6.14 Accélérations moyennes obtenues par les différentes agrégations créées

	Avec restriction d'arcs		Sans restriction d'arcs	
	A	B	A	B
Agrégations parfaites	77.9	79.9	63.3	60.0
Forêt d'arbres décisionnels	28.0	30.9	19.5	20.1
Bi-RNN	32.6	22.1	20.4	9.7
GNN	−40.9	−27.7	−57.6	−65.4

Nous constatons qu'un certain pourcentage du potentiel d'accélération présenté au chapitre 5 est atteint à l'aide d'algorithme d'apprentissage machine. L'algorithme de forêt d'arbres décisionnels obtient les meilleurs résultats sur l'ensemble de problèmes B et les Bi-RNN obtiennent les meilleurs résultats sur l'ensemble A. Il est toutefois probablement possible d'atteindre un meilleur résultat à l'aide des Bi-RNN en utilisant un hyperparamétrage plus exhaustif. Les GCN ne permettent pas d'obtenir d'accélération. Une nouvelle structure du réseau utilisé par le GCN ou changer la formule COMBINE (6.17) pourrait permettre d'atteindre de meilleurs résultats avec les GNN.

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS

Dans le cadre de ce projet, une méthode d'accélération de l'agrégation dynamique de contraintes reposant sur l'apprentissage automatique a été proposée pour la résolution de DSP. Cette méthode consiste à briser les agrégations faisant partie d'une partition initiale basée sur les sorties de bus déterminées lors de la résolution du VSP au préalable du DSP lui étant associé. Les agrégations sont brisées aux points de relève où l'algorithme d'apprentissage automatique détermine qu'il est probable qu'un échange de conducteurs devrait avoir lieu dans la solution de la relaxation linéaire au nœud 0.

7.1 Synthèse des travaux

Premièrement, un générateur de DSP a été créé. Celui-ci permet de créer un grand nombre de DSP pour lesquels les fréquences relatives entre les différentes lignes du réseau sont similaires aux fréquences que l'on retrouve dans les données réelles de la STM. De plus, les lignes et les arrêts composant le réseau sont aussi tirés de données réelles. Le générateur crée donc des problèmes ayant des caractéristiques similaires à ceux résolus en industrie. Deux ensembles de 12 problèmes ont été créés à l'aide de ce générateur afin de tester le potentiel de la nouvelle méthode proposée.

Deuxièmement, le potentiel de la méthode a été testé en utilisant des agrégations parfaites. Celles-ci ont été créées en résolvant les DSP créés par le générateur jusqu'à obtenir la solution de la relaxation linéaire au nœud 0. Les échanges de conducteurs faisant partie de cette solution ont été identifiés afin de briser les agrégations de la partition initiale à ces endroits. L'utilisation de ces partitions a engendré une accélération en moyenne de 63.3% sur l'ensemble de problèmes A et de 60.0% sur l'ensemble de problèmes B. Des méthodes d'accélération ont été testées afin de pousser le potentiel d'accélération de la nouvelle méthode. La restriction des arcs dans le sous-problème à ceux respectant les agrégations initiales en phase 0 de l'agrégation dynamique de contraintes à multiphases a permis d'atteindre une accélération en moyenne de 77.9% sur l'ensemble de problèmes A et de 79.9% sur l'ensemble de problèmes B en utilisant les agrégations parfaites. L'utilisation de la dominance exacte sur la ressource d'incompatibilité et briser les agrégations à d'autres endroits que les points de relève ayant un échange de conducteur dans la solution de la relaxation linéaire au nœud 0 n'ont pas apporté d'accélération.

Troisièmement, les caractéristiques associées à chaque point de relève ont été présentées

et leur pertinence dans le cadre d'une tâche de prédiction d'échange de conducteurs a été évaluée. Trois types de modèles d'apprentissage automatique et les performances de chacun ont été présentés. L'algorithme de forêt d'arbres décisionnels a obtenu les meilleurs résultats sur l'ensemble de problèmes B avec une accélération en moyenne de 20.1% sans restriction d'arcs et de 30.9% en utilisant la restriction d'arcs. Le Bi-RNN a obtenu les meilleurs résultats sur l'ensemble de problèmes A avec une accélération en moyenne de 20.4% sans restrictions d'arcs et de 32.6% en utilisant la restriction d'arcs.

7.2 Limitations de la solution proposée

Quelques limitations du projet doivent être mentionnées. Tout d'abord, la taille des problèmes résolus est beaucoup plus petite que celle des problèmes réels en industrie. En effet, les problèmes résolus ne contiennent que 8 lignes différentes et en moyenne 1075 voyages pour l'ensemble de problèmes A et 893 voyages pour l'ensemble de problèmes B. De plus, le potentiel d'accélération a seulement été évalué sur des SDVSP, les problèmes en industrie sont souvent composés de multiples dépôts.

7.3 Améliorations futures

De nombreuses améliorations pourraient être étudiées suite à ce projet. La considération d'autres caractéristiques sur les points de relève pourrait permettre aux modèles d'apprentissage automatique de mieux prédire les échanges de conducteurs. Différents modèles d'apprentissage automatique et d'architectures de réseau de neurones pourraient être testés afin de surpasser les performances obtenues à l'aide des modèles de forêts d'arbres décisionnels et des Bi-RNN. Une piste plus compliquée, mais qui pourrait apporter une accélération importante à la résolution de DSP serait d'essayer de prédire le prochain segment dans le quart de travail du conducteur sortant à l'échange de conducteurs au lieu de seulement prédire les échanges de conducteurs. Cette information supplémentaire permettrait de créer de meilleures agrégations sans augmenter le nombre d'agrégations faisant partie de la partition. En effet, les agrégations basées sur les sorties de bus ne seraient plus seulement brisées, elles seraient combinées selon le segment du conducteur sortant lors de l'échange de conducteur.

RÉFÉRENCES

[1] Société de transport de Montréal. (2022) Budget 2022. [En ligne]. Disponible : https://www.stm.info/sites/default/files/pdf/fr/budget2022.pdf

[2] G. Desaulniers et M. D. Hickman, "Public transit," dans *Transportation*, ser. Handbooks in Operations Research and Management Science, C. Barnhart et G. Laporte, édit. Elsevier, 2007, vol. 14, ch. 2, p. 69–127.

[3] S. Bunte et N. Kliewer, "An overview on vehicle scheduling models," *Public Transport*, vol. 1, p. 299–317, 2010.

[4] A. Bertossi, P. Carraresi et G. Gallo, "On some matching problems arising in vehicle scheduling models," *Networks*, vol. 17, p. 171–281, 1987.

[5] L. Bodin, B. Golden, A. Assad et M. Ball, "Routing and scheduling of vehicles and crews," *Computers & Operations Research*, vol. 10, p. 63–211, 1983.

[6] J. R. Daduna et J. M. Pinto Paixão, "Vehicle scheduling for public mass transit — an overview," dans *Computer-Aided Transit Scheduling*, J. R. Daduna, I. Branco et J. M. P. Paixão, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 1995, p. 76–90.

[7] G. Carpaneto, M. Dell'amico, M. Fischetti et P. Toth, "A branch and bound algorithm for the multiple depot vehicle scheduling problem," *Networks*, vol. 19, n°. 5, p. 531–548, 1989.

[8] M. Forbes, J. Holt et A. Watts, "An exact algorithm for multiple depot bus scheduling," *European Journal of Operational Research*, vol. 72, n°. 1, p. 115–124, 1994.

[9] A. Hadjar, O. Marcotte et F. Soumis, "A branch-and-cut algorithm for the multiple depot vehicle scheduling problem," *Operations Research*, vol. 54, n°. 1, p. 130–149, 2006.

[10] A. Oukil, H. B. Amor, J. Desrosiers et H. El Gueddari, "Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems," *Computers & Operations Research*, vol. 34, n°. 3, p. 817–834, 2007, logistics of Health Care Management.

[11] B. Smith et A. Wren, "A bus crew scheduling system using set covering formulation," *Transportation Research Part A : General*, vol. 22, p. 97–108, 1988.

[12] M. Desrochers et F. Soumis, "A column generation approach to the urban transit crew scheduling problem," *Transportation Science*, vol. 23, p. 1–13, 1989.

[13] A. de Silva, "Combining constraint programming and linear programming on an example of bus driver scheduling," *Annals OR*, vol. 108, p. 277–291, 11 2001.

[14] M. Grötschel, R. Borndörfer et A. Löbel, *Duty Scheduling in Public Transit*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2003, p. 653–674.

[15] T. H. Yunes, A. V. Moura et C. C. de Souza, "Hybrid column generation approaches for urban transit crew management problems," *Transportation Science*, vol. 39, n°. 2, p. 273–288, 2005.

[16] S. Fores, L. Proll et A. Wren, "Tracs ii : A hybrid ip/heuristic driver scheduling system for public transport," *The Journal of the Operational Research Society*, vol. 53, n°. 10, p. 1093–1100, 2002.

[17] S. Chen et Y. Shen, "An improved column generation algorithm for crew scheduling problems," *Journal of Information and Computational Science*, vol. 10, p. 175–183, 2013.

[18] M. Chen et H. Niu, "A model for bus crew scheduling problem with multiple duty types," *Discrete Dynamics in Nature and Society*, vol. 2012, p. 1–11, 2012.

[19] B. Kecskeméti et A. Bilics, "Bus driver duty optimization using an integer programming and evolutionary hybrid algorithm," *Central European Journal of Operations Research*, vol. 21, p. 745–755, 2013.

[20] E. Marchiori et A. Steenbeek, "An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling," dans *Real-World Applications of Evolutionary Computing*, S. Cagnoni, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2000, p. 370–384.

[21] R. Portugal, H. R. Lourenço et J. P. Paixão, "Driver scheduling problem modelling," *SSRN Electronic Journal*, vol. 1, p. 103–120, 2009.

[22] A. Tóth et M. Krész, "An efficient solution approach for real-world driver scheduling problems in urban bus transportation," *Central European Journal of Operations Research*, vol. 21, p. 75–94, 2013.

[23] G. Dantzig et P. Wolfe, "A decomposition principle for linear programs," *Operations Research*, vol. 8, p. 101–111, 1960.

[24] I. Elhallaoui, D. Villeneuve, F. Soumis et G. Desaulniers, "Dynamic aggregation of set-partitioning constraints in column generation," *Operations Research*, vol. 53, p. 632–645, 2005.

[25] A. H. Land et A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, n°. 3, p. 497–520, 1960.

[26] Y. Bengio, A. Lodi et A. Prouvost, "Machine learning for combinatorial optimization : a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, 2020.

[27] A. Alvarez, Q. Louveaux et L. Wehenkel, "A machine learning-based approximation of strong branching," *INFORMS Journal on Computing*, vol. 29, p. 185–195, 2017.

[28] N. Vesselinova, R. Steinert, D. F. Perez-Ramirez et M. Boman, "Learning combinatorial optimization on graphs : A survey with applications to networking," *IEEE Access*, vol. 8, p. 120 388–120 416, 2020.

[29] K. Haase, G. Desaulniers et J. Desrosiers, "Simultaneous vehicle and crew scheduling in urban mass transit systems," *Transportation Science*, vol. 35, p. 286–303, 2001.

[30] M. Fischetti, S. Martello et P. Toth, "The fixed job schedule problem with working-time constraints," *Operations Research*, vol. 37, p. 395–403, 1989.

[31] R. Freling, "Models and techniques for integrating vehicle and crew scheduling," Thèse de doctorat, Erasmus University Rotterdam, 1997.

[32] I. Elhallaoui, M. Abdelmoutalib, F. Soumis et G. Desaulniers, "Multi-phase dynamic constraint aggregation for set partitioning type problems," *Mathematical Programming*, vol. 123, p. 345–370, 2010.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot et E. Duchesnay, "Scikit-learn : Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, p. 2825–2830, 2011.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu et X. Zheng, "TensorFlow : Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [En ligne]. Disponible : https://www.tensorflow.org/

[35] I. Goodfellow, Y. Bengio et A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[36] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal et V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, n°. 3, 2019.

[37] B. Leo, "Random forests," *Machine Learning*, vol. 45, p. 5–32, 2001.

[38] H. Deng, G. Runger et E. Tuv, "Bias of importance measures for multi-valued attributes and solutions," dans *Artificial Neural Networks and Machine Learning – ICANN 2011*,

T. Honkela, W. Duch, M. Girolami et S. Kaski, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 293–300.

[39] C. Chen et L. Breiman, "Using random forest to learn imbalanced data," *University of California, Berkeley*, 2004.

[40] J. Bergstra et Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, p. 281–305, 2012.

[41] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[42] S. Hochreiter et J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, p. 1735–1780, 1997.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov, "Dropout : A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, p. 1929–1958, 2014.

[44] D. P. Kingma et J. Ba, "Adam : A method for stochastic optimization," 2014. [En ligne]. Disponible : https://arxiv.org/abs/1412.6980

[45] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang et P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, nº. 1, p. 4–24, 2021.

[46] T. N. Kipf et M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. [En ligne]. Disponible : https://arxiv.org/abs/1609.02907

[47] O. Tange, "Gnu parallel 20201222 ('vaccine')," 2020, GNU Parallel is a general parallelizer to run multiple serial command line programs in parallel without changing them. [En ligne]. Disponible : https://doi.org/10.5281/zenodo.4381888

ANNEXE A TABLEAUX DES ACCÉLÉRATIONS OBTENUES

Cette annexe présente les données présentées dans les tableaux du mémoire pour chaque instance des deux ensembles de problèmes. T orig. est la durée en secondes des résolutions utilisant les agrégations basées sur les sorties de bus.

Tableau A.1 Accélérations obtenues à l'aide d'agrégations parfaites

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	3 314.9	9 242.0	64.1	3 453.2	8 801.2	60.8
2	4 205.4	12 667.0	66.8	5 902.3	16 962.1	65.2
3	3 522.9	8 515.4	58.6	8 755.2	13 542.6	35.4
4	5 477.4	19 580.4	72.0	5 740.3	11 056.0	48.1
5	4 398.2	12 143.8	63.8	2 736.1	7 858.5	65.2
6	3 129.7	9 033.0	65.4	1 609.0	3 653.9	56.0
7	12 548.9	28 021.1	55.2	5 595.6	15 803.4	64.6
8	4 606.0	11 066.7	58.4	3 277.0	8 994.2	63.6
9	4 440.8	10 396.3	57.3	2 245.0	5 172.8	56.6
10	2 494.8	15 112.5	83.5	2 899.8	13 189.5	78.0
11	8 763.0	17 417.0	49.7	6 308.6	15 063.8	58.1
12	3 885.0	10 951.6	64.5	2 398.2	7 644.1	68.6
Maximum	12 548.9	28 021.1	83.5	8 755.2	16 962.1	78.0
Moyenne	5 065.6	13 678.9	63.3	4 243.4	10 645.2	60.0
Minimum	2 494.8	8 515.4	49.7	1 609.0	3 653.9	35.4

TABLEAU A.2 Accélérations obtenues à l'aide d'agrégations parfaites en utilisant la méthode avec restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	2 319.6	9 242.0	74.9	1 193.4	8 801.2	86.4
2	2 814.3	12 667.0	77.8	1 419.3	16 962.1	91.6
3	2 669.4	8 515.4	68.7	3 383.8	13 542.6	75.0
4	3 015.4	19 580.4	84.6	1 726.2	11 056.0	84.4
5	1 838.4	12 143.8	84.9	1 656.3	7 858.5	78.9
6	1 754.8	9 033.0	80.6	1 037.2	3 653.9	71.6
7	8 613.6	28 021.1	69.3	6 085.8	15 803.4	61.5
8	3 106.5	11 066.7	71.9	1 905.0	8 994.2	78.8
9	2 090.3	10 396.3	79.9	1 422.1	5 172.8	72.5
10	1 477.0	15 112.5	90.2	1 477.0	13 189.5	88.8
11	3 850.2	17 417.0	77.9	1 861.5	15 063.8	87.6
12	2 805.6	10 951.6	74.4	1 406.7	7 644.1	81.6
Maximum	8 613.6	28 021.1	90.2	6 085.8	16 962.1	91.6
Moyenne	3 029.6	13 678.9	77.9	2 047.9	10 645.2	79.9
Minimum	1 477.0	8 515.4	68.7	1 037.2	3 653.9	61.5

TABLEAU A.3 Accélérations obtenues à l'aide d'agrégations parfaites en utilisant la dominance exacte sur la ressource d'incompatibilité

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	6 972.5	9 242.0	24.6	3 477.2	8 801.2	60.5
2	8 143.6	12 667.0	35.7	4 621.8	16 962.1	72.8
3	7 545.6	8 515.4	11.4	5 375.9	13 542.6	60.3
4	8 309.0	19 580.4	57.6	4 912.4	11 056.0	55.6
5	6 676.1	12 143.8	45.0	4 989.9	7 858.5	36.5
6	6 499.0	9 033.0	28.1	3 080.1	3 653.9	15.7
7	12 769.3	28 021.1	54.4	6 744.6	15 803.4	57.3
8	10 412.9	11 066.7	5.9	4 876.9	8 994.2	45.8
9	6 287.1	10 396.3	39.5	4 337.3	5 172.8	16.2
10	7 612.6	15 112.5	49.6	4 687.3	13 189.5	64.5
11	10 231.3	17 417.0	41.3	5 625.9	15 063.8	62.7
12	6 054.0	10 951.6	44.7	4 229.1	7 644.1	44.7
Maximum	12 769.3	28 021.1	57.6	6 744.6	16 962.1	72.8
Moyenne	8 126.1	13 678.9	36.5	4 746.5	10 645.2	49.4
Minimum	6 054.0	8 515.4	5.9	3 080.1	3 653.9	15.7

Tableau A.4 Accélérations obtenues à l'aide d'agrégations brisées sur l'ensemble #2

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	15 411.4	9 242.0	−66.8	13 882.5	8 801.2	−57.7
2	12 704.5	12 667.0	−0.3	28 860.1	16 962.1	−70.1
3	14 233.3	8 515.4	−67.1	27 338.1	13 542.6	−101.9
4	28 825.4	19 580.4	−47.2	14 518.8	11 056.0	−31.3
5	12 900.0	12 143.8	−6.2	10 178.7	7 858.5	−29.5
6	12 067.7	9 033.0	−33.6	5 225.6	3 653.9	−43.0
7	28 836.1	28 021.1	−2.9	17 327.2	15 803.4	−9.6
8	19 678.3	11 066.7	−77.8	12 304.5	8 994.2	−36.8
9	16 503.4	10 396.3	−58.7	11 156.9	5 172.8	−115.7
10	20 108.9	15 112.5	−33.1	20 398.6	13 189.5	−54.7
11	28 843.0	17 417.0	−65.6	27 190.1	15 063.8	−80.5
12	16 652.2	10 951.6	−52.1	9 441.3	7 644.1	−23.5
Maximum	28 843.0	28 021.1	−0.3	28 860.1	16 962.1	−9.6
Moyenne	18 897.0	13 678.9	−42.6	16 485.2	10 645.2	−54.5
Minimum	12 067.7	8 515.4	−77.8	5 225.6	3 653.9	−115.7

Tableau A.5 Accélérations obtenues à l'aide d'agrégations brisées sur l'ensemble #3

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	21 654.0	9 242.0	−134.3	35 648.1	8 801.2	−305.0
2	23 224.4	12 667.0	−83.3	65 808.0	16 962.1	−288.0
3	21 022.8	8 515.4	−146.9	38 967.9	13 542.6	−187.7
4	45 220.5	19 580.4	−130.9	27 735.9	11 056.0	−150.9
5	33 302.2	12 143.8	−174.2	19 214.0	7 858.5	−144.5
6	18 356.8	9 033.0	−103.2	9 049.4	3 653.9	−147.7
7	56 895.2	28 021.1	−103.0	22 365.8	15 803.4	−41.5
8	25 509.7	11 066.7	−130.5	24 605.6	8 994.2	−173.6
9	21 119.6	10 396.3	−103.1	19 437.6	5 172.8	−275.8
10	38 664.8	15 112.5	−155.8	39 017.9	13 189.5	−195.8
11	54 477.0	17 417.0	−212.8	132 894.8	15 063.8	−782.2
12	20 641.9	10 951.6	−88.5	16 148.0	7 644.1	−111.2
Maximum	56 895.2	28 021.1	−83.3	132 894.8	16 962.1	−41.5
Moyenne	31 674.1	13 678.9	−130.5	37 574.4	10 645.2	−233.7
Minimum	18 356.8	8 515.4	−212.8	9 049.4	3 653.9	−782.2

TABLEAU A.6 Accélérations obtenues à l'aide d'agrégations parfaites dans l'atteinte de la solution en nombres entiers

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	5 195.7	13 151.8	60.5	3 195.3	12 303.5	74.0
2	8 208.5	17 496.8	53.1	13 686.0	24 582.5	44.3
3	9 364.1	19 509.9	52.0	11 588.6	15 869.4	27.0
4	13 368.5	35 636.1	62.5	6 359.6	18 761.1	66.1
5	5 434.2	23 957.5	77.3	6 530.6	13 500.4	51.6
6	5 428.8	16 095.0	66.3	3 134.5	8 103.0	61.3
7	22 433.2	36 368.8	38.3	16 259.6	23 436.3	30.6
8	4 089.0	15 487.4	73.6	8 072.9	17 303.9	53.3
9	5 851.8	19 748.4	70.4	5 479.3	9 146.4	40.1
10	7 837.7	15 114.4	48.1	13 156.1	19 083.9	31.1
11	5 455.5	30 551.9	82.1	9 253.6	27 340.9	66.2
12	10 789.8	20 295.3	46.8	5 495.3	12 781.8	57.0
Maximum	22 433.2	36 368.8	82.1	16 259.6	27 340.9	74.0
Moyenne	8 621.4	21 951.1	60.9	8 517.6	16 851.1	50.2
Minimum	4 089.0	13 151.8	38.3	3 134.5	8 103.0	27.0

TABLEAU A.7 Accélérations obtenues avec les agrégations créées par le modèle de forêt d'arbres décisionnels utilisant $\beta = 1$ sans restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	7 681.5	9 242.0	16.9	6 997.1	8 801.2	20.5
2	6 884.4	12 667.0	45.7	16 126.9	16 962.1	4.9
3	9 075.3	8 515.4	−6.6	13 795.4	13 542.6	−1.9
4	12 947.5	19 580.4	33.9	5 138.6	11 056.0	53.5
5	10 639.4	12 143.8	12.4	4 310.8	7 858.5	45.1
6	9 009.5	9 033.0	0.3	3 662.7	3 653.9	−0.2
7	19 808.1	28 021.1	29.3	13 163.4	15 803.4	16.7
8	9 308.9	11 066.7	15.9	7 575.6	8 994.2	15.8
9	9 033.6	10 396.3	13.1	5 246.9	5 172.8	−1.4
10	10 712.8	15 112.5	29.1	6 651.1	13 189.5	49.6
11	16 587.5	17 417.0	4.8	10 513.3	15 063.8	30.2
12	6 619.6	10 951.6	39.6	6 972.6	7 644.1	8.8
Maximum	19 808.1	28 021.1	45.7	16 126.9	16 962.1	53.5
Moyenne	10 692.3	13 678.9	19.5	8 346.2	10 645.2	20.1
Minimum	6 619.6	8 515.4	−6.6	3 662.7	3 653.9	−1.9

TABLEAU A.8 Accélérations obtenues avec les agrégations créées par le modèle de forêt d'arbres décisionnels utilisant $\beta = 1$ avec restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	10 302.2	9 242.0	−11.5	4 882.0	8 801.2	44.5
2	7 234.2	12 667.0	42.9	11 517.8	16 962.1	32.1
3	10 014.9	8 515.4	−17.6	12 160.6	13 542.6	10.2
4	10 633.7	19 580.4	45.7	4 700.2	11 056.0	57.5
5	5 611.8	12 143.8	53.8	4 616.6	7 858.5	41.3
6	4 710.3	9 033.0	47.9	2 779.2	3 653.9	23.9
7	20 058.7	28 021.1	28.4	11 109.5	15 803.4	29.7
8	7 777.0	11 066.7	29.7	7 636.0	8 994.2	15.1
9	8 622.5	10 396.3	17.1	5 487.7	5 172.8	−6.1
10	9 402.9	15 112.5	37.8	7 994.4	13 189.5	39.4
11	12 996.6	17 417.0	25.4	9 877.2	15 063.8	34.4
12	6 920.8	10 951.6	36.8	3 874.2	7 644.1	49.3
Maximum	20 058.7	28 021.1	53.8	12 160.6	16 962.1	57.5
Moyenne	9 523.8	13 678.9	28.0	7 219.6	10 645.2	30.9
Minimum	4 710.3	8 515.4	−17.6	2 779.2	3 653.9	−6.1

TABLEAU A.9 Accélérations obtenues avec les agrégations créées par le modèle de forêt d'arbres décisionnels utilisant $\beta = 2$ sans restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	6 148.4	9 242.0	33.5	4 326.6	8 801.2	50.8
2	7 852.4	12 667.0	38.0	7 416.1	16 962.1	56.3
3	13 635.9	8 515.4	−60.1	22 716.5	13 542.6	−67.7
4	9 852.3	19 580.4	49.7	4 772.8	11 056.0	56.8
5	7 161.2	12 143.8	41.0	5 253.7	7 858.5	33.1
6	12 265.4	9 033.0	−35.8	6 631.4	3 653.9	−81.5
7	32 616.0	28 021.1	−16.4	9 273.3	15 803.4	41.3
8	14 708.1	11 066.7	−32.9	5 898.6	8 994.2	34.4
9	15 380.4	10 396.3	−47.9	6 341.6	5 172.8	−22.6
10	12 586.6	15 112.5	16.7	19 886.4	13 189.5	−50.8
11	13 048.1	17 417.0	25.1	19 649.2	15 063.8	−30.4
12	12 552.3	10 951.6	−14.6	8 917.3	7 644.1	−16.7
Maximum	32 616.0	28 021.1	49.7	22 716.5	16 962.1	56.8
Moyenne	13 150.6	13 678.9	−0.3	10 090.3	10 645.2	0.2
Minimum	6 148.4	8 515.4	−60.1	4 326.6	3 653.9	−81.5

TABLEAU A.10 Accélérations obtenues avec les agrégations créées par le modèle de forêt d'arbres décisionnels utilisant $\beta = 2$ avec restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	4 161.6	9 242.0	55.0	2 762.8	8 801.2	68.6
2	5 550.2	12 667.0	56.2	3 490.6	16 962.1	79.4
3	10 770.6	8 515.4	−26.5	17 659.3	13 542.6	−30.4
4	6 158.5	19 580.4	68.5	3 882.6	11 056.0	64.9
5	5 683.4	12 143.8	53.2	2 851.3	7 858.5	63.7
6	9 817.9	9 033.0	−8.7	4 746.1	3 653.9	−29.9
7	29 200.8	28 021.1	−4.2	8 474.9	15 803.4	46.4
8	13 588.3	11 066.7	−22.8	4 570.9	8 994.2	49.2
9	14 976.2	10 396.3	−44.1	2 697.4	5 172.8	47.9
10	6 576.1	15 112.5	56.5	16 076.9	13 189.5	−21.9
11	7 699.9	17 417.0	55.8	20 925.2	15 063.8	−38.9
12	10 124.7	10 951.6	7.6	7 325.7	7 644.1	4.2
Maximum	29 200.8	28 021.1	68.5	20 925.2	16 962.1	79.4
Moyenne	10 359.0	13 678.9	20.5	7 955.3	10 645.2	25.3
Minimum	4 161.6	8 515.4	−44.1	2 697.4	3 653.9	−38.9

TABLEAU A.11 Accélérations obtenues avec les agrégations créées par le modèle de forêt d'arbres décisionnels utilisant $\beta = 0.5$ sans restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	5 537.1	9 242.0	40.1	7 901.8	8 801.2	10.2
2	7 603.7	12 667.0	40.0	12 827.2	16 962.1	24.4
3	7 052.9	8 515.4	17.2	14 222.3	13 542.6	−5.0
4	14 309.3	19 580.4	26.9	6 693.0	11 056.0	39.5
5	9 760.4	12 143.8	19.6	5 298.8	7 858.5	32.6
6	6 525.5	9 033.0	27.8	2 857.0	3 653.9	21.8
7	23 187.1	28 021.1	17.3	11 089.8	15 803.4	29.8
8	7 403.2	11 066.7	33.1	9 729.9	8 994.2	−8.2
9	9 099.4	10 396.3	12.5	5 313.8	5 172.8	−2.7
10	13 003.8	15 112.5	14.0	11 727.8	13 189.5	11.1
11	23 177.8	17 417.0	−33.1	14 181.8	15 063.8	5.9
12	8 048.1	10 951.6	26.5	4 773.3	7 644.1	37.6
Maximum	23 187.1	28 021.1	40.1	14 222.3	16 962.1	39.5
Moyenne	11 225.7	13 678.9	20.2	8 884.7	10 645.2	16.4
Minimum	5 537.1	8 515.4	−33.1	2 857.0	3 653.9	−8.2

Tableau A.12 Accélérations obtenues avec les agrégations créées par le modèle de forêt d'arbres décisionnels utilisant $\beta = 0.5$ avec restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	5 426.1	9 242.0	41.3	4 818.1	8 801.2	45.3
2	6 363.8	12 667.0	49.8	11 755.6	16 962.1	30.7
3	5 751.3	8 515.4	32.5	13 089.0	13 542.6	3.3
4	16 371.1	19 580.4	16.4	4 942.1	11 056.0	55.3
5	8 188.6	12 143.8	32.6	4 257.8	7 858.5	45.8
6	5 244.4	9 033.0	41.9	2 225.4	3 653.9	39.1
7	21 448.7	28 021.1	23.5	10 571.1	15 803.4	33.1
8	7 275.1	11 066.7	34.3	10 221.2	8 994.2	−13.6
9	6 713.4	10 396.3	35.4	3 577.4	5 172.8	30.8
10	15 456.8	15 112.5	−2.3	8 765.8	13 189.5	33.5
11	11 958.0	17 417.0	31.3	12 726.0	15 063.8	15.5
12	11 901.1	10 951.6	−8.7	4 955.7	7 644.1	35.2
Maximum	21 448.7	28 021.1	49.8	13 089.0	16 962.1	55.3
Moyenne	10 174.9	13 678.9	27.3	7 658.8	10 645.2	29.5
Minimum	5 244.4	8 515.4	−8.7	2 225.4	3 653.9	−13.6

Tableau A.13 Accélérations obtenues avec les agrégations créées par le Bi-RNN utilisant $\beta = 1$ sans restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	5 830.9	9 242.0	36.9	5 608.5	8 801.2	36.3
2	13 722.7	12 667.0	−8.3	22 569.9	16 962.1	−33.1
3	6 774.4	8 515.4	20.4	12 953.6	13 542.6	4.3
4	10 414.8	19 580.4	46.8	7 117.0	11 056.0	35.6
5	10 291.3	12 143.8	15.3	5 515.9	7 858.5	29.8
6	7 581.9	9 033.0	16.1	5 257.0	3 653.9	−43.9
7	20 428.5	28 021.1	27.1	10 811.8	15 803.4	31.6
8	9 937.0	11 066.7	10.2	7 635.0	8 994.2	15.1
9	10 489.7	10 396.3	−0.9	5 984.4	5 172.8	−15.7
10	8 904.2	15 112.5	41.1	8 904.2	13 189.5	32.5
11	16 525.1	17 417.0	5.1	12 546.0	15 063.8	16.7
12	7 057.3	10 951.6	35.6	7 057.3	7 644.1	7.7
Maximum	20 428.5	28 021.1	46.8	22 569.9	16 962.1	36.3
Moyenne	10 663.2	13 678.9	20.4	9 330.0	10 645.2	9.7
Minimum	5 830.9	8 515.4	−8.3	5 257.0	3 653.9	−43.9

TABLEAU A.14 Accélérations obtenues avec les agrégations créées par le Bi-RNN utilisant $\beta = 1$ avec restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	5 197.2	9 242.0	43.8	5 327.6	8 801.2	39.5
2	14 405.8	12 667.0	−13.7	21 479.5	16 962.1	−26.6
3	7 233.3	8 515.4	15.1	11 728.5	13 542.6	13.4
4	8 408.8	19 580.4	57.1	4 354.9	11 056.0	60.6
5	8 976.6	12 143.8	26.1	4 627.6	7 858.5	41.1
6	4 709.7	9 033.0	47.9	3 144.8	3 653.9	13.9
7	19 128.0	28 021.1	31.7	10 226.6	15 803.4	35.3
8	5 997.2	11 066.7	45.8	9 030.3	8 994.2	−0.4
9	7 142.4	10 396.3	31.3	4 750.8	5 172.8	8.2
10	9 606.2	15 112.5	36.4	8 691.2	13 189.5	34.1
11	13 130.7	17 417.0	24.6	11 295.4	15 063.8	25.0
12	6 049.2	10 951.6	44.8	6 049.2	7 644.1	20.9
Maximum	19 128.0	28 021.1	57.1	21 479.5	16 962.1	60.6
Moyenne	9 165.4	13 678.9	32.6	8 392.2	10 645.2	22.1
Minimum	4 709.7	8 515.4	−13.7	3 144.8	3 653.9	−26.6

TABLEAU A.15 Accélérations obtenues avec les agrégations créées par le GCN utilisant $\beta = 1$ sans restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	19 118.0	9 242.0	−106.9	11 516.8	8 801.2	−30.9
2	14 134.8	12 667.0	−11.6	22 215.7	16 962.1	−31.0
3	18 386.2	8 515.4	−115.9	23 037.2	13 542.6	−70.1
4	25 557.7	19 580.4	−30.5	20 132.1	11 056.0	−82.1
5	17 253.3	12 143.8	−42.1	12 887.8	7 858.5	−64.0
6	13 825.3	9 033.0	−53.1	8 447.5	3 653.9	−131.2
7	42 880.4	28 021.1	−53.0	17 205.3	15 803.4	−8.9
8	13 519.2	11 066.7	−22.2	14 230.9	8 994.2	−58.2
9	23 271.7	10 396.3	−123.8	15 991.4	5 172.8	−209.1
10	14 362.0	15 112.5	5.0	8 600.0	13 189.5	34.8
11	30 867.1	17 417.0	−77.2	24 669.2	15 063.8	−63.8
12	17 921.9	10 951.6	−63.6	12 991.5	7 644.1	−70.0
Maximum	42 880.4	28 021.1	5.0	24 669.2	16 962.1	34.8
Moyenne	20 924.8	13 678.9	−57.9	15 993.8	10 645.2	−65.4
Minimum	13 519.2	8 515.4	−123.8	8 447.5	3 653.9	−209.1

TABLEAU A.16 Accélérations obtenues avec les agrégations créées par le GCN utilisant $\beta = 1$ avec restriction d'arcs

Problème	T (s)	T orig. (s)	Gain (%)	T (s)	T orig. (s)	Gain (%)
	Ensemble A			Ensemble B		
1	18 789.2	9 242.0	−103.3	8 461.3	8 801.2	3.9
2	13 594.0	12 667.0	−7.3	26 007.5	16 962.1	−53.3
3	15 693.8	8 515.4	−84.3	18 242.8	13 542.6	−34.7
4	23 021.3	19 580.4	−17.6	16 978.0	11 056.0	−53.6
5	18 466.3	12 143.8	−52.1	9 956.6	7 858.5	−26.7
6	12 649.5	9 033.0	−40.0	2 964.5	3 653.9	18.9
7	29 327.6	28 021.1	−4.7	18 602.9	15 803.4	−17.7
8	14 569.1	11 066.7	−31.6	12 409.0	8 994.2	−38.0
9	22 834.6	10 396.3	−119.6	13 052.3	5 172.8	−152.3
10	10 647.1	15 112.5	29.5	7 359.0	13 189.5	44.2
11	22 851.0	17 417.0	−31.2	10 948.5	15 063.8	27.3
12	14 122.2	10 951.6	−29.0	11 454.4	7 644.1	−49.8
Maximum	29 327.6	28 021.1	29.5	26 007.5	16 962.1	44.2
Moyenne	18 047.1	13 678.9	−40.9	13 036.4	10 645.2	−27.6
Minimum	10 647.1	8 515.4	−119.6	2 964.5	3 653.9	−152.3

ANNEXE B TABLEAUX DES PERFORMANCES OBTENUES PAR CHAQUE MODÈLE

Cette annexe présente les performances obtenues par chaque modèle présenté dans les tableaux du mémoire pour chaque instance des deux ensembles de problèmes. Préc. est la précision et E. B. l'exactitude balancée.

TABLEAU B.1 Performances obtenues en pourcentages par la forêt d'arbres décisionnels en utilisant $\beta = 1$ et l'ensemble des caractéristiques

Prob.	Rappel	Préc.	TNR	E. B.	F_β	Rappel	Préc.	TNR	E. B.	F_β
	Ensemble A					Ensemble B				
1	61.5	42.2	92.5	77.0	50.1	57.7	28.9	90.6	74.1	38.5
2	48.4	32.9	92.2	70.3	39.2	52.3	28.0	90.5	71.4	36.5
3	60.1	27.9	90.5	75.3	38.2	60.4	12.8	87.7	74.1	21.2
4	51.4	32.1	92.3	71.9	39.6	48.5	37.0	91.9	70.2	42.0
5	51.1	33.7	93.3	72.2	40.7	40.5	32.5	92.2	66.4	36.0
6	57.8	32.1	90.0	73.9	41.3	62.5	39.1	91.1	76.8	48.1
7	45.0	24.0	90.1	67.5	31.3	46.2	29.9	90.5	68.4	36.3
8	56.5	33.4	91.4	73.9	42.0	63.1	32.7	89.4	76.3	43.0
9	62.4	32.5	90.9	76.7	42.8	48.6	30.0	89.7	69.1	37.1
10	59.1	24.3	89.6	74.4	34.4	57.7	30.6	89.7	73.7	40.0
11	50.3	18.4	90.6	70.4	27.0	47.0	28.5	90.5	68.7	35.5
12	46.0	19.2	91.7	68.8	27.1	50.9	35.7	92.4	71.7	42.0
Max.	62.4	42.2	93.3	77.0	50.1	63.1	39.1	92.4	76.8	48.1
Moy.	54.1	29.4	91.3	72.7	37.8	53.0	30.5	90.5	71.7	38.0
Min.	45.0	18.4	89.6	67.5	27.0	40.5	12.8	87.7	66.4	21.2

TABLEAU B.2 Performances obtenues en pourcentages par la forêt d'arbres décisionnels en utilisant $\beta = 1$ et les caractéristiques significatives

Prob.	Rappel	Préc.	TNR	E. B.	F_β	Rappel	Préc.	TNR	E. B.	F_β
	Ensemble A					Ensemble B				
1	61.5	40.2	91.4	78.2	49.7	50.0	25.0	90.0	70.0	33.3
2	55.0	30.5	90.1	72.6	39.2	61.4	29.0	89.4	75.4	39.4
3	58.7	25.1	89.2	74.0	35.2	61.5	12.4	87.0	74.3	20.7
4	58.8	29.9	90.3	74.5	39.7	54.7	33.7	89.4	72.1	41.7
5	56.2	30.4	91.4	73.8	39.5	49.6	32.1	90.3	70.0	38.9
6	57.8	32.7	90.2	74.0	41.7	61.6	37.4	90.6	76.1	46.6
7	56.9	25.3	88.3	72.6	35.0	52.5	29.7	89.2	70.8	37.9
8	60.7	31.1	89.7	75.2	41.1	60.1	30.4	88.8	74.5	40.4
9	71.6	29.7	88.1	79.8	42.0	54.7	29.4	88.1	71.4	38.2
10	63.7	24.7	89.1	76.4	35.7	59.5	32.9	90.5	75.0	42.4
11	59.4	17.2	87.9	73.6	26.7	62.5	31.1	88.8	75.6	41.5
12	54.0	18.1	89.5	71.7	27.1	60.7	34.9	90.6	75.7	44.3
Max.	71.6	40.2	91.4	79.8	49.7	62.5	37.4	90.6	76.1	46.6
Moy.	59.5	27.9	89.6	74.7	37.7	57.4	29.8	89.4	73.4	38.8
Min.	54.0	17.2	87.9	71.7	26.7	49.6	12.4	87.0	70.0	20.7

TABLEAU B.3 Performances obtenues en pourcentages par la forêt d'arbres décisionnels en utilisant $\beta = 0.5$ et les caractéristiques significatives

Prob.	Rappel	Préc.	TNR	E. B.	F_β	Rappel	Préc.	TNR	E. B.	F_β
	Ensemble A					Ensemble B				
1	46.8	48.5	95.6	71.2	57.8	36.8	27.5	93.5	65.2	34.7
2	40.7	35.0	94.0	67.4	43.2	49.7	34.3	93.3	71.5	43.8
3	38.0	32.7	95.2	66.6	40.3	42.9	16.0	93.3	68.1	21.9
4	38.4	34.6	94.9	66.6	42.3	33.6	39.1	94.9	64.2	45.5
5	33.3	36.0	96.0	64.7	42.5	29.4	40.4	96.0	62.7	45.1
6	37.8	38.2	95.0	66.4	45.7	44.4	48.1	95.6	70.0	56.8
7	28.7	25.3	94.1	61.4	31.1	33.2	35.1	94.6	63.9	41.6
8	41.6	36.2	94.4	68.0	44.6	43.3	37.0	94.0	68.7	45.7
9	36.7	35.1	95.2	66.0	42.5	36.6	37.9	94.6	65.6	45.1
10	28.0	28.7	96.1	62.0	34.3	30.2	41.9	96.7	63.4	46.6
11	25.8	19.2	95.4	60.6	24.3	24.6	32.8	95.9	60.2	36.9
12	31.3	23.3	95.6	63.4	29.4	25.0	41.8	97.1	61.1	44.2
Max.	46.8	48.5	96.1	71.2	57.8	49.7	48.1	97.1	71.5	56.8
Moy.	35.6	32.7	95.1	65.4	39.8	35.8	36.0	95.0	65.4	42.3
Min.	25.8	19.2	94.0	60.6	24.3	24.6	16.0	93.3	60.2	21.9

TABLEAU B.4 Performances obtenues en pourcentages par la forêt d'arbres décisionnels en utilisant $\beta = 2$ et les caractéristiques significatives

Prob.	Rappel	Préc.	TNR	E. B.	F_β	Rappel	Préc.	TNR	E. B.	F_β
	Ensemble A					Ensemble B				
1	88.8	29.9	81.4	85.1	38.2	88.5	21.4	78.3	83.4	32.6
2	83.7	22.8	77.6	80.7	32.7	85.3	24.0	81.0	83.1	33.9
3	82.6	19.8	79.4	81.0	30.3	86.8	11.0	79.0	82.9	21.9
4	88.2	22.5	78.6	83.4	33.4	83.2	28.7	79.6	81.4	36.2
5	83.6	22.4	80.6	82.1	32.4	79.0	26.1	79.4	79.2	33.7
6	78.9	26.4	81.9	80.4	33.8	80.6	27.9	80.9	80.8	35.1
7	74.6	21.5	81.0	77.8	29.9	69.1	24.6	81.5	75.3	30.4
8	80.2	25.2	81.9	81.0	33.5	82.8	24.8	79.6	81.2	33.8
9	87.8	22.7	79.0	83.4	33.5	72.0	23.8	79.1	75.5	30.7
10	90.2	17.4	75.9	83.0	29.5	82.4	21.3	75.9	79.2	31.4
11	80.6	12.3	75.6	78.1	22.9	81.9	23.8	78.7	80.3	33.0
12	76.7	15.1	81.4	79.0	25.3	80.8	28.8	83.4	82.1	35.6
Max.	90.2	29.9	81.9	85.1	38.2	88.5	28.8	83.4	83.4	36.2
Moy.	83.0	21.5	79.5	81.3	31.3	81.0	23.9	79.7	80.4	32.4
Min.	74.6	12.3	75.6	77.8	22.9	69.1	11.0	75.9	75.3	21.9

TABLEAU B.5 Performances obtenues en pourcentages par le Bi-RNN en utilisant $\beta = 1$ et les caractéristiques significatives

Prob.	Rappel	Préc.	TNR	E. B.	F_β	Rappel	Préc.	TNR	E. B.	F_β
	Ensemble A					Ensemble B				
1	50.4	42.3	93.9	72.1	46.0	37.4	24.3	92.2	64.8	29.4
2	66.3	25.1	84.4	75.4	36.4	75.6	23.3	82.4	79.0	35.6
3	57.7	25.6	89.7	73.7	35.4	54.9	11.2	87.0	71.0	18.6
4	49.8	28.3	91.1	70.4	36.1	48.2	34.7	91.1	69.6	40.4
5	51.6	26.2	90.2	70.9	34.7	46.8	30.2	90.0	68.4	36.7
6	54.6	32.1	90.5	72.6	40.4	54.7	33.2	89.9	72.3	41.3
7	38.3	22.9	91.1	64.7	28.7	42.6	32.0	92.1	67.3	36.5
8	34.0	36.9	95.6	64.8	35.4	38.6	37.0	94.7	66.6	37.8
9	55.5	27.5	89.7	72.6	36.8	47.3	29.7	89.8	68.6	36.5
10	47.2	21.3	90.2	68.7	29.3	42.8	26.5	90.7	66.7	32.8
11	52.3	15.9	88.3	70.3	24.4	47.8	25.8	88.9	68.3	33.5
12	48.7	14.8	88.0	68.3	22.7	59.8	33.2	90.0	74.9	42.7
Max.	66.3	42.3	95.6	75.4	46.0	75.6	37.0	94.7	79.0	42.7
Moy.	50.5	26.6	90.2	70.4	33.9	49.7	28.4	89.9	69.8	35.2
Min.	34.0	14.8	84.4	64.7	22.7	37.4	11.2	82.4	64.8	18.6

TABLEAU B.6 Performances obtenues en pourcentages par le GCN en utilisant $\beta = 1$ et les caractéristiques significatives

Prob.	Rappel	Préc.	TNR	E. B.	F_β	Rappel	Préc.	TNR	E. B.	F_β
	Ensemble A					Ensemble B				
1	49.3	16.6	78.0	63.6	24.9	59.9	13.6	74.6	67.2	22.1
2	33.7	13.8	83.4	58.5	19.6	44.7	13.9	80.5	62.6	21.2
3	37.1	10.6	80.8	58.9	16.5	61.5	6.6	73.8	67.7	11.9
4	39.6	13.1	81.4	60.5	19.7	55.5	17.3	73.9	64.7	26.4
5	43.4	14.0	82.1	62.7	21.1	45.6	16.7	79.1	62.4	24.5
6	35.5	14.1	82.3	58.9	20.2	41.4	15.9	80.0	60.7	23.0
7	30.6	10.1	81.1	55.9	15.2	43.0	14.8	78.4	60.7	22.0
8	37.4	16.6	85.7	61.5	23.0	42.5	14.6	79.9	61.2	21.8
9	39.3	11.3	78.2	58.8	17.5	43.2	12.7	73.0	58.1	19.6
10	31.1	16.0	90.8	60.9	21.1	27.5	20.3	91.5	59.5	23.3
11	38.7	8.8	83.0	60.8	14.3	50.9	14.1	74.8	62.8	22.1
12	42.0	9.2	82.1	62.0	15.0	44.6	15.6	79.9	62.3	23.1
Max.	49.3	16.6	90.8	63.6	24.9	61.5	20.3	91.5	67.7	26.4
Moy.	38.1	12.9	82.4	60.3	19.0	46.7	14.7	78.3	62.5	21.8
Min.	30.6	8.8	78.0	55.9	14.3	27.5	6.6	73.0	58.1	11.9

ANNEXE C AUTRES TABLEAUX

Tableau C.1 Nombre de voyages dans chaque problème

Problème	Ensemble A	Ensemble B
1	1012	864
2	1052	891
3	1091	932
4	1099	902
5	1037	886
6	985	822
7	957	825
8	1100	923
9	1036	866
10	1075	901
11	1136	918
12	1078	866

Tableau C.2 Nombre d'agrégations dans la partition initiale dépendamment du groupe de nœuds utilisé pour briser les agrégations basées sur les sorties de bus

Problème	Ensemble A			Ensemble B		
	#1	#2	#3	#1	#2	#3
1	408	1103	1370	285	992	1301
2	376	938	1244	300	1080	1457
3	353	1143	1354	201	1123	1515
4	375	1241	1549	381	1134	1379
5	337	1047	1412	349	965	1229
6	377	984	1316	336	821	1030
7	317	1260	1865	318	1000	1502
8	396	1098	1390	346	1004	1322
9	360	1078	1326	340	924	1140
10	325	1081	1399	328	1033	1421
11	290	1234	1506	330	1110	1477
12	296	1129	1406	332	944	1253