Titre: Title:	Combiner intelligence artificielle et programmation mathématique pour la planification des horaires des équipages en transport aérien
Auteur: Author:	Yassine Yaakoubi
Date:	2019
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Yaakoubi, Y. (2019). Combiner intelligence artificielle et programmation mathématique pour la planification des horaires des équipages en transport aérien [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/4137/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Combiner intelligence artificielle et programmation mathématique pour la planification des horaires des équipages en transport aérien

YASSINE YAAKOUBI
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Décembre 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Combiner intelligence artificielle et programmation mathématique pour la planification des horaires des équipages en transport aérien

présentée par **Yassine YAAKOUBI**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Guy DESAULNIERS, président, Ph. D.
François SOUMIS, membre et directeur de recherche, Ph. D.
Simon LACOSTE-JULIEN, membre et codirecteur de recherche, Ph. D.
Issmaïl EL HALLAOUI, membre, Ph. D.
Julien MAIRAL, membre externe, Ph. D.

REMERCIEMENTS

En premier lieu, je tiens à exprimer ma profonde gratitude à mon directeur de recherche : François Soumis, qui m'a ouvert les portes du GERAD pour pouvoir accomplir cette thèse. Je le remercie du fond du coeur, tout d'abord pour l'honneur qu'il m'a fait en me confiant la réalisation de ce projet, pour ses encouragements tout au long de cette thèse, son accompagnement remarquable et son soutien généreux. Il était prêt à tout moment à conseiller, soulager et semer de l'espoir même pendant les périodes les plus ardues. Sa disponibilité, ses qualités humaines, son sens humaniste, ses conseils éclairés et son encadrement judicieux se sont tous réunis pour me permettre d'accomplir le présent travail.

Je tiens à remercier mon co-directeur de recherche, Simon Lacoste-Julien, pour ses remarques pertinentes, sa grande rigueur et ses précieux conseils qui étaient incontestablement d'une utilité importante et dont j'ai grandement bénéficié.

J'adresse des sincères remerciements à François Lessard et Benoit Rochefort pour leur patience et leur dévouement pendant l'ensemble du projet. Ils m'ont aidé à résoudre de manière rigoureuse l'ensemble des problèmes qui se sont présentés sur notre chemin et ce fut un réel plaisir de travailler avec eux. Je remercie également Pierre Girard et Edoh Liagros Logo pour leur assistance informatique et leurs conseils techniques.

Je tiens à exprimer ma gratitude à Ghislaine Maury, ma professeure à Grenoble-INP. Je lui suis particulièrement reconnaissant pour avoir pris le temps de répondre à mes innombrables questions sur le programme de double-diplôme, ainsi que m'aider tout au long du processus qui m'a permis de faire un doctorat.

Je souhaite également remercier toutes les personnes du GERAD et de Polytechnique Montréal qui ont partagé mon quotidien pendant ces années passées à leurs côtés. Je remercie notamment Mohamed Amine Aboussalah, Mouad Morabit, Philippe Racette, Alice Wu, Adil Tahir, Abderrahman Bani et Rachid Hassani. Je remercie évidemment les membres de ma petite famille pour leur présence et support constants tout au long de cette thèse ainsi que ceux de ma famille élargie pour m'avoir accompagné dans ce projet et avoir participé de manière significative à la réussite de mes études depuis de longues années.

Que messieurs les membres de jury trouvent ici l'expression de ma reconnaissance pour avoir accepté de juger mon travail.

Je remercie enfin tous ceux qui m'ont soutenu tout au long de mes études ainsi que ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail.

RÉSUMÉ

La recherche opérationnelle est un élément central de l'amélioration des horaires d'équipage. L'objectif est d'appliquer des algorithmes de programmation mathématique pour trouver des solutions optimales. Toutefois, cette approche présente un inconvénient important : les temps d'exécution sont longs et nécessitent souvent plusieurs jours pour converger. Cela réduit la valeur pratique d'une solution optimale puisqu'il n'est pas possible d'effectuer une nouvelle exécution avec de nouveaux réglages de paramètres. Étant donné que les horaires des transporteurs aériens sont fréquemment perturbés par des événements météorologiques pendant toute l'année, il est souhaitable de chercher de nouveaux moyens de réduire les durées d'exécution.

Dans le cadre de cette thèse, on s'intéresse au problème de rotations d'équipage aériens ou CPP (*Crew Pairing Problem*), une des étapes de la planification des horaires d'équipage. Pour chaque catégorie d'équipage et chaque type de flotte d'aéronefs, le CPP consiste à trouver un ensemble de rotations à coût minimal afin que chaque vol actif soit effectué par un équipage, en respectant certaines conditions supplémentaires qui varient selon les applications et qui découlent généralement des accords de travail de chaque compagnie. Ce problème devient difficile à résoudre lorsque le nombre de vols augmente car le nombre de rotations possibles augmente de façon exponentielle (nombre de variables).

La méthode la plus répandue depuis les années 1990 a été de résoudre le problème de partitionnement d'ensemble avec génération de colonnes insérée dans un algorithme de séparation et évaluation ou B&B (*branch-&-bound*). Lorsque le nombre de vols augmente dans un problème de rotations d'équipage, le temps pour le résoudre par génération de colonnes devient important. Le nombre d'itérations de génération de colonnes, le temps par itération pour résoudre le problème maître et le nombre de nœuds de branchement augmentent. La méthode d'agrégation dynamique des contraintes (DCA) accélère le problème maître en réduisant le nombre de contraintes de partitionnement définies dans le problème maître restreint en agrégeant en une seule contrainte chaque groupe de tâches qui devraient être consécutives dans la solution optimale. Ceci correspond à fixer temporairement à 1 des variables de connexion de vol. Ceci permet de remplacer toutes les contraintes de couverture des vols d'une grappe par une contrainte unique. L'algorithme modifie dynamiquement ces grappes pour atteindre la solution optimale si certaines prédictions étaient fausses.

L'objectif de cette thèse est donc d'utiliser différentes méthodes d'apprentissage machine pour proposer des grappes de vols ayant une forte probabilité d'être effectués consécutivement par le même équipage, dans une solution optimale. Cette information alimente l'optimiseur de program-

mation mathématique pour terminer le travail en tenant compte de la fonction de coût exacte et des contraintes complexes.

Dans le premier sujet de cette thèse, nous présentons une étude de cas sur l'utilisation d'algorithmes d'apprentissage machine pour initialiser solveur commercial à base de génération de colonnes à grande échelle (GENCOL) dans le contexte d'un problème hebdomadaire de rotations d'équipage aérien, où de petites économies de 1.0 % se traduisent par une augmentation des revenus annuels de dizaines de millions de dollars dans une grande compagnie aérienne. Nous nous concentrons sur le problème de la prédiction du prochain vol de correspondance d'un équipage, défini comme un problème de classification multiclasse formé à partir de données historiques, et nous concevons une approche de réseaux de neurones adaptée qui atteint une grande précision (99.7% au total ou 82.5% sur les cas plus difficiles). Nous démontrons l'utilité de notre approche en utilisant une heuristique simple pour combiner les prédictions de connexion de vols afin de former des grappes initiales de vols qui sont fournis comme information initiale au solveur GENCOL, ce qui donne une amélioration de vitesse 10x et jusqu'à 0.2% d'économie.

Dans le second sujet de cette thèse, nous proposons de combiner de multiples méthodes d'optimisation mises en œuvre, développées et testées sur de petits ensembles de données, afin d'obtenir un nouveau solveur efficace pour le problème de rotations d'équipes à grande échelle. Nous utilisons l'apprentissage machine pour proposer des grappes initiales pour un problème de rotations d'équipage important : des problèmes mensuels comportant jusqu'à 50 000 vols. Nous utilisons l'apprentissage machine, pour produire des grappes de vols ayant une forte probabilité d'être effectués consécutivement par le même équipage, dans une solution optimale. Un nouvel algorithme combinant plusieurs techniques avancées de recherche opérationnelle sera utilisé pour assembler et modifier ces grappes, au besoin, afin de produire une bonne solution. Cette nouvelle approche, en commençant par l'apprentissage machine et en terminant l'optimisation par la programmation mathématique, permettra de résoudre des problèmes globalement plus importants et d'éviter la perte d'optimalité résultant de la décomposition heuristique en petites périodes de temps dans l'approche à horizon fuyant. Nous montrons que les grappes produites par l'heuristique à base d'apprentissage machine sont mieux adaptées aux problèmes de rotations d'équipage, ce qui se traduit par une réduction moyenne du coût de la solution entre 6.8 et 8.52 %, qui est principalement dû à la réduction du coût des contraintes globales entre 69.79 et 78.11 %, par rapport aux rotations obtenus avec une solution initiale standard.

Dans l'algorithme de génération de colonnes, une solution initiale réalisable est requise pour assurer la faisabilité du problème primal à chaque itération de génération de colonnes. De plus, il est évident, d'après les résultats expérimentaux dans la littérature, que si la qualité de la solution initiale est meilleure, la convergence de génération de colonnes est également plus rapide. Ainsi,

une solution initiale de haute qualité devrait être générée dans un laps de temps plus court. Pour pouvoir proposer une telle solution initiale, on a besoin d'un algorithme d'apprentissage machine capable d'incorporer les contraintes locales dans le processus d'entraînement.

Dans le troisième sujet de cette thèse, nous présentons donc les réseaux à noyaux convolutifs structurés (SCKN) qui combinent les propriétés des architectures d'apprentissage profond, la flexibilité non paramétrique des méthodes du noyau et les prédicteurs structurés. Plus précisément, nous montrons que l'utilisation supervisée de cette combinaison surpasse les méthodes de pointe en termes de sous-optimalité primale et de précision du test sur l'ensemble de données OCR. Nous appliquons cette méthode à un ensemble de données de prévision de connexions de vols pour proposer de bonnes solutions initiales à un solveur de planification des horaires d'équipage aérien. Les principaux résultats des calculs montrent que l'utilisation de l'approche proposée aboutit à de meilleures solutions avec des coûts significativement plus faibles, réduisant de 9.51 % le coût de la solution et de 80.25 % le coût des contraintes globales. De plus, l'utilisation de la solution obtenue pour relancer le processus d'optimisation donne de meilleurs résultats, réduisant encore le coût de la solution et fournissant une solution avec un coût très négligeable des contraintes globales et un nombre beaucoup plus réduit de repositionnements.

ABSTRACT

A focal point for improving crew scheduling is the study of operations research methods, in order to find optimal solutions. However, this approach has a major drawback. While optimal solutions are possible to achieve, the run times are lengthy, often requiring days for convergence. This reduces the practical value of an optimal solution because there is limited ability to complete a re-run with new parameter settings. Given that air carrier schedules experience frequent year-round disruption from weather events, it is desirable to look for new ways to reduce run times thus making schedule re-generation quicker and more interactive.

For each crew category and aircraft fleet type, the crew pairing problem (CPP) consists of finding a set of minimum-cost rotations so that each active flight is performed by a crew, under certain additional conditions that vary according to the applications and that generally result from the work agreements of each airline. This problem becomes difficult to solve when the number of flights increases because the number of possible rotations increases exponentially (number of variables).

The most prevalent method since the 1990s has been the set partitioning problem with column generation inserted in branch-&-bound. When the number of flights increases in a CPP, the time to solve it by column generation becomes important. Specifically, the number of iterations and the time per iteration to solve the master problem and the number of branching nodes increase. The dynamic constraint aggregation (DCA) method accelerates the master problem by reducing the number of partitioning constraints defined in the restricted master problem by aggregating into a single constraint each group of tasks that should be consecutive in the optimal solution. This corresponds to temporarily fixing to one the flight-connection variables. This allows all flight-covering constraints for flights in a cluster to be replaced by a single constraint. The algorithm modifies the clusters dynamically to reach an optimal solution if some predictions were wrong.

The objective of this thesis is therefore to use various machine learning methods to propose clusters of flights with a high probability of being performed consecutively by the same crew, in an optimal solution. This information feeds into the mathematical programming optimizer to complete the work taking into account the exact cost function and complex CPP constraints.

In the first subject of this thesis, we present a case study of using machine learning classification algorithms to initialize a large-scale commercial operations research solver (GENCOL) in the context of a weekly airline CPP, where small savings of as little as 1% translate to increasing annual revenue by dozens of millions of dollars in a large airline. We focus on the problem of predicting the next connecting flight of a crew, framed as a multiclass classification problem trained from historical data, and design an adapted neural network approach that achieves high accuracy (99.7%

overall or 82.5% on harder instances). We demonstrate the utility of our approach by using simple heuristics to combine the flight-connection predictions to form initial crew-pairing clusters that are provided as initial information to the GENCOL solver, yielding a 10x speed improvement and up to 0.2% cost saving.

In the second subject of this thesis, we propose to combine multiple optimization methods implemented, developed and tested on small datasets, in order to obtain an efficient new solver for large-scale CPPs. We use Machine Learning (ML) to propose a good initial partition for a large CPP: monthly problems with up to 50 000 flights. We use ML to produce clusters of flights having a high probability of being performed consecutively by the same crew, in an optimal solution. A new algorithm combining several advanced Operations Research techniques will be used to assemble and modify these clusters, when necessary, to produce a good solution. This new approach, starting with Machine Learning and finishing the optimization with Mathematical Programming will permit to solve globally larger problems and will avoid the loss of optimality resulting of heuristic decomposition in small time slices in the rolling horizon approach. We show that the clusters produced by ML-based heuristics are better suited for CPPs, resulting in an average reduction of solution cost between 6.8% and 8.52%, which is mainly due to the reduction in the cost of global constraints between 69.79% and 78.11%, when compared to pairings obtained with a standard initial solution.

In the column generation algorithm, an initial feasible solution is required to ensure the feasibility of the primal problem at each iteration of column generation. Moreover, it is clear from the computational experiments in the literature that if the quality of the initial solution is better, the convergence of column generation is also faster. Thus, a high quality initial solution should be generated in a shorter period of time. To be able to propose such an initial solution, we need a Machine Learning algorithm that is able to integrate local constraints into the training process.

In the third subject of this thesis, we therefore introduce a Structured Convolutional Kernel Network, or SCKN, which combines the properties of deep learning architectures, the non-parametric flexibility of kernel methods and the structured predictors. More precisely, we show that using this combination in a supervised fashion outperforms state of the art methods in terms of the primal sub-optimality as well as on the test accuracy on the OCR dataset. We apply this method on a Next-Flight-Prediction dataset to propose good initial solutions to an airline crew scheduling solver. The main computational results show that using our proposed approach yields better results with significantly smaller costs, reducing by 9.51% the solution cost and by 80.25% the cost of global constraints. Furthermore, using the obtained solution to re-launch the optimization process yields better results, further reducing the solution cost and providing a solution with a very negligible cost of global constraints and a much smaller number of deadheads.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ARMP	Aggregated Reduced Master Problem
BCFW	Block-Coordinate Frank-Wolfe
CG	Column Generation
CKN	Convolutional Kernel Networks
CNN	Convolutional Neural Networks
CP	Complementary Problem
CRF	Conditional Random Fields
CPP	Crew Pairing Problem
CRP	Crew Rostering Problem
DCA	Dynamic Constraint Aggregation
DH	DeadHead
FCD	Flight-Connection Dataset
GA	Genetic Algorithms
IPS	Improved Primal Simplex
LR	Learning Rate
MCK	Multi-layer Convolutional Kernel
MILP	Mixed Integer Linear Programming
MP	Master Problem
MPDCA	Multi-Phase Dynamic Constraint Aggregation
OCR	Optical Character Recognition
OEG	Online Exponentiated Gradient
RKHS	Reproducing Kernel Hilbert Space
RMP	Restricted Master Problem
RNN	Recurrent Neural Networks
SAG	Stochastic Average Gradient
SCKN	Structured Convolutional Kernel Networks
SCRBM	Sequence Classification Restricted Boltzmann Machines
SCP	Set Covering Problem
SDCA	Stochastic Dual Coordinate Ascent
SP	SubProblem
SPP	Set Partitioning Problem
SVM	Support Vector Machine
SSVM	Structured Support Vector Machine

CHAPITRE 1 INTRODUCTION

La planification est un outil de gestion puissant dont les applications sont variées dans l'industrie. Par exemple, la nécessité de planifier les tâches opérationnelles est à la base de presque toutes les entreprises de fabrication ou de services. Le problème de planification a pris de l'ampleur au cours de la deuxième étape de la révolution industrielle. Le plus grand défi des temps modernes en matière de planification est peut-être celui de l'industrie du transport aérien. Les coûts d'exploitation des compagnies aériennes proviennent principalement du carburant et des ressources humaines. Dans le cas du carburant, les coûts sont en grande partie incontrôlables, car l'offre et la demande de produits pétroliers jouent un rôle important. Par contre, les coûts de la planification des horaires des pilotes et du personnel de cabine peuvent être contrôlés au moyen d'horaires efficaces [1].

Au fur et à mesure que l'industrie mondiale du transport aérien prend de l'ampleur et du volume, la complexité des problèmes d'horaires s'accroît considérablement au fil du temps. La montée en puissance de la capacité de calcul capable de traiter de grandes tailles de problèmes mathématiques permet de résoudre des problèmes plus importants jusqu'à la quasi-optimalité. Cependant, la nature sans cesse croissante de l'industrie du transport aérien signifie qu'il faudra innover encore davantage pour garantir l'efficacité des opérations et la rentabilité. Ces dernières décennies, les compagnies aériennes ont commencé à mettre en place des départements de recherche opérationnelle afin de traiter leurs problèmes les plus fondamentaux [2], bien que la pratique de la recherche opérationelle (RO) sous une forme ou une autre par les compagnies remonte à 1950 [3]. Ce virage vers la RO comme planificateur des opérations des compagnies aériennes, combiné à de solides programmes de recherche universitaires, a donné lieu à une littérature riche sur le sujet.

Certains des problèmes de transport aérien les plus connus et les plus complexes ont été étudiés au sein du Groupe d'Études et de Recherche en Analyse des Décisions (GERAD). Ces problèmes sont liés aux défis associés à l'acheminement et à l'établissement des horaires d'équipage des compagnies aériennes. Au cours des 35 dernières années, plusieurs projets du GERAD ont contribué de façon importante à la création d'une grappe industrielle à Montréal par l'optimisation des horaires des véhicules et du personnel. Cela a également permis de développer des solutions robustes spécialement conçues pour cette famille de problèmes, ce qui a donné un avantage concurrentiel à deux entreprises : GIRO et AD-OPT. Les deux sociétés utilisent ces solutions dans leurs logiciels.

Dans le cadre de cette thèse, on s'intéresse au problème de rotations d'équipage aériens ou CPP (*Crew Pairing Problem*), une des étapes du problème de planification des horaires d'équipage ou CSP (*Crew Scheduling Problem*). Pour chaque catégorie d'équipage et chaque type de flotte d'aéronefs, le CPP consiste à trouver un ensemble de rotations à coût minimal afin que chaque vol actif

soit effectué par un équipage. Lors de l'élaboration de ces rotations, il faut également respecter certaines conditions supplémentaires qui varient selon les applications et qui découlent généralement des accords de travail de chaque compagnie : chaque équipe est associée à une ville (station) appelée base d'où elle doit partir et à laquelle elle doit retourner. Le CPP est généralement modélisé comme un problème de partitionnement d'ensemble ou SPP (*Set Partitioning Problem*), les vols doivent être répartis dans des rotations. Ce problème devient difficile à résoudre lorsque le nombre de vols augmente car le nombre de rotations possibles augmente de façon exponentielle (nombre de variables).

La méthode la plus répandue a été de résoudre le problème de recouvrement d'ensemble ou SCP (*Set Covering Problem*) par génération de colonnes ou CG (*Column Generation*) insérée dans un algorithme de séparation et évaluation ou B&B (*branch-&-bound*) [4]. Cette méthode, ainsi que d'autres, est décrite dans une étude sur le CPP par Cohn et al. [5] ; voir aussi Deveci et al. [6] pour un article synthèse récent. Selon cette dernière étude, la CG était l'approche la plus fréquemment utilisée.

Lorsque le nombre de vols augmente dans un CPP, le temps pour le résoudre par CG devient important. Le nombre d'itérations de CG, le temps par itération pour résoudre le problème maître et le nombre de nœuds de branchement augmentent. La méthode d'agrégation dynamique de contraintes (DCA) développée par Elhallaoui et al. [7] accélère le problème maître en réduisant le nombre de contraintes et la dégénérescence. La méthode DCA commence par une agrégation, en grappes, des vols ayant une bonne probabilité d'être effectués consécutivement par le même équipage, dans une solution optimale. Il correspond à fixer temporairement à 1 des variables de connexion de vol. Ceci permet de remplacer toutes les contraintes de couverture des vols d'une grappe par une contrainte unique. La méthode DCA utilise les coûts réduits pour identifier les grappes qui doivent être brisées pour atteindre une solution optimale et celles qui peuvent être conservées. En même temps, la méthode agrège des grappes reliées par des variables de connexion égales à 1. Cette gestion dynamique des grappes agrégeant les contraintes permet d'atteindre des solutions optimales avec un problème maître plus petit et moins dégénéré. Cette méthode produit de meilleures variables duales et réduit le nombre d'itérations de CG. De plus, la solution du programme linéaire est moins fractionnaire et réduit le nombre de nœuds à explorer dans le B&B.

Un inconvénient majeur de l'agrégation dynamique de contraintes, comme le soulignent Elhallaoui et al. [7], est qu'elle ne prend pas en compte l'agrégation dans la résolution du sous-problème. Par conséquent, de nombreuses variables incompatibles avec l'agrégation actuelle peuvent être produites, même s'il existe une variable *compatible* avec un coût réduit négatif. Lorsque cela se produit, les grappes incompatibles avec les variables générées sont désagrégées, pour permettre à ces variables d'entrer dans le problème maître restreint agrégé. Cela augmente la taille du problème

maître agrégé et ralentit l'algorithme.

Pour résoudre cela, ElHallaoui et al. [8] ont proposé une version améliorée de l'algorithme d'agrégation dynamique de contraintes, appelé l'algorithme d'agrégation dynamique de contraintes multiphase ou MPDCA (*Multiphase Dynamic Constraint Aggregation*). Dans cet algorithme, ils proposent une stratégie d'évaluation partielle des coûts réduits favorisant les colonnes compatibles avec l'agrégation de contraintes. En phase k, on ajoute dans le sous-problème une contrainte interdisant de briser les grappes plus que k fois. On débute ainsi une phase zéro qui interdit de briser les grappes courantes et k est augmenté graduellement pour explorer tout le domaine et atteindre une solution optimale. Semblable à DCA, MPDCA commence par une partition initiale de vols en grappes produite par un raisonnement logique ou une solution heuristique. Une telle partition initiale est appropriée si ses grappes combinent des composants qui ont une forte probabilité d'être dans une colonne d'une solution de relaxation linéaire optimale. ElHallaoui et al. [9] présentent une version améliorée de MPDCA en proposant l'algorithme d'agrégation bi-dynamique de contraintes ou BDCA (*Bi-Dynamic Constraint Aggregation*). Dans cet algorithme, l'agrégation est appliquée dynamiquement à la fois au problème maître et au sous-problème, réduisant ainsi la taille des sous-problèmes et donc le temps de résolution des sous-problèmes.

De là, et avec l'émergence de l'apprentissage machine ou ML (*Machine Learning*), une idée est née : utiliser les modèles ML pour trouver de bonnes grappes, qui seront fournies comme information initiale aux algorithmes de recherche opérationnelle (RO), améliorant ainsi la qualité des solutions et la rapidité avec laquelle ces solutions sont découvertes. L'objectif global serait donc de permettre le traitement de problèmes d'optimisation plus importants à l'aide de l'apprentissage machine. Cette thèse est donc une étude sur l'intégration de l'apprentissage machine dans le CPP en transport aérien, afin de proposer de meilleures solutions initiales et de meilleures grappes de DCA initiales pour le solveur RO. Plus spécifiquement, nous utilisons l'apprentissage machine pour apprendre à partir de solutions d'instances similaires afin de produire des prédictions sur certaines parties de la solution de la nouvelle instance. Cette information alimente l'optimiseur de programmation mathématique pour terminer le travail en tenant compte de la fonction de coût et des contraintes complexes.

Dans une première partie de ce travail, nous étudions la performance des algorithmes d'apprentissage machine pour résoudre le problème de prédiction de connexion de vols dans lequel l'objectif est de prédire le prochain vol qu'un équipage doit suivre dans son horaire avec seulement une information partielle (défini comme problème de classification multi-classes). Nous adaptons une solution basée sur les réseaux de neurones de plusieurs façons pour améliorer la précision des prédictions : en proposant de meilleurs encodages de caractéristiques de l'entrée, en réduisant le nombre de classes à prédire, en utilisant des contraintes appropriées au domaine et en proposant

des prédicteurs qui peuvent s'abstenir de faire une prédiction. Nous motivons le problème ci-dessus dans le contexte du CPP hebdomadaire. L'idée est d'exploiter un grand volume de données de vol, c'est-à-dire la solution existante comptant des dizaines de milliers de vols sur plusieurs mois, et d'appliquer des algorithmes d'apprentissage supervisés pour construire un prédicteur du problème de connexion de vol, afin d'obtenir une bonne information initiale pour accélérer un optimiseur RO (appelé GENCOL) qui doit être exécuté sur de nouvelles données de planification. Grâce à notre algorithme, nous pouvons traiter des milliers de vols, contrairement aux techniques standard décrites dans la littérature. Au meilleur de notre connaissance, il n'a jamais été question auparavant de trouver de bonnes informations initiales à l'aide d'algorithmes ML afin d'optimiser un horaire d'équipage d'une compagnie aérienne de cette ampleur. Comme preuve de concept, nous proposons une heuristique simple pour former des grappes initiales de rotations d'équipage qui peuvent être fournies au solveur GENCOL, ce qui donne une amélioration de vitesse 10x et une économie de 0.2% qui se traduirait par des millions de dollars d'économies annuelles pour une grande compagnie.

Dans la deuxième partie de ce travail, nous proposons de combiner plusieurs méthodes mises en œuvre, développées et testées sur de petits ensembles de données, afin d'obtenir un nouveau solveur Commercial-GENCOL-DCA pour les CPPs de grande taille : des problèmes mensuels comportant jusqu'à 50 000 vols. Nous utilisons l'apprentissage machine pour proposer des grappes initiales de vols ayant une forte probabilité d'être effectuées consécutivement par le même équipage, dans une solution optimale. Le nouvel algorithme Commercial-GENCOL-DCA combinant CG, DCA et l'algorithme de simplexe primal amélioré (IPS) sera utilisé pour assembler et modifier ces grappes, au besoin, afin de produire une bonne solution. On ajoute le nom "Commercial" car l'algorithme comprend la couche du système commercial qui modélise toutes les contraintes du problème industriel. Cette nouvelle approche, en commençant par l'apprentissage machine et en terminant l'optimisation par la programmation mathématique, permettra de résoudre des problèmes hebdomadaires comportant jusqu'à 10 000 vols et d'éviter la perte d'optimalité résultant de la décomposition heuristique en petites périodes de temps (2 jours) dans l'approche à horizon fuyant.

Dans la troisième partie de ce travail, nous proposons un nouveau prédicteur structuré appelé réseaux à noyaux convolutifs structurés ou SCKN (*Structured Convolutional Kernel Networks*), qui combine les propriétés des architectures d'apprentissage profond, la flexibilité non paramétrique des méthodes du noyau et les prédicteurs structurés. Le CKN (*Convolutional Kernel Networks*) apprend à approximer la carte des caractéristiques du noyau sur les données d'entraînement. Il peut être interprété comme un type particulier de CNN (*Convolutional Neural Networks*). Nous montrons que cette approche alternée offre un grand potentiel dans un cadre d'apprentissage supervisé et couplé. Nous montrons que l'utilisation de cette combinaison de manière supervisée surpasse les méthodes de pointe en termes de sous-optimalité primale et de précision du test sur le jeu de

données OCR (*Optical Character Recognition*). Ensuite, nous appliquons la méthode proposée à un ensemble de données de connectivité en vol ou FCD (*Flight-Connectivity Dataset*) pour proposer de bonnes solutions initiales à un solveur de CSP des compagnies aériennes. Plus précisément, nous utilisons une structure de réseau de vols modélisée comme un graphe CRF (*Conditional Random Fields*) général (dirigé), où les noeuds correspondent aux coordonnées spatio-temporelles et les arcs représentent les tâches effectuées par les membres d'équipage (vols, repositionnements, connexions, repos, etc.) capables d'incorporer les contraintes locales dans le processus d'entraînement des prédicteurs ML. Une fois que les poids du SCKN ont été formés, nous alimentons la solution produite au solveur Commercial-GENCOL-DCA comme solution initiale et comme grappes initiales pour l'agrégation dynamique de contraintes ou DCA (*Dynamic Constraint Aggregation*). Nous montrons que cette approche permet d'accélérer le processus d'optimisation du CPP, tout en offrant de meilleures solutions.

La présente thèse est organisée comme suit : tout d'abord nous situons le lecteur au chapitre 2 dans le contexte géneral de notre travail. Nous y décrivons le problème général CSP et le CPP, avant de présenter la méthode de décomposition de Dantzig-Wolfe sur laquelle la méthode de géneration de colonnes est basée. Une revue de littérature est présentée au chapitre 3, nous y rappelons les différentes variantes du SPP, les approches proposées de résolution, les méthodes et techniques utilisées dans la littérature pour combiner l'apprentissage machine et la recherche opérationnelle puis une revue de différentes méthodes d'apprentissage machine. Le chapitre 5 est consacré à notre première contribution intitulée : "Flight-Connection Prediction for Airline Crew Scheduling to Construct Initial Clusters for OR Optimizer". Ensuite, nous présentons au chapitre 6 notre deuxième contribution, intitulée "Machine Learning in Airline Crew Pairing to Construct Initial Clusters for Dynamic Constraint Aggregation". Nous présentons au chapitre 7 notre troisième travail, à savoir : "Réseaux à noyaux convolutifs structurés pour la reconnaissance optique de caractères et la planification des horaires d'équipage". En conclusion, nous faisons au chapitre 8 une synthèse globale de nos contributions et au chapitre 9 les différentes extensions de nos travaux de recherche ainsi que des propositions de nos perspectives de recherche.

CHAPITRE 2 CONTEXTE GÉNÉRAL

Cette thèse porte principalement sur le problème de CPP. Ce problème fait partie du CSP et est formulé en pratique comme un SPP résolu avec la méthode CG. Dans ce chapitre, nous présentons d'abord le CSP dans la section 2.1. Ensuite, nous présentons le CPP dans la section 2.2 et expliquons la décomposition de Dantzig-Wolfe [10] sur laquelle la méthode de CG est basée dans la section 3.1.2. Une description du mécanisme de fonctionnement de cette méthode clôt le chapitre.

2.1 Problème de planification des horaires d'équipage

Le problème de planification des horaires d'équipage des compagnies aériennes (CSP) est le problème de l'affectation d'un groupe de membres d'équipage à un ensemble de vols actifs de sorte que tous les vols actifs soient couverts, tout en respectant les règles et les conventions collectives, imposées principalement par les organisations syndicales et de sécurité. Les restrictions complexes en font l'un des problèmes d'affectation des équipages les plus difficiles de l'industrie du transport. Les coûts de carburant sont les coûts d'exploitation directs les plus élevés de la plupart des compagnies aériennes, mais les coûts liés à l'équipage viennent au deuxième rang. Pour cette raison, on s'est beaucoup concentré sur la planification efficace d'horaires d'équipage [11].

En raison de sa complexité et de sa taille, le CSP est généralement séparé en deux étapes : le problème de CPP et le problème d'horaires personnalisés (CRP) . Traditionnellement, les étapes se traitent séquentiellement.

Le CPP détermine les ensembles de segments de vol (*flight segments*) qui seront effectués par le même équipage. Différentes compagnies aériennes ont des règles différentes, mais les caractéristiques principales des rotations anonymes sont communes à toutes les compagnies aériennes.

Le CRP consiste à affecter un bloc mensuel à chaque membre d'équipage, un bloc étant une série de rotations entrecoupées de vacances, de jours de repos et d'entraînement. Des restrictions supplémentaires peuvent être imposées en fonction des besoins de chaque membre d'équipage. Les horaires doivent respecter l'ensemble des règles de sécurité et des règles contractuelles.

Formellement, le CPP et le CRP sont généralement modélisés par le biais du SPP ou du SCP avec des contraintes supplémentaires. Ces problèmes sont connus pour être assez difficiles à résoudre. La complexité d'un tel problème est due au nombre de variables et de contraintes impliquées dans la définition du problème. En bref, il s'agit de problèmes de programmation en nombres entiers de grande taille. Dans la littérature, deux méthodes courantes sont fréquemment employées : (i) B&P (*Branch-and-Price*) [12, 13, 14] ; et (ii) relaxation lagrangienne [15, 16, 17].

2.2 Problème de rotations d'équipage

Dans cette thèse, nous nous intéressons plus particulièrement au CPP. Le défi dans la construction des rotations consiste à établir de manière efficace et précise des horaires hebdomadaires pour les équipages, de façon à couvrir chaque vol exactement une fois à un coût minimum pour la compagnie aérienne. En développant ces rotations, il faut également respecter certaines conditions supplémentaires qui varient selon les applications et qui découlent généralement des ententes de travail de chaque équipage. Chaque équipage est associé à une ville appelée base d'où il doit partir et à laquelle il doit retourner. Par conséquent, pour une catégorie d'équipage et un type de flotte donnés, le CPP trouve un ensemble des rotations à coût minimum de sorte que chaque vol actif sur l'horizon est inclus dans exactement une rotation. L'approche de résolution dépend de la taille de la compagnie aérienne, de la structure du réseau (par exemple, en étoile), des règles et réglementations, ainsi que de la structure des coûts. Aux fins de ce travail, chaque jour ouvrable est appelé service de vol et se compose d'une séquence de segments de vol ; une rotation est une séquence de services de vol séparés par des périodes de repos quotidiennes ; et un DH est un vol au cours duquel un membre d'équipage vole en tant que passager à des fins de relocalisation [18].

Les contraintes de ce problème peuvent être divisées en quatre catégories :

- Contraintes de couverture : dans la solution, nous devons sélectionner les rotations pour que chaque vol soit couvert par un seul équipage.
- Contraintes globales supplémentaires liant un ensemble de rotations : heures de travail disponibles dans chaque base, limitation du pourcentage de rotations de certaines durées, etc.
- Contraintes de continuité : pour qu'une rotation soit réalisable, les tâches ou les vols qu'elle couvre doivent être enchaînés dans le temps et dans l'espace.
- Contraintes locales : liées à l'accord de travail (par exemple, heures de travail par jour, durée de vol par jour, nombre de jours d'absence de la base)

Trois horizons ou périodes de temps sont généralement étudiés : un jour, une semaine et un mois [14]. Le problème quotidien suppose que les vols sont identiques ou relativement similaires pour chaque jour de la période de planification. Le problème suppose également que des rotations de coûts minimum sont générés pour les vols programmés pour une journée. La solution cyclique est produite, c'est-à-dire que le nombre d'équipages présents dans chaque ville le soir est le même que le matin. Ainsi, les solutions hebdomadaires et mensuelles peuvent être produites en juxtaposant la solution quotidienne. Lorsque les horaires des vols ne sont pas exactement les mêmes tous les jours, les rotations non conformes aux règles de vol sont supprimées, car elles ne sont plus valables et peuvent être réoptimisées pour créer de nouvelles rotations couvrant des vols non précédemment couverts. Le problème hebdomadaire suppose que les vols sont identiques (ou relativement similaires) chaque semaine, et le CPP est résolu pour les vols actifs pendant une semaine typique.

Dans la littérature, le CPP a été modélisé comme un SCP ou un SPP, dans lequel chaque ligne est une contrainte associée à chaque vol et toutes les rotations réalisables sont traitées comme des variables [19, 20, 18]. Il existe généralement des contraintes supplémentaires qui imposent diverses restrictions, comme le temps de vol maximal pour chaque base, le nombre maximum d'équipage de chaque base utilisés chaque jour, etc. Le nombre de rotations réalisables est extrêmement élevé, il est donc souvent impossible de toutes les considérer.

CHAPITRE 3 REVUE DE LITTÉRATURE

Le CSP est généralement séparé en deux étapes qui sont résolues séquentiellement. La première consiste à construire les rotations d'équipage (CPP) et la seconde consiste à construire les blocs mensuels (CRP). Pour la plupart des applications, le CPP et le CRP sont résolus à l'aide de l'algorithme CG.

Dans ce chapitre, nous décrivons dans un premier lieu le CPP et les méthodes de résolution utilisées dans la litérature. Ensuite, nous détaillons les méthodes utilisées pour combiner l'apprentissage machine et la recherche opérationnelle, aussi bien dans le cadre de l'optimisation combinatoire que dans le cadre de la planification des horaires d'équipage. Finalement, nous présentons différents algorithmes d'apprentissage machine, en commençant par les méthodes standard, puis les méthodes d'apprentissage profond pour finir avec les méthodes de prédiction structurée et les méthodes de prédiction structurée profondes.

3.1 Problème de rotations d'équipage

3.1.1 Variantes et complexité

Pour chaque catégorie d'équipage et chaque type de flotte d'aéronefs, la formulation mathématique du CPP vise à trouver un ensemble de rotations à un coût minimal afin que chaque vol prévu soit effectué par un équipage dans le respect des contraintes des conventions collectives et des divers règlements de sécurité aérienne. Le CPP consiste à minimiser les coûts associés aux personnel d'équipage nécessaires à l'exploitation d'un type particulier d'avion. La fonction objectif minimise les coûts totaux des rotations. Les contraintes garantissent que chaque segment est couvert exactement une fois et imposent des exigences binaires sur les variables de rotations. Le CPP est un problème difficile. Sa complexité est due au nombre de variables et de contraintes impliquées dans la définition du problème, pouvant atteindre des millions de colonnes, ce qui ne tient pas dans la mémoire de l'ordinateur. Le CPP est donc un problème de programmation en nombres entiers de grande taille, l'un des plus importants sur le plan opérationnel. De plus, il existe un grand nombre de contraintes juridiques et réglementaires, comme le temps de repos minimal, le temps de connexion minimal entre deux services de vol consécutifs et le positionnement du nombre d'heures travaillées par base. Pour répondre aux contraintes complexes, les pilotes et les membres d'équipage de cabine doivent parfois être des passagers plutôt que des membres de l'équipage de conduite, ce qui signifie qu'ils sont des passagers plutôt que des travailleurs. Ce vol de repositionnement ou DH (*deadhead*) s'explique par l'inadéquation entre horaires d'équipage et horaires des vols et contribue générale-

ment à des coûts élevés aux opérations aériennes [21].

Pour les grandes flottes, résoudre globalement le problème hebdomadaire ou réoptimiser globalement le problème mensuel peut être beaucoup trop long. Ainsi, l'approche à horizon fuyant est utilisée pour accélérer le processus de résolution [22]. Des études récentes se sont concentrées sur les problèmes hebdomadaires et mensuels. En raison des périodes de vacances et des différences dans les horaires de vol, le problème mensuel est le plus précis [14].

3.1.2 Méthodes de résolution

Plusieurs approches heuristiques utilisant la programmation mathématique ont été développées pour résoudre les SPPs et les SCPs. On présente dans cette section une revue de ces méthodes en commençant par la programmation mixte en nombres entiers, puis la programmation linéaire, pour présenter par la suite la méthode CG.

Programmation mixte en nombres entiers

La programmation linéaire mixte en nombres entiers ou MIP (*Mixed-Integer Programming*) a été utilisée avec succès dans plusieurs domaines de la recherche opérationelle. Cependant, son efficacité dans le domaine de la planification des horaires a été limitée en raison des longs temps d'exécution pour trouver des solutions à des problèmes de taille même modeste. Le modèle utilisé lors de la résolution contraint certaines des variables de décision à être entières dans la solution optimale tandis que d'autres peuvent être fractionnaires. L'approche de résolution standard est généralement la technique de séparation et évaluation avec plans coupants.

Marsten et Shepardson [23] présentent ce qui est probablement le premier système commercial avec un solveur MIP spécialisé pour la formulation du SPP. Ils présentent les résultats de différentes entreprises et proposent une décomposition heuristique pour des problèmes plus importants. Anbil et al. [1] ont introduit une méthode intégrée avec un objectif de minimisation des coûts. Dans cette recherche impliquant *American Airlines Decision Technologies* et *IBM*, de nombreuses rotations possibles (colonnes) ont été générés a priori, et une partie est introduite dans le solveur MIP. À chaque itération, plusieurs rotations hors-base sont rejetées et de nouvelles rotations sont insérées. La procédure se poursuit pour les rotations générées jusqu'à ce que toutes les rotations aient été prises en considération pour obtenir une solution optimale pour les rotations générées [20].

Approches de programmation linéaire

Étant donné la difficulté d'obtenir des solutions optimales dans un délai réaliste à l'aide d'un MIP, le CPP peut être relaxé pour obtenir une solution de programmation linéaire (LP). Cependant, cette

approche LP ne peut pas être une méthode de résolution des CSP en soi. Elle est donc utilisée avec quelques heuristiques supplémentaires qui déduisent une solution entière de la solution relaxée. L'heuristique, dans certains cas, convertit les variables de décision en entiers (méthodes d'arrondissement). Lorsque c'est le cas, cela signifie qu'il ne s'agit pas d'une solution optimale, mais que le compromis est un temps de résolution beaucoup plus court (et des tailles de problèmes plus importants) plutôt que l'acceptation d'une solution presque optimale comme étant suffisante. L'autre aspect de la relaxation LP est qu'une solution est presque toujours garantie, ce qui n'est pas toujours le cas avec le MIP.

Comme cette approche LP est pratique et qu'elle permet de réduire les coûts pour les ordinateurs dotés d'une mémoire et d'une capacité de traitement extrêmement importantes, il existe de nombreuses applications documentées dans l'industrie, comme Cacchiani et al. [24] ou Rasmussen et al. [25] pour le CSP.

Cependant, pour les CPPs de grande taille, comme le nombre de rotations réalisables est extrêmement élevé, il est souvent impossible de toutes les considérer. Le problème a donc été traité de manière heuristique en deux étapes dans les premiers systèmes développés : la première étape génère un sous-ensemble de bonnes rotations par énumération, et la seconde utilise le SPP pour sélectionner les meilleures rotations de ce sous-ensemble.

Les recherches actuelles dans la littérature indiquent que les algorithmes heuristiques présentent trois inconvénients principaux. Premièrement, ils ne considèrent pas tous les vols actifs simultanément et doivent effectuer plusieurs itérations avant de pouvoir obtenir une solution de haute qualité [26]. Deuxièmement, les algorithmes ne considèrent pas toutes les rotations faisables [27]. Troisièmement, l'écart par rapport à la solution optimale est important, les nouvelles solutions peuvent être loin d'une solution optimale globale [28]. En conséquence, des approches plus sophistiquées ont été proposées au fil des ans. La méthode CG a donc été proposée, elle débute avec un sous-ensemble de colonnes pour ce problème [4]. Un SCP est résolu avec l'algorithme du simplexe et des colonnes supplémentaires sont générées en résolvant un problème du plus court chemin avec contraintes de ressources.

Décomposition Dantzig-Wolfe

La décomposition de Dantzig-Wolfe [10] est une méthode de reformulation classique permettant de résoudre un programme linéaire de grande taille dont la matrice de contraintes comprend un ensemble de blocs indépendants couplés par un ensemble de lignes de liaison.

Le problème initial est reformulé en un programme maître et plusieurs sous-problèmes indépendants. Cette reformulation repose sur le fait qu'un polyèdre convexe non vide et borné peut être

représenté comme une combinaison convexe de ses points extrêmes (ou, dans le cas d'un polyèdre sans borne, une combinaison convexe de ses points extrêmes et une combinaison pondérée de ses rayons extrêmes).

Étant donné que le nombre de points extrêmes des polytopes de sous-problèmes peut être exponentiellement grand, un problème maître restreint ou RMP (*Restricted Master Problem*) est résolu à la place, en utilisant seulement un sous-ensemble des points extrêmes (ou colonnes). En utilisant une solution duale optimale du RMP pour calculer les coûts réduits, chaque sous-problème est résolu indépendamment pour trouver un nouveau point extrême dont le coût réduit est négatif. Ce processus, appelé CG, est répété jusqu'à ce qu'il n'y ait plus de points extrêmes ou de colonnes avec des coûts réduits négatifs. Une solution au RMP fournit alors une solution optimale.

Génération de colonnes

La méthode la plus répandue depuis les années 1990 a été de résoudre le SPP avec l'algorithme CG insérée dans un algorithme de séparation et évaluation ou B&B (*branch-&-bound*) [4]. Cette méthode, ainsi que d'autres, sont décrites dans un article synthèse récent de Deveci et al. [6]. Selon cette dernière étude, l'algorithme CG était l'approche la plus fréquemment utilisée.

À chaque itération, l'algorithme CG considère un RMP qui contient un sous-ensemble des variables (colonnes). Le RMP optimal est résolu par un algorithme LP standard tel que la méthode simplexe et trouve une valeur de fonction objectif optimale et une paire de solutions primale et duale. Compte tenu de la solution duale optimale du RMP, le sous-problème actuel tente de trouver des colonnes avec des coûts réduits négatifs. Si de telles colonnes sont trouvées, elles sont ajoutées au RMP pour la prochaine itération. Pour le CPP, chaque sous-problème correspond à un problème du plus court chemin avec contraintes de ressources et est généralement résolu par programmation dynamique. Lorsqu'aucune variable ayant un coût réduit négatif ne peut être trouvée, la solution optimale pour le RMP est optimale pour le MP. En pratique, l'algorithme CG est souvent arrêté avant que l'optimalité ne soit atteinte en raison de la convergence lente (*tailing-off effect*).

En résumé, l'algorithme CG résout, pour chaque itération, un RMP et un ou plusieurs sous-problèmes. Cet algorithme est reconnue pour résoudre la relaxation linéaire d'un MILP dans de nombreuses applications, en particulier dans les domaines de la planification des horaires d'équipage et du routage des véhicules. Le lecteur est prié de se reporter aux articles fondateurs, Desrosiers et al. [29] et Desrosiers et al. [30].

3.2 Combiner Apprentissage Machine et Recherche Opérationnelle

3.2.1 Combiner Apprentissage Machine et Optimisation Combinatoire

L'optimisation combinatoire (OC) est une branche de la recherche opérationnelle (RO) qui consiste à trouver dans un ensemble discret un parmi les meilleurs sous-ensembles (ou solutions) réalisables, par rapport à une fonction objectif. Comme la majorité des problèmes intéressants sont NP-difficiles, l'utilisation de techniques d'apprentissage machine ou ML (*Machine Learning*) a été proposée avec succès dans plusieurs publications récentes portant sur divers types de problèmes. L'objectif de l'utilisation du ML est le plus souvent de réduire le temps de calcul des techniques traditionnelles pour résoudre des problèmes de tailles plus importantes. Toutefois, le ML peut aussi contribuer à trouver de meilleures solutions. Nous fournissons un bref aperçu des architectures dans lesquels le ML a été déployé pour résoudre les problèmes d'OC et nous présentons quelques résultats récents.

Dans les cas où on a une compréhension structurelle du problème d'OC et où on possède ainsi une connaissance théorique des décisions à prendre par l'algorithme d'optimisation, le ML peut être utilisé pour fournir des approximations rapides de ces décisions, réduisant ainsi le temps de calcul requis. C'est ce qu'on appelle dans la littérature *l'apprentissage par imitation* : on dispose d'échantillons du comportement attendu qui sont utilisés dans le modèle de ML comme démonstrations pour l'apprentissage. En tant que tel, le modèle ML a pour tâche de minimiser une fonction de coût comparant les décisions de l'expert et les siennes. Dans ce cas, l'application du ML peut être simple, mais l'hypothèse selon laquelle nous possédons suffisamment d'échantillons de démonstration pour l'apprentissage n'est pas toujours valide.

D'autre part, il se peut qu'on ne soit pas en mesure de comprendre suffisamment bien la structure du problème et, dans ces cas, le modèle ML peut être déployé pour découvrir la structure elle-même. Nous appelons ce cas d'utilisation *Apprentissage d'une politique*. Il s'agit d'une application de l'apprentissage par renforcement : dans chaque état, le modèle dispose d'un ensemble d'actions qu'il peut effectuer et chacune porte une récompense ; le modèle ML essaie alors de maximiser la somme attendue des récompenses obtenues par la succession de ses actions. Le défi commun à la mise en œuvre de cette approche est de trouver une fonction de récompense appropriée afin que l'algorithme d'apprentissage ne tombe pas dans les minima locaux ou ne se termine pas sans une couverture d'exploration suffisante. Un exemple clair est la résolution du problème du voyageur de commerce ou TSP (*Travelling Salesman Problem*). Nous pouvons attribuer une récompense positive pour la visite des nœuds qui n'ont pas encore été visités ; il s'agit d'une heuristique gloutonne qui aboutit bien à une permutation mais dont les performances s'avèrent assez médiocres. Une approche intéressante est présentée dans Khalil et al. [31] qui utilisent un réseau de plongement

de graphes ou GEN (*Graph Embedding Network*) et un apprentissage par renforcement pour apprendre une politique de sélection de noeud. L'architecture GEN permet une "caractéristation" des noeuds et de leurs propriétés par rapport à leurs voisins. Le modèle apprend ainsi une politique qui fait la distinction entre les nœuds en ce qui concerne l'utilité de faire une visite.

L'imitation et l'apprentissage d'une politique ne s'excluent pas mutuellement et, pour certaines tâches d'apprentissage, il peut être intéressant d'explorer une combinaison des deux approches : que les étapes initiales d'apprentissage du modèle de prédiction soient réalisées par des démonstrations avec une taille d'échantillons limitée, puis que la politique acquise soit améliorée par un apprentissage par renforcement avec une fonction de récompense.

Il existe diverses approches architecturales pour la mise en œuvre de modèles ML pour les problèmes d'OC. Dans le cas le plus simple, le modèle ML est le solveur lui-même. Cette approche a été utilisée avec succès pour résoudre certaines classes de TSP euclidiennes en utilisant l'apprentissage profond. Vinyals et al. [32] utilisent cette approche en combinaison avec *l'apprentissage par imitation* : en utilisant des solutions TSP exactes pour des graphes plus petits et des approximations pour les graphes plus grands, ils génèrent un ensemble de démonstrations qui sont ensuite codées par un RNN (*Recurrent Neural Networks*). Un autre RNN sert de décodeur qui peut être utilisé pour produire une permutation sur les noeuds TSP. Cette méthode permet de créer un modèle capable de traiter des entrées de différentes tailles.

Dans les cas plus complexes, le ML est utilisé pour enrichir le processus de décision d'un algorithme d'OC existant. On peut utiliser un modèle ML pour traiter la définition du problème afin d'en extraire une structure significative, ou réduire l'espace du problème. Dans ces cas, le ML jouera un rôle préparatoire pour l'algorithme d'OC. Un exemple peut être trouvé dans Kruber et al. [33] pour résoudre les problèmes MILP. En utilisant la décomposition Dantzig-Wolfe d'une instance MILP, ils entraînent des modèles ML afin de prédire lequel des deux solveurs va résoudre l'instance de manière optimale et plus rapide. Toutefois, les modèles ML ne sont pas utilisés pour trouver les solutions finales. De même, Lodi et al. [34] examinent le problème de l'emplacement d'installations où un modèle ML est utilisé pour prédire si une instance de problème dérivée aura une solution similaire à sa référence. Ils ajoutent ensuite ces données supplémentaires dans le solveur final, ce qui permet un temps de calcul plus rapide, permettant ainsi aux sociétés du secteur de l'énergie d'évaluer la viabilité des emplacements des installations sans effectuer le calcul possible mais coûteux.

En général, on peut implémenter une boucle entre le modèle ML et l'algorithme OC. L'algorithme maître combine les deux en appelant à plusieurs reprises le modèle ML afin de prendre une classe de décisions qui sont ensuite utilisées dans les itérations successives de l'algorithme OC. Ceci a été utilisé pour guider le processus de branchement dans l'algorithme de séparation et évaluation

ou B&B (*Branch & Bound*) en apprenant l'ordre dans lequel explorer les noeuds. Lui et al. [35] utilisent le ML à trois niveaux différents : apprendre une politique qui choisit parmi les différentes stratégies pour différents types de problèmes, puis au sein de chaque type de problème, la difficulté de trouver la solution est évaluée, et enfin, la stratégie de recherche est adaptée par niveau de l'arborescence B&B basée sur des caractéristiques spécifiques aux noeuds. Khalil et al. [36] utilisent le ML pour se rapprocher des règles de branchement performantes. Leur ensemble de données pour la formation est généré par la stratégie de branchement fort ou SB (*Strong Branching*), qui est efficace mais très coûteuse sur le plan informatique. Une fois que leur modèle ML est formé, ils renoncent à l'utilisation du SB et appellent leur modèle pour déterminer la variable de branchement suivante. Contrairement aux autres modèles qui supposent une connaissance préalable du domaine d'application, Václavík et al. [37] proposent une approche qui peut être déployée pour un problème combinatoire queconque qui peut être résolu par CG. un modèle de régression est utilisé pour resserrer la borne supérieure de la fonction objectif du problème de recherche de variables améliorantes (*pricing*). Cette borne supérieure est utilisée pour élaguer plus efficacement l'espace de recherche du problème en question, ce qui permet de réduire le nombre d'états sélectionnés à considérer et donc de réduire le temps de calcul, surtout pour les problèmes de grande taille, où il est difficile de calculer une borne supérieure serrée dans un délai raisonnable.

Nous renvoyons le lecteur intéressé à Bengio et al. [38] pour un aperçu plus détaillé de l'utilisation des modèles ML en recherche opérationnelle.

3.2.2 Apprentissage machine pour la planification des horaires de personnel aérien

L'application des techniques ML pour résoudre les problèmes d'OC a permis de réaliser des progrès dans la résolution des problèmes spécifiques aux compagnies aériennes. Les modèles ML utilisés sont fréquemment rencontrés dans d'autres domaines problématiques, par exemple, certaines variétés d'algorithmes génétiques ou GA (Genetic Algorithms). Le GA est un type d'algorithme évolutif qui utilise la mutation et croisement en tant que principaux opérateurs de recherche pour l'optimisation ; l'applicabilité et le succès du GA en optimisation dépendent en grande partie du choix judicieux de ces opérateurs. Elsayed et al. [39] étudient une variante du GA avec un croisement généré à partir de trois individus et introduisent un opérateur de diversité aléatoire au lieu d'utiliser la mutation. Ils rapportent qu'un GA avec ces opérateurs de recherche est en mesure de traiter une plus grande variété de problèmes. Dans une approche différente, Akbari et al. [40] conçoivent un GA spécifiquement pour résoudre les problèmes de planification des tâches. Ils mettent en place un nouvel opérateur qu'ils appellent "Inversion", qui vise à augmenter la diversité des échantillons générés au cours des générations suivantes, réduisant ainsi de manière méta-heuristique le nombre d'horaires répétés ou dupliqués lors de l'optimisation.

Les problèmes de planification pour les compagnies aériennes incluent la planification des vols, la planification de la flotte, l'établissement des itinéraires des aéronefs et l'établissement des horaires des équipages. Graf et al. [41] examinent la planification des horaires de vol et utilisent la programmation linéaire traditionnelle pour la résoudre. Le même problème est traité dans Tsai et al. [42] utilisant un GA avec des chromosomes définis comme objets bidimensionnels représentant des horaires possibles. Pandit et al. [43] étudient une nouvelle approche de la planification de la flotte d'aéronefs en utilisant un simple réseau de neurones à 3 couches.

En raison du coût total élevé associé, la planification des horaires d'équipages des compagnies aériennes est de la plus haute importance parmi les problèmes de planification des compagnies aériennes, en particulier le CPP. Le CPP consiste à trouver la solution à coût minimal qui couvre chaque vol actif, tout en respectant un grand nombre de contraintes. Diverses études ont abordé le problème à l'aide des GA. [44] introduisent un opérateur de "perturbation" pour leur GA. Étant donné une génération de rotations, cet opérateur élimine certaines des rotations d'équipages les plus coûteuses, ce qui rend la solution irréalisable. Une recherche est alors effectuée pour tenter de trouver une nouvelle solution avec de nouvelles rotations et à moindre coût ; si cela échoue, le chromosome est inversé. Comme l'opérateur de perturbation a tendance à converger très rapidement vers une solution, ils ne l'applique que pour certaines générations et affirment que son inclusion permet de trouver une meilleure solution. Une autre étude sur la résolution des CPP à l'aide d'algorithmes évolutifs se trouve dans Azadeh et al. [45]. Ils implémentent un algorithme d'optimisation par essaims particulaires (*particle swarm optimization*) qui vise à optimiser une solution via un parcours de particules dans un espace de recherche composé de toutes les solutions possibles. Les particules se déplacent à chaque itération par rapport à une dynamique définie et se communiquent les minima qu'elles ont trouvés afin de trouver une meilleure solution. Deveci et Demirel [46] adressent une variante de CPP qui consiste à trouver un ensemble de rotations à coût minimal couvrant tous les vols à l'aide de divers algorithmes évolutifs. Ils ont amélioré les solutions précédentes en utilisant un algorithme hybride comportant un GA avec la méthode de l'escalade (*hill-climbing method*). Ainsi, leur utilisation de ML a créé un ensemble de données plus riche pour les méta-heuristiques. Bien que l'approche proposée soit la plus performante, comparée à deux variantes de GA, les instances utilisées sont de petite taille, contenant jusqu'à 750 vols.

3.3 Algorithmes d'apprentissage machine

Les algorithmes d'apprentissage machine peuvent être regroupés en trois grandes catégories selon le type d'apprentissage : supervisé, non supervisé et semi-supervisé. L'apprentissage supervisé est la forme la plus courante utilisée pour résoudre différents problèmes [47]. On peut considérer qu'il s'agit de trouver la fonction la plus appropriée pour faire correspondre (mappe) un ensemble de

caractéristiques d'entrée à un ensemble de cibles basé sur des données d'entraînement étiquetées. La performance prédictive d'un algorithme d'apprentissage supervisé dépend du type de modèle utilisé [48], et le choix d'un modèle dépend principalement du type de données utilisées.

Un modèle ML est construit en se basant sur la relation entre les valeurs des attributs du prédicteur et la valeur cible [49]. Le défi des problèmes de classification consiste à prédire correctement la classe de nouvelles données à partir des données d'entraînement. Le surapprentissage est l'un des problèmes les plus difficiles des algorithmes de classification, lorsque le modèle offre de très bonnes performances pour les données d'entraînement, mais une performance médiocre pour les données de test. Cela s'explique principalement par la surcomplexité du modèle utilisé et le manque de données d'entraînement. Les algorithmes de classification tentent d'améliorer la généralisation du modèle en *(i)* réduisant la dépendance du modèle aux échantillons d'entraînement, en utilisant un paramètre de régularisation ; *(ii)* augmentant les données d'entraînement ; et *(iii)* augmentant artificiellement les données d'entraînement par un processus d'augmentation de la quantité de données.

3.3.1 Méthodes classiques

Arbres de décision

L'arbre de décision se compose de trois types de nœuds fondamentaux : le noeud racine, les nœuds internes et les nœuds feuilles. Le noeud supérieur est le nœud racine, les nœuds feuilles sont les nœuds terminaux de la structure et les nœuds intermédiaires sont appelés nœuds internes. Chaque nœud interne représente un test sur un attribut, chaque branche représente un résultat du test et chaque nœud feuille porte une étiquette de classe. De nombreux algorithmes, tels que ID3 (*Iterative Dichotomiser 3*), J48 et l'algorithme du forêt d'arbres décisionnels (*random forests*), sont utilisés pour concevoir l'ordre des caractéristiques de l'arbre de décision en fonction de l'importance de chaque caractéristique dans la tâche de classification.

ID3 L'algorithme d'arbre de décision ID3 a été développé par Stone et al. [50]. L'idée de base de l'algorithme ID3 est de construire l'arbre de décision en utilisant une recherche gloutonne descendante à travers les ensembles donnés pour tester chaque attribut à chaque noeud de l'arbre. ID3 utilise uniquement des attributs catégoriques pour créer un modèle d'arborescence. Les données bruyantes doivent être prétraitées pour augmenter la précision. J48 est une version avancée de ID3 et détermine la valeur cible des nouvelles données de test en fonction de différentes valeurs d'attributs des données d'entraînement [51]. Les nœuds internes d'un arbre de décision sont désignés par différents attributs, tandis que les branches fournissent les valeurs possibles de ces attributs. Les nœuds internes représentent les valeurs des variables dépendantes.

Forêt d'arbres décisionnels La forêt d'arbres décisionnels [52] est une méthode d'apprentissage d'ensemble pour la classification, la régression et d'autres tâches qui construit un arbre de décision au moment de l'apprentissage et l'utilise pour prédire les classes au moment de l'inférence. Cet algorithme permet de résoudre le problème de surapprentissage des arbres de décision. En tant que méthode de méta-apprentissage composé de multiples arbres individuels, la forêt d'arbres décisionnels peut être entrainé rapidement sur de vastes ensembles de données et, plus important encore, peut être diversifiée en utilisant des échantillons aléatoires pour construire chaque arbre dans la forêt.

Réseaux bayésiens

Les réseaux bayésiens ont été introduits par Pearl [53]. Un réseau bayésien fournit une représentation compacte d'une distribution de probabilité trop complexe pour être gérée à l'aide de spécifications traditionnelles et fournit une méthode systématique et localisée pour incorporer des informations probabilistes sur une situation. Le terme "réseaux bayésiens" couvre plusieurs domaines de problèmes et de techniques d'analyse de données et de raisonnement probabiliste, dans lesquels les données sont collectées pour un grand nombre de variables. L'objectif est de factoriser la distribution, de la représenter graphiquement et d'exploiter la représentation graphique.

SVM

Les machines à vecteurs de support (SVM) [54] sont l'un des classificateurs les plus répandus qui cherchent la ligne qui sépare les données en deux classes dans les problèmes de classification binaire. Dans les problèmes de classifications multiclasses, la frontière de décision devient un hyperplan. Bien qu'il s'agisse d'un classificateur linéaire, SVM peut être appliqué à des données séparables de façon non linéaire en transformant les données dans un espace de dimension supérieure où elles deviennent linéairement séparables. Il est intuitif qu'il existe une infinité de frontières de décision possibles qui séparent les classes. Cependant, l'algorithme de SVM tente de maximiser la marge entre la frontière de décision et les points les plus proches de chaque classe comme le montre la figure 3.1. Ces points sont appelés vecteurs de support. La frontière avec la distance maximale par rapport aux vecteurs de support est le meilleur hyperplan renvoyé par SVM. Cet hyperplan est utilisé ultérieurement pour classer les points de test en fonction de leur position par rapport à la frontière de décision. Le paramètre de régularisation reflète l'importance accordée aux classifications erronées.

Les SVM posent un problème d'optimisation quadratique qui vise à maximiser la marge entre les deux classes et à minimiser le nombre de classifications erronées. Cependant, pour les problèmes non séparables, afin de trouver une solution, la contrainte de classification erronée doit être as-

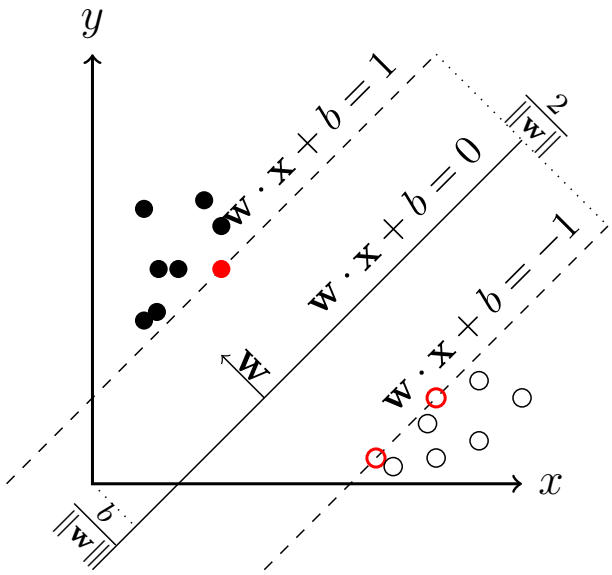


Figure 3.1 Illustration d'un classificateur SVM dans le cadre d'une classification binaire où $\mathbf{w}$ est le paramètre du SVM. $\frac{2}{\|\mathbf{w}\|}$ est la marge maximale entre les deux classes, $\frac{b}{\|\mathbf{w}\|}$ est l'hyperplan optimal renvoyé par SVM et les points en rouge sont les vecteurs de support.

souplie, et cela se fait en fixant un paramètre de régularisation. Plus la valeur de ce paramètre est élevée, mieux sera la performance du prédicteur sur l'ensemble d'entraînement, avec des frontières de décision plus complexes, ce qui signifie que le prédicteur risque de surapprendre. Inversement, une sélection judicieuse de ces valeurs manquera quelques points d'entraînement, au profit d'une meilleure régularisation sur l'ensemble de test.

3.3.2 Méthodes d'apprentissage profond

Les méthodes standard ont été utilisées avec une performance décente dans des tâches telles que la classification, la segmentation et la détection. Cependant, ces méthodes impliquent des ajustements empiriques des paramètres, ce qui les rend plus sujets aux erreurs [55]. D'autre part, les méthodes basées sur les réseaux de neurones profonds se sont révélées être une excellente alternative à l'ingénierie de caractéristiques en proposant des techniques d'extraction automatique des caractéristiques. Les modèles d'apprentissage profond extraient automatiquement les caractéristiques pertinentes pour une tâche donnée en analysant une grande quantité de données. L'énorme quantité de données disponibles et les avancées récentes en matière de ressources de calcul GPU ont révolutionné l'application des modèles d'apprentissage profond dans différents domaines [56, 57, 58, 59].

Apprentissage profond

Les réseaux de neurones convolutifs (CNN) sont un type particulier de réseaux de neurones pour le traitement de données de type grille. Par exemple, les données de séries chronologiques peuvent être considérées comme une grille 1D, les données d'image comme une grille 2D de pixels et les scans médicaux de scanner (CT) ou d'IRM comme des grilles 3D. La couche de convolution scanne (convolve) la matrice d'entrée avec un filtre (noyau) de taille prédéfinie. Les poids du noyau sont mis à jour pendant le processus d'apprentissage. L'architecture de CNN comme le montre la figure 3.2 est relativement simple et se compose de couches successives organisées de manière hiérarchique ; chaque couche comporte des convolutions avec des filtres appris, suivis d'une non-linéarité ponctuelle et d'une opération de sous-échantillonnage appelée "regroupement de caractéristiques" ou POOL (*Pooling*). Les couches de convolution successives extraient automatiquement les caractéristiques les plus pertinentes des données d'entrée et les insèrent dans les dernières couches de classification du réseau.

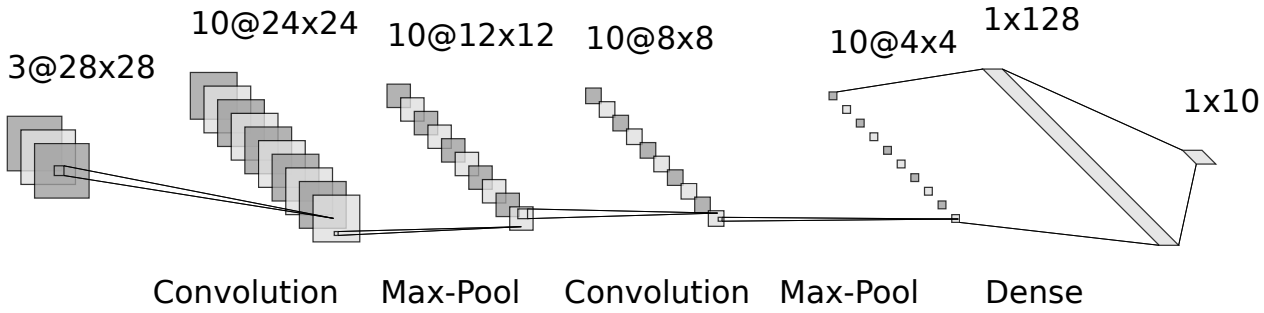


Figure 3.2 Ce classificateur reçoit comme entrée une image RVB (3 canaux) de taille 28×28 et se compose de deux couches de convolution, d'une couche cachée et d'une couche de sortie avec 10 classes (problème de classification multi-classes). Chaque couche de convolution est caractérisée par le nombre de filtres (10), la taille des filtres (5×5), la forme de l'opération Max-Pool (2×2) et le pas (le saut) dans l'application de la convolution (1). Chaque couche de convolution est suivie d'une non-linéarité ponctuelle (généralement sous forme d'une fonction d'activation *relu*). La couche cachée est caractérisée par le nombre de neurones (128) et d'une fonction d'activation *relu* et la couche de sortie dispose de 10 neurones (chacune produit la probabilité d'une parmi les 10 classes) et d'une fonction d'activation *softmax*.

Les réseaux de neurones récurrents ou RNN [60] constituent une autre famille de réseaux de neurones utilisés pour le traitement séquentiel des données. Des exemples de séquences comprennent des textes et des séquences d'ADN. Contrairement à CNN, les RNN modélisent les relations séquentielles entre une séquence de mots pour prédire ce qui va suivre. Comme le mot *recurrent* l'implique, la sortie de l'étape courante devient l'entrée de l'étape suivante. Cela permet au modèle de prendre en compte toutes les étapes précédentes dans sa prédiction, et pas seulement l'étape en

cours. Ceci est d'une grande importance dans la modélisation de texte car le mot suivant dépendra probablement des phrases précédentes, pas seulement du mot précédent. Des réseaux récurrents à mémoire court et long terme ou LSTM (*Long Short-Term Memory*) ont été introduits pour atténuer le problème de la disparition du gradient (*vanishing gradient*) des réseaux de neurones denses.

Une application des réseaux de neurones profonds consiste à mapper (projeter) les données d'entrée dans un espace de dimension supérieure, ce processus est appelé plongement de données (*embedding*). Les réprésentations résultantes peuvent être utilisées pour la visualisation des caractéristiques ou le partitionnement de données (*clustering*) non supervisé. Il peut également être utilisé comme entrée pour les modèles d'apprentissage machine supervisé.

Les modèles d'apprentissage profond incluent des millions de paramètres et représentent une fonction non convexe de grande dimension, ce qui rend l'optimisation plus difficile. Les fonctions non convexes contiennent de nombreux minima et points de selle locaux, en particulier dans les grandes dimensions. La fonction de coût est calculée pour les échantillons d'entraînement et inclut un terme de régularisation pour assurer la généralisation à de nouvelles données. Les algorithmes d'optimisation optimisent cette fonction et cherchent l'ensemble de paramètres le plus performant sur l'ensemble de test.

Les méthodes de descente de gradient sont largement utilisées dans de tels algorithmes comme technique d'optimisation de premier ordre. Cependant, l'évaluation de la première dérivée exacte est très coûteuse car elle nécessite d'évaluer le modèle sur chaque échantillon des données d'entraînement. Sans quoi, on peut aussi calculer la dérivée en échantillonnant un petit nombre d'exemples à partir du jeu de données, puis en prenant la moyenne sur ces exemples. Un tel algorithme d'approximation pour évaluer les dérivées est connu sous le nom d'algorithme du gradient stochastique ou SGD (*Stochastic Gradient Descent*). Les dérivées (donc l'erreur) sont ensuite rétropropagées sur le réseau pour mettre à jour les paramètres du modèle en conséquence.

Le taux d'apprentissage ou LR (*Learning Rate*) est une quantité multipliée par les gradients avant la mise à jour des paramètres du modèle et représente le pas effectué par l'algorithme pour réduire l'erreur. Un LR réduit conduit à une convergence lente, alors qu'un LR supérieur peut ne pas atteindre le point optimal. Par conséquent, le LR est l'un des hyperparamètres critiques dans la configuration du réseau. Différents optimiseurs ont différentes méthodes de réglage du LR. En SGD, le LR est statique et reste le même tout au long de l'entraînement, alors que les LR adaptatifs sont adoptés dans d'autres optimiseurs tels que : AdaGrad [61], RMSProp [62] et Adam [63]. De tels optimiseurs mettent à jour le LR initial pour s'adapter au processus d'apprentissage afin d'obtenir une convergence rapide vers le point optimal. L'optimisation des modèles profonds reste un domaine de recherche actif pour accélérer la convergence et minimiser la sensibilité aux hyperparamètres.

Réseaux à noyaux convolutifs

Malgré la bonne performance de CNN, leur entraînement reste difficile, car les réseaux à grande capacité peuvent nécessiter l'apprentissage de milliards de paramètres, ce qui nécessite à la fois une grande puissance de calcul (par exemple, des GPU) et des techniques de régularisation appropriées. Un schéma d'approximation appelé réseaux à noyaux convolutifs ou CKN (*Convolutional Kernel Networks*) a été introduit par Mairal et al. [64, 65, 66], ce qui rend l'approche du noyau (*kernel trick*) réalisable sur le plan du calcul. Ce schéma est un type de réseau de neurones à convolution supervisée formé pour approximer l'image du noyau. Semblable aux descripteurs hiérarchiques du noyau [67], des caractéristiques ou voisinages locaux de l'image (*local image neighborhoods*) sont projetés à des points dans un "espace de Hilbert à noyau reproduisant" ou RKHS (*Reproducing Kernel Hilbert Space*) via l'astuce du noyau. Ensuite, des représentations hiérarchiques sont construites via les compositions du noyau, produisant une séquence de "cartes caractéristiques" semblables à CNN, mais de dimension infinie. Pour rendre l'algorithme pratique et efficace, CKN fournit un schéma d'approximation qui peut être interprété comme un type particulier de CNN appris sans supervision. Pour effectuer un apprentissage de bout en bout à partir de données étiquetées, nous utilisons un principe simple mais efficace qui consiste à apprendre des sous-espaces discriminants dans les RKHS, où nous projetons les données. Les CKN supervisés sont introduits par Mairal [65], où les sous-espaces linéaires (un par couche) sont optimisés conjointement en minimisant une fonction de perte supervisée. La formulation s'avère être un nouveau type de réseau de neurones à convolution avec une paramétrisation non standard. Le réseau admet également des principes simples à apprendre sans supervision : apprendre les sous-espaces peut en effet être réalisé efficacement avec les techniques classiques d'approximation du noyau [68, 69]. Un CKN partage les propriétés caractéristiques du CNN en matière de dispersion et de partage de paramètres, en plus de ses hyperparamètres (par exemple, la taille des filtres, le nombre de filtres). Les CKN et les CNN diffèrent par la fonction de coût optimisée pour l'apprentissage des filtres et par le choix des non-linéarités.

3.3.3 Prédiction stucturée

Les réseaux de neurones profonds produisent un vecteur de variables aléatoires sans aucune dépendance entre ces variables aléatoires. L'apprentissage structuré, quant à lui, est un cadre pour résoudre des problèmes de classification et de régression en produisant des variables aléatoires dépendantes. De telles relations entre les quantités produites sont importantes dans de nombreux problèmes tels que la traduction automatique, le traitement du langage naturel et la segmentation des objets. Dans cette section, nous passons en revue certains des algorithmes de prédiction structurée et concluons. Dans la section suivante, nous explorons les études récentes proposant une

combinaison d'apprentissage profond et de prédiction structurée.

Champ aléatoire conditionnel

Les champs aléatoires conditionnels ou CRF (*Conditional Random Fields*) modélisent la probabilité conditionnelle d'une sortie structurée $y \in \mathcal{Y}$ (telle qu'une séquence d'étiquettes) à partir d'une entrée $x \in \mathcal{X}$ (telle qu'une séquence de mots) basée sur les caractéristiques $F(x, y)$ et paramètres w en utilisant :

$$p(y|x, w) = \frac{exp(w^T F(x, y))}{\sum_{y' \in \mathcal{Y}} exp(w^T F(x, y'))} \tag{3.1}$$

Étant donné n paires $\{x_i, y_i\}$ d'entraînement, l'approche standard pour la formation du CRF est de minimiser "log-vraisemblance négative".

$$\min_w f(w) = \frac{1}{n} \sum_{i=1}^{n} -log(p(y_i|x_i, w)) + \frac{\lambda}{2} \|w\|^2 \tag{3.2}$$

où $\lambda \geq 0$ est un paramètre de régularisation. Malheureusement, évaluer $log(p(y_i|x_i, w))$ est coûteux en raison de l'impossibilité de considérer toutes les configurations possibles y' dans la somme. Par exemple, dans les modèles structurés en chaîne, l'algorithme progressif-rétrogressif (*forward-backward algorithm*) est utilisé pour calculer $log(p(y_i|x_i), w)$. Un deuxième problème avec la résolution de l'équation (3.2) est que le nombre d'exemples d'entraînement n dans les applications est en croissance constante, et nous aimerions donc utiliser des méthodes qui ne nécessitent que quelques passages dans le jeu de données.

Optimisation des champs aléatoires conditionnels

Lafferty et al. [70] ont proposé un algorithme itératif de mise à l'échelle pour résoudre le problème (3.2), mais cela a donné de moins bons résultats que les stratégies déterministes génériques d'optimisation comme L-BFGS [71]. Le problème de ces méthodes est que nous devons évaluer $log(p(y_i|x_i), w)$, et son gradient pour tous les exemples d'entraînement n à chaque itération. C'est très coûteux pour des problèmes où n est très grand, donc pour traiter ce problème, des méthodes de gradients stochastiques ont été examinées [72, 73]. Cependant, les méthodes traditionnelles de gradient stochastique exigent des itérations $O(1/\epsilon)$ plutôt que des itérations plus courtes de $O(log(1/\epsilon))$ requises par les méthodes déterministes.

Plusieurs tentatives ont été faites pour améliorer le coût des méthodes déterministes ou le taux de convergence des méthodes stochastiques. Par exemple, l'algorithme EG (*Exponentiated Gradient*)

de Collins et al. [74] traite les données en ligne et ne nécessite que des itérations $O(log(1/\epsilon))$ pour atteindre une précision de ϵ pour la valeur objectif duale. Cependant, cela ne garantit pas de bonnes performances en termes d'objectif primal. Bien que cette méthode soit efficace si λ est très grand, la performance de la version en ligne du EG (*Online EG*) se dégrade considérablement si une petite valeur de λ est utilisée (ce qui peut être nécessaire pour obtenir la meilleure erreur de test) [74, 75].

En revanche, SAG (*Stochastic Average Gradient*) se dégrade moins lentement lorsque λ devient petit, atteignant même un taux de convergence plus rapide que les méthodes du gradient stochastique classiques lorsque $\lambda = 0$ [76]. Lavergne et al. [77] envisagent d'utiliser plusieurs processeurs et le calcul vectorisé (*vectorized computation*) pour réduire le coût d'itération élevé des méthodes quasi-Newton, mais lorsque n est énorme, ces méthodes ont toujours un coût élevé par itération. Friedlander et Schmidt [78] explorent une méthode déterministe-stochastique hybride qui augmente lentement le nombre d'exemples qui sont considérés afin d'atteindre un taux de convergence $O(log(1/\epsilon))$ avec un coût moindre comparé aux méthodes déterministes.

Le-Roux et al. [79] introduisent l'algorithme SAG, une méthode simple avec un faible coût par itération des méthodes de gradients stochastiques, mais qui nécessite seulement des itérations $O(log(1/\epsilon))$. Pour mettre en avant l'aspect unique de cet algorithme, nous écrivons d'abord l'itération classique de descente de gradient comme suit :

$$w^{t+1} = w^t - \frac{\alpha}{n} \sum_{i=1}^{n} s_i^t \tag{3.3}$$

où α est la taille du pas et à chaque itération nous réglons les variables de pente (*slope variable*) s_i^t au gradient par rapport à l'ensemble d'entraînement x_i à w^t, pour que $s_i^t = -\nabla log(p(y_i|x_i, w_t)) + \lambda w^t$. L'algorithme SAG utilise cette même itération, mais au lieu de mettre à jour s_i^t pour tous les n points de données à chaque itération, il définit simplement $s_i^t = -\nabla log(p(y_i|x_i, w^t)) + \lambda w^t$ pour un point de données choisi aléatoirement et conserve les s_i^t restants à leur valeur de l'itération précédente. Ainsi, l'algorithme SAG est une version "randomisée" de l'algorithme de gradient où nous utilisons le gradient de chaque exemple de la dernière itération où il a été sélectionné. L'aspect surprenant du travail de Le-Roux et al. [79] est que cet algorithme de gradient retardé simple atteint un taux de convergence similaire à l'algorithme classique de gradient malgré que les itérations soient n fois plus rapides. L'intuition peut être que les gradients stockés en mémoire fournissent une meilleure approximation que la méthode du gradient stochastique, qui n'évalue qu'un seul gradient par itération.

L'optimisation de ce modèle est notoirement difficile. Schmidt et al. [80] décrit une implémentation pratique de SAG [79] pour les CRF et propose un plan d'échantillonnage non uniforme qui améliore les performances. Cet algorithme (SAG-NUS) est l'une des méthodes les plus performantes

pour l'optimisation CRF, et nous nous référons à Schmidt et al. [80] pour une étude détaillée des méthodes concurrentes.

Les méthodes déterministes (par batch) telles que L-BFGS [71, 81] ont un taux de convergence linéaire, mais le coût par itération est élevé. D'autre part, la méthode OEG (*Online Exponentiated Gradient*) [74] et SAG font partie d'une famille d'algorithmes avec des mises à jour stochastiques peu coûteuses et des taux de convergence linéaires, et ils ont tous deux été appliqués à l'entraînement des CRF. On les appelle algorithmes à variance réduite, car leur point commun est d'utiliser la mémoire pour réduire la variance de la direction de mise à jour stochastique quand ils s'approchent de l'optimum.

L'algorithme SDCA (*Stochastic Dual Coordinate Ascent*) proposé par Shalev-Shwartz et Zhang [82, 83], et appliqué aux CRF par Le-Priol [84] appartient à cette famille. Il est étroitement lié à l'algorithme OEG dans la mesure où il effectue également une ascension sur les blocs de coordonnées sur la fonction objectif duale. Un avantage intéressant de SDCA par rapport à l'OEG (et à SAG) est que la forme de sa mise à jour permet d'effectuer une recherche linéaire "précise" avec un seul appel à l'oracle de marginalisation, c'est-à-dire le calcul des probabilités marginales du CRF. Cela contraste avec le SAG et l'OEG, où chaque changement de taille de pas nécessite un nouvel appel à l'oracle de marginalisation.

Algorithme SVM structuré

De nombreux problèmes de vision par ordinateur ont une formulation naturelle en tant que tâches de prédiction structurées : étant donné une image, le but est de prédire un objet de sortie structuré, par exemple, un masque de segmentation ou une pose humaine. Les machines à vecteurs de support structurel (SSVM) [85, 86] sont actuellement l'une des méthodes les plus populaires pour entraîner des modèles capables d'exécuter cette tâche à partir de données d'entraînement. Contrairement aux machines à vecteurs de support (SVM) standards [87], qui ne prédisent que des valeurs uniques, par exemple, une étiquette de classe, les SSVM sont conçus de telle sorte qu'ils peuvent prévoir des objets structurés arbitraires. Cependant, cette flexibilité a un coût : l'entraînement d'un SSVM nécessite de résoudre un problème d'optimisation plus difficile que l'entraînement d'un SVM ordinaire. En particulier, l'entraînement d'un SSVM nécessite des exécutions répétées de l'étape d'inférence structurée (le max-oracle) sur l'ensemble d'entraînement. Chacune de ces étapes est un problème d'optimisation en soi, par exemple, trouver l'étiquetage à énergie minimale d'un graphe est souvent coûteux sur le plan calculatoire. En fait, plus le problème est difficile, plus les appels à l'oracle deviennent coûteux. C'est aussi un facteur important qui explique pourquoi les SSVM ne sont généralement utilisés que pour des problèmes avec des jeux de données de petite et moyenne taille, et non pour l'apprentissage à grande échelle [88].

Le but de la prédiction structurée est de prédire les objets structurés, $y \in \mathcal{Y}$, pour des entrées données, $x \in \mathcal{X}$. Les SSVM [85, 86] offrent des méthodes d'apprentissage d'une fonction de prédiction structurée, $h : \mathcal{X} \to \mathcal{Y}$, à partir de données d'entraînement dans le cadre de la marge maximale. Nous paramétrisons $h(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \Phi(x, y) \rangle$, où $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ est une fonction de caractéristiques conjointe des entrées et sorties, et $\langle ., . \rangle$ désigne le produit scalaire en $\mathbb{R}^d$. Le vecteur de poids w est appris à partir d'un ensemble d'entraînement $\{(x_1, y_1), ..., (x_n, y_n)\}$ en résolvant le problème d'optimisation convexe suivant :

$$\min_{w} \quad \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{n} H_i(w) \tag{3.4}$$

où $\lambda \geq 0$ est un paramètre de régularisation. $H_i(w)$ est la fonction de perte SVM multi-classe structurée (*structured hinge-loss*), mise à l'échelle, définie comme suit :

$$H_i(w) = \frac{1}{n} \max_{y \in \mathcal{Y}} \Delta(y_i, y) - \langle w, \Phi(x_i, y_i) - \Phi(x_i, y) \rangle \tag{3.5}$$

où $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ est une fonction de perte spécifique à la tâche, par exemple la perte de Hamming pour les tâches de segmentation d'image. Calculer la valeur de $H_i(w)$, ou l'étiquette qui réalise cette valeur, nécessite de résoudre un problème d'optimisation sur l'ensemble des étiquettes. Pour ce faire, nous appelons la procédure $oracle - de - max$, ou simplement $oracle$. D'autres noms pour ceci dans la littérature sont LAI (*Loss-Augmented Inference*), ou l'étape $argmax$. Cela dépend du problème à résoudre et de la manière dont $l'oracle - de - max$ est implémentée. Les SSVM se sont avérés utiles pour de nombreuses tâches complexes de vision par ordinateur, y compris l'estimation de la pose humaine [89, 90], la détection d'objets [91], la segmentation sémantique des images [92, 93, 94], la reconstruction de scène [95, 96] et le repérage [97, 98].

Algorithmes d'optimisation

De nombreux algorithmes ont été proposés pour résoudre le problème d'optimisation (3.4) ou des formulations équivalentes. Dans Taskar et al. [85] et Tsochantaridis et al. [86], où le problème a été initialement introduit, les auteurs dérivent un problème d'optimisation quadratique équivalent à (3.4), mais semblable au problème de l'optimisation SVM avec des variables d'écart et une grande quantité de contraintes linéaires. Ce problème d'optimisation peut être résolu par une méthode des plans sécants qui alterne entre l'appel de l'oracle-de-max une fois pour chaque exemple d'apprentissage et la résolution d'un problème quadratique avec un sous-ensemble de contraintes (plans sécants) obtenues de l'oracle. L'algorithme a atteint une solution ϵ proche de la solution optimale dans l'étape $O(\frac{1}{\epsilon^2})$, c'est-à-dire $O(\frac{n}{\epsilon^2})$ appels vers l'oracle-de-max.

Joachims et al. [99] ont amélioré cette limite en introduisant la formulation one-slack. La méthode proposée est également basée sur la recherche de plans sécants, mais garde leur nombre beaucoup plus petit, obtenant un taux de convergence amélioré de $O(\frac{n}{\epsilon})$. Le même taux de convergence peut également être atteint en utilisant des méthodes de faisceaux [100].

On peut aussi appliquer la méthode des sous-gradients directement à l'objectif (3.4), qui permet également un apprentissage stochastique en ligne [101]. L'un des inconvénients est que la vitesse de convergence dépend essentiellement du choix d'un taux d'apprentissage approprié, ce qui rend souvent les méthodes utilisant les sous-gradients pour entraîner les SSVM moins attrayantes pour les tâches pratiques.

L'algorithme Frank-Wolfe (FW) [102] est une alternative élégante : il ressemble aux méthodes de sous-gradient par la simplicité de ses mises à jour mais ne nécessite pas une sélection manuelle d'un taux d'apprentissage. Lacoste-Julien et al. [75] ont introduit une variante de l'algorithme de Frank-Wolfe appelée BCFW (*Block-Coordinate Frank-Wolfe*) [75]. L'algorithme proposé est plus efficace que l'algorithme FW standard, en exploitant le fait que l'objectif du SSVM peut être décomposé en n termes, tous structurés. Les expériences de Lacoste-Julien et al. [75] montrent une accélération significative du BCFW par rapport à l'algorithme FW standard ainsi que les techniques proposées précédemment.

Presque simultanément, Branson et al. [103] proposent une méthode "SVM-IS". La méthode proposée utilise une procédure spécifique au cas d'application appelée "Échantillonnage préférentiel" qui retourne K solutions au lieu d'une seule. Pour $K = 1$, la méthode est similaire à la méthode BCFW dans Lacoste-Julien et al. [75], à l'exception de différences mineures dans la formulation du problème duale.

Outre les techniques ci-dessus qui utilisent l'oracle-de-max comme une boîte noire, plusieurs autres approches pour accélérer l'entraînement des SSVM ont été développées, par exemple, en utilisant la décomposition duale [104], en limitant l'espace de recherche aux sous-classes traitables [105], en résolvant des problèmes d'optimisation relaxés ou lissés [106, 107] ou en exploitant la similarité entre solutions [108]. De telles techniques peuvent également produire une accélération significative, mais leur applicabilité dépend généralement de la forme concrète de l'oracle-de-max.

3.3.4 Combiner apprentissage profond et prédiction structurée

Nous passons d'abord en revue les travaux qui combinent les réseaux de neurones à des modèles structurés à énergie. L'idée de combiner les réseaux et les modèles à énergie était bien connue dans les années 1990. Par exemple, Bottou [109] et LeCun et al. [110] ont introduit les transformateurs graphiques (une combinaison de modèles de Markov cachés et de réseaux de neurones) entrainés

de bout en bout et déployés en tant que systèmes de reconnaissance de caractères manuscrits.

Certaines études ont suggéré la combinaison de modèles d'apprentissage profond et de modèles graphiques pour saisir les dépendances statistiques entre les variables de sortie. Ces modèles profonds structurés devraient permettre d'obtenir des modèles plus efficaces tant pour l'entraînement que pour l'inférence. Tompson et al. [111], Jaderberg et al. [112], Vu et al. [113] et Chen et al. [114] ont utilisé l'approche séquentielle pour les tâches de vision par ordinateur pour l'estimation de la pose humaine, la reconnaissance de texte libre, la détection d'objets multiples et le marquage d'images. [115] et Chen et al. [116] ont utilisé l'approche de formation conjointe pour les modèles BiLSTM-CRF combinant des techniques de réseaux de neurones, à savoir le plongement de mots ou plongement lexical (*word embeddings*) et réseaux récurrents avec des unités LSTM, avec le modèle LCCRF au-dessus pour diverses tâches de traitement automatique du langage naturel.

Huang et al. [117] ont proposé d'utiliser des modèles structurés profonds pour la recherche sur le Web afin de mapper les requêtes de recherche aux documents pertinents à un niveau sémantique. Ils ont évalué leur approche sur des données de recherche réelles et ont montré qu'elles surpassent l'appariement traditionnel par mots-clés et les autres modèles sémantiques latents.

CHAPITRE 4 ORGANISATION DE LA THÈSE

Comme il a été mentionné dans la revue de la littérature, le CPP est généralement modélisé comme un SPP, à résoudre avec la méthode CG. Ce problème devient difficile à résoudre lorsque le nombre de vols augmente car le nombre de rotations possibles augmente de façon exponentielle (nombre de variables). Ainsi, on utilise DCA pour réduire le nombre de contraintes dans le problème maître restreint. Toutefois, DCA a besoin d'une partition initiale de vols en grappes (*clusters*) qui peut être donnée par une heuristique ou en résolvant une approximation du problème.

La contribution principale de cette thèse est donc de proposer différentes méthodes d'apprentissage machine pour proposer une partition initiale pour DCA ainsi qu'une solution initiale pour le processus d'optimisation du CPP. Cette thèse comporte trois chapitres principaux, chacun présentant les résultats obtenus pour un objectif de recherche.

Le chapitre 5 présente l'article *Flight-Connection Prediction for Airline Crew Scheduling to Construct Initial Clusters for OR Optimizer* où nous proposons d'abord une reformulation du CPP hebdomadaire comme problème de prédiction de connexion de vol. Ensuite, pour produire des grappes initiales pour DCA, on utilise des techniques d'apprentissage machine pour déterminer quelles variables de connexion de vols sont susceptibles d'être égales à un. Nous comparons la performance des réseaux de neurones à celle des méthodes classiques et utilisons l'apprentissage profond pour génerer la partition initiale à l'aide d'une heuristique de construction qui évite d'avoir des rotations où l'équipage finit loin de la base.

La première contribution de ce chapitre est la présentation du problème de connexion de vol comme un problème de prédiction multi-classes où on peut mesurer la performance de différents algorithmes d'apprentissage machine. La seconde est l'adaptation des réseaux de neurones pour améliorer la performance du modèle en encodant les entrées, en réduisant le nombre de candidats (classes) et en ajoutant l'option "abstention". En effet, au lieu de tenir compte de toutes les prédictions, il est préférable de rejeter un pourcentage de ces prédictions pour améliorer la précision. La troisième contribution de ce chapitre est l'intégration du modèle de prédiction dans l'optimiseur sous forme d'une partition initiale pour DCA. Avec la nouvelle approche, on n'a plus besoin de passer un temps important pour trouver une partition initiale et on divise le temps de calcul par 10, tout en faisant une économie pouvant atteindre 0.2%.

L'article *Machine Learning in Airline Crew Pairing to Construct Initial Clusters for Dynamic Constraint Aggregation* est présenté dans le chapitre 6, où l'objectif est d'intégrer les méthodes d'apprentissage machine dans la résolution d'un CPP mensuel comprenant 50 000 vols. Dans le processus de développement d'un solveur capable de traiter des problèmes plus importants, GEN-

COL, un solveur utilisant la méthode CG a été développé avec une approche à horizon fuyant. Comme le solveur ne peut traiter que quelques milliers de vols par fenêtre, les fenêtres sont limitées à deux jours. La longueur des fenêtres est trop courte pour produire de bonnes solutions aux problèmes mensuels, car les contraintes couvrent de longues périodes de temps. Pour permettre de résoudre des CPP de grande taille, DCA diminue le nombre de contraintes de partitionnement définies dans le problème maître restreint. On propose donc de combiner plusieurs méthodes mises en œuvre, développées et testées sur de petits ensembles de données, afin d'obtenir un algorithme efficace pour les CPP de grande taille. Ensuite, on utilise les méthodes d'apprentissage machine pour proposer des grappes initiales pour DCA.

On propose donc un nouvel algorithme, Commercial-GENCOL-DCA, combinant plusieurs techniques avancées de recherche opérationnelle pour assembler et modifier ces grappes, au besoin, pour produire une bonne solution. Cette nouvelle approche permettra de résoudre des problèmes globalement plus importants et d'éviter la perte d'optimalité résultant de la décomposition heuristique en petites tranches de temps dans l'approche à horizon fuyant. Encore plus, puisqu'on utilise l'approche à horizon fuyant pour la résolution avec des fenêtres d'une semaine et une période de chevauchement de deux jours, la partition initiale présentera de nombreuses incohérences avec la solution de la fenêtre précédente trouvée par l'optimiseur, en particulier pendant la période de chevauchement. Ainsi, pour proposer des grappes pour la fenêtre courante, on propose une méthode d'adaptation qui construit ces grappes en évitant toute incohérence avec la solution de la fenêtre précédante trouvée par l'optimiseur.

La nouvelle approche donne de meilleurs résultats que l'approche standard en réduisant le coût de la solution par 8.52% et le coût des contraintes globales par 78.11%. Cette amélioration s'accompagne d'une augmentation du temps de calcul, en raison des incohérences entre la solution initiale donnée au processus d'optimisation et les grappes de DCA construites par le modèle d'apprentissage machine et les heuristiques de construction des rotations.

Le chapitre 7 quant à lui est intitulé *Réseaux à Noyaux Convolutifs Structurés pour la Reconnaissance Optique de Caractères et la Planification des Horaires d'Équipages*. Comme la méthode CG a besoin d'une solution initiale réalisable pour assurer la faisabilité du problème primal à chaque itération, on cherche un prédicteur capable d'incorporer les contraintes locales dans le processus d'entraînement dans le but de produire non seulement une partition initiale pour DCA, mais aussi une solution initiale quasi-réalisable. Cette approche a pour but d'éviter les incohérences entre la solution initiale et la grappes de DCA. Comme première contribution, nous proposons un nouveau prédicteur structuré appelé réseaux à noyaux convolutifs structurés, qui combine les propriétés des architectures d'apprentissage profond, la flexibilité non paramétrique des méthodes du noyau et les prédicteurs structurés. Nous montrons que l'utilisation de cette combinaison de manière supervisée

surpasse les méthodes de pointe en termes de sous-optimalité primale et de précision du test sur le jeu de données OCR. Comme deuxième contribution, nous appliquons la méthode proposée au jeu de données de connexion de vols ou FCD (*Flight-Connection Dataset*) pour proposer de bonnes solutions initiales à un solveur de planification des horaires d'équipage des compagnies aériennes. Nous montrons que cette approche permet d'accélérer le processus d'optimisation du CPP, tout en offrant de meilleures solutions.

CHAPITRE 5 ARTICLE 1 : FLIGHT-CONNECTION PREDICTION FOR AIRLINE CREW SCHEDULING TO CONSTRUCT INITIAL CLUSTERS FOR OR OPTIMIZER

Y. Yaakoubi, S. Lacoste-Julien, F. Soumis (2019), *CPAIOR - International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, soumis le 11 Novembre 2019.

Abstract

We present a case study of using machine learning classification algorithms to initialize a large scale commercial operations research solver (GENCOL) in the context of the airline crew pairing problem, where small savings of as little as 1% translate to increasing annual revenue by dozens of millions of dollars in a large airline. We focus on the problem of predicting the next connecting flight of a crew, framed as a multiclass classification problem trained from historical data, and design an adapted neural network approach that achieves high accuracy (99.7% overall or 82.5% on harder instances). We demonstrate the usefulness of our approach by using simple heuristics to combine the flight-connection predictions to form initial crew-pairing clusters that can be fed in the GENCOL solver, yielding a 10x speed improvement and up to 0.2% cost saving.

Keywords : Flight Connection Prediction, Crew Pairing, Crew Scheduling, Neural Networks, Column Generation, Constraint Aggregation.

5.1 Introduction

Airlines need to construct crew pairings to cover their flights. A pairing is a sequence of flights starting and finishing at a base and satisfying complex collective agreement constraints. Figure 5.1 presents a pairing that contains two duties (days of work), separated by a rest period. For major airlines which handle more than 10k flights on a weekly basis, this becomes an important and difficult problem to solve. Efficient solutions are required since savings as low as 1% represent many dozens of millions saved every year. The complexity of the problem lies in the large number of possible pairings, and the selection of the set of pairings of minimal cost, which is a large integer programming problem impossible to solve with standard solvers [7, 18].

Because the crew pairing problem is similar to other scheduling problems, exploring more efficient methods to solve such problems becomes even more justified. Indeed, in contrast to problem-specific heuristics, a solution method for the above problem can be generalized to other problems

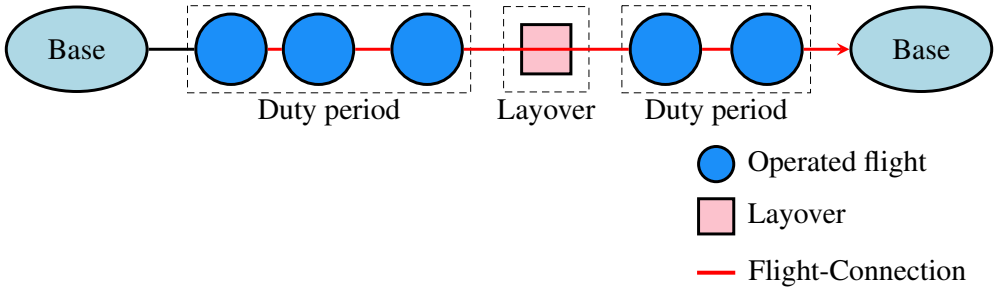


Figure 5.1 Illustration of a crew pairing. Here the flight-connection variable (in red) is defined to determine the next flight that a crew is going to perform, given an incoming flight.

in many airlines: covering pairings with monthly personalized schedules or covering flights with aircraft routes. Furthermore, similar covering problems appear in different transportation systems such as bus, trucking, and rail industries.

In our review of related work, we address some advanced optimization techniques that reduce the number of variables and the number of constraints to solve it. The main drawback of these techniques, however, is that they require days to compute, while airlines are often given all the scheduling data only a few days before having to build the schedule. The objective of this paper is to use machine learning (ML) techniques to improve the algorithmic efficiency and solve this problem in a more feasible time horizon. Unfortunately, ML alone can not do it: Solving a 10 000 flights problem needs to find 10 000 crew connections. Even if ML was 99.9% accurate for each connection, the probability of finding a good feasible solution is $(.999)^{10000} \approx 10^{-5}$. Furthermore, learning from previous months' solutions, the structure of the 10k best **flight-connection variables** [1] is a complex object to identify. For each flight, there is a large number of possible followers and very complex costs and constraints on the sequence of flights in a pairing. Finding these variables with high accuracy for the solution would require more data than what is available.

Instead of using only the operations research (OR) optimizer to solve the crew pairing problem, which takes tens of hours, we propose to use ML to obtain good initial information in a few seconds to start the optimizer, thus reducing the solution time by an order of magnitude. The information provided by ML does not need to be an entirely feasible solution [7]. Consequently, it can contain some weak points, which the OR optimizer can fine tune into a feasible solution.

It is possible to construct simple heuristics to build an initial solution for the optimizer such as rule-based heuristics (defining rules on maximum number of working hours, connection time, etc.). Nevertheless, because solutions differ and evolve over time, these heuristics require costly fine-tuning, rendering them obsolete over time. Because the past instances contain information

1. The following flight that a crew is going to perform after each flight.

-although not always updated-, our approach takes into consideration these instances and updates dynamically the weights of the neural network to propose an initial real-time solution without the need for manually fine-tuning the hyperparameters.

Contributions. We investigate the performance of machine learning algorithms in Section 5.5 to solve the **flight-connection** prediction problem in which the goal is to predict the next flight that an airline crew should follow in their schedule with only partial information (framed as multiclass classification, cf. Section 5.3). We adapt a neural network solution in several ways to improve the accuracy of the predictions: proposing improved feature encodings of the input, reducing the number of classes to predict using domain-appropriate constraints, and testing solutions that can abstain to make a prediction (cf. Section 5.4).

We motivate the above problem in the context of solving the airline crew pairing problem. The idea is to exploit a large volume of flight data, that is, the existing solution counting thousands of flights over several months, and apply supervised learning algorithms to build a predictor for the flight-connection problem, in order to obtain good initial information to speed up an OR optimizer (called GENCOL) that needs to be run on new schedule data. Using our algorithm, we can process tens of thousands of flights, in contrast to current techniques reported in the literature. To the best of our knowledge, finding good initial information with ML algorithms in order to optimize an airline crew schedule of this magnitude has not been addressed before. As a proof of concept, we propose in Section 5.6 simple heuristics to form initial crew-pairing clusters that can be fed in the GENCOL solver, yielding a 10x speed improvement and up to 0.2% cost saving which would translate in millions of dollars saved for a large airline.

5.2 Related Work

5.2.1 Machine Learning for Scheduling

Priore et al. [118] provides an overview of the dynamic scheduling for manufacturing systems using machine learning techniques. The classical problem involving job scheduling is to allocate the time, for each job, for each specific machine, for satisfying a set of predefined constraints. The machine learning approach uses a set of previous system simulations (as training samples) in order to determine which rule is optimal for the current system state. The described approaches include a variety of techniques such as inductive learning, case-based reasoning, support vector machines, reinforcement learning, and other approaches. Unlike the dynamic scheduling for manufacturing systems, we do not have access to these predefined constraints, as each airline company has its own collective agreement and we would like to build a model that can be extended to *other* airline

companies.

Other works have used neural networks (NNs) to solve scheduling problems. Wang et al. [119] applies standard NN to predict the anticipated bus traffic, whereas Doherty et al. [120] describes an application of NN to activity scheduling time horizon. These approaches are not used in combination with an OR optimizer, though, and the flight-connection problem contains a larger number of classes as well as more complicated constraints to satisfy.

5.2.2 Previous Work on Crew Pairing

Those problems are known to be quite hard to solve. The complexity of such a problem is due to the number of variables and constraints involved in the problem definition. In short, they are large-scale integer programming problems. In the literature, there are two common methods frequently employed: (i) branch-and-price (B&P) [12, 13, 14]; and (ii) Lagrangian relaxation [15, 16, 17].

The most prevalent method since the 1990s has been the set covering problem with column generation inserted in branch-&-bound (see Desrochers et al. [4]). This method, along with others, is described in a survey on crew pairing problems by Cohn et al. [5]; see also Deveci et al. [6] for a more recent survey. According to this latest survey, column generation was the most frequently used approach.

Column generation uses a master problem and a sub-problem. On the one hand, the master problem shall be referred to as the restricted master problem; here, restricted indicates that we are using a subset of all possible pairings. The objective function seeks to minimize the total cost for the current columns. This problem is solved with the Simplex algorithm. On the other hand, the sub-problem generates promising pairings that are more likely to improve the solution of the master problem. In the sub-problem, the constraints of flow continuity and local constraint ensure that flights in the pairing will be chained in time and space. Furthermore, rules and collective agreements have to be met as well; examples include minimum connection time between two consecutive flights, minimum rest time and maximum number of duties in a pairing. This problem is solved by dynamic programming.

To reduce the number of constraints considered simultaneously, the work reported by Elhallaoui et al. [7] introduced a *dynamic constraint aggregation* technique that decreases the amount of the set partitioning constraints in the restricted master problem and reduces the solution time. This is achieved by combining a few of them as per a correspondence relationship. Dynamic constraint aggregation begins with an initial aggregation partition and aggregates in a single covering constraint each cluster of flights expected to be consecutive in an optimal solution. The algorithm modifies the clusters dynamically to reach an optimal solution if some expectations were wrong. In previous

work, the initial clusters were constructed with heuristic rules based on the user's knowledge.

In this work, we use ML techniques to learn from previous solutions in order to construct clusters of flights. To produce such initial clusters, we ought to determine which flight-connection variables are likely to be equal to one. These clusters are not a feasible solution, but the dynamic constraint aggregation method will repair it with phase 1 and phase 2 of the simplex to reach a good feasible solution.

5.3 Problem Setting

5.3.1 Crew Pairing Motivation

In our solution method for the overall crew pairing problem (cf. Section 5.6), we want to provide the OR optimizer with an initialization which uses a partition of the flights into clusters. Each cluster represents a sequence of flights with a strong probability to be consecutive in the same pairing in the solution. To construct each cluster, we will need the following:

— Information on where and when a crew begins a pairing. This makes it possible to identify whether a flight is the beginning of a pairing;
— For each incoming flight to a connecting city, predict what the crew is going to do next: layover, flight, or end of a pairing. If it is the second case (flight), then we further predict which flight the crew will undertake.

Since the end of a pairing depends on the maximum number of days in a pairing permitted by the airline company, we solely rely on this number as a hyperparameter. In other words, there is no need to predict the end of the pairing. Therefore, with regards to the pairing construction, we propose to decompose the second task into two sub-tasks. The first is predicting whether the crew makes a layover; the second is predicting the next flight, under the assumption that the crew always takes another flight.

On the one hand, layovers are highly correlated with the duration of the connection, although there is no fixed threshold for the number of hours a crew must stay in an airport to make a layover. On the other hand, predicting the next flight is a much more complicated prediction problem. Indeed, for each incoming flight, there may be hundreds of flights departing in the next hour and thousands of flight codes. We call this the **flight-connection problem** which is the focus of our ML approach described in the next sections.

5.3.2 Flight-connection Prediction Dataset

We obtained several months of historical crew pairing data from an anonymous airline company covering approximately 200 cities and tens of thousands of flights per month. We will transform this data to build a flight-connection prediction subproblem (a multiclass classification problem) where the goal is to predict the next flight that an airline crew should follow in their schedule given only partial information about the beginning of their schedule. To avoid error propagation through the whole sequence of flights (and because the OR solver is able to use partially correct plans), we will only use information about the crew's *incoming flight* (we will not use earlier information in the pairing) to predict the next flight.

The classification problem is thus the following: given the information about an incoming flight in a specific connecting city, choose among all the possible departing flights from this city (which can be restricted to the next 48 hours) the one that the crew should follow. These departing flights will be identified by their flight code (about K=2 000 possible flight codes in our dataset, K will represent the number of classes). Different flights may share the same flight codes in some airline companies, as flights performed multiple times a week usually use the same flight code. Nevertheless, a flight is uniquely identified by its flight code, departing city and day; information which in practice can be deduced from the information about the incoming flight and our 48 hour-window.

Each flight will give the following 5 features that we can use in our classification algorithm:

— City of origin and of destination ($\sim$200 categories);
— Aircraft type (5 types);
— Duration of flight (in minutes)
— Time (with date) of arrival (for an incoming flight) or of departure (for a departing flight).

Our transformed dataset contains about 300k flight prediction examples (organized in different months), giving for each example these features for both the incoming flight as well as the possible departing flights from the same city. This dataset is used for training and validation. We keep a separate 10k flights for testing which are derived from a different weekly crew pairing schedule. [2]

5.4 Algorithms

In this section, we describe the different methods and models that we investigated to solve the flight-connection problem.

2. The code and the dataset are available at: `https://www.gerad.ca/fr/papers/G-2019-26`

5.4.1 Basic Neural Network (NN) Model

As neural networks have shown great potential to extract meaningful features from complex inputs and obtain good accuracies through different classification tasks, we propose to use them and compare their results with classical machine learning methods. In both cases, the input contains the following information on the previous flight as well as the next departing flight:

— Information on the aircraft: type of the aircraft (categorical)
— Information on the flight: duration of flight (numeric, in minutes) and time of arrival to the arrival city (Date, Day, Hour, Minutes)
— Information on the city: codes of the departure and the arrival city (both categorical)

We use a standard neural network with two dense hidden layers, each with 1 000 neurons and `relu` as the activation function. The output layer is another dense layer with K=2 000 neurons and `softmax` as the activation function. We use K neurons in the output layer because we want to predict the flight code. We also use dropout [121] with a probability of 0.2 to prevent over-fitting as well as batch normalization [122] between the activation function and the dropout

5.4.2 Adding an Embedding Layer

The city code and aircraft type features mentioned in Section 5.3.2 are categorical features which are simply treated as numeric values in our basic NN model. This means that cities with nearby codes are treated similarly by the NN even though the codes are somewhat arbitrary. A more meaningful encoding for the different categorical features is to use a one-hot encoding. By fully connecting each one-hot encoding of a categorical feature to a separate hidden layer, we basically get an embedding layer of dimension d for this feature (d is a hyperparameter). The $C \times d$ parameter matrix (where C is the number of possible categorical values) represents the d-dimensional encoding for each of the C values, and this encoding is learned during the NN training. The embedding layer approach thus learns a d-dimensional representation of each city, and one could explore which city is similar to another one in this space. By concatenating the embedding layer for each categorical feature with the other numeric features (such as hours and minutes), we get an n_d-vector, where n_d is the dimensionality of the representation of one flight that is fed into the NN. For the results given in Table 5.3, we feed the NN with the concatenation of the n_d encodings of the incoming flight as well as the first next flight (using more next flights did not improve results but just took longer to learn).

5.4.3 Evaluation Metrics

We now present the evaluation metrics that we will consider; the feasibility one will motivate further model refinements described next.

— Categorical accuracy:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{\arg\max_j(y_{i,j}) \in \arg\max_j(p_{i,j})\} \tag{5.1}$$

where i is the sample index, j refers to the class index, and $y_{i,\cdot}$ denotes the sample label encoded as a one-hot vector. $p_{i,j}$ is the classifier score for class j for input i.

— top-k categorical accuracy:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{\arg\max_j(y_{i,j}) \in \text{top-}k_j(p_{i,j})\} \tag{5.2}$$

— Different aircraft accuracy: Throughout the months, in 88% to 95% of the instances, the crew arrives at an airport and follows the aircraft, i.e. takes the next departing flight that the same aircraft makes (note that the classifier cannot know which next flight uses the same aircraft though as we removed this information). To consider more difficult flight-connection instances, we therefore report the accuracy only on the instances where the crew changes aircraft.

— Feasibility: Given that the aircrew arrived at a specific airport at a given time, we can use *a priori* knowledge to define which flights are possible. For example, as shown in Figure 5.2, it is neither possible to make a flight that starts ten minutes after the arrival, nor it is possible five days later. Furthermore, it is rare that the type of aircraft changes between flights since each aircrew is formed to use one or two types of aircraft at most. The reader is referred to Kasirzadeh et al. [18] for further details on the likelihood of these scenarios. In this paper, we use the following conditions that are always satisfied for the next flight taken by the crew:

  — The departure time of the next flight should follow the arrival time of the previous flight to the connecting city;

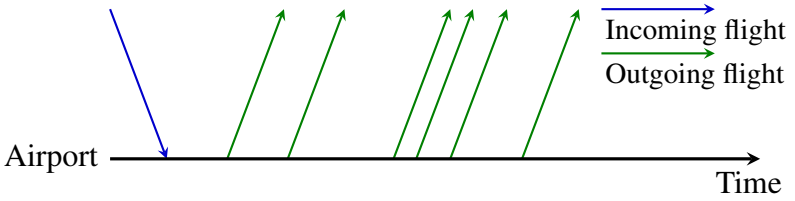
  — The departure time of the next flight should not exceed 48 hours following the arrival time of the previous flight to the connecting city;

  — The departure city of the next flight should be identical to the connecting city in the previous flight;

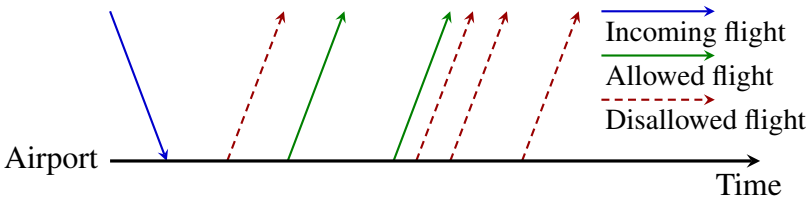
  — The aircraft type should be the same. Indeed, crew scheduling is separable by crew category and aircraft type or family [18].

We use these conditions to define a mask $m_{i,j} = 1$ only when flight j satisfies these conditions for input i, and 0 otherwise. For the basic classifiers, we can evaluate the proportion of time they predict a feasible next flight with the "feasibility" metric:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{\arg\max_j(p_{i,j}) \in \arg\max_j(m_{i,j})\} \tag{5.3}$$



(a) Different possibilities for an incoming flight



(b) Different **feasible** possibilities for an incoming flight

Figure 5.2 Illustration of an incoming flight scenario.

5.4.4 Masked Output Neural Network

To reduce the number of possible outputs, we can use the feasibility function $m_{i,j}$ to restrict our classifier to only predict a class among the feasible next flights for a specific incoming flight i. We can use such a mask to define a new output layer L_m, such that in the output layer, for the softmax function, we only take into account the probabilities for the feasible next flights.

5.4.5 Transformed Classification Problem

By using the feasibility information about next flights, we can design a more useful encoding for the output classes by normalizing their representation across different instances.

Therefore, for each incoming flight, we consider all departing flights in the next 48 hours as a set. Then, we consider only those that are feasible according to our masking constraints (defined in Section 5.4.3) to filter this set. Then, we sort these flights based on the time of departure and limit the maximal number of possible flights to 20, as it is sufficient in the airline industry. Thus, we predict the rank of the true label among that set with a 20-dimensional softmax output layer.

We use the embedding layer described earlier to construct a feature representation for each of these 20 flights. Then, we concatenate them to have a matrix $n_d \times 20$ input for the next layers, where n_d is the embedded representation of information on one flight. Consequently, we do not only reduce the number of possible next flights but also construct a similarity-based input, where the neighboring factors have similar features, which in turn allows the usage of convolutional neural networks. The intuition here is that we can consider each next flight as a different time step, enabling the use of convolutional architecture across time [123, 124].

5.4.6 Post-processing Step: Abstention Methods

Instead of taking all the predictions into account, it is preferable in our case to discard a percentage of these predictions to enhance accuracy. We consider augmenting our classifier with an "abstain" option [125]. Since the OR solver takes more time to break a connection in a cluster (links) than to build one, removing low confidence links (flights) using this "abstain" option can be advantageous. We observed that our solver is able to carry out the optimization problem much faster when taking into account 99% of our predictions instead of 100%. We describe below three abstention methods that we investigated in our experiments (Section 5.5.3).

Abstain when low confidence

The simplest abstention technique would be to ignore the samples whose winning class confidence falls below a certain threshold. Indeed, Hendrycks and Gimpel [126] show that the lower the softmax probability of the most probable class, the greater the likelihood that the prediction is incorrect. Therefore, we use a threshold for the probabilities given by the NN. We also consider an RBF kernel SVM, to which we give as input the probabilities given by the NN, as well as the rectangular input of the NN, where the categorical features are one-hot encoded. The binary label is 1 when the NN predicted correctly, 0 otherwise. Once trained, we put a threshold on the probabilities given by the SVM, exactly as we did for the probabilities given by the NN.

Abstain when high entropy

One issue with the approach of Hendrycks and Gimpel [126] is that modern neural networks are often notoriously miscalibrated [127]. Methods such as temperature scaling [127] can be applied to correct this miscalibration. Unfortunately, even when we consider using calibration for selective classification, we still encounter problems in the presence of class imbalance (which is the case in the flight-connection dataset) [128]. The idea here, in short, is that we choose not to predict if the predictive entropy of the output probability vector p (given by $H(p) = -\sum_i p_i \log p_i$) is too

high [126]. Entropy is an uncertainty measure that takes into consideration the confidence scores of all the classes, in contrast to the confidence abstention method described in the previous paragraph which only takes into account the winning class confidence.

Estimate confidence with dropout

The NN literature considers various techniques to estimate the uncertainty of a prediction. In one technique, the prediction tasks can be carried N times while applying dropout in the entire layers of the neural networks [129], yielding N probability vectors for each test sample. A rough estimate of certainty of prediction is obtained by computing the mean of these N probability vectors and subtracting their component-wise estimated standard deviation (computed from the same N vectors). This gives a rough lower bound on the certainty of our prediction. If the maximum value of these confidences is too low, we decide to abstain.

5.5 Experiments

In this section, we illustrate the numerical examples employing the dataset described in subsection 5.3.2. Furthermore, this section describes the specifications of the employed hardware and software to realize the algorithm, in addition to the implementation details, and the main results of the experiments.

5.5.1 Hardware and Software

The experiments were executed on a 40-core machine with 384 GB of memory. Each method is executed in an asynchronously parallel set up of 2-4 GPUs. That is, it can evaluate multiple models in parallel, with each model on a single GPU. When the evaluation of one model is completed, the methods can incorporate the result and immediately re-deploy the next job without waiting for the others to be finalized. We use four K80 (12 GB) GPUs with a time allocation of 10 hours. All algorithms were implemented in Python using Keras [130] and Tensorflow [131] libraries. For classical machine learning methods, we use Scikit-learn [132].

5.5.2 Hyperparameter Selection

Random search

As a first phase exploration of the hyperparameters space, we use random search [133] with k-fold cross-validation on the training set to choose the best hyperparameters for each method we consider.

We use different months for different folds (6 folds) to simulate the more realistic scenario where we make a prediction on a new time period. We keep the weekly problem of 10k flights for testing.

Bayesian optimization

After having identified the best predictor with random search, we refine further the choice of hyperparameters by using Bayesian optimization with k-fold cross-validation to measure the configuration quality. Bayesian optimization builds a probabilistic model of the function mapping from hyperparameter settings to the model performance and provide a systematic way to explore the space more efficiently [134].

We optimize the hyperparameters listed in Table 5.1 [135] with an implementation of Gaussian process-based Bayesian optimization provided by the GPyOpt Python library version 1.2.1 [136]. In our approach, the optimization was initialized with 50 random search iterations, followed by up to 450 iterations of standard Gaussian process optimization. Here, the total return is used as the surrogate function and Expected Improvement ($\mathcal{EI}$) as the acquisition function.

Table 5.1 Hyperparameters used in optimization

Parameters	**Search space**	**Type**
Optimizer	Adadelta; Adam; Adagrad; Rmsprop	Categorical
Learning rate	0.001, 0.002, . . . , 0.01	Float
Dimensions of the embeddings	5, 10, 15, . . . , 50	Integer
Number of dense layers	1, 2, 3, 4, 5	Integer
Neurons per layer	100, 200, . . . , 1000	Integer
Dropout rate	0.1, 0.2, . . . , 0.9	Float
Convolutional layers	0, 1, 2, 3	Integer
Filters n	50, 100, 250, 500, 1000	Integer
Filter size h	3,4,5	Integer

5.5.3 Results

Classical machine learning methods

Using features described in Section 5.4, we considered several classical machine learning methods for our comparison, such as a logistic regression model and Multinomial Naive Bayes methods. As demonstrated in Table 5.2, we have non-linear decision boundaries. The data points are not easily separated by a single hyperplane, this is due to the fact, that the train classification accuracy is greater than test classification accuracy, in which we are slightly over-fitting the data.

Table 5.2 Performance of classical machine learning algorithms

Method	Train accuracy (%)	Test accuracy (%)	Different aircraft accuracy (%)	Time (s)
Decision trees	99.99	97.44	61.84	70
One-Vs-Rest Decision trees	99.99	92.66	66.53	320
Logistic regression	11.89	11.71	1.90	1990
Multinomial Naive Bayes	6.61	6.32	0.62	3

Neural networks

After testing classical machine learning methods, we show results for NN with and without embedding, Masked output NN with embeddings as described in section 5.4.4, Transformed input with and without convolutional layers and Transformed input with convolutional layers after performing Gaussian process as described in section 5.5.3. We report in Table 5.3 for each predictor the test accuracy, the "Different aircraft accuracy", the top-3 accuracy and the feasibility metric as described in section 5.4.3 as well as the total computational time. As speculated, we show that the computational effort (CPU training time) of executing the model without the embedding layer is much higher, and provides a lower accuracy rate. When using the mask to guide the gradient descent during training jointly with the usage of the embedding layer, our model takes less than one-third of the training time and gives a better result put in comparison with the standalone embedding layer.

Table 5.3 Comparison of results for different methods

Method	Test accuracy (%)	Different aircraft accuracy (%)	Top-3 accuracy (%)	Feasibility (%)	Time (s)
NN without embeddings	88.03	60.52	98.95	92.25	4000
NN with embeddings	99.11	70.85	99.47	98.68	1850
Masked output NN with embeddings	99.20	71.24	100	100	500
Transf. input without convol. layers	99.18	70.32	100	100	2300
Transf. input with convol. layers	99.35	71.79	100	100	700
Transf. input with convol. layers (after GP)	99.68	82.53	100	100	600

Gaussian process

We perform the Gaussian process on "Transformed input with convolutional layers" to search for the best configuration of hyperparameters. After a few iterations, we are able to get an accuracy of 99.35%. Then, random search boosts the total return very quickly up to 99.62% after 18 iterations and thus remains until the end of the random search cycle (iteration number 50). Using Gaussian

process, we can show that we constantly improve our process of searching for the best architecture that maximizes the overall return. In our case, we stopped at iteration number 500 with the best architecture providing an accuracy of 99.68%. One should notice that there was no limitation that prevents our algorithmic procedure from exploring, even more, the configuration space; in other words, the algorithm was stopped manually. The final training was done with the best-identified architecture found by Bayesian optimization.

Abstention

Figure 5.3 shows the accuracy-abstention tradeoff curves for the abstention methods proposed in Section 5.4.6 on the best architecture on the test set giving a 99.62% accuracy initially; using these methods, we can discard some of our predictions and enhance the accuracy. For example, if we discard only 1% of our predictions, the accuracy increases from 99.62% to 99.90% abstaining when low confidence and to 99.94% estimating confidence with dropout. For the next Section 5.6, we will use a 1% rejection rate.

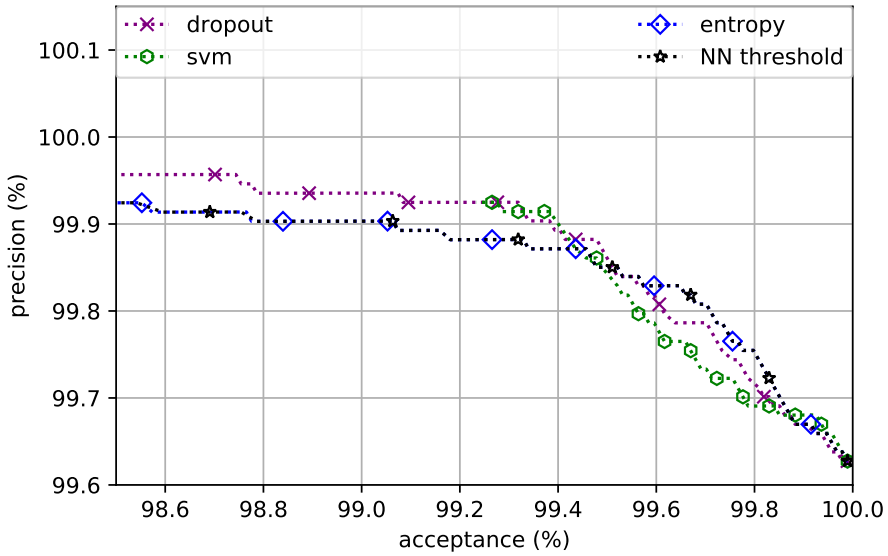


Figure 5.3 Results of the different abstention methods. NN threshold and Entropy use probabilities given by the NN. SVM takes into consideration a confidence score given by a RBF-SVM. Dropout uses the lower confidence bound of the prediction confidence. For more information on these methods, see Section 5.4.6

5.6 Integration of the Flight-Connection Predictor Into Crew Pairing

GENCOL [3] (GENeration de COLonnes) is a commercial toolbox that produces real-world airline crew scheduling solutions. The toolbox uses column generation as an optimization strategy that allows a large number of variables to be considered in solving large-scale problems, with up to 3 000 flights. We access this solver through private communication with the company. As described in section 5.2.2, to solve larger problems, we use DCA (Dynamic Constraint Aggregation) to reduce the number of constraints by combining a few of them. This solver (GENCOL-DCA) starts with an initial aggregation partition, in clusters, of flights having a good probability of being done consecutively by the same crew, in an optimal solution. As more than 20 airlines companies use GENCOL for their crew scheduling, we explore in this section whether our machine learning approach can yield gains on this state-of-the-art solver (through private communication with the company).

Upon the finalization of the flights-connection prediction model, we can use the same architecture to solve two other prediction problems on the test set (10k flights): (i) predict if each of the scheduled flights is the beginning of a pairing or not; and (ii) predict whether each flight is performed after a layover or not. In reality, the three predictors share the same representation. To solve these independent classification problems, we sum the three prediction problems' cross-entropy losses when learning, therefore performing a multi-output classification.

We use a greedy heuristic to build the crew pairing. Specifically, we consider each flight that the model predicts at the beginning of a pairing as a first flight. Given this incoming flight, we predict whether the crew is making a layover or not. In both cases, we consider the incoming flight and predict the next one. The pairing ends when the crew has arrived at the base and when the maximal number of days permitted per pairing is approached.

We can use the above heuristic to construct a weekly solution for the testing data, obtaining a crew pairing that can be fed as an initial partition for the GENCOL-DCA solver. Unfortunately, if one flight in the pairing is poorly predicted, as the flights are predefined, the crew can finish its pairing away from the base. Therefore, we consider different heuristics to correct the pairings:

- **Heuristic 1 (H1):** includes pairings where the crews finish away from the base, but it erases what occurs after the last time the crew arrives at the base, or deletes the whole pairing if it never passes to the base again.
- **Heuristic 2 (H2):** deletes all pairings where the crews finish away from the base;
- **Heuristic 3 (H3):** like heuristic 1, erases what occurs after the last time the crew arrives at the base but also cuts the pairings into smaller ones if we pass multiple times at the base.

As a baseline, we consider a benchmark called "GENCOL init" which is a standard initial solution

3. `http://www.ad-opt.com/optimization/why-optimization/column-generation/`

approach obtained with the GENCOL solver. In this approach, the weekly problem is solved by a "rolling time horizon" approach with one-day windows. This means that the week is divided into overlapping time slices of equal length (slightly more than one day). Then a solution is constructed greedily in chronological order by solving the problem *restricted* to each time slice sequentially, taking into account the solution of the previous slice through additional constraints.

We can feed the "GENCOL init" solution to the GENCOL-DCA solver as an initial partition for the constraint aggregation approach to further improve the solution globally, obtaining a solution that we call "GENCOL-DCA from GENCOL init". We can also use ML-based heuristics H1, H2, and H3 to construct clusters that we provide to the GENCOL-DCA solver as an initial partition, obtaining solutions that we respectively call "GENCOL-DCA from H1", "GENCOL-DCA from H2" and "GENCOL-DCA from H3". We report in Table 5.4 the savings w.r.t the cost of the "GENCOL init" solution and the total computational time.

Table 5.4 Final crew pairing costs after running the GENCOL-DCA solver with different initialization clusters. We compare the solution cost to the cost of the "GENCOL init" solution.

Resolution method	**Cost diff. vs "GENCOL init" (%)**	**Total time**
GENCOL-DCA from GENCOL init	1.21	$\approx$ 45h (40h + 5h)
GENCOL-DCA from H1	**1,45**	6h30
GENCOL-DCA from H2	1.38	5h
GENCOL-DCA from H3	0.97	6h30

As reported in Table 5.4, two of the three heuristics that we propose outperform the benchmark solution "GENCOL init" as well as the "GENCOL-DCA from GENCOL init" solution, and we can conclude that for the pairings that finish away from the base, it is better to allow the solver to cover the flights than to propose smaller pairings. Therefore, instead of performing the optimization process for 40 hours to obtain the "GENCOL init" solution and then for another 5 hours to obtain the "GENCOL-DCA from GENCOL init" solution, our proposed method gives better costs after a few seconds to predict the flight connections, and optimized results after five to six hours.

Note that the heuristics that we propose to build crew pairings are still limited, even though they do give better costs in two of the three cases. Further study and experimentation are required to explore other heuristics to construct the crew pairings. Further study is also needed to solve *monthly* problems by rolling horizon with one-week windows. The fast machine learning predictor will permit to construct in a few seconds clusters for each window of a week, customized according to the flight schedule of this week. We expect improvements in particular when the monthly schedule

is irregular from week to week. It is the case nearly every month: Christmas, Easter, Thanksgiving, National Holiday, Mother and Father days, big sports events, etc. It was out of the question to use five times 40 hours to produce customized clusters for each window with the "GENCOL init" solution. Future studies will explore the generalization of the proposed approach to similar decision problems in different systems, such as bus, road, and rail traffic management.

5.7 Conclusion

It is rather difficult to assign the crew workers to a range of tasks while taking into account all the variables and constraints associated with the process. In this paper, we incorporate different algorithms and methods, which include neural networks in conjunction with abstention techniques, such as dropout, in order to prevent overfitting, or masks to ensure guidance of the gradient descent. These networks allow our solver to achieve high accuracy for the flight-connection problem.

This paper introduces a new paradigm for solving a large combinatorial problem: "Start with ML — Finish with OR optimizer". This paradigm, first, use ML to learn from solutions of similar instances to produce predictions on some parts of the solution of the new instance. This information feeds the OR optimizer to finish the work taking account of the exact cost function and the complex constraints. This approach reduces significantly the solution time without loosing on the quality of the solution. This new paradigm can be applied on many types of problems solved recursively.

CHAPITRE 6 ARTICLE 2 : MACHINE LEARNING IN AIRLINE CREW PAIRING TO CONSTRUCT INITIAL CLUSTERS FOR DYNAMIC CONSTRAINT AGGREGATION

Y. Yaakoubi, F. Soumis, S. Lacoste-Julien (2019), *EURO Journal on Transportation and Logistics, Special issue : Combining optimization and machine learning : applications in vehicle routing, network design and crew scheduling*, soumis le 10 Septembre 2019.

Abstract

The crew pairing problem is generally modelled as a set partitioning problem where the flights have to be partitioned in pairings. A pairing is a sequence of flight legs separated by connection-time and rest-periods that starts and ends at the same base. This problem becomes difficult to solve when the number of flights increases because the number of feasible pairings (number of variables) grows exponentially. This paper introduces a new paradigm for solving this large combinatorial problem: "Machine Learning $\rightarrow$ Mathematical Programming". This paradigm uses Machine Learning to learn from solutions of similar instances to produce predictions on some parts of the solution of the new instance. This information feeds the Mathematical Programming optimizer that connects these parts of the solution by modifying them as needed, taking into account of the exact cost function and the complex constraints. We show that the clusters produced by ML-based heuristics are better suited for crew pairing problems, resulting in an average reduction of solution cost between 6.8% and 8.52%, mainly due to the reduction of the cost of global constraints between 69.79% and 78.11%, when compared to pairings obtained with a standard initial solution.

Keywords : Machine Learning, Mathematical Programming, Airline Crew Scheduling, Crew Pairing.

6.1 Introduction

Crew scheduling is of crucial importance for airlines since the crew cost represents their second-highest source of spending, after fuel costs. The first step of crew scheduling is the construction of the pairings. A pairing is a sequence of flight legs separated by connections and rest-periods that starts and ends at the same crew base (an airport where crews are stationed). A pairing may also contain legs that some crew members take as passengers to be relocated, called deadheads. A pairing contains multiple duties — sequences of flights and deadheads that form a day of work. Two consecutive duties inside a pairing are separated by a layover. Pairings must comply with airline regulations as well as collective agreements. The cost of a pairing approximates the crew's

salary as well as other expenditures, such as hotel costs. Since pilots and copilots are trained to operate on a single type of aircraft, the crew pairing problem (CPP) can be decomposed by aircraft type. The CPP is to find a set of pairings at minimal cost so that a pairing covers each planned flight.

This problem is known to be quite hard to solve. The complexity is due to the number of variables and constraints involved in the problem definition. In short, it is a large-scale integer programming problem. The most prevalent method since the 1990s has been the branch-and-price: column generation embedded into a branch-and-bound structure [4]. This method is described with others in a survey on CPPs by Cohn and Barnhart [5]; see also Deveci and Demirel [46] for a more recent survey.

In the process of developing a solver capable of dealing with much larger problems than what had been dealt with so far, GENCOL, a commercial operations research solver was integrated into a rolling horizon approach. Because the original solver can handle only a few thousand flights per window, windows are constrained to be two-days long in the case of a large CPP problem. The length of the windows is too short to produce good solutions to monthly problems, as the constraints cover long periods of time.

To permit to solve large-scale pairing problems, a dynamic constraint aggregation approach was introduced, which improves upon column generation by reducing the number of constraints in the problem. Indeed, to reduce the number of constraints considered simultaneously, the work reported by Elhallaoui et al. [7] introduced the dynamic constraint aggregation (DCA) technique that decreases the number of the set partitioning constraints in the restricted master problem and the degeneracy. The GENCOL-DCA solver starts with an aggregation, in clusters, of flights having a good probability of being done consecutively by the same crew, in an optimal solution. The initial aggregation partition is produced with a heuristic solver producing pairings. It corresponds to fixing some flight connection variables to one temporarily, which permits to replace all the flight-covering constraints of the flights in a cluster by a single constraint. GENCOL-DCA uses reduced costs to identify flight connection variables that can be unfixed to improve the solution by breaking clusters. The drawback of this method is that it takes more time to produce the initial clusters than re-optimize with GENCOL-DCA.

This paper proposes to combine multiple methods implemented, developed and tested on small datasets, in order to obtain an efficient algorithm for large-scale CPPs. We use Machine Learning (ML) to propose a good initial partition for a large CPP: monthly problems with up to 50 000 flights. ML alone cannot solve CPP: Solving a 50 000 flights problem needs to find 50 000 crew connections. Even if ML was 99.9% accurate for each connection, the probability of finding a good feasible solution is $(.999)^{50000} \approx 10^{-22}$. We use ML to produce clusters of flights having a high

probability of being done consecutively by the same crew, in an optimal solution. A new algorithm combining many advanced Operations Research techniques will be used to assemble and modify these clusters, when necessary, to produce a good solution. This new approach, starting with Machine Learning and finishing the optimization with Mathematical Programming will permit to solve globally larger problems and will avoid the loss of optimality resulting of heuristic decomposition in small time slices in the rolling horizon approach.

In this paper, we show that modern ML techniques can help to automatically learn highly efficient crew pairing schedules and develop initial clusters for DCA. We describe both the Machine Learning and the Operations Research methodology and detail a proof-of-concept study on a huge monthly problem with up to 50 000 flights.

The remainder of this article is structured as follows. A literature review on crew pairing is presented in Section 6.2. Section 6.3 presents an overview of multiple methods that will be combined to obtain an efficient algorithm for very large CPPs and describes the new algorithm. Next, Section 6.4 describes the Machine Learning predictor and introduces the cluster construction methodology, including different heuristics to avoid constructing infeasible pairings. Section 6.5 reports computational results. Finally, a short conclusion is drawn in Section 6.6.

6.2 Crew pairing

This section reviews a collection of relevant literature on crew pairing to provide an overview of how researchers suggest approaching this problem. We focus on approaches developed to solve large-scale industrial problems.

According to Desaulniers et al. [19], for each category of the crew and each type of aircraft fleet, the construction problem of crew pairing finds a set of pairings at minimal cost so that each planned flight is performed by a crew. The methodology for solving the problem depends on the size of the airline's network structure, rules, collective agreements, and cost structure [137].

In the literature, the CPP has been traditionally modelled as a set covering problem (SCP) or a set partitioning problem (SPP), with a covering constraint for each flight and a variable for each feasible pairing [19, 20, 18]. Formally, let F be a set of legs that must be operated during a given period and let Ω be the set of all feasible pairings that can be used to cover these legs. Let a_{fp}, $f \in F$, $p \in \Omega$, be a constant equal to 1 if pairing p contains leg f, and 0 otherwise, and let c_p be the cost of this pairing. For each $p \in \Omega$, define x_p as a binary variable that takes value 1 if pairing p is selected, and 0 otherwise. Using a set-partitioning formulation, the CPP can be modelled as

follows:

$$\underset{x}{\text{minimize}} \quad \sum_{p \in \Omega} c_p x_p \tag{6.1}$$

$$\text{subject to} \quad \sum_{p \in \Omega} a_{fp} x_p = 1 \qquad \forall f \in F \tag{6.2}$$

$$x_p \in \{0, 1\} \qquad \forall p \in \Omega \tag{6.3}$$

The objective function (6.1) minimizes the total pairing costs. Constraints (6.2) ensure that each leg is covered exactly once, and constraints (6.3) enforce binary requirements on the pairing variables.

Marsten et al. [23] present a first commercial system with a specialized Mixed Integer Linear Program (MILP) solver for the set partitioning formulation. They present results for various companies and propose a heuristic decomposition for larger problems. Anbil et al. [1] introduced a comprehensive method with a cost-minimization goal. In this research, presenting collaborative research between American Airlines Decision Technologies and IBM, numerous possible pairings (columns) were generated as an apriori, and a considerable amount is fed into the MILP solver. At every repetition, several of the non-basic pairings are rejected, and new pairings are being inserted. The procedure keeps its execution until all the pairings have been taken into consideration [20].

Overall, the conclusion that can be drawn from literature is that heuristic algorithms have three main shortcomings. Firstly, they do not consider all scheduled flights simultaneously and must perform many iterations before they can achieve a solution with a high-quality [26]. Secondly, the algorithms do not consider all viable pairings [27]. Thirdly, the divergence from the optimal solution is significant, resulting in new solutions which can be far from the globally optimal solution [28]. To overcome the above limitations, more sophisticated approaches have been proposed over the years, which are presented in Section 6.3.

The most prevalent method since the 1990s to solve this large SCP is the column generation inserted in branch-&-bound [19, 3]. This method is described with others in a recent survey [46] concluding that column generation is the most frequently used approach. The column generation method iterates between pairing generation (sub-problem (SP)) and pairing selection (master problem (MP)). For the SP, many authors have used shortest-path in a network with resource constraints, and the MP is a SPP [30].

In practice, three time-horizons are generally studied: a daily, a weekly, and a monthly horizon [14]. The daily problem assumes that the flights are identical or relatively quite similar, for every day of the planning horizon. The problem also assumes that minimum cost pairings are generated for flights scheduled for a day. A cyclic solution is produced, where the number of crews present in every city in the evening is the same as in the morning. Accordingly, the weekly and monthly

solutions can be produced by juxtaposing the daily solution. When the schedules of flights are not exactly the same every day, the pairings that do not fit with the flight rules are removed, as they no longer apply and can be re-optimized to create new pairings covering flights that are not covered previously.

For large fleets, globally solving the weekly problem or globally re-optimizing the monthly problem can be too long. The rolling horizon approach is used to speed up the solution process [22]. The horizon is divided into time slices of equal length (except maybe the last one), each one overlapping partially with the previous one. Then a solution is constructed greedily in chronological order by solving the problem restricted to each time slice sequentially, taking into account the solution of the previous slice, and the next one if it is a re-optimization, through additional constraints. When the size of the problem increases, the time slices need to be shorter to obtain problems that can be resolved within a reasonable time. When the time slices become shorter than the pairings, the quality of the solution is deeply impacted. The pairings partially outside of the time slice cannot be moved to another base, and many deadheads are used to balance the workload between bases. The weekly problem assumes that flights are identical (or somewhat similar) each week, and the pairing problem is solved for scheduled flights for a typical week. The monthly problem globally deals with a full month. Recent studies have concentrated on weekly and monthly problems. Owing to the vacation period and differences in flight schedules, the monthly time horizon is the most accurate [14]. To solve a monthly CPP, a one-week window with two-days overlapping is usually used.

In recent years, personnel scheduling has become more advanced, using different multidimensional approaches and techniques. A frequently used method is constraint programming, where we seek a good feasible solution that satisfies a specific set of constraints [138]. The most common objective when using constraint programming is to minimize the weighted quantity of late jobs. Several scheduling algorithms are used to optimize the set of schedules which occur in an airport or airline company, from flight scheduling to personnel and equipment scheduling, and researchers often tend to use heuristics instead of exact solution techniques [139]. Nevertheless, solutions provided by Machine Learning methods are often infeasible and need to be refined further by a Mathematical Programming optimizer, taking into account the exact cost function and the complex constraints.

6.3 Solution methods

We first describe the most complex methods that will be combined to obtain an efficient algorithm for very large CPPs. We then present the structure of the new algorithm, describing the other procedures that compose it and the relations between the procedures.

6.3.1 Column Generation method

Column Generation (CG) is an iterative method that solves, for each iteration, a restricted master problem and one or numerous sub-problems [140]. Column generation is well-recognized for solving *relaxation* of a MILP in a range of applications, particularly in the fields of crew scheduling and vehicle-routing. For this type of applications, several problems can be framed as set-partitioning-type problems [141]. The reader is referred to Desrosiers et al. [29] and Desrosiers et al. [30], the founding papers of the domain.

When the number of columns is large, to the extent that they can no longer be considered at the initial stage, column generation decomposition can be employed to solve a restricted master problem, that is, a linear program limited to a subgroup of columns (variables), resulting in a plausible and viable primal solution, along with dual variables. The dual vector is used by an oracle algorithm. This algorithm resolves one or several sub-problems, which in turn produces columns with negative reduced costs and expands the restricted master problem. The procedure is executed until no more variables with negative reduced costs can be recognized.

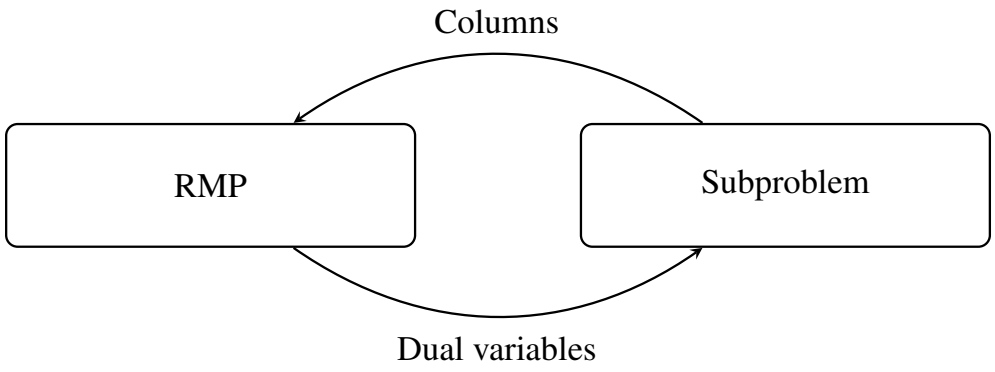


Figure 6.1 Standard column generation decomposition [141]

Like the simplex method, the column generation method is a convergent and exact method [142]. The speed of convergence depends on the number of iterations necessary to get the proof of convergence. The time by iteration depends on the time to solve the restricted master problem and the sub-problem. It turns out that in practice, a compromise must be found between the speed of convergence and the complexity of the generator algorithm. The generator algorithm can solve a restricted subproblem, or use approximate methods to reduce the computational time [143].

Nevertheless, one should note that the convergence of the iterative process requires an exact algorithm in the last iterations, in order to prove that there is no negative reduced cost column involved. A commonly used method for accelerating column generation is the simultaneous generation of several improving columns at every iteration [144]. This method makes it possible to reduce the

number of iterations necessary to generate the columns of the optimal basis.

6.3.2 Dynamic Constraint Aggregation method

When the number of flights in a CPP increases, the resolution time by column generation becomes large. The larger problems appear in a short-haul network where the crews make many flights per day. In this case, there are many flights per pairing, and the SPP has more dense columns that yield high degeneracy of the simplex algorithm. In the late 1990s, Pan [145] developed some acceleration techniques for SPPs. Elhallaoui et al. [7] improves the Pan's work and adapts it for column generation, which resulted in the DCA algorithm. The DCA algorithm speeds up the master problem by reducing the number of constraints and degeneracy. The DCA method starts with an aggregation, in clusters, of flights having a high probability of being done consecutively by the same crew, in an optimal solution.

Let Q be the set of clusters. The aggregation corresponds to temporarily fix to one the flight connection variables between the flights in a cluster, thus permitting to replace all the flight-covering constraints of the flights in a cluster by a single constraint. This constraint elimination removes degenerate constraints. On the other hand, the variables are partitioned in compatible and incompatible variables. A variable is said to be compatible with the set of clusters Q if the set of flights covered by the corresponding schedule is the union of some clusters in Q. Otherwise, this variable is declared incompatible. Observe that the constraints in a cluster are identical for the compatible variables and in particular the non-degenerate basic variables, justifying keeping only one of them. The Aggregated Reduced Master Problem (ARMP) with only $|Q|$ constraints and only the compatible variables is solved efficiently with the primal simplex. This smaller non-degenerate problem permits to improve the solution rapidly.

To reach optimality, the incompatible variables with negative reduced cost, need to be considered. The ARMP provides dual variables only for the aggregated set-partitioning constraints. To compute the dual variables for all set-partitioning constraints of the original model, a procedure based on shortest-path calculations is invoked by DCA to identify negative reduced cost variables. To add a group of incompatible variables to the ARMP, the clusters need to be adapted. Some clusters are broken to make the added variables compatible while others are merged when they are connected by a variable equal to one. This dynamic management of clusters permits to reach an optimal solution with a smaller and less degenerate master problem.

6.3.3 Improved Primal Simplex

Elhallaoui et al. [146] and Omer al. [147] proposed Improved Primal Simplex (IPS), a generalization of DCA, applicable to any linear programming problem. Both IPS and DCA consider a problem with a reduced number of constraints and are restricted to compatible variables. While DCA groups in clusters the constraints that are identical on the non-degenerate variables and keeps only one copy of the constraints in each cluster, IPS keeps a maximum subset of independent constraints on the non-degenerate variables. Even though many dependent constraints are not just copies in linear programming, the vast majority of dependent constraints are copies for SPPs. In IPS, the compatible variables are the variables in the span of the non-degenerate variables. The compatible variables of DCA are a special case of this definition.

In addition to being applicable to any linear programming problem, the main innovation of IPS is the introduction of the Complementary Problem (CP). The CP is formulated with the following notations. Let Q be the set of constraints in the ARMP, $\bar{Q}$ the remaining constraints, C the set of compatible variables and I the set of incompatible variables. The sub-matrix of A with columns indexed by J is denoted $A_{\cdot J}$ and the sub-matrix of A with rows indexed by I and columns indexed by J is denoted A_{IJ}, then the CP is formulated as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \bar{c}_I^T x_I \\ \text{subject to} \quad & \bar{A}_{\bar{Q}I} x_I = 0 \\ & w_I^T x_I = 1 \\ & x_I \geq 0 \end{aligned} \tag{CP}$$

This CP finds a normalized combination of incompatible variables with minimum reduced cost. The first constraint requests that the solution x_I^* is in the subspace of the compatible variables where $\bar{A} = B^{-1}A$ is the matrix of the simplex tableau. This ensures that entering the surrogate variable $\bar{A}_{\cdot I} x_I^*$ in the basis of the ARMP permits a non-degenerate pivot and improves the solution of $\bar{c}_I^T x_I^*$, which is negative. The sequence of pivots on the non-zero variables x_I^* produces the same improvement [147].

The second constraint is a normalization constraint to obtain a bounded solution instead of an extreme ray. The selection of a good normalization is discussed in [147]. This problem is very primal degenerate, but can be solved efficiently with the dual simplex. The dual is not degenerate because the reduced costs in the objective are real numbers. The probability of having two subsets of variables with the same cost is very low.

This CP produces variables to enter in the ARMP to improve the solution and dual variables for all

set-partitioning constraints of the model. There is a huge number of dual solutions for the primal solution of the ARMP. The optimal dual solution of the CP is an interior point in the complementary subspace associated with constraints in $\bar{Q}$. Unlike the extreme dual solution of the original problem, this more central solution does not zigzag from an extreme point to another of the dual polyhedron. The distance between dual solutions used at successive iterations of column generation is reduced by a factor of more than 2 and reduces the number of iterations by a factor of 4 [141].

6.3.4 The new algorithm for the Set Partitioning Problem

In this section, we present the structure of the new SPP solver Commercial-GENCOL-DCA, providing a short description of the procedures that compose it and the relations between the procedures. Figure 6.2 presents the structure of the new algorithm.

In box 1, the aggregation of flights in a set of Q clusters is done with Machine Learning and will be explained in Section 6.4. The mathematical programming part of the algorithm is the Branch-&-Bound using CG at each node of the branching tree (see Section 6.3.1)

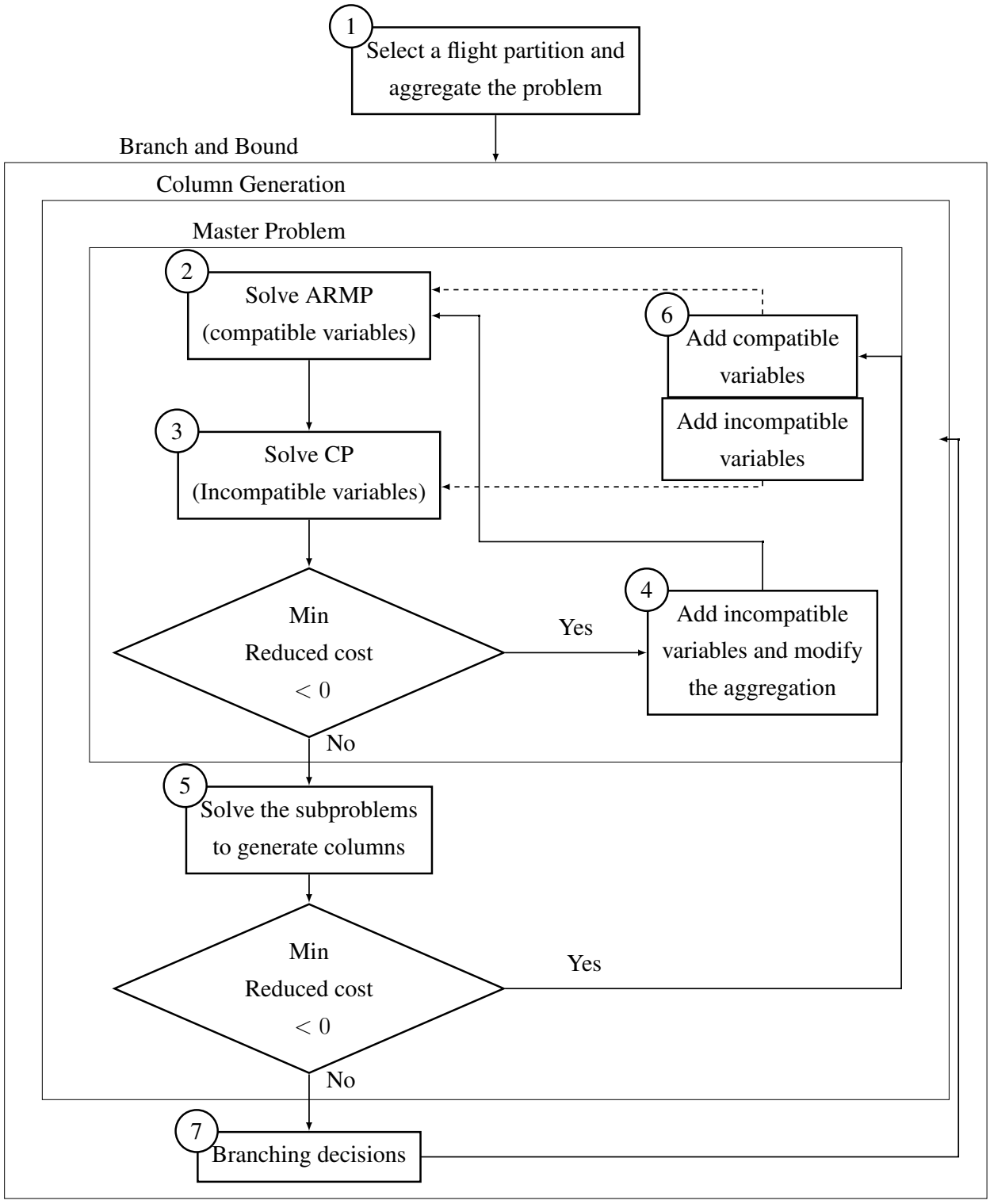


Figure 6.2 The new algorithm for the SPP type

In boxes 2 and 3, the compatible and incompatible variables are identified with the network techniques of DCA (see Sections 6.3.2 and 6.3.3). This applies easily in CPPs where the flights can be ordered by beginning-time in the clusters and in the path corresponding to the variables. It suffices to follow each path corresponding to a variable. The variable is incompatible if the path enters or exits in the middle of a cluster. It is easy to update C and I when Q is modified by splitting clusters. Pointers from flights to columns covering these flights permit to rapidly re-classify previous incompatible columns covering flights in these clusters by verifying if they enter in the middle of the divided clusters. It is less time consuming than computing B^{-1} and $\bar{A}$ to verify if $\bar{A}\bar{Q}_{.j} = 0$ or not.

In box 4, when the solution of CP has negative reduced cost, the non-null columns of this solution are added to the ARMP. Some clusters are broken to make compatible the added variables. The incompatible variables becoming compatible are also added to the ARMP.

In box 5, when the solution of CP has null reduced cost, the sub-problem needs to be solved to generate columns with negative reduced cost, if possible. To be more efficient, we go to the sub-problem when the reduced cost is above a negative threshold. Instead of adding existing incompatible columns with a small potential of solution improvement, we give priority to generating columns with a better potential for improvement. At each column generation iteration, this threshold is increased up to zero to reach the optimality criterion. There is a sub-problem per crew base and per starting day for pairings. Each sub-problem has a shortest path problem structure with resource constraints modelling the rules, ensuring that the paths are feasible pairings.

In box 6, the variables compatible with the actual partition are added in ARMP. The incompatible ones are added to the CP. The variables are easily classified using the method described for boxes 2 and 3. After adding new variables, ARMP is solved in priority without modifying the partition. ARMP has priority, as long as the reduced cost of its least-reduced cost variable is less than that of the least reduced cost of incompatible variables multiplied by a predetermined multiplier (less than 1).

The column generation combining DCA and IPS is accelerated with the multi-phase dynamic constraint aggregation (MPDCA). Elhallaoui et al. [8] present results on bus and airline crew scheduling with 2000 tasks where the time for the LP solution is reduced by a factor of 4.5 compared to DCA. In this algorithm, a partial pricing strategy, favouring the compatible columns with the aggregation is employed. In phase k, we add in the sub-problem a resource constraint forbidding to generate a pairing breaking clusters more than k times. This method reduces the search space and CPU time. Furthermore, it gives priority to less dense columns in $\bar{A}\bar{Q}_{.I}$ and produces less fractional solutions by combining less dense columns. We see in the computational results LP solutions with less than 1400 fractional variables for a problem with 10 000 constraints.

In box 7, we use a partial exploration of the branching tree. With the column generation solving sub-problem at integrality and having access to a huge number of columns, the LP relaxation provides a very good lower bound. Furthermore, MPDCA provides a primal solution with a small number of fractional variables. The lower bound and the primal solution gives useful information to explore the branching tree efficiently. We first use the column fixing algorithm as follows (for instance, see [22]): At each node of the search tree, several columns taking a fractional-value in the current ARMP solution are selected, and their values are set to one. These columns are selected in decreasing order of their values because columns with larger values, in general, increase less the value of the lower bound when they are set to one than columns with smaller values. To impose a pairing p, we remove from the ARMP and CP all columns containing the flights covered by p, and from the sub-problem networks all arcs representing those flights. These decisions rapidly reduce the size of the problems to solve. When there are no variables large enough, we use the arc fixing strategy. We fix to one the arcs with a large value. Unfortunately, such diving heuristic may sometimes make bad decisions and produce a poor integer solution. To reduce the weakness of the diving branching, we use the Retrospective Branching [148]. This algorithm detects and revises, without backtracking, poor decisions made previously in the search tree. These decisions are selected from a list of risky decisions that is maintained during the branching process. A risky decision is a column, or an arc fixed even if their value was smaller than a threshold. When the relative gap q_i between the values z_i and z_0 of the solutions computed at a node i in the search tree and at the root node (i.e., $q_i = \frac{z_i - z_0}{z_0}$) exceeds a relative estimate gap for a good integer solution, the algorithm ejects risky decisions from the current solution without backtracking. This ejection is performed by adding a constraint on the number of risky decisions in the solution.

Several strategies of partial pricing on the variables to enter in the ARMP and in the basis can be used to enter first the most promising variables. As such, a heuristic can be applied in the pricing steps as long as the last iteration uses a complete model. Due to this, partial pricing is appealing as it can apply different sets of compatible and incompatible variables to lower the density of the current pricing problem. For example, one can use partial cost, residual capacity, or rank-incompatibility. Partial cost bias uses a threshold value to discard incompatible variables temporarily based on their partial reduced cost. Residual capacity guarantees a minimum step size. The last type of bias is the rank incompatibility in which all the incompatible values are assigned a rank according to their extent of incompatibility. The pricing model first takes consideration of the low ranked incompatible values. The aim of this bias is not to stray too far from the existing definition of compatibility. The above idea was first developed by Elhallaoui et al. [8] and used in MPDCA.

6.4 Machine Learning model

This section describes the Machine Learning predictor constructed to provide the probabilities of flights being performed consecutively by the same crew. The prediction problem is first stated and formulated in Section 6.4.1. The proposed prediction model is disclosed in Section 6.4.2. Section 6.4.3 presents the input transformations needed to get an effective prediction model. Upon the finalization of the model training and in order to build pairings, we use different heuristics presented in Section 6.4.4. Finally, we present additional improvements to the crew pairing construction methodology in Section 6.4.5.

6.4.1 Prediction problem formulation

In our solution method for the overall CPP, we provide the optimizer with an initial partition of flights into clusters. Each cluster represents a sequence of flights with a high probability to be consecutive in the same pairing in the solution. To construct each cluster, we need the following:

— Information on where and when a crew begins a pairing. This information makes it possible to identify whether a flight is the beginning of a pairing;
— For each incoming flight to a connecting city, predict what the crew is going to do next: layover, flight, or end of a pairing. If it is the second case (flight), then we further predict which flight the crew will undertake.

Since the end of a pairing depends on the maximum number of days in a pairing permitted by the airline company, we will solely rely on this number as a hyperparameter. In other words, there is no need to predict the end of the pairing. Therefore, with regards to the pairing construction, we propose to decompose the second task into two sub-tasks. The first is predicting whether the crew will make a layover; the second is predicting the next flight, under the assumption that the crew will always take another flight.

On the one hand, layovers are highly correlated with the duration of the connection, although there is no fixed threshold for the number of hours a crew must stay in an airport to make a layover. On the other hand, predicting the next flight is a much more complicated prediction problem. We call this the **flight-connection problem**, which is the focus of our ML approach described in the next sections (see Figure 6.3).

We will transform this data to build a flight-connection prediction sub-problem (a supervised ML multiclass classification problem) where the goal is to predict the next flight that an airline crew should follow in their schedule given the previous flight.

The classification problem is thus the following: given the information about an incoming flight in

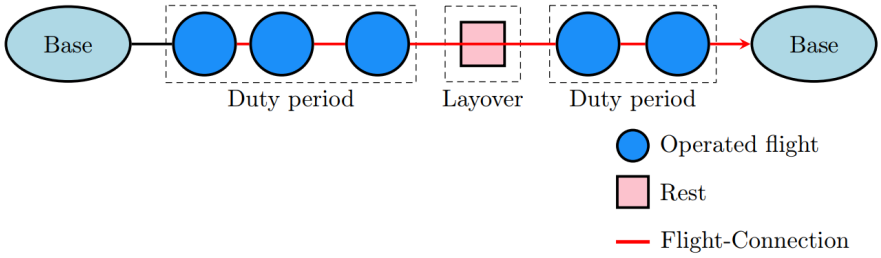


Figure 6.3 Illustration of a crew pairing. Here the flight-connection variable (in red) is defined to determine the next flight that a crew is going to perform, given an incoming flight

a specific connecting city, choose among all the possible departing flights from this city (which can be restricted to the next 48 hours) the one that the crew should follow. These departing flights will be identified by their flight code and departing city and day (about 2 000 possible flight codes in our dataset).

Each flight gives the following 5 features that we can use in our classification algorithm:

— City of origin and city of destination (∼200 choices);
— Aircraft type (5 types);
— Duration of flight (in minutes)
— Time (with date) of arrival (for an incoming flight) or of departure (for a departing flight).

6.4.2 Neural Network architecture

As Neural Networks (NN) have shown great potential to extract meaningful features from complex inputs and obtain good accuracies through different classification tasks, we use in our work the predictor described in Yaakoubi et al. [149]. In more detail, we use a multi-layer convolutional neural network. The output layer is a dense layer with `softmax` as the activation function. We also use dropout [121] to prevent overfitting as well as batch normalization [122] between the activation function and the dropout.

Using standard encoding, the city code and aircraft type features are treated as numeric values. This means that cities with close values are treated similarly by the NN even though the codes are somewhat arbitrary. A more meaningful encoding for such categorical features is to use one-hot encoding. By fully connecting each one-hot encoding of a categorical feature to a separate hidden layer, we get an embedding layer of dimension d for this feature (d is a hyperparameter). The $C \times d$ parameter matrix (where C is the number of possible categorical values) represents the d-dimensional encoding for each of the C values, and this encoding is learned during the NN training. The embedding layer approach thus learns a d-dimensional representation of each city, and one could explore which city is similar to another one in this space. By concatenating the

embedding layer for each categorical feature with the other numeric features (such as hours and minutes), we get an n_d-vector, where n_d is the dimensionality of the representation of one flight that is fed into the NN.

6.4.3 Transformed input

Throughout the months, in 74% to 76% of the connections, the crew arrives at an airport and follows the aircraft, i.e., takes the next departing flight that the same aircraft makes. To consider more difficult next-flight prediction instances, we, therefore, report "the accuracy only on the instances where the crew changes aircraft."

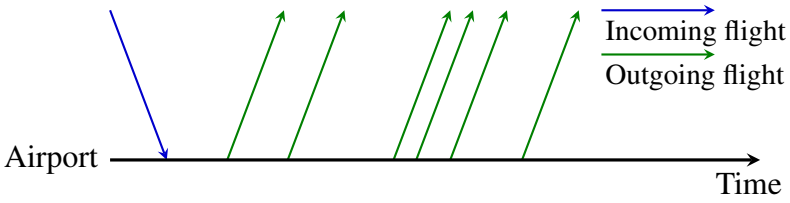
Given that the aircrew arrived at a specific airport at a given time, we can use *a priori* knowledge to define which flights are possible. For example, as shown in Figure 6.4, it is not possible to make a flight that starts ten minutes after the arrival, nor it is possible five days later. Furthermore, it is rare that the type of aircraft changes between flights since each aircrew is formed to use one or two types of aircraft at most. The reader is referred to [18] for further details on the likelihood of these scenarios.

In our work, we use the following conditions, that need to be always satisfied for the next flight performed by the crew:

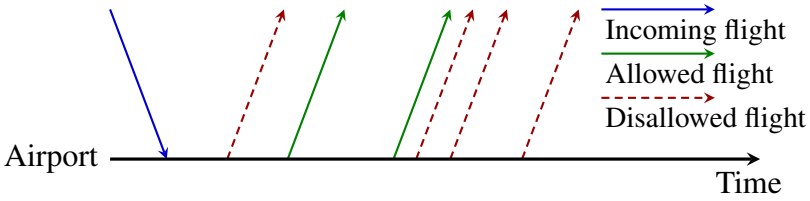
— The departure time of the next flight should follow the arrival time of the previous flight to the connecting city;
— The departure time of the next flight should not exceed 48 hours following the arrival time of the previous flight to the connecting city;
— The departure city of the next flight should be identical to the connecting city in the previous flight;
— The aircraft type should be the same. Indeed, crew scheduling is separable by crew category and aircraft type or family [18].

We can, therefore, design a more useful encoding for the output classes by normalizing their representation across different instances. For each incoming flight, we consider all departing flights in the next 48 hours as a set. Then, we consider only those that are feasible according to our masking constraints to filter this set. Then, we sort these flights based on the time of departure and limit the maximal number of possible flights to 20, as it is sufficient in the airline industry. Thus, we predict the rank of the true label among that set with a 20-dimensional softmax output layer.

We use the embedding layer described earlier to construct a feature representation for each of these 20 flights. Then, we concatenate them to have a matrix $n_d \times 20$ input for the next layers, where n_d is the embedded representation of information on one flight. Consequently, we not only reduce the number of possible next flights but also construct a similarity-based input, where the neighbouring



(a) Different possibilities for an incoming flight



(b) Different **feasible** possibilities for an incoming flight

Figure 6.4 Illustration of an incoming flight scenario

factors have similar features, which in turn allows the usage of convolutional neural networks. The intuition here is that we can consider each next flight as a different time step, enabling the use of convolutional architecture across time [123, 124].

6.4.4 Cluster construction

Upon the finalization of the flights-connection prediction model training, we can use the same architecture to solve two other prediction problems on the test set (50 000 flights): (i) predict if each of the scheduled flights is the beginning of a pairing or not; and (ii) predict whether each flight is performed after a layover or not. In reality, the three predictors share the same representation. To solve these independent classification problems, we sum the three prediction problems' cross-entropy losses when learning, therefore performing a multi-output classification. To build the crew pairing, one can use the following heuristics:

— Heuristic 1 (H1): We use a greedy heuristic to build a crew pairing. Specifically, we consider each flight that the model predicts at the beginning of a pairing as a first flight. Given this incoming flight, we predict whether the crew is making a layover or not. In both cases, we consider the incoming flight and predict the next one. The pairing ends when the maximal number of days permitted per pairing is approached.

We can use the above heuristic to construct a solution for the testing data, obtaining a crew pairing that can be fed as an initial cluster for the solver. Unfortunately, if one flight in the pairing is poorly predicted, as the flights are predefined, the crew can finish its pairing away from the base. To correct the pairings, we define a heuristic (H1) in which all pairings

where the crews finish away from the base are deleted.

— Heuristic 2 (H2): like H1, consider each flight that the model predicts at the beginning of a pairing as a first flight and for each incoming flight, consider the incoming flight and predict the next one, but deletes all pairings where the crews finish away from the base and all pairings where the predictor abstains.

Indeed, instead of taking all the predictions into account in constructing the initial clusters for DCA, one can also discard a percentage of these predictions to enhance accuracy. We consider augmenting our classifier with an "abstain" option [125]. Since MPDCA adds in phase k in the sub-problem a constraint forbidding to generate a pairing breaking clusters more than k times, breaking a connection in a cluster takes more time than building one. Therefore, removing low confidence links (flights) using this "abstain" option can be advantageous.

6.4.5 Further improvements

Start in a base. While training the NN predictor to recognize whether a flight is the beginning of a pairing or not, it is possible that it misclassifies that a flight departing from a non-base city is indeed the beginning of a pairing. It is imperative to correct such false prediction in order to avoid pairings starting away from the base. Even though it is possible to construct a predictor to which only flights departing from the bases are given, it is more efficient and robust to use all flights in the training step. That way, the predictor learns a better representation of the input.

No Layover below a threshold. While training the NN predictor to recognize whether there is a layover or not between two flights, and since this decision is independent of finding the next flight, we can use a threshold on the number of hours between the previous and the next flight, below which it does not make sense to make a layover. This threshold should be defined considering previous solutions or by a practitioner.

Adaptable DCA partition generation. Because the monthly problem is solved using a rolling-horizon approach with one-week windows and two days of overlap period, constructing an initial partition for the entire month and using the subset in each window to feed DCA can be a major flaw. Such initial partition will have many inconsistencies with the solution of the previous window, particularly during the overlap period, such as legs belonging to two different clusters. We propose to adapt the proposed clusters to the solution of the previous window using the heuristics proposed in Section 6.4.4 to construct the clusters of the current window in accordance with the solution found for the previous window and any inconsistency with the previous window is avoided, so the

proposed partition is adapted to the current resolution. In addition, instead of only considering the flights that the model predicts at the beginning of a pairing as a first flight, incomplete clusters from the solution of the previous window starting during the overlap period are completed. For the next section, we denote by Adapt-H1 and Adapt-H2 the heuristics proposing such adaptation to the partitions produced with H1 and H2.

6.5 Computational experiments

In this section, we report the results of the computational experiments we conducted using the proposed new algorithm for large-scale CPPs to compare the performance of the proposed heuristics described in Section 6.4.4 with that of using a cyclic weekly solution as initial clusters. First, we present the instances used for training and testing in Section 6.5.1, then the hyperparameters and hardware used in the experimentation in Section 6.5.2. Finally, Sections 6.5.3 and 6.5.4 report our ML prediction results and the crew-pairing problem resolution using Commercial-GENCOL-DCA.

6.5.1 Instances description

Our instances originate from a major airline and consist of six monthly crew pairing solutions for approximately 200 cities and approximately 50 000 flights per month. Each instance contains a one-month flight schedule, crew pairings, as well as a list of airports and bases. This information is used to create input for the learning phase. The dataset consists of approximately 1 000 different source-destination pairings, 2 000 different flight codes and pairings start from 7 different bases. Figure 6.5 shows the data format for a single pairing.

As shown in Figure 6.5, this entry describes a pairing that takes place every Tuesday, Thursday, Friday, and Saturday between August 9, 2018 and September 3, 2018, except for the dates: 08/10, 08/20, and 08/29. The pairing contains flights between Buffalo Niagara International Airport (BUF) and O'Hare International Airport (ORD). Apart from this, we know the distance between the two airports is 761 Km, from which we can extrapolate the duration of the flight, based on an average aircraft flight speed, which is equivalent to roughly 1 hour and 26 minutes.

```
O T 1821 0 CC 1 CAT 1 () "a" "" _2_456_
2018/08/09 2018/09/03 X 2018/08/10
2018/08/20 2018/08/29
{
RPT ;
ABC05914 ORD 1 02:10 BUF;
ABC06161 BUF 1 09:50 ORD;
RLS;
}
```

Figure 6.5 Data format for aircrew pairings

6.5.2 Parameters setting

The parameters setting is relevant to the hyperparameter optimization, evaluation process, and hardware description. To find the best configuration of hyperparameters, we use Bayesian optimization with k-fold cross-validation on the training set to measure the configuration quality. These are presented in Table 6.1. We use different months for different folds (6 folds) to simulate the more realistic scenario where we make a prediction over a new period of time.

More specifically, we optimize the hyperparameters listed in Table 6.1 [135] with an implementation of Gaussian process-based Bayesian optimization provided by the GPyOpt Python library version 1.2.1 [136]. Bayesian optimization constructs a probabilistic model of the function mapping from hyperparameter settings to the model performance and provides a systematic way to explore the space more efficiently [134]. In our approach, the optimization was initialized with 50 random search iterations, followed by up to 450 iterations of standard Gaussian process optimization. Here, the total return is used as the surrogate function and Expected Improvement ($\mathcal{EI}$) as the acquisition function.

The experiments were executed on a 40-core machine with 384 GB of memory. Each method is executed in an asynchronously parallel set up of 2-4 GPUs. That is, it can evaluate multiple models in parallel, with each model on a single GPU. When the evaluation of one model is completed, the methods can incorporate the result and immediately re-deploy the next job without waiting for the others to be finalized. We use four K80 (12 GB) GPUs with a time allocation of 10 hours. All algorithms were implemented in Python using Keras [130] and Tensorflow [131] libraries.

Table 6.1 Hyperparameters used in optimization

Parameters	**Search space**	**Type**
Optimizer	Adadelta; Adam; Adagrad; Rmsprop	Categorical
Learning rate	0.001, 0.002, . . . , 0.01	Float
Dimensions of the embeddings	5, 10, 15, . . . , 50	Integer
Number of dense layers	1, 2, 3, 4, 5	Integer
Neurons per layer	100, 200, . . . , 1000	Integer
Dropout rate	0.1, 0.2, . . . , 0.9	Float
Convolutional layers	0, 1, 2, 3	Integer
Filters n	50, 100, 250, 500, 1000	Integer
Filter size h	3, 4, 5	Integer

6.5.3 Results on Next-Flight-Prediction

We perform the Gaussian process to search for the best configuration of hyperparameters. After a few iterations, we are able to get an accuracy of 99.35%. Then, random search boosts the total return very quickly up to 99.62% after 18 iterations and thus remains until the end of the random search cycle (iteration number 50). Using Gaussian process, we can show that we continuously improve our process of searching for the best architecture that maximizes the overall return. In our case, we stopped at iteration number 500 with the best architecture providing an accuracy of 99.68%. One should notice, that no limitation prevents our algorithmic procedure from exploring, even more, the configuration space; in other words, the algorithm was stopped manually. The final training was done with the best-identified architecture found by Bayesian optimization.

Using the "abstain" option, the accuracy increases from 99.62% to 99.94%, estimating confidence with dropout: The prediction tasks can be carried N times while applying dropout in the entire layers of the neural networks [129], yielding N probability vectors for each test sample. A rough estimate of certainty of prediction is obtained by computing the mean of these N probability vectors and subtracting their component-wise estimated standard deviation (computed from the same N vectors). This gives a lower bound on the certainty of our prediction. If the maximum value of these confidences is too low, we decide to abstain. For the next subsection, we will use a 0.5% rejection rate for Heuristic 2 (H2).

6.5.4 Results on crew pairing problems

GENCOL init We consider a standard monthly solution called "GENCOL init", obtained with the GENCOL solver (without DCA). In this approach, the problem is solved by a "rolling time horizon" approach. Because the GENCOL solver (without DCA) is able to solve up to a few

thousand flights per window, we are constrained to use two-days windows and one-day overlap period. This means that the month is divided into overlapping time slices of equal length. Then a solution is constructed greedily in chronological order by solving the problem *restricted* to each time slice sequentially, taking into account the solution of the previous slice through additional constraints.

Baseline Using the constraint aggregation approach, GENCOL-DCA is fed with the pairings of the solution "GENCOL init" as initial clusters to solve the monthly pairing problem, thus obtaining a solution that we consider as a baseline for comparison. Then, for each experiment, we use one of the heuristics previously defined to construct an initial DCA partition and feed it to Commercial-GENCOL-DCA as initial clusters to solve the monthly pairing problem.

Computational results per window are reported in Table 6.2 for all algorithms, namely, baseline, H1, H2, Adapt-H1 and Adapt-H2, where Adapt-H1 and Adapt-H2 are adaptation-based versions of H1 and H2 (see Section 6.4.5). For each window and each algorithm, we provide the LP value at the root node of the search tree N0 (LP-N0), the computational time at N0 (N0 time), the number of fractional variables (# VF-N0) in the current MP solution at N0, the number of branching nodes resolved (# Nodes), the best LP value found (Best-LP), the pairing cost of the best feasible solution (INT) and finally the total computational time (T time); times are in seconds. Furthermore, for all ML algorithms, and for LP-N0, Best-LP and INT, we indicate the relative difference between the result obtained with this algorithm and that with "GENCOL-DCA" (Baseline).

We start by comparing Baseline, H1 and H2. Observe first that H2 gives better LP-N0 and Best-LP values with an average reduction factor of 14.32% and 14.24% respectively, compared to Baseline, while H1 gives lees significant reductions in LP-N0 and Best-LP values with a reduction factor of only 0.57% and 0.45% respectively. The same can be observed for the cost of the best feasible solution (INT), where H2 has an average reduction factor of 9.06% while H1 gives worse results with an increase factor of 18.05%. H1 does not perform well because the proposed clusters are not adapted to the solution of the previous window found by the optimizer, while this problem seems to be partially avoided when using the abstention method. This is explained by the ability of H2 to discard poorly predicted clusters by using the "abstain" option. Since the optimizer takes more time to break a connection in a cluster (links) than to build one, removing low confidence links (flights) using this "abstain" option can be advantageous.

On the other hand, note that for H1 and H2, the average total computational times per window are between 9.57% and 133.85% larger than those of Baseline. This time increase is due to the large number of fractional variables at the root node of the search tree with an increase factor between 5.55% and 129.18%. This is explained by the fact that, when base constraints are too restrictive,

Table 6.2 Computational results per window

Win.	Alg.	LP-N0	Diff. (%)	#VF-N0	#Nodes	Best-LP	Diff. (%)	INT	Diff. (%)	T time (s)
	Baseline	10122035		2719	159	10025487.73		10276344.14		7891
	H1	9904828	-2.15	2870	203	9891390.17	-1.34	10136106.73	-1.36	10405
1	H2	9889525	-2.30	3014	249	9877050.50	-1.48	10300447.14	0.23	11455
	Adapt H1	9904828	-2.15	2870	203	9891390.17	-1.34	10136106.73	-1.36	10691
	Adapt H2	9889525	-2.30	3014	249	9877050.50	-1.48	10300447.14	0.23	13642
	Baseline	11396498		2519	162	11225022.17		11501016.42		27778
	H1	10589590	-7.08	4771	446	10426771.03	-7.11	13080746.04	13.74	63186
2	H2	10528757	-7.61	5048	381	10386235.55	-7.47	11133714.80	-3.19	63462
	Adapt H1	10319719	-9.45	5848	419	10215252.88	-9.00	10865255.80	-5.53	94004
	Adapt H2	10271980	-9.87	6182	492	10188780.86	-9.23	10990621.86	-4.44	126761
	Baseline	10740127		2330	177	10641496.00		11107990.12		30421
	H1	10590614	-1.39	5340	366	10439285.84	-1.90	10926909.75	-1.63	67217
3	H2	9850067	-8.29	4561	307	9659421.20	-9.23	10272961.85	-7.52	52748
	Adapt H1	9596326	-10.65	4838	294	9432461.69	-11.36	10051850.40	-9.51	53736
	Adapt H2	9685782	-9.82	4574	308	9483718.96	-10.88	10160044.85	-8.53	68597
	Baseline	9764727		2606	229	9591592.13		9968081.73		33898
	H1	9164173	-6.15	3808	314	9015056.00	-6.01	13618635.11	36.62	37142
4	H2	8007573	-17.99	5264	358	7872923.59	-17.92	9619055.54	-3.50	79271
	Adapt H1	8063084	-17.43	4763	394	7868177.74	-17.97	8791799.94	-11.80	56291
	Adapt H2	7574006	-22.44	6473	463	7516289.41	-21.64	8333643.43	-16.40	152781
	Baseline	8095063		3150	188	7899149.01		8102703.93		35948
	H1	9375047	15.81	3571	303	9193080.40	16.38	10730905.63	32.44	34123
5	H2	6305624	-22.11	4558	362	6146413.51	-22.19	6746656.99	-16.74	50138
	Adapt H1	6076461	-24.94	5333	417	5953932.68	-24.63	6453605.80	-20.35	74622
	Adapt H2	5706073	-29.51	6008	480	5646359.35	-28.52	6464805.10	-20.21	133395
	Baseline	6778925		2513	187	6610562.32		6939096.41		29368
	H1	6611841	-2.46	3960	324	6432282.33	-2.70	8915306.53	28.48	39626
6	H2	4905952	-27.63	4882	422	4814868.16	-27.16	5297148.54	-23.66	61760
	Adapt H1	4763520	-29.73	4940	318	4697990.54	-28.93	4947363.87	-28.70	55263
	Adapt H2	4763509	-29.73	5713	437	4726686.40	-28.50	5114636.73	-26.29	94092
	Baseline	9482896		2640	183	9332218.23		9649205.46		27551
	H1	9372682	-0.57	4053	326	9232977.63	-0.45	11234768.30	18.05	41950
Mean	H2	8247916	-14.32	4555	346	8126152.09	-14.24	8894997.48	-9.06	53139
	Adapt H1	8120656	-15.73	4765	340	8009867.62	-15.54	8540997.09	-12.88	57435
	Adapt H2	7981813	-17.28	5327	404	7906480.91	-16.71	8560699.85	-12.61	98211

the root node solutions contain a more significant number of fractional-valued pairing variables in order to split the worked time between the bases evenly. This causes an increase in the number of branching nodes required to obtain a good integer solution. Indeed, H1 and H2 present an increase factor in the number of branching nodes between 27.67% and 175.30%.

Next, we compare Baseline, Adapt-H1 and Adapt-H2. For the root node (N0), observe that both adaptation-based heuristics give better LP-N0 values for all windows providing an average reduction factor of 15.73% and 17.28% respectively. Likewise, both heuristics provide better Best-LP values with an average reduction factor of 15.54% and 16.71% and better feasible solutions with a reduction factor of 12.88% and 12.61%. This is explained by the ability of Adapt-H1 and Adapt-H2 to propose custom-made clusters in an online manner adapted to the solution of the previous window, completing clusters that start in the overlap period and proposing new unseen clusters for the non-overlap period.

On the other hand, note that the average computational time for Adapt-H1 is similar to H2 while that of Adapt-H2 is on average 76.46% larger. This is due to the larger number of nodes, caused by the larger number of fractional variables at the root node of the search tree. This can be explained by the fact that Adapt-H2 discards between 30 and 50 clusters per window, providing fewer clusters than Adapt-H1. Therefore, the adaptation scheme is capable of proposing suitable clusters and the shifting scheme to use the next available flight if the predicted next flight is covered by another crew makes the abstention option unnecessary.

Computational results on monthly solutions are reported in Table 6.3. We report the total solution cost, the cost of global constraints and the number of deadheads. For all heuristics algorithms, we also indicate the relative difference between the result obtained with this algorithm and that with GENCOL-DCA (Baseline).

Table 6.3 Computational results on monthly solution

	Solution cost	Diff. vs Baseline (%)	Cost of global constraints	Diff. vs Baseline (%)	Number of deadheads
GENCOL init	30 681 120.5		9 465 982.28		1725
Baseline	20 639 814.6	-32.73 (vs GENCOL init)	2 127 086.77	-77.53 (vs GENCOL init)	992
H1	21 118 006.16	2.32	2 202 610.31	3.55	1136
H2	19 235 343.4	-6.80	642 599.29	-69.79	1059
Adapt-H1	18 881 977.89	-8.52	465 687.94	-78.11	1014
Adapt-H2	19 104 804.62	-7.44	490 787.75	-76.93	1097

We start by comparing GENCOL init and Baseline. Observe first that a significantly better solution was found using Commercial-GENCOL-DCA, compared to the initial solution GENCOL init. Indeed, the solver significantly reduced both the solution cost and the cost of global constraints with a reduction factor of 32.73% and 77.53% respectively, while reducing the number of deadheads by

42.49%. This attests to the optimizer's capacity to tackle larger windows finding better solutions and improving industrial-scale solutions.

Next, we compare Baseline, H1 and H2. While the solution cost and the cost of global constraints found with H1 are slightly worse than Baseline with an increase factor of 2.32% and 3.55%, H2 outperforms Baseline reducing the solution cost by 6.8%. Furthermore, H2 reduced the cost of global constraints by 69.79%, which supports and justifies the large number of fractional variables at the root node of the search tree and the number of nodes, yielding larger computational times. We believe that this trade-off is acceptable since the improvement in the cost of global constraints is significant and that the larger computational times are due to the tight constraints on the number of worked hours per base. Relaxing these constraints may yield better results than Baseline while reducing the computational time. Finally, note that the poor results of H1 and good results of H2 are explained by the ability of H2 to tackle and revise poor predictions and discard poorly constructed clusters.

Then, we compare Baseline, Adapt-H1 and Adapt-H2. The solutions found by both heuristics present better statistics than Baseline, H1 and H2. Adapt-H1 yields better solutions than any other heuristic, with a reduction factor in the solution cost and cost of global constraints of 8.52% and 78.11%, respectively. Adapt-H2 presents similar results while providing a reduction factor of 7.44% and 76.93%. It is also worth noting that, for all heuristics, the number of deadheads used is slightly larger with an increase factor between 2.22% and 14.52%, compared to Baseline. The cost of deadheads is accounted for in the solution cost. Because the solution cost is a multi-objective function, we believe that using slightly more deadheads permitted to get better solutions, enhancing both the solution cost and the cost of global constraints.

6.6 Conclusion

In this paper, we present Commercial-GENCOL-DCA, a new efficient SPP sovler that relies on column generation and constraint aggregation. We developed ML-based heuristics capable of constructing adapted initial clusters for the optimizer, taking into account multiple past solutions. This work is the first attempt to embed into a column generation framework recently developed ML methods. We also proposed an adaptation mechanism to propose clusters in an online manner, taking into account the solution of the previous window.

We compared the performance of Commercial-GENCOL-DCA either using a standard initial solution or with clusters proposed by ML-based heuristics with an existing standard solution used in the airline industry. The main computational results show that Commercial-GENCOL-DCA yields better results than the old GENCOL solver using a rolling-horizon approach with narrow windows.

In addition, ML-based heuristics, along with either abstention or adaptation yields better results with significantly smaller costs reducing by 8.52% the solution cost and by 78.11% the cost of global constraints. This improvement comes with an increase in computational time.

We believe that the proposed solution is able to handle larger problems as well as learn from multiple airline companies to construct initial clusters for a new company since it does not use flight codes in any part of the pre-processing, learning or prediction process. We believe it can easily be adapted to other types of optimization problems, such as railway or bus shift scheduling. Finally, our long-term goal is to develop efficient new learning techniques that could handle and learn from the flight-based network structure, where nodes correspond to time-space coordinates and arcs represent tasks performed by crew members (legs, deadheads, connections, rests, etc.) capable of incorporating global and local constraints in the ML-predictor learning process.

CHAPITRE 7 RÉSEAUX À NOYAUX CONVOLUTIFS STRUCTURÉS POUR LA RECONNAISSANCE OPTIQUE DE CARACTÈRES ET LA PLANIFICATION DES HORAIRES D'ÉQUIPAGE

7.1 Introduction

Nous avons récemment assisté à un regain d'attention porté aux réseaux de neurones à convolution (CNN) en raison de leurs performances élevées dans les tâches de reconnaissance visuelle à grande échelle. L'architecture des CNN est relativement simple et se compose de couches successives organisées de manière hiérarchique ; chaque couche est composée de convolutions avec des filtres appris, suivies d'une fonction d'activation non-linéaire et d'une opération de sous-échantillonnage appelée "regroupement des caractéristiques" ou POOL (*feature pooling*). L'entraînement des CNN reste toutefois difficile, car les réseaux à haute capacité peuvent nécessiter l'apprentissage de milliards de paramètres, ce qui nécessite à la fois une grande puissance de calcul (GPU, par exemple) et des techniques de régularisation appropriées.

Nous étudions un schéma d'approximation appelé réseaux à noyaux convolutifs ou CKN (*Convolutional Kernel Networks*) proposé par Mairal et al. [64], qui rend l'approche du noyau réalisable sur le plan du calcul. Ce schéma est un type de réseau de neurones à convolution supervisée formé pour approximer l'image du noyau. Pour introduire le CKN, nous devons revoir certaines de ses notions et méthodes sous-jacentes, telles que l'astuce du noyau, l'espace d'Hilbert à noyau reproduisant ou RKHS (Reproducing Kernel Hilbert Space), la méthode d'approximation de Nyström de rang faible et le noyau convolutionnel multicouche ou MCK (*Multi-layer Convolutional Kernel*), objet abstrait adapté à des fins théoriques. Le CKN est une approximation en dimension finie d'un MCK, ainsi qu'un type particulier de CNN (c'est-à-dire une succession de convolutions, de fonctions d'activation non-linéaire et de POOL linéaire). Le CKN partage les propriétés caractéristiques du CNN en matière de dispersion et de partage de paramètres, en plus de ses hyperparamètres (par exemple, la taille des filtres, le nombre de filtres). Les CKN et les CNN diffèrent par la fonction de coût à optimiser pour l'apprentissage des filtres et par le choix des non-linéarités.

Les réseaux à noyaux (*kernel networks*) n'ont pas encore été appliqués aux méthodes de prédiction structurée tels que les Champs aléatoires conditionnels ou CRF (*Conditional Random Fields*). Les CRF sont des modèles par paires sur des noeuds (c.-à-d. qu'ils modélisent l'étiquetage d'un graphe, en utilisant les caractéristiques des noeuds et la relation entre les noeuds voisins). L'hypothèse principale de ces modèles est que les noeuds sont homogènes : ils ont tous la même signification. Par conséquent, chaque noeud a le même nombre de classes, et ces classes signifient la même chose.

En pratique, cela signifie que les poids sont partagées entre tous les noeuds et toutes les arêtes et que le modèle s'adapte via des fonctions. L'un des cas d'utilisation les plus courants de la prédiction structurée est celui des sorties structurées en chaîne, utilisant le CRF en chaîne. Ces sorties se produisent naturellement dans les tâches d'étiquetage de séquence, telles que le marquage partiel de la parole ou la reconnaissance d'entités nommées dans le traitement du langage naturel ou la segmentation et la reconnaissance de phonèmes dans le traitement de la parole. À titre d'exemple, nous utilisons la base de données OCR pour la reconnaissance de caractères manuscrites (*Optical Character Recognition*) [85]. Chaque lettre est un noeud de notre chaîne, et les lettres voisines sont reliées par un arête. La longueur de la chaîne varie selon le nombre de lettres du mot. Comme dans tous les modèles de type CRF, les noeuds de la chaîne CRF ont tous la même signification et partagent les mêmes paramètres.

Contributions Pour combler l'écart entre les méthodes du noyau et les réseaux de neurones, nous proposons un nouveau prédicteur structuré appelé réseaux à noyau convolutifs structurés, qui combine les propriétés des architectures d'apprentissage profond, la flexibilité non paramétrique des méthodes du noyau et les prédicteurs structurés. Le CKN apprend à approximer la carte des caractéristiques du noyau sur les données d'entraînement. Il peut être interprété comme un type particulier de CNN. Nous montrons que cette approche alternée offre un grand potentiel dans un cadre d'apprentissage supervisé et couplé.

Nous montrons que l'utilisation de cette combinaison de manière supervisée surpasse les méthodes de pointe en termes de sous-optimalité primale et de précision du test sur le jeu de données OCR. Ensuite, nous appliquons la méthode proposée à un ensemble de données de connexion de vols ou FCD (*Flight-Connection Dataset*) pour proposer de bonnes solutions initiales à un solveur de planification des horaires d'équipage des compagnies aériennes. Plus précisément, nous utilisons une structure de réseau de vols modélisée comme un graphe CRF général (dirigé), où les noeuds correspondent aux coordonnées spatio-temporelles et les arcs représentent les tâches effectuées par les membres d'équipage (vols, repositionnements, connexions, repos, etc.) capables d'incorporer les contraintes locales dans le processus d'entraînement des prédicteurs ML. Une fois que les poids SCKN ont été formés, nous alimentons la solution produite à Commercial-GENCOL-DCA comme solution initiale et comme grappes (*clusters*) initiales pour l'agrégation dynamique de contraintes ou DCA (*Dynamic Constraint Agregation*). Nous montrons que cette approche permet d'accélérer le processus d'optimisation du problème de rotations d'équipage, tout en offrant de meilleures solutions.

Travaux connexes Nous passons d'abord en revue les travaux qui combinent les réseaux de neurones à des modèles structurés à énergie. L'idée de combiner les réseaux et les modèles à énergie

était bien connue dans les années 1990. Par exemple, Bottou [109] et LeCun et al. [110] ont introduit les transformateurs graphiques (une combinaison de modèles de Markov cachés et de réseaux de neurones) entraînés de bout en bout et déployés en tant que systèmes de reconnaissance de caractères manuscrits.

Plus récemment, Tompson et al. [111], Jaderberg et al. [112], Vu et al. [113] et Chen et al. [114] ont utilisé l'approche séquentielle (avec un réglage final de bout en bout au stade final) pour les tâches de vision par ordinateur pour l'estimation de la pose humaine, la reconnaissance de texte libre, la détection d'objets multiples et le marquage d'images. Yang et al. [115] et Chen et al. [116] ont utilisé l'approche de formation conjointe pour les modèles BiLSTM-CRF combinant des techniques de réseaux de neurones, à savoir le plongement de mots ou plongement lexical (word embeddings) et réseaux récurrents avec des unités LSTM, avec le modèle LCCRF au-dessus pour diverses tâches de traitement automatique du langage naturel.

Récemment, le modèle DenseCRF [150] a attiré beaucoup d'attention car il a dépassé par une grande marge les méthodes CRF en chaîne offrant les meilleurs performances [151] sur les tâches d'étiquetage d'images denses, par exemple, la segmentation d'image sémantique. Naturellement, de nombreux travaux ont combiné DenseCRF à des CNN pour améliorer encore davantage les performances. En particulier, Zheng et al. [152] et Schwing et al. [153] ont utilisé la procédure de formation conjointe, mais Chen et al. [154, 155] ont utilisé différentes variantes de la procédure par étapes. En raison de la complexité du réglage des paramètres de DenseCRF, Chen et al. [156] ont remplacé le modèle DenseCRF dans leur système DeepLabv3 par un CRF gaussien continu, car il permettait une inférence exacte relativement rapide en résolvant un système d'équations linéaires.

Le reste de ce chapitre est structuré comme suit. Nous présentons d'abord les CKN non supervisés et supervisés dans la section 7.2. Nous présentons dans la section 7.3 le problème d'optimisation pour les CRF ainsi que deux méthodes d'apprentissage : BCFW (*Block-Coordinate Frank-Wolfe*) et SDCA (*Stochastic Dual Coordinate Ascent*) pour les CRF. Ensuite, nous présentons le CKN structuré ou SCKN et discutons des aspects importants d'implémentation dans la section 7.4. Section 7.5 rapporte les résultats de calcul sur le jeu de données OCR, le jeu de données FCD et le processus d'optimisation de rotations d'équipages. Enfin, une brève conclusion est tirée dans la section 7.6.

7.2 Réseaux à noyaux convolutifs

Pour introduire les CKN, nous devons revoir certaines de ses notions et méthodes sous-jacentes, telles que l'astuce du noyau, l'espace d'Hilbert à noyau reproduisant (RKHS), l'approximation de Nyström de rang faible et le MCK, qui est un objet abstrait approprié à des fins théoriques. Le CKN, une approximation en dimension finie d'un MCK, et un type particulier de CNN, différant

par la fonction de coût à optimiser pour l'apprentissage des filtres et par le choix des non-linéarités. Notons qu'on utilise dans ce chapitre la même notation utilisée dans Mairal et al. [64], Mairal [65] et Bietti et Mairal [66].

7.2.1 L'astuce du noyau

L'astuce du noyau consiste à incorporer un "espace d'entrée brut" de X dans un espace d'Hilbert H de haute dimension, par un mappage (fonction ou *mapping*) éventuellement non linéaire $\varphi : X \to H$, où le produit scalaire dans H admet la formule de calcul suivante :

$$\langle \varphi(x), \varphi(x') \rangle_H = K(x, x'), \quad x, x' \in X \tag{7.1}$$

avec $K : X \times X \mapsto \mathbb{R}$.

La "fonction noyau" K est considérée comme spécifiant la similarité entre les éléments de X. En revanche, les similitudes en H sont exprimées en tant que produits scalaires simples. Le choix d'un φ approprié permet de rendre l'image du sous-ensemble de données de X quasi-linéairement séparable, dans le sens où on pourrait (presque) séparer l'image X et l'image d'un autre sous-ensemble disjoint de X avec un prédicteur linéaire. Lorsqu'un prédicteur linéaire sur H n'utilise que des produits scalaires d'éléments H, la formule (7.1) rend l'apprentissage de ce prédicteur linéaire faisable sur le plan du calcul.

La section suivante passe en revue une classe spécifique d'espaces de fonctionnalités construits à partir de noyaux définis positifs.

7.2.2 Espace d'Hilbert à noyau reproduisant

Definition 1 (Noyau défini positif) *Un noyau défini positif sur un ensemble X est une fonction $K : X \times X \to \mathbb{R}$ qui est symétrique (i.e. $K(x, x') = K(x', x), \forall x, x' \in X$), et pour laquelle, pour tout $N \in \mathbb{N}$ et pour tout $(x_1, ..., x_N) \in X^N$, la matrice de Gram (ou matrice de similarité) $[K]_{ij} = K(x_i, x_j)$ est une matrice semi-définie ou autoadjointe positive (ce qui signifie que pour tout $(a_1, ..., a_N) \in \mathbb{R}^N$, $\sum_{i=1}^N \sum_{j=1}^N a_i a_j K(x_i, x_j) \geq 0$).*

Théorème 1 (RKHS) *Soit K un noyau défini positif sur l'ensemble X. Il existe un espace Hilbert unique $H \subset X^{\mathbb{R}}$ tel que :*

1. $\{K_x : y \mapsto K(x, y), X \mapsto \mathcal{R}\}_{x \in X} \subset H$
2. $\forall f \in H, \forall x \in X, f(x) = \langle f, K_x \rangle_H$

En particulier, la fonction :

$$\begin{aligned} \Phi : X &\mapsto H \\ x &\mapsto K_x \end{aligned}$$

qui projette X dans "l'espace objet" H satisfait :

$$\langle \Phi(x), \Phi(x') \rangle_H = K(x, x') \quad x, x' \in X \tag{7.2}$$

Deux exemples classiques de RHKS associés à l'espace euclidien $X = \mathbb{R}^d$ sont le noyau linéaire et le noyau gaussien. Le noyau linéaire sur X est défini comme étant :

$$K_{lin}(x, x') = \langle x, x' \rangle_{\mathbb{R}^d} \tag{7.3}$$

Le RKHS associé au (X, K_{lin}) est l'espace Hilbert suivant : $H = \{f_w = \langle \bullet, w \rangle_{\mathbb{R}^d}\}_{w \in \mathbb{R}^d}$ muni du produit scalaire : $\langle f_w, f_v \rangle_H = \langle w, v \rangle_{\mathbb{R}^d}$. Ce RKHS est isométrique au $\mathbb{R}^d$. Le noyau gaussien est muni d'un paramètre de lissage σ sur X et est défini comme :

$$K_{Gauss}(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}} \tag{7.4}$$

Le RKHS H associé à (X, K_{Gauss}) est un espace objet de dimension infinie et tous les points de X sont projetés à la sphère unité de H ($\|\Phi(x)\|_H^2 = K(x, x) = 1$).

Théorème 2 (théorème de représentation) *Soit X un ensemble muni d'un noyau positif défini K, H son RKHS et $S = \{x_1, ..., x_n\} \subset X$ l'ensemble d'entraînement. Si $\Psi : \mathbb{R}^{n+1} \mapsto \mathbb{R}$ est strictement croissante, alors :*

$$\begin{aligned} f^* &= \arg\min_{f \in H} \Psi\left(f(x_1), ..., f(x_n), \|f\|_H\right) \\ &\Longrightarrow f^* \in \mathit{Span}\left(K_{x_1}, ..., K_{x_n}\right) \end{aligned}$$

Notons d'ailleurs que f^* est dans un sous-espace de dimension n, bien que H peut être de dimension infinie.

7.2.3 L'approximation de Nyström de rang faible

Considérons un noyau défini positif $K : X \times X \mapsto \mathbb{R}$ et son RKHS H, avec la projection $\varphi : X \mapsto H$ tel que $K(x, x') = \langle \varphi(x), \varphi(x') \rangle_H$. L'approximation de Nyström consiste à remplacer tout point

$\varphi(x)$ dans H par sa projection orhogonale $\Pi_{\mathcal{F}}(x)$ sur un sous-espace de dimension finie

$$\mathcal{F} = \text{Vect}(f_1, ..., f_p) \text{ avec } p \ll n \tag{7.5}$$

où les f_i' sont des *points d'ancrage* dans H.

Cette projection est équivalente à :

$$\Pi_{\mathcal{F}}(x) = \sum_{j=1}^{p} \beta_j^*(x) f_j \tag{7.6}$$

avec

$$\beta^*(x) = \underset{\beta \in \mathbb{R}^p}{\arg\min} \left\| \varphi(x) - \sum_{j=1}^{p} \beta_j f_j \right\|_H^2 \tag{7.7}$$

Sachant que $[K_f]_{jl} = \langle f_j, f_l \rangle_H$ et que $f(x) = (f_1(x), ..., f_p(x)) \in \mathbb{R}^p$, on peut vérifier rapidement que :

$$\beta^*(x) = K_f^{-1} f(x) \tag{7.8}$$

Ainsi, on obtient :

$$\varphi(x) \approx \sum_{j=1}^{p} \beta_j^*(x) f_j \tag{7.9}$$

$$\langle \varphi(x), \varphi(x') \rangle_H \approx \beta^*(x)^T K_f \beta^*(x') \tag{7.10}$$

Soit la projection

$$\begin{aligned} \psi : X &\mapsto \mathbb{R}^p \\ x &\mapsto \psi(x) = K_f^{-1/2} f(x) \end{aligned}$$

tel que $\psi(x) = K_f^{1/2} \beta^*(x)$, on trouve que

$$K(x, x') \approx \langle \psi(x), \psi(x') \rangle_{\mathbb{R}^d}$$

Par conséquent, la projection $\varphi : X \mapsto H$ est approximé par une projection $\psi : X \mapsto \mathbb{R}^p$ tel que

$$\langle \varphi(x), \varphi(x') \rangle_H \approx \langle \psi(x), \psi(x') \rangle_{\mathbb{R}^d}$$

Il existe différentes méthodes pour choisir les points d'ancrage f_j's. Quand $X = \mathbb{R}^d$, une de ces méthodes consiste à utiliser l'algorithme K-means sur un ensemble de données d'entraînement $S = \{x_1, ..., x_n\} \subset X$, pour avoir p centroides $z_1, ..., z_p \in \mathbb{R}^d$, et définir les points d'ancrage comme étant : $f_j = \varphi(z_j)$, for $j = 1, ..., p$.

On se réfère à Williams et Seeger [157] et Zhang et al. [158] pour plus de détails.

7.2.4 Réseaux à noyaux convolutifs multicouches

Les deux principaux avantages d'un réseau CNN par rapport à un réseau de neurones standard entièrement connecté (dense) sont les suivants : (i) faible densité - chaque filtre convolutif non-linéaire n'agit que sur un patch local de l'entrée, (ii) partage des paramètres - le même filtre est appliqué à chaque patch.

Definition 2 (carte de caractéristiques d'une image et d'un patch) *Une carte de caractéristiques d'une image (image feature map) φ est une fonction $\varphi : \Omega \to \mathcal{H}$, où Ω est un sous-ensemble discret de $[0,1]^2$ représentant un ensemble d'emplacements de pixels, et $\mathcal{H}$ est un espace d'Hilbert représentant un espace objet (feature space). Une carte de caractéristiques d'un patch (patch feature map) est identique à celle d'une image avec Ω remplacé par un patch de pixels $\mathcal{P}$ centrés à 0.*

Definition 3 (Noyau convolutif) *On considère deux images représentées par deux carte de caractéristiques, φ et $\varphi' : \Omega \to \mathcal{H}$. Le noyau convolutif à une couche entre φ et φ' est défini comme :*

$$K(\varphi, \varphi') = \sum_{z \in \Omega} \sum_{z' \in \Omega} \Big\{ \|\varphi(z)\|_{\mathcal{H}} \|\varphi'(z')\|_{\mathcal{H}} e^{-\frac{1}{2\beta^2}\|z - z'\|_2^2} \\ e^{-\frac{1}{2\sigma^2}\|\tilde{\varphi}(z) - \tilde{\varphi}'(z')\|_2^2} \Big\} \tag{7.11}$$

où $\tilde{\varphi}(z) = (1/\|\varphi(z)\|_{\mathcal{H}})\, \varphi(z)$ si $\varphi(z) \neq 0$ et $\tilde{\varphi}(z) = 0$ sinon. Pareil pour $\tilde{\varphi}'(z)$.

Notons que le rôle de β est de contrôler combien le noyau est localement invariant par rapport au décalage (*shift-invariant*), et que K est construit sur un noyau k sur H de la forme :

$$k(h, h') = \|h\|_H \|h'\|_H \kappa\left(\frac{\langle h, h' \rangle}{\|h\|_H \|h'\|_H}\right), \quad h, h' \in H \tag{7.12}$$

Selon un résultat classique [159], lorsque κ est lisse avec des coefficients de développement de Taylor non négatifs, k est un noyau positif défini. Par conséquent, K est aussi un noyau positif défini.

Definition 4 (Noyau convolutif multicouche) *Considérons un ensemble discret d'emplacements de pixels $\Omega_{k-1} \subseteq [0,1]^2$ et un espace d'Hilbert $\mathcal{H}_{k-1}$. On construit un nouvel ensemble Ω_k et un nouvel espace d'Hilbert $\mathcal{H}_k$ comme suit : (i) Choisir une forme de patch $\mathcal{P}_k$ défini comme un sous-ensemble symétrique discret de $[-1,1]^2$, et un ensemble d'emplacements de pixels Ω_k tel que pour*

tout emplacement $\mathbf{z}_k$ dans Ω_k, le patch $\{\mathbf{z}_k\} + \mathcal{P}_k$ est un sous-ensemble de Ω_{k-1}; En d'autres termes, chaque emplacement de pixel $\mathbf{z}_k$ in Ω_k correspond à un patch valide d'emplacements de pixels dans Ω_{k-1} centré sur $\mathbf{z}_k$.

(ii) Définir le noyau convolutif K_k sur la carte de caractéristiques du patch $\mathcal{P}_k \to \mathcal{H}_{k-1}$, en remplaçant Ω par $\mathcal{P}_k$, $\mathcal{H}$ par $\mathcal{H}_{k-1}$, et σ, β par les paramètres de lissage appropriés σ_k, β_k (7.11). On note par $\mathcal{H}_k$ l'espace d'Hilbert pourlequel le noyau défini positif K_k est reproduisant.

Une image représentée par une carte de caractéristiques $\varphi_{k-1} : \Omega_{k-1} \to \mathcal{H}_{k-1}$ à la couche k–1 est maintenant encodée dans la k-ème couche comme $\varphi_k : \Omega_k \to \mathcal{H}_k$, où pour tous $\mathbf{z}_k$ dans Ω_k, $\varphi_k(\mathbf{z}_k)$ est la représentation dans $\mathcal{H}_k$ de la carte de caractéristiques du patch $\mathbf{z} \mapsto \varphi_{k-1}(\mathbf{z}_k + \mathbf{z})$ pour $\mathbf{z}$ in $\mathcal{P}_k$.

En d'autres termes, étant donné $\varphi_{k-1} : \Omega_{k-1} \to \mathcal{H}_{k-1}$, on commence par définir Ω_k et $\mathcal{P}_k$ tel que pour tout $z_k \in \Omega_k$, $z_k + \mathcal{P}_k \subset \Omega_{k-1}$. Ensuite, on définit $\mathcal{H}_k$ comme étnat le RKHS de l'ensemble des cartes de caractéristiques du patch.

$$\{\mathbf{z} \mapsto \varphi(\mathbf{z}_k + \mathbf{z}), \mathcal{P}_k \mapsto \mathcal{H}_{k-1} : \varphi : \Omega_{k-1} \to \mathcal{H}_{k-1}, z_k \in \Omega_k\}$$

doté du noyau défini positif K_k adapté de (7.11), et on fixe $\varphi_k(\mathbf{z}_k)$ comme étant la réprésentation dans $\mathcal{H}_k$ de la carte caractéristique patch $\mathbf{z} \mapsto \varphi_{k-1}(\mathbf{z}_k + \mathbf{z})$. Ainsi, on obtient $\varphi_k : \Omega_k \to \mathcal{H}_k$, et pour tout $\varphi_{k-1}, \varphi'_{k-1} : \Omega_{k-1} \to \mathcal{H}_{k-1}$, on a

$$K_k\left(\varphi_{k-1}, \varphi'_{k-1}\right) = \langle \varphi_k, \varphi'_k \rangle_{H_k}$$

7.2.5 Réseaux à noyaux convolutifs non-supervisés

Les réseaux à noyaux convolutifs, ou CKN (*Convolutional Kernel Networks*), sont des approximations des MCK par la méthode de Nyström. Dans cette section, nous présentons comment construire des CKN. Le CKN à une seule couche est illustré à la figure 7.1.

(1) Considérons une image $I_0 : \Omega_0 \to \mathbb{R}^{p_0}$, où p_0 est le nombre canaux, *e.g.*, $p_0 = 3$ pour RVB, et $\Omega_0 \subset [0, 1]^2$ est un ensemble discret d'emplacements de pixels.

(2) On construit une base de données de n patchs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ extraits aléatoirement de diverses images et normalisés pour avoir 1 comme norme ℓ_2.

(3) On utilise un algorithme de K-means sphérique pour obtenir p_1 centroïdes $\mathbf{z}_1, \ldots, \mathbf{z}_{p_1}$ avec une norme ℓ_2 de 1.

(4) Étant donné un patch $\mathbf{x}$ of I_0, la projection de $\varphi_1(\mathbf{x})$ sur $\mathcal{F}_1 = \text{Span}(\varphi_1(z_1), ..., \varphi_1(z_p))$ admet

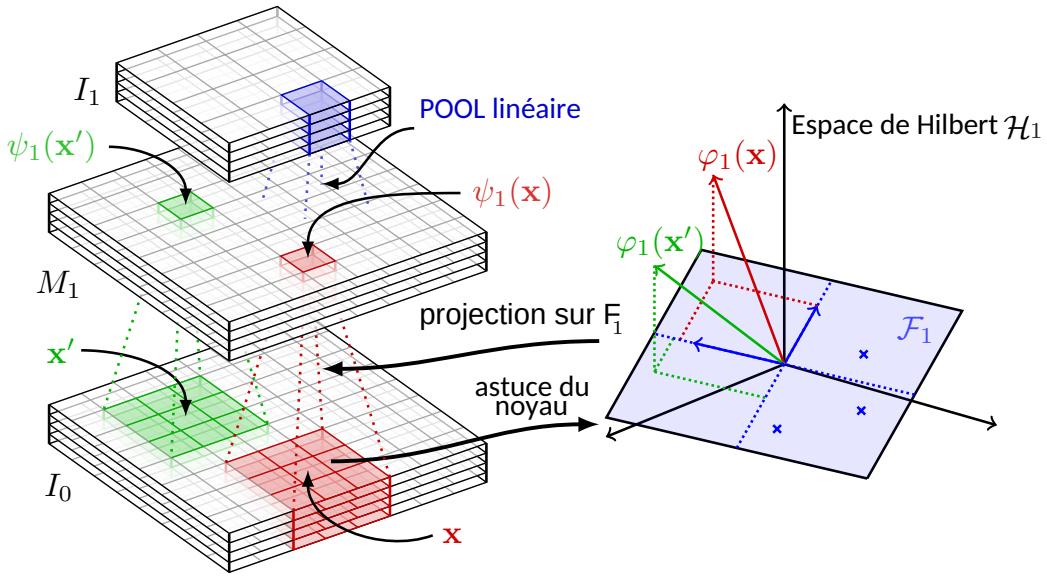


Figure 7.1 Illustration du CKN entre les couches 0 et 1. Les patchs locaux sont projetés sur le RKHS $\mathcal{H}_1$ via l'astuce du noyau et ensuite projetés dans le sous-espace fini $\mathcal{F}_1 =$ Span$(\varphi(\mathbf{z}_1), \ldots, \varphi(\mathbf{z}_{p_1}))$. Les croix bleues à droite représentent les points $\varphi(\mathbf{z}_1), \ldots, \varphi(\mathbf{z}_{p_1})$. Sans supervision, optimiser $\mathcal{F}_1$ consiste à minimiser les résidus de projection [64]

une paramétrisation naturelle conduisant à

$$\psi_1(\mathbf{x}) := \|\mathbf{x}\| \kappa_1(\mathbf{Z}^\top \mathbf{Z})^{-1/2} \kappa_1 \left(\mathbf{Z}^\top \frac{\mathbf{x}}{\|\mathbf{x}\|} \right)$$
$$\text{si } \mathbf{x} \neq 0 \text{ et } 0 \text{ sinon}$$

où nous avons introduit la matrice $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{p_1}]$, et où la fonctio κ_1 est appliquée point par point à ses arguments.

(5) On considère tous les patchs qui se chevauchent de l'image d'entrée I_0. On fixe :

$$M_1(z) = \psi_1(\mathbf{x}_z), \quad z \in \Omega_0$$

où $\mathbf{x}_z$ est le patch provenant de I_0 centré à l'emplacement du pixel z. La carte spatiale $M_1 : \Omega_0 \to \mathbb{R}^{p_1}$ effectue ainsi :
(i) calculer les quantités $\mathbf{Z}^\top \mathbf{x}$ pour tous les patchs $\mathbf{x}$ de l'image I (convolution spatiale après la mise en miroir des filtres $\mathbf{z}_j$) ;
(ii) application de la fonction non linéaire ponctuelle κ_1.

(6) Les étapes précédentes transforment une image $I_0 : \Omega_0 \to \mathbb{R}^{p_0}$ à une carte $M_1 : \Omega_0 \to$

Algorithme 1 : Réseaux à noyaux convolutifs - entraîner les paramètres de la k-ème couche .

1 **Entrées:** $\xi_{k-1}^1, \xi_{k-1}^2, \ldots : \Omega'_{k-1} \to \mathbb{R}^{p_{k-1}}$ (séquence de $(k–1)$ caractéristiques obtenue à partir d'images d'entraînement) ; $\mathcal{P}'_{k-1}$ (forme du patch) ; p_k (nombre de filtres) ; n (nombre d'entrées d'entraînement);

2 extraire aléatoirement des paires n $(\mathbf{x}_i, \mathbf{y}_i)$ de patches avec la forme $\mathcal{P}'_{k-1}$ des cartes $\xi_{k-1}^1, \xi_{k-1}^2, \ldots$;

3 s'il n'est pas fourni par l'utilisateur, définir σ_k sur le quantile 0.1 des données $(\|\mathbf{x}_i - \mathbf{y}_i\|_2)_{i=1}^n$;

4 apprentissage non supervisé : pour obtenir les filtres $\mathbf{W}_k$ in $\mathbb{R}^{|\mathcal{P}'_{k-1}| p_{k-1} \times p_k}$ et $\boldsymbol{\eta}_k$ dans $\mathbb{R}^{p_k}$, résoudre :

$$\min_{\boldsymbol{\eta}, \mathbf{W}} [\frac{1}{n} \sum_{i=1}^{n} (e^{-\frac{1}{2\sigma^2}\|x_i - y_i\|_2^2} - \sum_{l=1}^{p} \boldsymbol{\eta}_l e^{-\frac{1}{\sigma^2}\|x_i - w_l\|_2^2} e^{-\frac{1}{\sigma^2}\|y_i - w_l\|_2^2})^2] \tag{7.13}$$

5 **Sorties:** $\mathbf{W}_k$, $\boldsymbol{\eta}_k$, et σ_k (paramètre de lissage)

$\mathbb{R}^{p_1}$. Ensuite, les CKN contient une étape de POOL pour obtenir l'invariance aux transformations mineurs, conduisant à une autre carte en dimension finie $I_1 : \Omega_1 \to \mathbb{R}^{p_1}$ avec une résolution inférieure :

$$I_1(z) = \sum_{z' \in \Omega_0} M_1(z') e^{-\beta_1 \|z' - z\|_2^2}, \quad z \in \Omega_1$$

(7) On construit une représentation multicouche d'une image en empilant et en composant des noyaux (c'est-à-dire en répétant les étapes précédentes), comme illustré à la figure 7.2.

7.2.6 Réseaux à noyaux convolutifs supervisés

Nous avons décrit une variante de CKN où les sous-espaces linéaires sont appris à chaque couche. Ceci est réalisé de manière non-supervisée au bias de l'algorithme K-means conduisant à des petits résidus de projection. On présente par la suite l'approche supervisée. On considère qu'on dispose d'images d'entraînement I_0^1, I_0^2, ... I_0^n avec les étiquettes y_1, ... y_n dans {-1 ; +1} pour une classification binaire. On dispose aussi de $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, une fonction de perte lisse convexe qui mesure l'ajustement d'une prédiction sur l'étiquette y.

Étant donné un noyau K défini positif sur les images, la formulation empirique classique de minimisation du risque consiste à trouver une fonction de prédiction dans le RKHS $\mathcal{H}$ associé à $\mathcal{K}$ en minimisant l'objectif :

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(I_0^i)) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \tag{7.16}$$

Algorithme 2 : Réseaux à noyaux convolutifs - calcul de la k-ième carte de la $(k–1)$-ième carte.

1 **Entrées:** $\xi_{k-1} : \Omega'_{k-1} \to \mathbb{R}^{p_{k-1}}$ (carte d'entrée) ; $\mathcal{P}'_{k-1}$ (forme du patch) ; $\gamma_k \geq 1$ (facteur de sous-échantillonnage) ; p_k (nombre de filtres) ; σ_k (paramètre de lissage) ; $\mathbf{W}_k = [\mathbf{w}_{kl}]_{l=1}^{p_k}$ et $\boldsymbol{\eta}_k = [\eta_{kl}]_{l=1}^{p_k}$ (paramètres des couches);

2 convolution et non-linéarité : définir la carte d'activation $\zeta_k : \Omega_{k-1} \to \mathbb{R}^{p_k}$ comme :

$$\zeta_k : \mathbf{z} \mapsto \|\psi_{k-1}(\mathbf{z})\|_2 \left[\sqrt{\eta_{kl}} e^{-\frac{1}{\sigma_k^2}\left\|\tilde{\psi}_{k-1}(\mathbf{z})-\mathbf{w}_{kl}\right\|_2^2} \right]_{l=1}^{p_k}, \tag{7.14}$$

où $\psi_{k-1}(\mathbf{z})$ est un vecteur représentant un patch de ξ_{k-1} centré à $\mathbf{z}$ avec la forme $\mathcal{P}'_{k-1}$, et le vecteur $\tilde{\psi}_{k-1}(\mathbf{z})$ est une version normalisée selon la norme ℓ_2 de $\psi_{k-1}(\mathbf{z})$. Cette opération peut être interprétée comme une convolution spatiale de la carte ξ_{k-1} avec les filtres $\mathbf{w}_{kl}$ suivie de non-linéarités ponctuelles;

3 Définir β_k comme étant γ_k fois l'espacement entre deux pixels dans Ω_{k-1};

4 regroupement des caractéristiques POOL : Ω'_k est obtenu en sous-échantillonnant Ω_{k-1} avec un facteur γ_k et on définir une nouvelle carte$\xi_k : \Omega'_k \to \mathbb{R}^{p_k}$ obtenu à partir de ζ_k par une opération de POOL linéaire avec des poids gaussiens :

$$\xi_k : \mathbf{z} \mapsto \sqrt{2/\pi} \sum_{\mathbf{u} \in \Omega_{k-1}} e^{-\frac{1}{\beta_k^2}\|\mathbf{u}-\mathbf{z}\|_2^2} \zeta_k(\mathbf{u}). \tag{7.15}$$

5 **Sorties:** $\xi_k : \Omega'_k \to \mathbb{R}^{p_k}$ (nouvelle carte) ;

où le paramètre λ contrôle la régularité de la fonction de prédiction f par rapport à la géométrie induite par le noyau, régularisant et réduisant ainsi le sur-apprentissage. Après l'entraînement du CKN avec k couches, un noyau défini positif peut être défini comme :

$$K_{\mathcal{Z}}(I_0, I'_0) = \sum_{z \in \Omega_k} \langle f_k(z), f'_k(z) \rangle_{\mathcal{H}_k} = \sum_{z \in \Omega_k} \langle I_k(z), I'_k(z) \rangle \tag{7.17}$$

où I_k, I'_k sont les cartes caractéristiques de taille finie à la couche k de I_0 et I'_0, respectivement, et f_k, f'_k les cartes caractéristiques correspondantes dans $\Omega_k \to \mathcal{H}_k$.

Le noyau est également indexé par $\mathcal{Z}$, qui représente les paramètres réseau, c'est-à-dire les sous-espaces $\mathcal{F}_1$,...., $\mathcal{F}_k$, ou de manière équivalente l'ensemble des filtres Z_1,..., Z_k. Ainsi, la formulation devient équivalente à :

$$\min_{W \in \mathbb{R}^{p_k \times |\Omega_k|}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \langle W, I_k^i \rangle) + \frac{\lambda}{2} \|W\|_F^2 \tag{7.18}$$

où $\|.\|_F$ est la norme de Frobenius qui étend la norme euclidienne aux matrices. Ainsi, la formula-

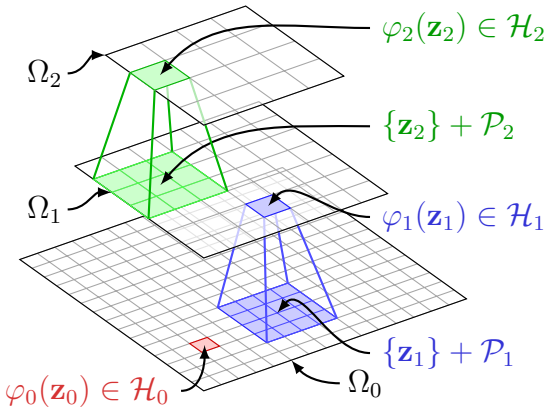


Figure 7.2 Représentation concrète des couches successives du noyau multicouche à convolution (MCK) [64]

tion du CKN supervisé consiste à minimiser conjointement l'équation (7.18) par rapport à W dans $\mathbb{R}^{p_k \times |\Omega_k|}$ et par rapport à l'ensemble des filtres Z_1,...., Z_k, dont les colonnes sont contraintes d'être sur la sphère euclidienne.

Étant donné que nous considérons une fonction de perte lisse, l'optimisation de (7.18) par rapport à W peut être réalisée avec n'importe quelle méthode à base de gradient. Optimiser par rapport aux filtres Z_j , j = 1, . . . , k est plus compliqué puisqu'on manque de convexité. Cependant, et comme la fonction objectif est différentiable, on espère touver un bon point stationnaire en utilisant des techniques classiques d'optimisation stochastique qui ont fait leurs preuves pour l'entraînement de réseaux profonds. Pour le faire, Mairal et al. [64] et Mairal [65] utilisent l'algorithme du gradient stochastique et L-BFGS-B [160]. Dans ce chapitre, on utilise l'algorithme SVRG (Stochastic Variance Reduced Gradient) [161] qui dispose d'une convergence plus rapide que l'algorithme du gradient, sans avoir à stocker les gradients.

7.3 Structure dans les champs aléatoires conditionnels

7.3.1 Champs aléatoires conditionnels

Les Champs aléatoires conditionnels ou CRF (*Conditional Random Field*) [70] sont des modèles graphiques probabilistes utilisés dans le traitement du langage naturel et la vision par ordinateur pour la prédiction structurée.

La probabilité d'observer l'étiquette y quand x est observé est donnée par :

$$p(y|x; w) \propto \exp(\langle w, \phi(x, y)\rangle) \tag{7.19}$$

où ϕ est un mappage de caractéristiques, et w le vecteur de paramètres ou de poids (à apprendre). Le prédicteur CRF de l'étiquette y quand x est observé est donc :

$$h_w(x) = \arg\max_{y \in Y} \langle w, \phi(x, y) \rangle \tag{7.20}$$

Étant donné un ensemble de données d'entraînement étiquetées $D = \{(x_n, y_n)\}_{n=1}^{N}$, nous pouvons calculer le postérieur sur les poids en utilisant la règle de Bayes :

$$p(w|Y, X) = \frac{p(Y|w, X)p(w)}{p(Y|X)} \tag{7.21}$$

où $Y = [y_1, ..., y_N]$, $X = [x_1, ..., x_N]$ et $p(Y|w, X) = \prod_n p(y_n|x_n, w)$. La distribution a-priori de w est géneralement supposée être une distribution multinormale isotrope de moyenne nulle avec une variance paramétrisée par λ. Ainsi, l'apprentissage des paramètres du modèle CRF consiste à déterminer :

$$\begin{aligned} w_{MAP} &= \arg\max_w p(w|Y, X) \\ &= \arg\min_w \{-\log p(w) - \log p(Y|w, X)\} \\ &= \arg\min_w \left\{\lambda \|w\|^2 - \log p(Y|w, X)\right\} \end{aligned}$$

qui est la formulation du problème primal du CRF :

$$\min_\omega \lambda \|\omega\|^2 + \frac{1}{n} \sum_{i=1}^{N} \mathcal{L}^{CRF}(x_i, y_i; \omega) \tag{7.22}$$

avec "log-vraisemblance négative" comme fonction de perte :

$$\mathcal{L}^{CRF}(x_i, y_i; \omega) = \log \left(\sum_{y \in Y} e^{\langle w, \phi(x_i, y) \rangle} \right) - \langle \omega, \phi(x_i, y_i) \rangle \tag{7.23}$$

7.3.2 L'algorithme SVM Structuré

Il existe deux formulations alternatives pour les SVM multi-classes : Crammer et Singer [162] définissent une variante qui ne pénalise que le plus grand étiquetage incorrect, tandis que Weston et Watkins [163] pénalisent tout étiquetage qui fournit éventuellement un rendement supérieur à celui de l'étiquetage réel.

Nous définissons un SVM multi-classes comme suit :

— **entrée :** $(x_1, y_1), ..., (x_m, y_m)$

— **paramètres :**
paramètre de régularisation $\lambda > 0$
fonction d'erreur ou de perte $\Delta : Y \times Y \mapsto \mathbb{R}_+$
mappage des caractéristiques sensibles à la classe $\phi : X \times Y \mapsto \mathbb{R}^d$
— **résoudre :** $\min_{\omega \in \mathbb{R}^d} \lambda \|\omega\|^2 + \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{SVM}(x_i, y_i; \omega)$
— **sortie :** Le prédicteur $h_w(x) = \arg\max_{y \in Y} \langle w, \phi(x, y) \rangle$

où $\mathcal{L}^{SVM}$, appelée "marge maximale généralisée" (*generalized hinge loss*), est définie comme

$$\mathcal{L}^{SVM}(x_i, y_i; \omega) = \max_{y' \in Y} \{\Delta(y', y_i) + \langle w, \phi(x_i, y') - \phi(x_i, y_i) \rangle\} \tag{7.24}$$

L'implémentation d'un SVM Structuré (SSVM) consiste à prendre en compte la " structure " de l'espace de données $X \times Y$ en définissant une fonction d'erreur structurée appropriée Δ, et un mappage approprié des fonctionnalités ϕ.

Par exemple, dans le contexte du jeu de données OCR, l'erreur structurée $\Delta(y, y')$ de deux séquences de lettres y et y' de même longueur r est généralement définie comme étant la "perte moyenne de Hamming" de y et y', c'est-à-dire $\Delta(y, y') = \frac{1}{r} \sum_{i=1}^{r} \mathbb{1}_{y_i = y'_i}$. De plus, dans un contexte structuré, lorsque toutes les étiquettes en Y sont *a priori* non acceptables pour un x_i donné, le $\max$ de la fonction de perte "marge maximale généralisée" $\mathcal{L}^{SVM}$ (7.24) est indiqué comme étant calculé sur le sous-ensemble des étiquettes $Y(x_i) \subset Y$ acceptables. Notez que si nous remplaçons la fonction de perte "marge maximale généralisée" par la fonction de perte de "log-vraisemblance négative" dans la définition du SVM multi-classes ci-dessus, on retrouve le prédicteur CRF.

7.3.3 L'algorithme de Frank-Wolfe

Étant donné l'ensemble d'entraînement $\{(x_1, y_1), ...(x_n, y_n)\} \in X \times Y$, la fonction de perte structurée $\Delta : Y \times Y \to \mathbb{R}$ et une fonction $\Phi : X \times Y \to \mathbb{R}$ qui encode l'information entrée/sortie nécessaire à la prédiction. Nous cherchons à trouver le paramètre w de $f(x) = argmax_y \langle w, \Phi(x, y) \rangle$ qui minimise la perte prévue ou estimée sur les données futures. Pour ce faire, nous utilisons des prédicteurs SSVM dérivés du cadre de la marge maximale. En effet, nous imposons que la sortie correcte soit meilleure que d'autres par une marge :

$$\langle w, \Phi(x_n, y_n) \rangle \geq \Delta(y_n, y) + \langle w, \Phi(x_n, y) \rangle; \ \forall \, y \in Y$$

C'est un problème d'optimisation convexe, mais non-différentiable. L'entraînement nécessite une prédiction-argmax répétée sans inférence et nous avons de nombreuses formulations équivalentes,

donc des algorithmes d'entraînement différents.

L'algorithme Frank-Wolfe, un de ces algorithmes d'entraînement, considère le problème de minimisation convexe $min_{\alpha \in M} f(\alpha)$, où M est compact, et f est continue différentiable. L'algorithme de Frank-Wolfe ne nécessite que l'optimisation des fonctions linéaires sur M. Puisque nous expérimentons sur des ensembles de données relativement volumineux, nous utilisons BCFW (*Block-Coordinate Frank Wolfe*) dans lequel chaque itération utilise un appel à l'oracle de maximisation et montre une meilleure convergence que la version standard ou la version en batch de l'algorithme.

Algorithme 3 : Algorithme de BCFW pour la résolution du SVM Structuré

Initialiser $w^{(0)} = {w_i}^{(0)} = \overline{w}^{(0)} = 0, \;\; \ell^{(0)} = \ell_i^{(0)} = 0$;
Pour $k = 0 \dots K$ **faire**
  Choisir i aléatoirement de $\{1, \dots, n\}$;
  Résoudre $y_i^* = \arg\max_{y \in Y_i} \; H_i(y; w^{(k)}) = \arg\max_{y \in Y_i} \; \Delta_i(y) - \langle w^{(k)}, \Psi_i(y) \rangle$
  où $\Delta_i(y) = \Delta(y_i, y)$ et $\Psi_i(y) = \Phi(x_i, y_i) - \Phi(x_i, y)$;
  Soit $w_s = \frac{1}{\lambda n} \Psi_i(y_i^*) \quad$ et $\quad \ell_s = \frac{1}{n} \Delta_i(y_i^*)$;
  Soit $\gamma = \frac{\lambda(w_i^{(k)} - w_s)^T w^{(k)} - \ell_i^{(k)} + \ell_s}{\lambda \| w_i^{(k)} - w_s \|^2}$ et limiter sur $[0, 1]$;
  Mettre à jour ${w_i}^{(k+1)} = (1 - \gamma) {w_i}^{(k)} + \gamma \, w_s \qquad$ et $\; {\ell_i}^{(k+1)} = (1 - \gamma) {\ell_i}^{(k)} + \gamma \, \ell_s$;
  Mettre à jour $w^{(k+1)} \; = w^{(k)} + {w_i}^{(k+1)} - {w_i}^{(k)} \qquad$ et $\; \ell^{(k+1)} = \; \ell^{(k)} + {\ell_i}^{(k+1)} \; - {\ell_i}^{(k)}$;
  (Facultatif : Mettre à jour $\overline{w}^{(k+1)} = \frac{k}{k+2} \overline{w}^{(k)} + \frac{2}{k+2} w^{(k+1)}$);

7.3.4 L'algorithme SDCA

Une autre alternative pour entraîner les CRF consiste à optimiser directement le modèle associé. L'optimisation de ces modèles est notoirement difficile. Schmidt et al. [80] décrivent une implémentation pratique de l'algorithme SAG (*Stochastic Average Gradient*) [164] pour les CRF et proposent une approche d'échantillonnage non-uniforme qui améliore les performances.

L'algorithme SDCA (*Stochastic Dual Coordinate Ascent*) proposé par Shalev-Shwartz et Zhang [165, 83] appartient à cette famille d'optimiseurs déstinés à entraîner les CRF. Il est étroitement lié à l'algorithme OEG (*Online Exponentiated Gradient*) dans la mesure où il effectue également une ascension de coordonnées par blocs sur la fonction objectif duale. Un avantage intéressant de SDCA par rapport à OEG (et SAG) est que la forme de sa mise à jour permet d'effectuer une recherche linéaire "exacte" avec un seul appel à l'oracle de marginalisation, à savoir le calcul des probabilités marginales pour le CRF. Cela contraste avec SAG et OEG, où chaque changement de taille du pas nécessite un nouvel appel à l'oracle de la marginalisation.

En plus, l'algorithme SDCA bénéficie d'un taux de convergence linéaire et d'une bonne perfor-

mance empirique pour les problèmes de classification binaire et bénéficie d'une recherche linéaire "exacte" avec un seul appel à l'oracle de marginalisation, contrairement aux approches précédentes. Il met à jour une coordonnée duale à la fois afin de maximiser la valeur objectif duale. SDCA a été initialement proposé pour la classification binaire [165] où chaque variable duale α_i est dans $\Delta_2 = [0, 1]$. Dans ce cas, il est possible de faire une maximisation exacte des coordonnées de la fonction objectif duale sur un seul α_i avec une optimisation unidimensionnelle standard.

Le-Priol et al. [166] ont proposé une adaptation de SDCA au cadre du CRF en considérant les probabilités marginales sur les cliques du modèle graphique. Ils fournissent une nouvelle interprétation de SDCA en tant qu'une méthode d'entraînement relaxé à point fixe et soulignent la séparabilité du gap duale par bloc. Dans les CRF, la variable duale α_i est exponentiellement grande dans la taille d'entrée x_i. Pour une séquence x_i de longueur T où chaque noeud peut prendre jusqu'à K valeurs, le nombre d'étiquettes possibles est $|\mathbb{Y}_i| = K^T$, ce qui peut même ne pas tenir en mémoire. Au lieu de cela, l'approche standard utilisée dans OEG et SAG consiste à considérer les probabilités marginales $(\mu_C)_{C\in\mathbb{C}}$ sur les cliques du modèle graphique $\mathbb{C}$. De même, on remplace α par $\mu = (\mu_1, ..., \mu_n)$, où $\mu_i \in \Pi_C^{\Delta_C}$ est la concaténation de tous les vecteurs marginaux des cliques de l'échantillon i.

L'algorithme 4 décrit cette variante du SDCA, avec une stratégie d'échantillonnage non-uniforme. La stratégie d'échantillonnage de gap est similaire à celle d'Osokin et al. [167] dans le contexte de SDCA appliquée à l'objectif SSVM. En supposant une connaissance complète des sauts de dualité (*duality gaps*), la décision optimale consiste à échantillonner le point avec l'écart de dualité maximal, en biaisant l'échantillonnage vers des exemples dont les écarts de dualité sont grands.

Algorithme 4 : SDCA pour CRF

Initialiser $\mu_i^{(0)} \in \Pi_C^{\Delta_C}, \forall i$;
Initialiser $w^{(0)} = \hat{w}(\mu^{(0)}) = \frac{1}{\lambda n}\sum_i B_i \mu_i^{(0)}$
  où B_i est la concaténation horizontale des vecteurs caractéristiques des cliques ;
(Facultatif) Initialiser le le saut de dualité $g_i = 100, \forall i$;
Pour *t=1 :* $n_{Epoques\_SCDA}$ **faire**
  Choisir i aléatoirement de {1, . . . , n};
  (Alternativement) Choisir i avec proportionnellement à g_i;
  Soit $\nu_{i,C}(y_C) = p(y_C|x_i; w^{(t)}), \forall C \in \mathbb{C}$;
  (Facultatif) Soit $g_i = \tilde{D}(\mu_i||\nu_i)$ (7.27);
  Définir la direction de montée comme : $\delta_i = \nu_i - \mu_i^t$;
  Définir la direction primale comme : $v_i = \frac{1}{\lambda n}\hat{w}(\delta_i)$;
  Utiliser la recherche linéaire pour obtenir la taille de pas optimale (7.28);
  Mettre à jour $\mu_i^{(t+1)} = \mu_i^{(t)} + \gamma^* \delta_i$;
  Mettre à jour $w^{(t+1)} = \hat{w}(\mu^{(t+1)}) = w^{(t)} + \gamma^* v_i$

Si nous supposons que le graphe a une structure d'arbre de jonction $\mathbb{T} = (\mathbb{C}, \mathbb{S})$ [168], où $\mathbb{C}$ est

l'ensemble des cliques maximum et $\mathbb{S}$ l'ensemble des séparateurs. Nous pouvons alors exécuter la méthode de passage de messages sur l'arbre de jonction pour déduire les nouveaux marginaux donnés par les poids w : $\hat{\mu}_i(w) = p(y_C = .|x_i; w)$. Nous pouvons donc récupérer la probabilité commune $\alpha_i(y)$ en fonction de ses marginaux $\mu_{i,C}$ [168] :

$$\alpha_i(y) = \frac{\prod_{C \in \mathbb{C}} \mu_{i,C}(y_C)}{\prod_{S \in \mathbb{S}} \mu_{i,S}(y_S)} \tag{7.25}$$

L'équation (7.25) permet à son tour de calculer l'entropie et les divergences des probabilités jointes (divergences of the joints) en utilisant uniquement les marginaux. Soit μ_i et ν_i les marginaux de α_i et β_i respectivement, l'entropie et la divergence Kullback-Leibler sont données donc par :

$$\widetilde{H}(\mu_i) = H(\alpha_i) = \sum_C H(\mu_{i,C}) - \sum_S H(\mu_{i,S}) \tag{7.26}$$

$$\widetilde{D}(\mu_i||\nu_i) = D_{KL}(\alpha_i||\beta_i) = \sum_C D_{KL}(\mu_{i,C}||\nu_{i,C}) - \sum_S D_{KL}(\mu_{i,S}||\nu_{i,S}) \tag{7.27}$$

Avec cette expression de l'entropie (7.26), on peut calculer la valeur objectif duale, et donc effectuer la recherche linéaire :

$$\gamma^* = \text{argmax}_{\gamma \in [0,1]} \widetilde{H}(\mu_i^{(t)} + \gamma \delta_i) - \frac{\lambda n}{2} \|w^{(t)} + \gamma v_i\|^2 \tag{7.28}$$

7.4 L'algorithme SCKN

Nous décrivons dans cette section le modèle des réseaux à noyaux convolutifs structurés (SCKN). Nous décrivons d'abord le modèle et l'algorithme d'apprentissage dans la section 7.4.1, puis certains aspects importants de l'implémentation sont résumés dans la section 7.4.2.

7.4.1 Modèle

Le modèle SCKN est composé de deux composantes différentes, le CKN et le prédicteur structuré (soit SSVM-BCFW soit SDCA), tous deux destinés à entraîner des CRF. On commence par initialiser les couches CKN et le prédicteur structuré. Ensuite, pour chaque itération, l'image d'entrée est passée à travers les multicouches CKN. La dernière carte du CKN est transmise au prédicteur structuré pour inférer les probabilités et ensuite entraîner les paramètres du prédicteur structuré. Les probabilités inférées sont utilisées pour former les poids CKN par rétropropropagation selon les règles décrites dans la section 7.2.6.

Algorithme 5 : Apprentissage des paramètres du SCKN

Initialiser les poids CKN de manière non-supervisée comme décrit dans la section 7.2.5;
Initialiser le prédicteur structuré et le modèle CRF comme décrit dans les sections 7.3.3 et 7.3.4;
Pour $k = 0 \dots$ **faire**
    Pour chaque entrée, construire une carte de caractéristiques unaire, comme décrit dans la section 7.2.5;
    (Facultatif) Utiliser une des techniques de mise à l'échelle pour centrer et redimensionner ces représentations afin d'obtenir une norme ℓ_2 égale à 1 en moyenne (*Normalizer*) ou redimensionner dans $[0, 1]$ (Min-Max) ;
    Inférer les probabilités en fournissant la carte-image en entrée au prédicteur structuré, comme décrit dans les sections 7.3.3 et 7.3.4;
    Entraîner le prédicteur structuré en utilisant la carte de caractéristiques comme entrée pour $n_{Epoques}$ époques (mises à jours des paramètres), comme décrit dans les sections 7.3.3 et 7.3.4;
    Utiliser les probabilités inférées pour calculer le gradient et mettre à jour les poids CKN en utilisant la règle de chaîne (rétropropagation), comme décrit dans la section 7.2.6

7.4.2 Implémentation

Nous présentons divers aspects importants d'implémentation dans les points suivants :

- méthode d'inférence : nous utilisons l'algorithme du "max-produit" pour la propagation des convictions (*belief propagation*). Nous utilisons cet algorithme car les chaînes peuvent être résolues de manière précise et efficace.
- Comme mentionné dans Chandra et al. [169], lors de l'utilisation de prédicteurs structurés profonds, il peut être nécessaire d'utiliser la mise à l'échelle (*scalers*) dans la mesure où la dernière carte (*map*) de la couche "profonde" doit être redimensionnée avant d'être passée à la couche "structurée". Par conséquent, nous proposons d'utiliser l'une des techniques de mise à l'échelle suivantes, dans la bibliothèque scikit-learn [132] : *Min-Max scaler*, *Normalizer scaler*, *Standard scaler* et *Robust scaler*.
- Lors de l'utilisation de l'optimiseur SDCA, le stockage de la variable duale peut s'avérer coûteux et il est préferable d'allouer une quantité suffisante de mémoire.
- Lors de l'utilisation de l'optimiseur SDCA, la recherche linéaire nécessite le calcul de l'entropie des marginaux. C'est coûteux et nous avons utilisé l'algorithme de Newton-Raphson pour minimiser le nombre d'itérations. Cela nécessite à son tour de stocker le logarithme de la variable duale.
- Lorsque cela est nécessaire, il peut être utile d'utiliser un plongement de mots ou une couche de plongement ou d'incorporation (*embedding layer*) avant de transmettre l'entrée

aux couches CKN. En effet, pour les variables catégorielles comportant un grand nombre de catégories telles que des mots, la matrice d'entrée est peu dense, ce qui rend le processus d'apprentissage difficile, car les patchs extraits ont pour la plupart des valeurs nulles. C'est d'ailleurs également le cas pour l'ensemble de données de rotations d'équipage que nous proposons, car il contient de multiples valeurs catégorielles, telles que les codes de ville. La sortie de la couche de plongement donne la représentation d'entrée introduite dans les couches CKN.

7.5 Résultats expérimentaux

7.5.1 Configuration expérimentale

Nous présentons maintenant les expériences que nous avons réalisées avec notre modèle SCKN. Nous utilisons Pytorch pour déclarer ledit modèle et effectuer des opérations sur GPU. Le modèle CRF est implémenté en utilisant la bibliothèque PyStruct et l'optimiseur SDCA est implémenté en utilisant la bibliothèque SDCA4CRF. Les techniques de mise à l'échelle (*scalers*) sont implémentés en utilisant Scikit-learn.

7.5.2 Reconnaissance optique de caractères

Les CRF en chaîne sont des modèles dits "couples" (*pairwise models*) sur des noeuds, c'est-à-dire qu'ils modélisent l'étiquetage d'un graphe, en utilisant les caractéristiques des noeuds et les relations entre les noeuds voisins. L'hypothèse principale de ces modèles est que les noeuds sont homogènes, c'est-à-dire qu'ils ont tous la même signification. Cela signifie que chaque noeud a le même nombre de classes, et que ces classes ont la même signification. En pratique, cela signifie que les poids sont partagés entre tous les noeuds et toutes les arêtes et que le modèle s'adapte via des caractéristiques (*features*).

L'un des cas d'utilisation les plus courants pour la prédiction structurée est celui des sorties structurées en chaîne, utilisant les CRF en chaîne. Celles-ci sont utilisées naturellement dans les tâches d'étiquetage de séquence, telles que l'extraction de groupes nominaux ou la reconnaissance d'entités nommées dans le traitement du langage naturel, ou la segmentation et la reconnaissance de phonèmes dans le traitement de la parole.

Comme exemple de jeu de données, nous utilisons OCR [85] pour la reconnaissance de caractères manuscrites. Chaque lettre est un noeud de notre chaîne, et les lettres voisines sont reliées par un arête. La longueur de la chaîne varie en fonction du nombre de lettres du mot. Comme dans tous les modèles de type CRF, les noeuds du CRF en chaîne ont tous la même signification et par-

tagent les mêmes paramètres. Le jeu de données de lettres contient plusieurs sous-ensemble (*folds*) prédéfinis, nous considérons un pli pour être l'ensemble de test, et le reste pour être l'ensemble de formation, comme dans les réseaux de Markov à Maximum de Marge ou M3N (*Max-Margin Markov Networks*) [85].

— BCFW (Block-Coordinate Frank-Wolfe) implémenté par Lacoste-Julien et al. [170]
— SAG-NUS et SAG-NUS* : implémenté par Schmidt et al. [80]
— OEG : proposé par Collins et al. [74] et implémenté par Schmidt et al. [80]
— SDCA et SDCA-GAP : implémenté par Le-Priol et al. [166]
— RNN : Réseaux de neurones récurrents standard
— Neural-CRF : implémenté par Artieres et al. [171]
— SCRBM : implémenté par Tran et al. [172], il s'agit d'une variante des machines de Boltzmann restreintes ou RBM (*Restrictred Boltzmann Machines*) conçue pour la classification de séquences en roulant les RBMs et les variables de classe dans le temps.

Nous rapportons dans le tableau 7.1 l'erreur de test obtenue avec différents algorithmes. Alors que BCFW a une erreur de test de 17%, toutes les méthodes d'optimisation du CRF (SAG-NUS, SAG-NUS* SDCA, SDCA-GAP, OEG) produisent toutes des erreurs de test similaires entre 11.8% et 12%. Les réseaux de neurones récurrents et Neural-CRF donnent des résultats moins bons, avec des taux d'erreur de test de 12% et 13.3% respectivement. À notre connaissance, SCRBM fournit le meilleur résultat dans la litérature, générant une erreur de test de 4.44%. L'utilisation de notre méthode SCKN-BCFW donne de meilleurs résultats en réduisant l'erreur de test à 3.42% tandis que SCKN-SDCA réduit l'erreur de test à 3.40%.

Tableau 7.1 Erreur de test sur le jeu de données OCR

	BCFW	SAG NUS	SDCA	RNN	Neural-CRF	SCRBM	SCKN-BCFW	SCKN-SDCA
Erreur de test (%)	17	11.8	12.0	13.3	4.44	4.03	3.42	**3.40**

Ensuite, nous rapportons la sous-optimalité primale en fournissant d'abord une comparaison dans la Figure 7.3 en utilisant différentes valeurs $n_{Epoques\_SCDA}$. Ensuite, dans la Figure 7.4, nous comparons selon deux mesures de complexité différentes.

Les appels à l'oracle Schmidt et al. [80] ont comparé les algorithmes sur la base du nombre d'appels à l'oracle. Cette mesure convenait aux méthodes qu'ils ont comparées. OEG et SAG-NUS utilisent tous deux une recherche linéaire où ils appellent un oracle à chaque étape. SDCA, SDCA-GAP et SCKN-SDCA n'ont pas besoin d'oracle pour effectuer leur recherche linéaire. L'oracle est

un algorithme de transmission de message sur un arbre de jonction. Son coût est proportionnel à la taille des marginaux. Chaque itération de la recherche linéaire nécessite le calcul de l'entropie de ces marginaux, ou de leurs dérivées. Ces coûts sont à peu près les mêmes.

Mises à jour des paramètres Pour donner une perspective différente, nous rapportons aussi le log de la sous-optimalité par rapport au nombre de mises à jour des paramètres. Ceci supprime le coût supplémentaire de la recherche linéaire pour toutes les méthodes.

Notons que la précision de la sous-optimalité primale est négligeable en dessous de 10^{-5} puisque nous n'entraînons habituellement pas le prédicteur structuré plus de quelques époques (mise à jour des paramètres). En effet, nous obtenons la meilleure erreur de test avec nos méthodes après seulement quelques mises à jour de paramètres (5-10). Par conséquent, nous ne sommes pas tenus d'avoir une précision de 10^{-6} comme critère d'arrêt.

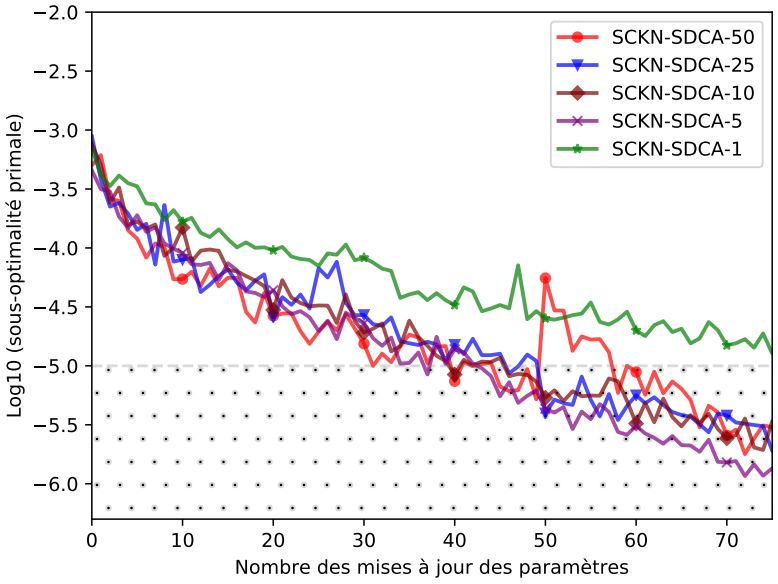


Figure 7.3 SCKN avec échantillonnage basé sur le gap de dualité avec différentes valeurs $n_{Epoques\_SCDA}$, comme indiqué par les valeurs dans la légende.

Comme le montre la Figure 7.3, nous obtenons des résultats similaires si nous utilisons des valeurs $n_{Epoques\_SCDA}$ entre 5 et 25. Effectuer une seule mise à jour de paramètre de SDCA avant de mettre à jour les poids du CKN rend le processus d'apprentissage difficile et l'utilisation d'une valeur de 50 entraîne un surapprentissage de l'optimiseur, ce qui rend le processus d'apprentissage plus difficile par la suite.

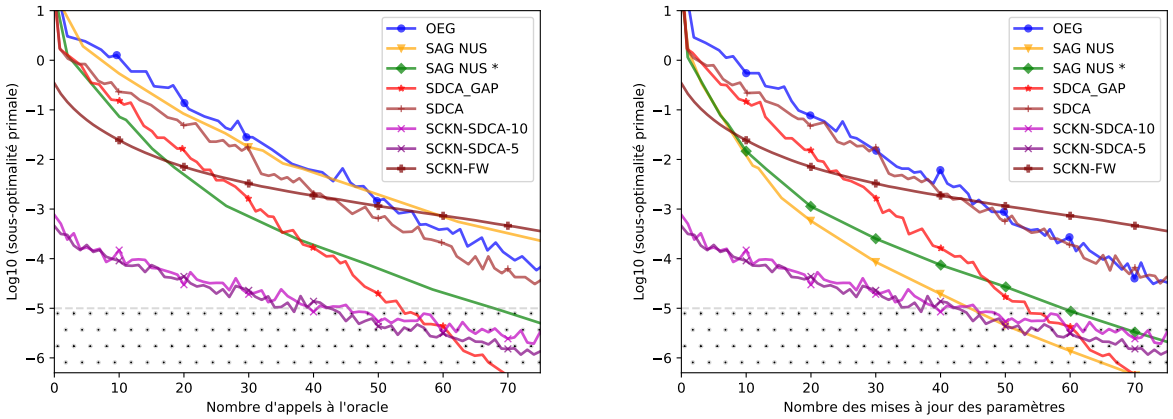


Figure 7.4 Comparaison de la sous-optimalité primale en fonction du nombre d'appels à l'oracle (à gauche) ou du nombre des mises à jour des paramètres (à droite). SDCA fait référence à un échantillonnage uniforme. SDCA-GAP fait référence à l'échantillonnage basé sur le gap de dualité avec un un taux de 80%. SAG-NUS effectue une recherche linéaire à chaque itération. SAG-NUS* met en oeuvre une stratégie de recherche linéaire munie d'une liste à saut (*line-search skipping strategy*). SCKN-FW et SCKN-SDCA utilisent respectivement BCFW et SDCA comme prédicteurs structurés. SCKN-SDCA-5 et SCKN-SDCA-10 utilisent respectivement 5 et 10 époques pour entraîner les poids SDCA pour chaque itération.

Comme le montre la figure 7.4, SCKN-SDCA surpasse les autres méthodes pour les 50 premières époques et est comparable aux autres méthodes pour les époques suivantes. Par ailleurs, notons que le prédicteur SDCA est désavantagé dans ce cas puisque nous n'utilisons pas le démarrage à chaud (*warm-start*), en ce sens que nous initialisons ses poids à chaque époque (chaque $n_{Epoques\_SCDA}$ mises à jour des paramètres du SDCA). Pour obtenir la meilleure sous-optimalité primale, on peut arrêter la mise à jour des poids CKN après 30-40 époques. Mais, comme mentionné précedemment, il n'était ni crucial ni nécessaire d'implémenter cette technique, puisque nous nous arrêtons après seulement quelques époques dans des cas réels d'utilisation.

7.5.3 Résolution du problème de rotations d'équipage

Dans cette section, nous présentons les expériences réalisées à l'aide du prédicteur SCKN proposé pour le problème de rotations d'équipage de grande taille. On commence par présenter le problème de prédiction et décrivons le modèle CRF construit, puis on présente une brève description des instances que nous avons utilisées pour construire le jeu de données FCD. Par la suite, on présente les résultats des prédictions, le processus d'optimisation ainsi que des statistiques sur la faisabilité

de la solution mensuelle proposée. Enfin, on présente les résultats de la résolution du problème de rotations d'équipage à l'aide de l'optimiseur Commercial-GENCOL-DCA.

Formulation du problème de prédiction

Dans notre méthode de résolution du problème de rotations d'équipage, nous cherchons à fournir à l'optimiseur une première partition des vols en grappes ainsi qu'une solution initiale. Chaque grappe représente une séquence de vols avec une forte probabilité d'être consécutifs dans la même rotation dans la solution. Par conséquent, nous transformons les données pour construire un problème de prédiction de connexion de vols (un problème de classification multi-classes supervisé), dont l'objectif est de prédire le prochain vol qu'un équipage de compagnie aérienne devrait suivre dans son horaire, en ne donnant que des informations partielles sur le début de son horaire.

Le problème de classification est le suivant : étant donné les informations relatives à un vol entrant dans une ville de correspondance, il faudrait choisir parmi tous les vols au départ de cette ville (qui peuvent être limités aux 48 prochaines heures) celui que l'équipage doit suivre. Ces vols au départ seront identifiés par leur code de vols (environ 2 000 codes de vols possibles dans notre jeu de données). Différents vols peuvent partager les mêmes codes de vols dans certaines compagnies aériennes, car les vols effectués plusieurs fois par semaine utilisent généralement le même code de vol. Néanmoins, un vol est identifié de manière unique par son code de vol, sa ville et son jour de départ ; informations qui -en pratique- peuvent être déduites des informations sur le vol entrant et de la fenêtre de 48 heures.

Chaque vol possède les 5 caractéristiques suivantes que nous pouvons utiliser dans notre algorithme de classification :

- Ville d'origine et ville de destination ($\sim$ 200 villes) ;
- Type d'aéronef (5 types) ;
- Durée du vol (en minutes) ;
- Temps d'arrivée (pour un vol entrant) ou de départ (pour un vol sortant).

Tout au long des mois, dans 74% à 76% des cas, l'équipage arrive à l'aéroport et suit l'avion, c'est-à-dire qu'il prend le prochain vol que le même avion effectue. Étant donné que l'équipage est arrivé à un aéroport spécifique à un moment donné, nous pouvons utiliser ces informations pour définir quels vols sont possibles. Par exemple, il n'est pas possible d'effectuer un vol qui commence dix minutes après l'arrivée, ni cinq jours plus tard. De plus, il est rare que le type d'aéronef change entre les vols puisque chaque équipage est formé pour utiliser un ou deux types d'avions au maximum. Le lecteur est invité à consulter Kasirzadeh et al. [18] pour plus de détails sur la probabilité de ces scénarios. Notez que même si nous utilisons l'itinéraire des avions dans les étapes de prétraitement, le classificateur ne peut pas savoir quel prochain vol utilise le même avion, car les performances du

prédicteur sont similaires, que nous utilisions cette information comme une caractéristique ou non. Le problème de prédiction n'est donc pas sensible à cette information et peut s'en passer.

Dans notre travail, nous utilisons les conditions suivantes, qui doivent toujours être satisfaites pour le prochain vol effectué par l'équipage :

- L'heure de départ du prochain vol doit être ultérieure à l'heure d'arrivée du vol précédent vers la ville de correspondance ;
- L'heure de départ du vol suivant ne doit pas dépasser 48 heures après l'heure d'arrivée du vol précédent vers la ville de correspondance ;
- La ville de départ du prochain vol doit être identique à la ville de correspondance du vol précédent ;
- Le type d'aéronef devrait être le même. En effet, le problème d'affectation des équipages est séparable par catégorie d'équipage et type d'aéronef [18].

Pour chaque vol entrant, nous considérons tous les vols au départ dans les prochaines 48 heures. Ensuite, nous ne considérons que ceux qui sont réalisables selon nos contraintes de masquage afin de filtrer cet ensemble. Nous limitons le nombre maximal de vols possibles à 20, comme il est suffisant dans le secteur du transport aérien, puis prédisons le rang du prochain vol dans cet ensemble de vols partants.

En utilisant le jeu de données construit, nous définissons un CRF par paire (*pairwise-CRF*) sur un graphe général : une structure de réseau basée sur les vols, où les nœuds correspondent aux tâches (vols) ayant été effectués et les arcs représentent la faisabilité de deux vols successifs. Par conséquent, chaque vol f correspond à un nœud n_f. Le nœud n_f est connecté aux nœuds correspondant aux successeurs éventuels (et prédécesseurs) de n_f et l'étiquette de n_f est le rang du prochain vol dans l'ensemble des successeurs possibles triés. Notez que le graphe défini peut être construit comme directionnel. Nous n'explorons pas cette propriété afin d'avoir une matrice symétrique pour les potentiels par paires (utilisant donc $n_{classes} \cdot \frac{n_{classes}+1}{2}$ paramètres pour les potentiels par paires plutôt que $n_{classes} \cdot n_{classes}$).

Description des instances

Nos instances proviennent d'une grande compagnie aérienne et consistent en six solutions de rotations d'équipage mensuelles pour environ 200 villes et environ 50 000 vols par mois. Chaque instance contient une liste de vols d'un mois, les routes des avions et les rotations d'équipage, ainsi qu'une liste des aéroports et des bases. Cette information est utilisée pour créer des données d'entrée pour la phase d'apprentissage. Le jeu de données comprend environ 1 000 paires source-destination différentes, 2 000 codes de vols et rotations différents à partir de 7 bases différentes.

Comme le montre la figure 7.5, cette entrée décrit une rotation qui a lieu les mardi, jeudi, vendredi

et samedi entre le 9 août 2018 et le 3 septembre 2018, à l'exception des dates : 08/10, 08/20 et 08/29. Cette rotation contient des vols entre l'aéroport international de Buffalo Niagara (BUF) et l'aéroport international d'O'Hare (ORD). En plus de ces informations, nous savons que la distance entre les deux aéroports est de 761 km, à partir de laquelle nous pouvons extrapoler la durée du vol, sur la base d'une vitesse moyenne de vol de l'aéronef, ce qui équivaut à environ 1 heure et 26 minutes.

```
O T 1821 0 CC 1 CAT 1 () "a" "" _2_456_
2018/08/09 2018/09/03 X 2018/08/10
2018/08/20 2018/08/29
{
RPT;
ABC05914 ORD 1 02 :10 BUF;
ABC06161 BUF 1 09 :50 ORD;
RLS;
}
```

Figure 7.5 Format de données pour les rotations d'équipage

Résultats de prédiction

Contrairement aux réseaux de neurones où il faudrait exécuter le processus gaussien pour rechercher la meilleure configuration d'hyperparamètres, le prédicteur SCKN est plus stable et nécessite moins d'ajustement pour trouver une bonne architecture car nous avons moins d'hyperparamètres. Pour la couche de prédiction structurée, donc pour BCFW et SDCA, nous fixons le paramètre de régularisation à $C = \frac{1}{\lambda \ n_{samples}} = 1$, donc nous utilisons $\lambda = \frac{1}{n_{samples}}$. De plus, pour SDCA, nous utilisons un échantillonnage de gap avec un taux de non-uniformité de 80% et on utilise l'algorithme de Newton-Raphson pour effectuer la recherche linéaire.

Pour les couches CKN, les hyperparamètres sont le nombre de couches, le nombre de filtres et la taille des patchs. Nous utilisons une couche et effectuons une recherche en grille pour déterminer le nombre de filtres (200) et la taille du patch (5x5). La recherche en grille est effectuée à l'aide de la validation croisée sur le jeu d'entraînement pour mesurer la qualité de la configuration. Nous utilisons différents mois pour différents échantillons afin de simuler le scénario plus réaliste dans lequel nous faisons une prévision sur une nouvelle période. Avec ce choix d'hyperparamètres, nous obtenons une précision de 99.65% pour SCKN-BCFW et de 99.72% pour SCKN-SDCA.

Construction et faisabilité d'une solution initiale

Après la finalisation d'apprentissage des poids du prédicteur SCKN, nous construisons une solution mensuelle que nous pouvons fournir au processus d'optimisation comme solution initiale. En effet, le processus d'optimisation nécessite une solution initiale standard qui est généralement une solution hebdomadaire "cyclique" qui pourra être recopiée à chaque semaine pour couvrir tout le mois, appelée *"Standard - solution initiale"*. Dans cette solution mensuelle, les rotations contenant des vols qui ont disparu ou qui ont changé d'heure sont brisées ; de nouvelles rotations sont alors construites pour couvrir ces vols ainsi que les nouveaux vols apparus.

Au lieu d'utiliser *"Standard - solution initiale"*, nous utilisons le prédicteur SCKN pour construire la solution initiale. On peut utiliser une heuristique gloutonne pour construire les rotations d'équipage. Plus précisément, nous considérons chaque vol que le modèle prédit au début d'une rotation comme un premier vol. Compte tenu de ce vol entrant, nous prédisons le prochain. La rotation prend fin lorsque le nombre maximal de jours autorisé par rotation est atteint. Nous pouvons utiliser l'heuristique ci-dessus pour construire une solution pour les données de test, en obtenant une solution mensuelle qui peut être utilisée à la fois comme solution initiale et comme partition initiale pour le solveur (*DCA clusters*). Malheureusement, si un vol dans la rotation est mal prédit, l'équipage peut finir la rotation loin de la base. Par conséquent, pour toutes les rotations où l'équipage finit loin de la base, nous supprimons tous les vols effectués après la dernière fois que l'équipage arrive à la base. Notez que les rotations peuvent commencer avant la période de l'offre (*bid period*) pour couvrir tous les vols actifs. Par conséquent, en utilisant le prédicteur SCKN, nous proposons des rotations sur une période prolongée pour couvrir le nombre maximal de vols. Cela conduit à une solution mensuelle que nous considérons comme une solution initiale appelée *"SCKN - solution initiale - période prolongée"* . Nous pouvons également limiter les rotations à la période de l'offre. Ceci conduit à une autre solution mensuelle qui peut également être considérée comme une solution initiale appelée *"SCKN - solution initiale - période de l'offre"*.

Par la suite, on gèle la partie légale des rotations contenue dans la solution initiale, en brisant toutes les rotations illégales. Notez que les rotations commençant avant la période de l'offre sont considérées comme illégales, car on voudrait éviter que le solveur génère des rotations avant que la période de l'offre ne commence. En brisant toutes les rotations illégales de *"Standard - solution initiale"*, on obtient une solution mensuelle, appelée *"Standard - réalisable"*, comportant des rotations réalisables, ainsi que des singletons de vols. De même, en brisant toutes les rotations illégales soit à partir de *"SCKN - solution initiale - période prolongée"* ou de *"SCKN - solution initiale - période de l'offre"*, on obtient la même solution mensuelle appelée *"SCKN - réalisable"*.

Pour couvrir les vols devenus non-couverts, nous générons des repositionnements ou DH (*deadheads*) sur tous les vols et utilisons le solveur GENCOL (sans DCA) pour optimiser avec les sous-blocs lé-

gaux imposés couvrant les vols actifs. Dans cette approche, le problème est résolu par une approche à “horizon fuyant”. Parce que le solveur GENCOL (sans DCA) est capable de résoudre quelques milliers de vols par fenêtre, on est contraint à utiliser des fenêtres de deux jours (GENCOL-2 jours). Cela signifie que le mois est divisé en tranches de temps se chevauchant de même longueur. Ensuite, une solution est construite de façon gloutonne dans l’ordre chronologique en résolvant le problème limité à chaque période de temps séquentiellement, en tenant compte de la solution de la période précédente par le biais de contraintes supplémentaires. La solution mensuelle obtenue est ensuite transmise à Commercial-GENCOL-DCA. Les grappes de DCA peuvent être soit extraites de cette solution (comme cela est habituellement fait), soit données séparément (comme c’est le cas dans notre approche). Par conséquent, en fournissant une solution initiale, nous pouvons non seulement accélérer le processus d’optimisation, calculer le nombre de rotations considérés comme légales, le nombre de vols couverts, mais également proposer des grappes de DCA qui sont similaires à la solution initiale, réduisant ainsi le degré d’incompatibilité entre la solution cumulative et les rotations proposées. En utilisant l’approche standard, nous réoptimisons le mois en proposant *“Standard - réalisable”* comme solution initiale à GENCOL-2 jours et obtenons une solution appelée *“GENCOL init”*. De même, en utilisant le prédicteur SCKN, nous proposons *“SCKN - réalisable”* comme solution initiale à GENCOL-2 jours et obtenons une solution appelée *“SCKN + GENCOL”*.

Tableau 7.2 Caractéristiques des solutions mensuelles

	Durée des vols	#vols non-couverts	#vols surcouverts	#vols couverts	#rotations	Coût (non-couverts. + surcouverts. + sol.)	#DH
Standard - solution initiale	81804h36	4 960	3 114	45 929	6 525	37 927 328 596	475
Standard - réalisable	45644h21	25 838	24	25 051	3 226	29 753 057 875	92
GENCOL init	82905h03	4 304	23	46 585	6 951	9 199 014 453	1 718
SCKN - solution initiale - période prolongée	81503h33	5 100	614	45 789	4 916	15 893 302 291	266
SCKN - solution initiale - période de l’offre	75292h22	8 535	599	42 354	4 515	20 294 606 643	241
SCKN - réalisable	70672h27	11 148	0	39 741	4 015	16 45 833 4260	220
SCKN + GENCOL	82886h49	4 313	23	46 576	5 993	9 198 925 708	1360

Les résultats sur la faisabilité et les caractéristiques des solutions initiales sont présentés dans le tableau 7.2. Nous rapportons le temps de vol pour les vols couverts, le nombre de vols non-couverts et surcouverts, le nombre de rotations, le nombre de repositionnements (DH) dans la solution et le coût de la solution mensuelle (coût des vols non-couverts + coût des vols surcouverts + coût de la solution des vols couverts). Notez que le coût de la solution mensuelle est gonflé (*inflated*) et ne représente pas le coût réel de la solution car le coût des vols non-couverts et surcouverts est exprimé en valeurs élevées. Notez aussi que nous effectuons ces expériences sur un mois isolé. Nous ne disposons pas des rotations commencées le mois précédent qui chevauchent sur le début de ce mois. Ainsi plusieurs vols le 1^{er} du mois qui commencent hors d’une base ne peuvent pas être couverts, car les équipages ne peuvent pas les atteindre en partant de leur base durant le mois.

Une situation symétrique apparaît à la fin du mois. Certains vols de la dernière journée du mois ne peuvent pas être couverts par une rotation revenant à une base durant le mois. En fonctionnement normal, ces non-couvertures n'apparaîssent pas, car on dispose de rotations chevauchant sur le début du mois (*carry-in*), et on peut construire des rotations qui chevauchent sur la fin du mois (*carry-out*), car on a dans le réseau les vols de la première semaine du mois suivant.

Notons tout d'abord que la solution initiale standard contient 45 929 vols couverts et 6 525 rotations. Une fois que toutes les rotations illégales ont été brisées, 51% des rotations et 45% des vols couverts sont retirées. La solution *"GENCOL init"* obtenue avec GENCOL-2 jours augmente le nombre de vols couverts à 46 585 et réduit le coût de la solution de 69%.

Ensuite, nous comparons l'approche standard à l'approche proposée utilisant SCKN. La solution initiale sur la période prolongée *"SCKN - solution initiale - période prolongée"* et sur la période de l'offre *"SCKN - solution initiale - période de l'offre"* présentent de meilleurs coûts avec un facteur de réduction de 58% et 47%, respectivement. Une fois que toutes les rotations illégales ont été brisées, seuls 18% des rotations et 13% des vols couverts sont supprimés sur la période prolongée pour la solution *"SCKN - solution initiale - période prolongée"*. Encore plus surprenant, seuls 11% des rotations et 6% des vols couverts sont retirés en utilisant la période de l'offre pour la solution *"SCKN - solution initiale - période de l'offre"*. Une fois que cette solution mensuelle est optimisée à l'aide de GENCOL-2 jours, le nombre de vols couverts est comparable à celui de "GENCOL init", le nombre de repositionnements est inférieur de 21% et le coût de la solution mensuelle est légèrement inférieur. Aussi, comme le coût des vols non-couverts et des vols surcouverts est exprimé en valeurs élevées, la différence du coût de la solution entre les deux approches ne peut pas être mise en évidence par le coût de la solution mensuelle, mais par le coût de la solution des vols couverts, qui est réduit de 20.5%. Nous ferons référence à cette valeur en tant que "coût de la solution" dans les sections suivantes.

Résultats sur les problème de rotations d'équipage

Scénario-Réf En utilisant l'approche d'agrégation de contraintes, Commercial-GENCOL-DCA reçoit les grappes initiales de la solution "GENCOL init" pour résoudre le problème de rotations d'équipage mensuel, obtenant ainsi une solution que nous considérons comme une base de comparaison ou scénario de réference appelée *"Scénario-Réf"*.

NN-Adapt Nous construisons et formons un prédicteur de réseau de neurones tel que décrit dans le chapitre 5. En utilisant le prédicteur entraîné et une heuristique gloutonne telle que décrite dans Soumis et al. [173] et au chapitre 6, nous construisons des rotations qui peuvent être fournies à l'optimiseur comme grappes de DCA initiales. De plus, comme le problème mensuel est résolu en

utilisant une approche à “horizon fuyant” avec des fenêtres d’une semaine et deux jours de chevauchement, la construction de la partition DCA pour chaque fenêtre avant de commencer la résolution peut constituer un défaut majeur. Si une partition DCA initiale est construite pour tout l’horizon avant de commencer la résolution, la partition initiale comportera de nombreuses incohérences par rapport à la solution de la fenêtre précédente, en particulier pendant la période de chevauchement, telles que des vols appartenant à deux grappes différentes. Par conséquent, les grappes proposées sont adaptées à la solution de la fenêtre précédente, de manière à éviter toute incohérence. De plus, au lieu de considérer seulement les vols prédits par le modèle comme début d’une rotation, les grappes incomplètes de la solution dans la fenêtre précédente commençant pendant la période de chevauchement sont également complétées.

S1 et S2 En utilisant “SCKN + GENCOL” comme solution initiale, Commercial-GENCOL-DCA reçoit les grappes initiales fournies par le prédicteur SCKN pour résoudre le problème de rotations d’équipage, obtenant ainsi une solution mensuelle appelée S1, meilleure que “SCKN + GENCOL”. En effet, bien que Commercial-GENCOL-DCA et GENCOL-2 jours aient tous deux accès à toutes les contraintes du CPP, Commercial-GENCOL-DCA utilise l’approche à horizon fuyant avec des fenêtres d’une semaine, au lieu de 2 jours, en réduisant la taille du problème avec l’agrégation dynamique de contraintes. Ensuite, pour chercher davantage une meilleure solution mensuelle, nous utilisons S1 comme solution initiale. Commercial-GENCOL-DCA est alimenté par des grappes initiales fournies par S1 pour résoudre le problème de rotations d’équipage, obtenant ainsi une solution mensuelle appelée S2. L’idée derrière S2 est de surmonter les limites de l’utilisation de “l’horizon fuyant” et d’exploiter la possibilité de trouver une meilleure solution. Dans l’algorithme CG, une solution initiale réalisable est requise pour assurer la faisabilité du problème primal à chaque itération de CG. Le solveur Commercial-GENCOL-DCA utilise une exploration limitée en limitant un voisignage de la solution initiale. Ainsi, si la qualité de la solution initiale est meilleure, la convergence du solveur est également plus rapide et on converge vers une meilleure solution. Ainsi, une solution de haute qualité devrait être générée dans un laps de temps plus court.

Les résultats par fenêtre sont rapportés dans le tableau 7.3 pour tous les algorithmes, à savoir Scénario-Réf, Adapt-NN, S1 et S2. Pour chaque fenêtre et chaque algorithme, nous fournissons la valeur LP au noeud racine de l’arbre de recherche N0 (LP-N0), le temps de calcul à N0 (temps N0), le nombre de variables fractionnaires (# VF-N0) dans la solution MP actuelle à N0, le nombre de noeuds de branchement résolus (# noeuds), la meilleure valeur LP trouvée (Meilleur-LP), le coût de la meilleure solution réalisable (INT) et enfin le temps de calcul total (Temps) ; tous les temps sont en secondes. De plus, pour tous les algorithmes ML, et pour LP-N0, Best-LP et INT, nous indiquons la différence relative entre le résultat obtenu avec cet algorithme et celui avec “Commercial-GENCOL-DCA” (Scénario-Réf).

Tableau 7.3 Résultats de l'optimisation par fenêtre

Fenêtre	Alg.	LP-N0	Diff. (%)	#VF-N0	#Noeuds	Meilleur-LP	Diff. (%)	INT	Diff. (%)	Temps (s)
	Scénario-Réf	10 122 035		2719	159	10 025 487.73		10 276 344.14		7 891
	Adapt-NN	9 904 828	-2.15	2870	203	9 891 390.17	-1.34	10 136 106.73	-1.36	10 691
1	S1	7 099 264	-29.86	2929	200	7 094 683.52	-29.23	7 298 476.40	-28.98	14 855
	S2	2 432 353	-75.97	2413	170	2 423 143.75	-75.83	2 501 388.70	-75.66	14 503
	Scénario-Réf	11 396 498		2519	162	11 225 022.17		11 501 016.42		27 778
	Adapt-NN	10 319 719	-9.45	5848	419	10 215 252.88	-9.00	10 865 255.80	-5.53	94 004
2	S1	7 547 443	-33.77	3134	340	7 430 486.16	-33.80	7 915 983.69	-31.17	41 890
	S2	4 533 093	-60.22	2011	113	4 518 960.16	-59.74	4 595 477.05	-60.04	28 210
	Scénario-Réf	10 740 127		2330	177	10 641 496.00		11 107 990.12		30 421
	Adapt-NN	9 596 326	-10.65	4838	294	9 432 461.69	-11.36	10 051 850.40	-9.51	53 736
3	S1	7 129 557	-33.62	3045	198	6 977 006.00	-34.44	7 400 639.68	-33.38	34 255
	S2	4 588 242	-57.28	1876	83	4 571 315.93	-57.04	4 650 601.49	-58.13	27 003
	Scénario-Réf	9 764 727		2606	229	9 591 592.13		9 968 081.73		33 898
	Adapt-NN	8 063 084	-17.43	4763	394	7 868 177.74	-17.97	8 791 799.94	-11.80	56 291
4	S1	6 427 422	-34.18	3387	183	6 291 641.09	-34.40	6 439 030.07	-35.40	45 430
	S2	4 617 230	-52.72	1938	134	4 601 886.81	-52.02	4 697 567.30	-52.87	27 737
	Scénario-Réf	8 095 063		3150	188	7 899 149.01		8 102 703.93		35 948
	Adapt-NN	6 076 461	-24.94	5333	417	5 953 932.68	-24.63	6 453 605.80	-20.35	74 622
5	S1	5 383 225	-33.50	3038	227	5 280 707.35	-33.15	5 560 969.73	-31.37	44 548
	S2	4 331 589	-46.49	1803	89	4 323 521.96	-45.27	4 386 135.29	-45.87	25 196
	Scénario-Réf	6 778 925		2513	187	6 610 562.32		6 939 096.41		29 368
	Adapt-NN	4 763 520	-29.73	4940	318	4 697 990.54	-28.93	4 947 363.87	-28.70	55 263
6	S1	5 074 211	-25.15	3222	278	5 003 421.94	-24.31	5 368 766.13	-22.63	51 166
	S2	4 356 910	-35.73	1961	126	4 349 221.21	-34.21	4 491 319.98	-35.28	26 529
	Scénario-Réf	9 482 896		2640	184	9 332 218.23		9 649 205.46		27 551
	Adapt-NN	8 120 656	-14.37	4765	341	8 009 867.62	-14.17	8 540 997.09	-11.48	57 435
Moyenne	S1	6 443 520	-32.05	3126	238	6 346 324.34	-32.00	6 663 977.62	-30.94	38 691
	S2	4 143 236	-56.31	2000	119	4 131 341.64	-55.73	4 220 414.97	-56.26	24 863

Nous commençons par comparer Scénario-Réf, Adapt-NN et S1. Notons d'abord que Adapt-NN donne de meilleures valeurs LP-N0 et Meilleur-LP avec un facteur de réduction moyen de 14.37% et 14.17% respectivement, par rapport aux résultats de réferences Scénario-Réf, tandis que S1 donne de meilleures réductions des valeurs LP-N0 et Meilleur-LP avec un facteur de réduction de 32.05% et 32% respectivement. Il en va de même pour le coût de la meilleure solution réalisable (INT), où Adapt-NN a un facteur de réduction moyen de 11.48% tandis que S1 donne de meilleures réductions avec un facteur de réduction moyen de 30.94%. Alors que le schéma d'adaptation est nécessaire pour les heuristiques NN pour proposer des grappes adaptées à la solution de la fenêtre précédente trouvée par l'optimiseur, il n'est pas nécessaire pour les heuristiques basées sur SCKN qui proposent à la fois une solution initiale et des grappes de DCA. En effet, même s'il existe des incohérences entre la solution mensuelle cumulative et les grappes proposées, elles sont limitées puisqu'elles sont principalement dues à la rupture et à la fusion des grappes par l'optimiseur, alors que la plupart des incohérences dans le cas des heuristiques NN sont dues à une solution initiale standard.

D'autre part, notons que Adapt-NN a un temps de calcul en moyenne supérieur de 108% par rapport à celui de Scénario-Réf, S1 a un temps de calcul seulement 40% supérieur en moyenne. Cette augmentation du temps est due au nombre de variables fractionnaires au noeud racine de l'arbre de recherche avec un facteur d'augmentation de 81% pour Adapt-NN et de 18% pour S1, par rapport à Scénario-Réf. Lorsque les contraintes de base sont restrictives, les solutions de noeud racine contiennent un nombre plus important de variables fractionnaires afin de répartir le temps de travail entre les bases uniformément. Cela entraîne une augmentation du nombre de nœuds de branchement nécessaires pour obtenir une bonne solution entière.

Alors que l'augmentation du temps dans Adapt-NN est due au grand nombre de variables fractionnaires au nœud racine de l'arbre de recherche, causé par des incohérences entre la solution initiale et les grappes de DCA, S1 a un nombre de nœuds et un temps de calcul inférieurs à Adapt-NN pour deux raisons. Premièrement, le processus de prédiction n'est pas un processus glouton, dans le sens où les liens (*connections*) entre les vols sont prédits conjointement. Cela rend la solution plus réalisable, comme le montre la section 7.5.3. Deuxièmement, et contrairement aux heuristiques NN, nous utilisons non seulement les rotations proposées en tant que grappes de DCA, mais également en tant que solution initiale.

Ensuite, nous comparons Scénario-Réf et S2. Notons d'abord que S2 donne de meilleures valeurs LP-N0 et Meilleur-LP pour toutes les fenêtres avec un facteur de réduction moyen de 56.31% et 55.73% respectivement. De même, S2 donne de meilleures solutions réalisables avec un facteur de réduction de 56.26%. Ceci s'explique par la capacité de S2 à proposer une meilleure solution initiale, meilleure que celles utilisées comme ensemble d'apprentissage pour le prédicteur SCKN. D'autre part, en moyenne, le temps de calcul pour S2 est inférieur de 10% à celui de Scénario-Réf, en raison d'un nombre faible de noeuds et d'un nombre faible de variables fractionnaires au noeud racine de l'arbre de branchement, soit respectivement 8.15% et 24.24% inférieurs, en moyenne.

Les résultats pour les solutions mensuelles obtenues à la fin du processus d'optimisation sont rapportés dans le tableau 7.4. Nous indiquons le coût total de la solution, le coût des contraintes globales et le nombre de repositionnements. Pour tous les algorithmes heuristiques, nous indiquons également la différence relative entre les résultats obtenus avec Commercial-GENCOL-DCA et ceux obtenus avec Adapt-NN, S1 et S2.

Nous commençons par comparer GENCOL init et Scénario-Réf. Notons d'abord qu'une solution nettement meilleure a été trouvée avec Commercial-GENCOL-DCA, par rapport à la solution initiale GENCOL init. En effet, le solveur a considérablement réduit le coût de la solution et le coût des contraintes globales avec un facteur de réduction de 32.73% et 77.53% respectivement, tout en réduisant de 42.49% le nombre de repositionnements (DH). Ceci témoigne de la capacité de l'optimiseur à résoudre des problèmes de grande taille, en trouvant de meilleures solutions et en

Tableau 7.4 Résultats de l'optimisation pour la solution finale

	Coût de la solution	Diff. vs Scénario-Réf (%)	Coût des contraintes globales	Diff. vs Scénario-Réf (%)	#DH
GENCOL init	30 681 120.5		9 465 982.28		1725
Scénario-Réf	20 639 814.6	-32.73 (vs GENCOL init)	2 127 086.77	-77.53 (vs GENCOL init)	992
Adapt-NN	18 881 977.89	-8.52	465 687.94	-78.11	1014
S1	18 677 641.1	-9.51	420 095.87	-80.25	915
S2	17 145 543.4	-16.93	58 797.21	-97.24	583

améliorant des solutions à l'échelle industrielle.

Ensuite, nous comparons Scénario-Réf, Adapt-NN et S1. Alors que Adapt-NN surpasse Scénario-Réf en réduisant le coût de la solution et le coût des contraintes globales de 8.52% et 78.11%, S1 surpasse Scénario-Réf et Adapt-NN en réduisant le coût de la solution et le coût des contraintes globales de 9.51% et 80.25%, respectivement. De plus, alors que S1 réduit le nombre de repositionnements dans la solution de 7.76%, notez que pour Adapt-NN, le nombre de repositionnements dans la solution est légèrement supérieur, avec un facteur d'augmentation de 2.17%, par rapport à Scénario-Réf. Le coût des repositionnements est comptabilisé dans le coût de la solution. Parce que le coût de la solution est une fonction multi-objectifs, nous pensons que l'utilisation d'un peu plus de repositionnements a permis d'obtenir de meilleures solutions, améliorant à la fois le coût de la solution et le coût des contraintes globales.

Ensuite, nous comparons Scénario-Réf et S2. Les solutions trouvées par S2 présentent de meilleures statistiques que Scénario-Réf, Adapt-NN et S1, avec un facteur de réduction du coût de la solution et du coût des contraintes globales de 16.93% et 97.24%, respectivement. Encore plus intéressant, le nombre de repositionnements utilisé est réduit de 41.23% par rapport à Scénario-Réf. Ceci atteste tout d'abord que la solution mensuelle S1 peut encore être optimisée et que des recherches plus poussées pour éviter l'approche à "horizon fuyant" et utiliser une fenêtre d'un mois peuvent présenter de meilleurs résultats que la version actuelle de Commercial-GENCOL-DCA. De plus, les solutions mensuelles utilisées pour génerer l'ensemble d'apprentissage peuvent être remplacées par différentes solutions obtenues avec la nouvelle approche. Ceci permettrait non seulement d'obtenir des solutions initiales meilleures et plus réalisables, mais également de meilleures solutions mensuelles après l'optimisation que S1.

7.6 Conclusion

Nous introduisons un nouveau prédicteur structuré profond combinant la haute performance des méthodes d'apprentissage profond, la flexibilité non paramétrique des méthodes du noyau et les prédicteurs structurés. Nous montrons que l'utilisation de cette combinaison de manière supervisée

surpasse les méthodes de pointe en termes de sous-optimalité primale et de précision du test sur le jeu de données OCR.

Ensuite, nous appliquons la méthode proposée sur un jeu de données de connexions de vols FCD (*Flight-Connection Dataset*) afin de proposer de bonnes solutions initiales au solveur pour résoudre un problème de rotations d'équipage mensuel. Nous transformons le problème de rotations en un problème de prédiction de connexion de vols et formons un graphe CRF général, où les noeuds correspondent aux coordonnées spatio-temporelles et les arcs représentent les tâches effectuées par les membres d'équipage. Une telle approche est capable d'incorporer les contraintes globales et locales dans le processus d'apprentissage du prédicteur et fournit une solution mensuelle. La solution proposée est "plus réalisable" que la solution initiale standard qui est géneralement une solution hebdomadaire cyclique recopiée pour couvrir tout le mois puis réparée pour enlever les vols disparus et ajouter les nouveaux vols apparus. Nous utilisons donc la solution proposée à la fois comme solution initiale et comme grappes initiales au processus d'optimisation. Nous avons comparé les performances de Commercial-GENCOL-DCA, soit en utilisant la solution initiale proposée et les grappes initiales, soit en utilisant une solution initiale standard. Les principaux résultats montrent que l'utilisation de l'approche proposée donne de meilleurs résultats réduisant de 9.51% le coût de la solution et de 80.25% le coût des contraintes globales. En outre, l'utilisation de la solution obtenue pour relancer le processus d'optimisation donne de meilleurs résultats, réduisant encore le coût de la solution et fournissant une solution avec un coût des contraintes globales très négligeable et un nombre de repositionnements beaucoup plus faible. En effet, nous commençons le processus d'optimisation avec une solution mensuelle optimisée et réalisable. En fait, comme le coût de la solution est une fonction multi-objectifs, et parce que la solution initiale a un coût de contraintes globales déjà négligeable, le solveur peut se concentrer sur l'exploration du voisinage des solutions permettant de réduire le coût réel, par opposition au coût des contraintes globales.

La solution proposée au problème de rotations d'équipage est capable de traiter des problèmes plus grands et d'apprendre de plusieurs compagnies aériennes à construire des grappes initiales pour une nouvelle compagnie, puisqu'elle n'utilise les codes des vols dans aucune partie du processus de prétraitement, d'entraînement ou de prévision. En plus, la méthode proposée peut facilement être adaptée à d'autres types de problèmes d'optimisation, tels que la planification des horaires de trains ou de bus. Enfin, pour utiliser l'approche proposée à son plein potentiel, notre objectif à long terme est d'intégrer le prédicteur d'apprentissage machine en tant que composant standard du processus d'optimisation, tel que décrit par Bengio et al. [38]. Similaire à l'approche utilisée dans l'algorithme AlphaGO pour jouer au jeu de Go [174, 175], on peut utiliser le temps d'inactivité des ordinateurs au cours du mois pour entraîner le prédicteur ML, puis utiliser Commercial-GENCOL-DCA pour obtenir différentes bonnes solutions en ajustant les paramètres d'optimisation, puis utiliser ces solutions pour re-entraîner le prédicteur et proposer de nouvelles meilleures solutions initiales. Ces

itérations entre la production de meilleures solutions et l'entraînement du prédicteur ML ont un potentiel d'amélioration important. Les résultats des trois chapitres utilisent un entraînement sur des solutions historiques, produites avec la génération précédente du système d'optimisation. Le nouveau système produit de bien meilleures solutions. On peut donc entraîner le prédicteur sur ces meilleures solutions pour obtenir des solutions encore meilleures. De plus, le temps de résolution devrait être réduit car l'optimiseur aura moins de travail à faire en partant avec une meilleure information.

CHAPITRE 8 DISCUSSION GÉNÉRALE

Dans cette thèse, nous avons étudié le CPP essentiellement modélisé comme un problème de partitionnement d'ensemble, à résoudre avec la méthode CG. Étant donné sa complexité due au grand nombre de rotations possibles et à la complexité des contraintes, nous utilisons l'agrégation dynamique des contraintes pour réduire le nombre de contraintes dans le problème maître restreint. Toutefois, DCA a besoin d'une partition initiale de vols en grappes (*clusters*) qui peut être donnée par une heuristique ou en résolvant une approximation du problème.

L'objectif principal de ce travail était d'utiliser l'information contenue dans des mois précédents pour construire des grappes initiales DCA pour le mois courant. Le défi que nous avons relevé était alors de proposer de nouvelles méthodes ML, tirant profit des avantages de la structure du problème et des caractéristiques de l'optimiseur adaptées.

Pour ce faire, nous avons d'abord transformé le problème de construction de rotations en deux sous-problèmes : identifier si un vol est le début d'une rotation et identifer pour chaque vol entrant ce que l'équipage va faire par la suite (pause, vol ou fin de rotation). Puisque la fin d'une rotation dépend du nombre maximum de jours permis par la compagnie aérienne, nous considérons ce nombre de jours maximal comme hyperparamètre. En d'autres termes, il n'est pas nécessaire de prédire la fin de la rotation. Par conséquent, pour construire les rotations, nous proposons de décomposer le deuxième sous-problème en deux sous-problèmes. Le premier consiste à prédire si l'équipage prendra un repos de fin de journée (*layover*) ; le second consiste à prédire le prochain vol, en supposant que l'équipage ne sera pas en repos. D'une part, les escales sont fortement corrélées à la durée de la connexion, bien qu'il n'y ait pas de seuil fixe pour le nombre d'heures qu'un équipage doit rester dans un aéroport pour prendre un repos. Cependant, prédire le prochain vol est un problème de prédiction beaucoup plus complexe. En effet, pour chaque vol entrant, il peut y avoir des centaines de vols au départ dans l'heure qui suit et des milliers de vols pour les 48 prochaines heures. C'est ce qu'on appelle le problème de connexion de vol.

Dans un premier volet de ce travail, pour produire des grappes initiales pour DCA, nous utilisons des techniques d'apprentissage machine pour déterminer quelles variables de connexion de vols sont susceptibles d'être égales à un. Comme les réseaux de neurones ont montré un grand potentiel pour extraire des caractéristiques significatives d'entrées complexes et obtenir de bonnes précisions à travers différentes tâches de classification, nous proposons de les utiliser et de comparer leurs résultats avec les méthodes classiques d'apprentissage machine. Dans les deux cas, l'entrée contient des informations sur le vol précédent ainsi que sur les vols suivants. Une fois les poids des réseaux de neurones entraînés, et pour éviter qu'un équipage finisse la rotation loin de la base, nous utilisons

des heuristiques qui construisent les rotations à partir des probabilités données par les réseaux de neurones. Ces grappes ne représentent pas des solutions réalisables, mais DCA les répare avec la phase 1 et la phase 2 de l'algorithme du simplexe pour atteindre une bonne solution réalisable. Encore plus, au lieu de tenir compte de toutes les prédictions, nous envisageons d'augmenter notre classificateur avec une option "abstention". Puisque le solveur prend plus de temps pour rompre une connexion dans une grappe que pour en construire une, il peut être avantageux de supprimer les liens à faible confiance en utilisant cette option "abstention".

Ainsi, au lieu d'effectuer le processus d'optimisation standard qui dure 45 heures au total, notre méthode proposée donne de meilleurs coûts après quelques secondes pour prédire les connexions de vol, et des résultats optimisés après cinq heures. Cette approche permet donc de réduire considérablement le temps de résolution sans perdre en qualité. Cependant, nous constatons que les heuristiques que nous proposons pour construire les rotations d'équipage sont limitées, même si elle donne de meilleurs coûts. Encore plus, il y a nécessité de tester la nouvelle approche sur un problème mensuel.

C'est ainsi que nous nous sommes intéressés dans un deuxième volet de ce travail à combiner plusieurs méthodes mises en œuvre, développées et testées sur de petits ensembles de données, afin d'obtenir un algorithme efficace pour le CPP de grande taille. Comme GENCOL ne peut traiter plus que quelques milliers de vols, nous proposons Commercial-GENCOL-DCA, un nouveau solveur pour le problème de partitionnement d'ensemble combinant l'aggrégation de contraintes (DCA et MPDCA) et l'algorithme de simplexe primal amélioré (IPS). Ensuite, nous utilisons l'apprentissage machine pour proposer une bonne partition initiale pour un CPP de grande taille : des problèmes mensuels comportant jusqu'à 50 000 vols. Un nouvel algorithme combinant plusieurs techniques avancées de recherche opérationnelle sera utilisé pour assembler et modifier ces grappes, au besoin, afin de produire une bonne solution. Cette nouvelle approche, en commençant par l'apprentissage machine et en terminant l'optimisation par la programmation mathématique, permettra de résoudre des problèmes globalement plus importants et d'éviter la perte d'optimalité résultant de la décomposition heuristique en petites tranches de temps dans l'approche à horizon fuyant.

Parce que le problème mensuel est résolu en utilisant une approche à horizon fuyant avec des fenêtres d'une semaine et deux jours de période de chevauchement, la construction de la partition DCA pour chaque fenêtre et la provision de ces partitions avant résolution peut être un défaut majeur. Si une partition DCA initiale est construite pour l'ensemble de l'horizon avant de commencer la résolution, la partition initiale présentera de nombreuses incohérences avec la solution de la fenêtre précédente, en particulier pendant la période de chevauchement. Nous adaptons les grappes proposées à la solution de la fenêtre précédente en utilisant l'heuristique pour construire les grappes de la fenêtre courante conformément à la solution trouvée pour la fenêtre précédente et

toute incohérence avec ladite fenêtre est évitée, donc la partition proposée est adaptée à la résolution courante. De plus, au lieu de considérer les vols que le modèle prédit au début d'une rotation comme un premier vol, les grappes incomplètes de la solution de la fenêtre précédente commençant pendant la période de chevauchement sont complétées.

Les principaux résultats de calcul montrent que Commercial-GENCOL-DCA donne de meilleurs résultats que les solutions utilisées en industrie par les compagnies aériennes. En outre, l'heuristique basée sur l'apprentissage machine, utilisant soit l'abstention ou l'adaptation donne de meilleurs résultats avec des coûts nettement plus faibles, réduisant de 8.52% le coût de la solution et de 78.11% le coût des contraintes globales.

Cette amélioration s'accompagne d'une augmentation du temps de calcul. En effet, un plus grand nombre de variables fractionnaires au noeud racine de l'arbre de branchement peut être expliqué par les contraintes de base restrictives ainsi que l'incohérence entre la solution initiale standard utilisée et les grappes proposées par l'heuristique de construction basée sur le prédicteur. En effet, dans l'algorithme CG, une solution initiale réalisable est nécessaire pour assurer la faisabilité du problème primal à chaque itération de CG.

Pour réduire les temps de calcul, et parce que l'incohérence entre la solution initiale et les grappes cause un grand nombre de variables fractionnaires, nous cherchons dans le troisième volet de ce travail à construire un prédicteur capable d'intégrer les contraintes locales dans l'étape d'entraînement. Les rotations construites servent aussi bien comme solution initiale au processus d'optimisation que comme une partition initiale pour DCA.

Nous introduisons les réseaux à noyaux convolutifs structurés, ou SCKN, qui combinent les propriétés des architectures d'apprentissage profond, la flexibilité non paramétrique des méthodes du noyau et les prédicteurs structurés. Le CKN apprend à approximer la carte des caractéristiques du noyau sur les données de formation. Il peut être interprété comme un type particulier de CNN. Nous montrons que cette approche alternée offre un grand potentiel dans un cadre d'apprentissage supervisé/couplé. Plus précisément, nous montrons que l'utilisation de cette approche de façon supervisée surpasse les méthodes de pointe en termes de sous-optimalité primale et de précision du test sur l'ensemble de données OCR et appliquons cette méthode au problème de prédiction des connexions de vol. On montre que l'utilisation de cette approche donne des meilleurs résultats en réduisant de 9.51% le coût de la solution et de 80.25% le coût des contraintes globales.

En outre, l'utilisation de la solution mensuelle obtenue pour relancer le processus d'optimisation donne de meilleurs résultats, réduisant le coût de la solution et fournissant une solution avec un coût très négligeable des contraintes globales et un nombre beaucoup plus réduit de repositionnements, puisque le processus d'optimisation est initialisé avec une solution mensuelle optimisée et réalisable.

CHAPITRE 9 CONCLUSION ET RECOMMANDATIONS

La recherche effectuée dans le cadre de cette thèse a été fructueuse dans le sens que les résultats obtenus montrent clairement que l'approche proposée de commencer par l'apprentissage machine et finir par la programmation mathématique fixée au début de cette thèse permet d'avoir de meilleures solutions en beaucoup moins de temps que les méthodes traitant de plus petits problèmes dans chaque fenêtre de l'horizon fuyant.

Ce paradigme ouvre de nouvelles pistes de recherche de grande envergure. On peut implémenter et tester d'autres heuristiques de construction de rotations à partir des probabilités produites par le prédicteur ainsi que tester d'autres algorithmes de prédiction structurée dans le prédicteur SCKN que nous proposons. La solution proposée au problème de rotations d'équipage est capable de traiter des problèmes plus importants et d'apprendre de plusieurs compagnies aériennes à construire des grappes initiales pour une nouvelle compagnie. En plus, la méthode proposée peut facilement être adaptée à d'autres types de problèmes d'optimisation, tels que le problème d'horaires personnalisés ou la planification des horaires de trains ou de bus. On peut aussi étendre les approches et méthodes proposées à d'autres algorithmes d'optimisation, tel que les méthodes primales de génération de colonnes en nombres entiers.

Enfin, pour utiliser l'approche proposée à son plein potentiel, notre objectif à long terme est d'intégrer le prédicteur d'apprentissage machine en tant que composant standard du processus d'optimisation. Comme les ordinateurs qui produisent les solutions pour les compagnies aériennes travaillent seulement une semaine par mois, on pourrait les utiliser pour perfectionner le processus. On pourrait utiliser Commercial-GENCOL-DCA pour obtenir différentes bonnes solutions en ajustant les paramètres d'optimisation, puis utiliser cette banque de solutions pour re-entraîner le prédicteur et proposer de nouvelles meilleures solutions initiales pour le mois suivant et de meilleures solutions finales. Cette boucle de production de bonnes solutions, d'apprentissage sur ces solutions pour obtenir un meilleur prédicteur et de l'utiliser pour produire de meilleures solutions est semblable à l'approche utilisée dans l'algorithme AlphaGO pour jouer au jeu de Go et battre le champion mondial.

RÉFÉRENCES

[1] R. Anbil, R. Tanga et E. L. Johnson, “A global approach to crew-pairing optimization,” *IBM Systems Journal*, vol. 31, n°. 1, p. 71–78, 1992.

[2] R. A. Rushmeier et S. A. Kontogiorgis, “Advances in the optimization of airline fleet assignment,” *Transportation science*, vol. 31, n°. 2, p. 159–169, 1997.

[3] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh et P. H. Vance, “Branch-and-price : Column generation for solving huge integer programs,” *Operations research*, vol. 46, n°. 3, p. 316–329, 1998.

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

[5] A. M. Cohn et C. Barnhart, “Improving crew scheduling by incorporating key maintenance routing decisions,” *Operations Research*, vol. 51, n°. 3, p. 387–396, 2003.

[6] M. Deveci et N. Ç. Demirel, “A survey of the literature on airline crew scheduling,” *Engineering Applications of Artificial Intelligence*, vol. 74, p. 54–69, 2018.

[7] I. Elhallaoui, D. Villeneuve, F. Soumis et G. Desaulniers, “Dynamic aggregation of set-partitioning constraints in column generation,” *Operations Research*, vol. 53, n°. 4, p. 632–645, 2005.

[8] I. Elhallaoui, A. Metrane, F. Soumis et G. Desaulniers, “Multi-phase dynamic constraint aggregation for set partitioning type problems,” *Mathematical Programming*, vol. 123, n°. 2, p. 345–370, 2010.

[9] I. Elhallaoui, G. Desaulniers, A. Metrane et F. Soumis, “Bi-dynamic constraint aggregation and subproblem reduction,” *Computers & Operations Research*, vol. 35, n°. 5, p. 1713–1724, 2008.

[10] G. B. Dantzig et P. Wolfe, “Decomposition principle for linear programs,” *Operations research*, vol. 8, n°. 1, p. 101–111, 1960.

[11] E. Andersson, E. Housos, N. Kohl et D. Wedelin, “Crew pairing optimization,” dans *Operations Research in the Airline Industry*, G. Yu, édit. vol 9. Springer, Boston, MA : International Series in Operations Research & Management Science, 1998.

[12] R. Freling, R. M. Lentink et A. P. Wagelmans, “A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm,” *Annals of Operations Research*, vol. 127, n°. 1-4, p. 203–222, 2004.

[13] O. Günlük, L. Ladányi et S. De Vries, “A branch-and-price algorithm and new test problems for spectrum auctions,” *Management Science*, vol. 51, n°. 3, p. 391–406, 2005.

[14] J. O. Brunner et J. F. Bard, “Flexible weekly tour scheduling for postal service workers using a branch and price,” *Journal of Scheduling*, vol. 16, n°. 1, p. 129–149, 2013.

[15] S. Ceria, P. Nobili et A. Sassano, “A lagrangian-based heuristic for large-scale set covering problems,” *Mathematical Programming*, vol. 81, n°. 2, p. 215–228, 1998.

[16] A. Caprara, M. Fischetti et P. Toth, “A heuristic method for the set covering problem,” *Operations research*, vol. 47, n°. 5, p. 730–743, 1999.

[17] J.-Q. Li, P. B. Mirchandani et D. Borenstein, “A lagrangian heuristic for the real-time vehicle rescheduling problem,” *Transportation Research Part E : Logistics and Transportation Review*, vol. 45, n°. 3, p. 419–433, 2009.

[18] A. Kasirzadeh, M. Saddoune et F. Soumis, “Airline crew scheduling : models, algorithms, and data sets,” *EURO Journal on Transportation and Logistics*, vol. 6, n°. 2, p. 111–137, 2017.

[19] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon et F. Soumis, “Crew pairing at air france,” *European journal of operational research*, vol. 97, n°. 2, p. 245–259, 1997.

[20] S. Qiu, “Airline crew pairing optimization problems and capacitated vehicle routing problems,” Thèse de doctorat, Georgia Institute of Technology, 2012.

[21] B. Zeren et I. Özkol, “A novel column generation strategy for large scale airline crew pairing problems,” *Expert Systems with Applications*, vol. 55, p. 133–144, 2016.

[22] M. Saddoune, G. Desaulniers et F. Soumis, “Aircrew pairings with possible repetitions of the same flight number,” *Computers & Operations Research*, vol. 40, n°. 3, p. 805–814, 2013.

[23] R. E. Marsten et F. Shepardson, “Exact solution of crew scheduling problems using the set partitioning model : Recent successful applications.” *Networks*, vol. 11, p. 165–177, 1981.

[24] V. Cacchiani et J.-J. Salazar-González, “A heuristic approach for an integrated fleet-assignment, aircraft-routing and crew-pairing problem,” *Electronic Notes in Discrete Mathematics*, vol. 41, p. 391–398, 2013.

[25] M. Rasmussen, R. Lusby, D. Ryan et J. Larsen, “Subsequence generation for the airline crew pairing problem,” *51st AGIFORS Annual Proceedings*, vol. 1, 01 2011.

[26] E. K. Baker, “Efficient heuristic algorithms for the weighted set covering problem,” *Computers & Operations Research*, vol. 8, n°. 4, p. 303–310, 1981.

[27] L. Bianco, M. Bielli, A. Mingozzi, S. Ricciardelli et M. Spadoni, "A heuristic procedure for the crew rostering problem," *European journal of operational research*, vol. 58, n°. 2, p. 272–283, 1992.

[28] S. Voß, *Meta-Heuristics : Advances and Trends in Local Search Paradigms for Optimization*. USA : Kluwer Academic Publishers, 1999.

[29] J. Desrosiers, F. Soumis et M. Desrochers, "Routing with time windows by column generation," *Networks*, vol. 14, n°. 4, p. 545–565, 1984.

[30] J. Desrosiers, Y. Dumas, M. M. Solomon et F. Soumis, "Chapter 2 time constrained routing and scheduling," dans *Handbooks in Operations Research and Management Science*. Canada : Elsevier, 1995, p. 35–139.

[31] E. Khalil, H. Dai, Y. Zhang, B. Dilkina et L. Song, "Learning combinatorial optimization algorithms over graphs," dans *Advances in Neural Information Processing Systems*, 2017, p. 6348–6358.

[32] O. Vinyals, M. Fortunato et N. Jaitly, "Pointer networks," dans *Advances in Neural Information Processing Systems*, 2015, p. 2692–2700.

[33] M. Kruber, M. E. Lübbecke et A. Parmentier, "Learning when to use a decomposition," dans *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, 2017, p. 202–210.

[34] A. Lodi, L. Mossina et E. Rachelson, "Learning to handle parameter perturbations in combinatorial optimization : an application to facility location," *arXiv preprint arXiv :1907.05765*, 2019.

[35] H. He, H. Daume III et J. M. Eisner, "Learning to search in branch and bound algorithms," dans *Advances in neural information processing systems*, 2014, p. 3293–3301.

[36] E. B. Khalil, P. Le Bodic, L. Song, G. Nemhauser et B. Dilkina, "Learning to branch in mixed integer programming," dans *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[37] R. Václavík, A. Novák, P. Šůcha et Z. Hanzálek, "Accelerating the Branch-and-Price Algorithm Using Machine Learning," *European Journal of Operational Research*, vol. 271, n°. 3, p. 1055 – 1069, 2018.

[38] Y. Bengio, A. Lodi et A. Prouvost, "Machine learning for combinatorial optimization : a methodological tour d'horizon," *arXiv preprint arXiv :1811.06128*, 2018.

[39] S. M. Elsayed, R. A. Sarker et D. L. Essam, "A new genetic algorithm for solving optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 27, p. 57–69, 2014.

[40] M. Akbari, H. Rashidi et S. H. Alizadeh, “An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,” *Engineering Applications of Artificial Intelligence*, vol. 61, p. 35–46, 2017.

[41] V. Graf, D. Teichmann et M. Dorda, “Mathematical model for charter flights planning at airports with high air traffic volume,” *Communications-Scientific letters of the University of Zilina*, vol. 18, n°. 3, p. 22–27, 2016.

[42] M.-W. Tsai, T.-P. Hong et W.-T. Lin, “A two-dimensional genetic algorithm and its application to aircraft scheduling problem,” *Mathematical Problems in Engineering*, vol. 2015, 2015.

[43] P. K. Pandit, M. Hasin et A. Akhtar, “Business model of aircraft fleet planning using ann,” dans *Hamburg International Conference of Logistics (HICL) 2018*. epubli, 2018, p. 221–247.

[44] B. Zeren et İ. Özkol, “An improved genetic algorithm for crew pairing optimization,” *Journal of Intelligent Learning Systems and Applications*, vol. 4, n°. 01, p. 70, 2012.

[45] A. Azadeh, M. H. Farahani, H. Eivazy, S. Nazari-Shirkouhi et G. Asadipour, “A hybrid meta-heuristic algorithm for optimization of crew scheduling,” *Applied Soft Computing*, vol. 13, n°. 1, p. 158–164, 2013.

[46] M. Deveci et N. Ç. Demirel, “Evolutionary algorithms for solving the airline crew pairing problem,” *Computers & Industrial Engineering*, vol. 115, p. 389–406, 2018.

[47] X. Xu et G. Yang, “Robust manifold classification based on semi supervised learning,” *International Journal of Advancements in Computing Technology*, vol. 5 (8), p. 174–183, 2013.

[48] N. Alajlan, Y. Bazi, F. Melgani et R. R. Yager, “Fusion of supervised and unsupervised learning for improved classification of hyperspectral images,” *Information Sciences*, vol. 217, p. 39–55, 2012.

[49] J. D. A. Silva et E. R. Hruschka, “An experimental study on the use of nearest neighbor based imputation algorithms for classification tasks,” *Data and Knowledge Engineering*, vol. 84, p. 47–58, 2013.

[50] C. J. Stone, J. H. Friedman, L. Breiman et R. A. Olshen, *Classification and Regression Trees*. Belmont, California : Wadsworth International Group, 1984.

[51] J. R. Quinlan, *C4. 5 : programs for machine learning*. Elsevier, 2014.

[52] L. Breiman, “Random forests, mach,” *Learn*, vol. 45, p. 532, 2001.

[53] J. Pearl, “Reverend bayes on inference engines : A distributed hierarchical approach,” *AAAI*, vol. 82, p. 133–136, 1982.

[54] T. Evgeniou et M. Pontil, "Support vector machines : Theory and applications," dans *Advanced Course on Artificial Intelligence*. Springer, 1999, p. 249–257.

[55] N. Sharma et L. M. Aggarwal, "Automated medical image segmentation techniques," *Journal of medical physics/Association of Medical Physicists of India*, vol. 35, n°. 1, p. 3, 2010.

[56] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau et S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, n°. 7639, p. 115, 2017.

[57] Y. LeCun, Y. Bengio et G. Hinton, "Deep learning," *nature*, vol. 521, n°. 7553, p. 436, 2015.

[58] O. Ronneberger, P. Fischer et T. Brox, "U-net : Convolutional networks for biomedical image segmentation," dans *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, p. 234–241.

[59] K. Simonyan et A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv :1409.1556*, 2014.

[60] D. E. Rumelhart, G. E. Hinton et R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, n°. 6088, p. 533–536, 1986.

[61] J. Duchi, E. Hazan et Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, n°. Jul, p. 2121–2159, 2011.

[62] T. Tieleman et G. Hinton, "Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude," *COURSERA : Neural networks for machine learning*, vol. 4, n°. 2, p. 26–31, 2012.

[63] D. P. Kingma et J. Ba, "Adam : A method for stochastic optimization," *arXiv preprint arXiv :1412.6980*, 2014.

[64] J. Mairal, P. Koniusz, Z. Harchaoui et C. Schmid, "Convolutional kernel networks," *Advances in Neural Information Processing Systems (NIPS)*, vol. 2014, 2014.

[65] J. Mairal, "End-to-end kernel learning with supervised convolutional networks," *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[66] A. Bietti et J. Mairal, "Group invariance, stability to deformations, and complexity of deep convolutional representations," *The Journal of Machine Learning Research*, vol. 20, n°. 1, p. 876–924, 2019.

[67] L. Bo, K. Lai, X. Ren et D. Fox, "Object recognition with hierarchical kernel descriptors," dans *CVPR 2011*. IEEE, 2011, p. 1729–1736.

[68] C. K. Williams et M. Seeger, "Using the nyström method to speed up kernel machines," dans *Advances in neural information processing systems*, 2001, p. 682–688.

[69] K. Zhang, I. W. Tsang et J. T. Kwok, “Improved nyström low-rank approximation and error analysis,” dans *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, p. 1232–1239.

[70] J. Lafferty, A. McCallum et F.Pereira, “Conditional random fields : Probabilistic models for segmenting and labelling sequence data,” *Proceedings of the Eighteenth International Conference on Machine Learning*, p. 28–289, 2001.

[71] F. Sha et F. Pereira, “Shallow parsing with conditional random fields,” dans *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2003, p. 213–220. [En ligne]. Disponible : https://www.aclweb.org/anthology/N03-1028

[72] J. R. Finkel, A. Kleeman et C. D. Manning, “Efficient, feature-based, conditional random field parsing,” dans *Proceedings of ACL-08 : HLT*, 2008, p. 959–967.

[73] S. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, K. P. Murphy *et al.*, “Accelerated training of conditional random fields with stochastic gradient methods,” dans *ICML*, vol. 6. Citeseer, 2006, p. 969–976.

[74] M. Collins, A. Globerson, T. Koo, X. Carreras et P. L. Bartlett, “Exponentiated gradient algorithms for conditional random fields and max-margin markov networks,” *Journal of Machine Learning Research*, vol. 9, n°. Aug, p. 1775–1822, 2008.

[75] S. Lacoste-Julien, M. Jaggi, M. Schmidt et P. Pletscher, “Block-coordinate frank-wolfe optimization for structural svms,” *arXiv preprint arXiv :1207.4747*, 2012.

[76] M. Schmidt, N. Le Roux et F. Bach, “Minimizing finite sums with the stochastic average gradient,” *Mathematical Programming*, vol. 162, n°. 1-2, p. 83–112, 2017.

[77] T. Lavergne, O. Cappé et F. Yvon, “Practical very large scale crfs,” dans *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, p. 504–513.

[78] M. P. Friedlander et M. Schmidt, “Hybrid deterministic-stochastic methods for data fitting,” *SIAM Journal on Scientific Computing*, vol. 34, n°. 3, p. A1380–A1405, 2012.

[79] N. Le-Roux, M. Schmidt et F. R. Bach, “A stochastic gradient method with an exponential convergence _rate for finite training sets,” dans *Advances in neural information processing systems*, 2012, p. 2663–2671.

[80] M. Schmidt, R. Babanezhad, M. Ahmed, A. Defazio, A. Clifton et A. Sarkar, “Non-uniform stochastic average gradient method for training conditional random fields,” dans *artificial intelligence and statistics*, 2015, p. 819–828.

[81] H. Wallach, “Efficient training of conditional random fields,” Thèse de doctorat, Master’s thesis, University of Edinburgh, 2002.

[82] S. Shalev-Shwartz et T. Le, “Stochastic dual coordinate ascent methods for regularized loss minimization,” *Journal of Machine Learning Research*, vol. 14, n°. Feb, p. 567–599, 2013.

[83] S. Shalev-Shwartz et T. Zhang, “Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization,” dans *International conference on machine learning*, 2014, p. 64–72.

[84] R. Le-Priol, A. Piché et S. Lacoste-Julien, “Adaptive stochastic dual coordinate ascent for conditional random fields,” *arXiv preprint arXiv :1712.08577*, 2018.

[85] B. Taskar, C. Guestrin et D. Koller, “Max-margin markov networks,” *Advances in Neural Information Processing Systems*, vol. 2004, 2004.

[86] I. Tsochantaridis, T. Hofmann, T. Joachims et Y. Altun, “Large margin methods for structured and interdependent output variables,” *Journal of Machine Learning Research*, vol. 6, p. 1453–1484, 2005.

[87] C. Cortes et V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, n°. 3, p. 273–297, 1995.

[88] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li et L. Fei-Fei, “Imagenet : A large-scale hierarchical image database,” dans *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, p. 248–255.

[89] C. Ionescu, L. Bo et C. Sminchisescu, “Structural svm for visual localization and continuous state estimation,” dans *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, p. 1157–1164.

[90] W. Yang, Y. Wang et G. Mori, “Recognizing human actions from still images with latent poses,” dans *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, p. 2030–2037.

[91] M. B. Blaschko et C. H. Lampert, “Learning to localize objects with structured output regression,” dans *European conference on computer vision*. Springer, 2008, p. 2–15.

[92] L. Bertelli, T. Yu, D. Vu et B. Gokturk, “Kernelized structural svm learning for supervised object segmentation,” dans *CVPR 2011*. IEEE, 2011, p. 2153–2160.

[93] S. Nowozin, P. V. Gehler et C. H. Lampert, “On parameter learning in crf-based approaches to object class image segmentation,” dans *European conference on computer vision*. Springer, 2010, p. 98–111.

[94] M. Szummer, P. Kohli et D. Hoiem, “Learning crfs using graph cuts,” dans *European conference on computer vision*. Springer, 2008, p. 582–595.

[95] A. Gupta, M. Hebert, T. Kanade et D. M. Blei, "Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces," dans *Advances in neural information processing systems*, 2010, p. 1288–1296.

[96] A. G. Schwing, T. Hazan, M. Pollefeys et R. Urtasun, "Efficient structured prediction for 3d indoor scene understanding," dans *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, p. 2815–2822.

[97] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks et P. H. Torr, "Struck : Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, n°. 10, p. 2096–2109, 2015.

[98] X. Lou et F. A. Hamprecht, "Structured learning for cell tracking," dans *Advances in neural information processing systems*, 2011, p. 1296–1304.

[99] T. Joachims, T. Finley et C.-n. J. Yu, "Cutting-plane training of structural svms," *Mach. Learn*, vol. 77, p. 127–59, 2009.

[100] C. H. Teo, S. Vishwanthan, A. J. Smola et Q. V. Le, "Bundle methods for regularized risk minimization," *Journal of Machine Learning Research*, vol. 11, n°. Jan, p. 311–365, 2010.

[101] N. D. Ratliff, J. A. Bagnell et M. A. Zinkevich, "(online) subgradient methods for structured prediction," *Robotics Institute*, p. 55, 2007.

[102] M. Frank et P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, n°. 1-2, p. 95–110, 1956.

[103] S. Branson, O. Beijbom et S. Belongie, "Efficient large-scale structured learning," dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, p. 1806–1813.

[104] N. Komodakis, "Efficient training for pairwise or higher order crfs via dual decomposition," dans *CVPR 2011*. IEEE, 2011, p. 1841–1848.

[105] V. Franc et B. Savchynskyy, "Discriminative learning of max-sum classifiers," *Journal of Machine Learning Research*, vol. 9, n°. Jan, p. 67–104, 2008.

[106] T. Hazan et R. Urtasun, "A primal-dual message-passing algorithm for approximated large scale structured prediction," dans *Advances in Neural Information Processing Systems*, 2010, p. 838–846.

[107] O. Meshi, D. Sontag, T. Jaakkola et A. Globerson, "Learning efficiently with approximate inference via dual losses," *International Conference on Machine Learning*, 2010.

[108] K.-W. Chang, S. Upadhyay, G. Kundu et D. Roth, "Structural learning with amortized inference," dans *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[109] L. Bottou, “Stochastic gradient descent tricks,” dans *Neural networks : Tricks of the trade*. Springer, 2012, p. 421–436.

[110] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, n°. 11, p. 2278–2324, 1998.

[111] J. J. Tompson, A. Jain, Y. LeCun et C. Bregler, “Joint training of a convolutional network and a graphical model for human pose estimation,” dans *Advances in neural information processing systems*, 2014, p. 1799–1807.

[112] M. Jaderberg, K. Simonyan, A. Vedaldi et A. Zisserman, “Deep structured output learning for unconstrained text recognition,” *arXiv preprint arXiv :1412.5903*, 2014.

[113] T.-H. Vu, A. Osokin et I. Laptev, “Context-aware cnns for person head detection,” dans *Proceedings of the IEEE International Conference on Computer Vision*, 2015, p. 2893–2901.

[114] L.-C. Chen, A. Schwing, A. Yuille et R. Urtasun, “Learning deep structured models,” dans *International Conference on Machine Learning*, 2015, p. 1785–1794.

[115] Z. Yang, R. Salakhutdinov et W. Cohen, “Multi-task cross-lingual sequence tagging from scratch,” *arXiv preprint arXiv :1603.06270*, 2016.

[116] T. Chen, R. Xu, Y. He et X. Wang, “Improving sentiment analysis via sentence type classification using bilstm-crf and cnn,” *Expert Systems with Applications*, vol. 72, p. 221–230, 2017.

[117] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero et L. Heck, “Learning deep structured semantic models for web search using clickthrough data,” dans *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, p. 2333–2338.

[118] P. Priore, A. Gómez, R. Pino et R. Rosillo, “Dynamic scheduling of manufacturing systems using machine learning : An updated review,” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 28, n°. 1, p. 83–97, 2014.

[119] P. Wang, G. Zhao et X. Yao, “Applying back-propagation neural network to predict bus traffic,” dans *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*, IEEE. China : ICNC-FSKD, 2016, p. 752–756.

[120] S. Doherty et A. Mohammadian, “Application of artificial neural network models to activity scheduling time horizon,” *Transportation Research Record*, vol. 1, n°. 1854, p. 43–49, 12 2003.

[121] 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.

[122] S. Ioffe et C. Szegedy, “Batch normalization : Accelerating deep network training by reducing internal covariate shift,” dans *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML’15. Lille, France : JMLR.org, 2015.

[123] N. Hatami, Y. Gavet et J. Debayle, “Classification of time-series images using deep convolutional neural networks,” dans *Tenth International Conference on Machine Vision (ICMV 2017)*, vol. 10696, 2018, p. 10 696 – 10 696 – 8.

[124] A. Borovykh, S. Bohte et K. Oosterlee, “Conditional time series forecasting with convolutional neural networks,” dans *Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence*. USA : Springer, sept. 2017, p. 729–730.

[125] M. S. A. Nadeem, J.-D. Zucker et B. Hanczar, “Accuracy-rejection curves (arcs) for comparing classification methods with a reject option,” dans *Proceedings of the third International Workshop on Machine Learning in Systems Biology*, ser. Proceedings of Machine Learning Research, S. Džeroski, P. Guerts et J. Rousu, édit., vol. 8. Ljubljana, Slovenia : PMLR, 05–06 Sep 2009, p. 65–81.

[126] D. Hendrycks et K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *Proceedings of International Conference on Learning Representations*, vol. 5, 2017.

[127] C. Guo, G. Pleiss, Y. Sun et K. Q. Weinberger, “On calibration of modern neural networks,” dans *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup et Y. W. Teh, édit., vol. 70. International Convention Centre, Sydney, Australia : PMLR, 06–11 Aug 2017, p. 1321–1330.

[128] G. Fumera, F. Roli et G. Giacinto, “Reject option with multiple thresholds,” *Pattern Recognition*, vol. 33, p. 2099–2101, 2000.

[129] Y. Gal et Z. Ghahramani, “Dropout as a Bayesian approximation : Representing model uncertainty in deep learning,” dans *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML’16. New York, NY, USA : JMLR.org, 2016.

[130] F. Chollet *et al.*, “Keras,” 2015.

[131] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, “Tensorflow : Large-scale machine learning on heterogeneous systems, 2015,” *Software available from tensorflow. org*, vol. 1, n°. 2, 2015.

[132] 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.

[133] J. Bergstra et Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, févr. 2012.

[134] F. Hutter, H. H. Hoos et K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” dans *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, ser. LION’05. Berlin, Heidelberg : Springer-Verlag, 2011.

[135] F. Dernoncourt et J. Y. Lee, “Optimizing neural network hyperparameters with gaussian processes for dialog act classification,” *2016 IEEE Spoken Language Technology Workshop (SLT)*, vol. 2016, p. 406–413, 2016.

[136] The GPyOpt authors, “GPyOpt : A bayesian optimization framework in python,” http://github.com/SheffieldML/GPyOpt, 2016.

[137] J. W. Yen et J. R. Birge, “A stochastic programming approach to the airline crew scheduling problem,” *Transportation Science*, vol. 40, n°. 1, p. 3–14, 2006.

[138] M. L. Pinedo, *Scheduling*. Springer International Publishing, 2016. [En ligne]. Disponible : https://doi.org/10.1007/978-3-319-26580-3

[139] P. D. Bruecker, J. V. den Bergh, J. Belien et E. Demeulemeester, “ A two-stage mixed integer programming approach for optimizing the skill mix and training schedules for aircraft maintenance,” KU Leuven, Faculty of Economics and Business, Department of Decision Sciences and Information Management, Working Papers Department of Decision Sciences and Information Management 516588, nov. 2015. [En ligne]. Disponible : https://ideas.repec.org/p/ete/kbiper/516588.html

[140] R. Sadykov, F. Vanderbeck, A. A. Pessoa, E. Uchoa et I. Tahiri, “Recent results for column generation based diving heuristics.” dans *ColGen*. Buzios, Brazil : INRIA, mai 2016, p. 1–33.

[141] H. Bouarab, I. El Hallaoui, A. Metrane et F. Soumis, “Dynamic constraint and variable aggregation in column generation,” *European Journal of Operational Research*, vol. 262, n°. 3, p. 835–850, 2017.

[142] O. du Merle, D. Villeneuve, J. Desrosiers et P. Hansen, “Stabilized column generation,” *Discrete Mathematics*, vol. 194, n°. 1, p. 229 – 237, 1999.

[143] M. Gamache, F. Soumis, G. Marquis et J. Desrosiers, “A column generation approach for large-scale aircrew rostering problems,” *Operations Research*, vol. 47, n°. 2, p. 247–263, 1999.

[144] G. Desaulniers, J. Desrosiers et M. M. Solomon, *Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems*. Boston, MA : Springer US, 2002, p. 309–324.

[145] P.-Q. Pan, "A basis-deficiency-allowing variation of the simplex method for linear programming," *Computers & Mathematics with Applications*, vol. 36, n°. 3, p. 33–53, 1998.

[146] I. Elhallaoui, A. Metrane, G. Desaulniers et F. Soumis, "An improved primal simplex algorithm for degenerate linear programs," *INFORMS Journal on Computing*, vol. 4, p. 569–577, 2011.

[147] J. Omer, S. Rosat, V. Raymond et F. Soumis, "Improved primal simplex : a more general theoretical framework and an extended experimental analysis," *INFORMS Journal on Computing*, vol. 27, n°. 4, p. 773–787, 2015.

[148] F. Quesnel, G. Desaulniers et F. Soumis, "A new heuristic branching scheme for the crew pairing problem with base constraints," *Computers & Operations Research*, vol. 80, p. 159–172, 2017.

[149] Y. Yaakoubi, F. Soumis et S. Lacoste-Julien, "Flight-connection prediction for airline crew scheduling to construct initial clusters for OR optimizer," *Les Cahiers du GERAD*, vol. G–2019, n°. 26, 2019.

[150] P. Krähenbühl et V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," dans *Advances in neural information processing systems*, 2011, p. 109–117.

[151] J. Shotton, J. Winn, C. Rother et A. Criminisi, "Textonboost for image understanding : Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *International Journal of Computer Vision*, vol. 81, n°. 1, p. 2–23, 2009.

[152] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang et P. H. Torr, "Conditional random fields as recurrent neural networks," dans *Proceedings of the IEEE international conference on computer vision*, 2015, p. 1529–1537.

[153] A. G. Schwing et R. Urtasun, "Fully connected deep structured networks," *arXiv preprint arXiv :1503.02351*, 2015.

[154] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy et A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv :1412.7062*, 2014.

[155] ——, "Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, n°. 4, p. 834–848, 2017.

[156] L.-C. Chen, G. Papandreou, F. Schroff et H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint arXiv :1706.05587*, 2017.

[157] C. Williams et M. Seeger, “Using the nystrom method to speed up kernel machines,” *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[158] K. Zhang, I. Tsang et J. Kwok, “Improved nyström low-rank approximation and error analysis,” *Proc. ICML*, 2008.

[159] I. Schoenberg, “Positive definite functions on spheres,” *Duke Math. J*, vol. 1, p. 172, 1942.

[160] R. H. Byrd, P. Lu, J. Nocedal et C. Zhu, “A limited memory algorithm for bound constrained optimization,” *SIAM Journal on Scientific Computing*, vol. 16, n°. 5, p. 1190–1208, 1995.

[161] R. Johnson et T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” dans *Advances in neural information processing systems*, 2013, p. 315–323.

[162] K. Crammer et Y. Singer, “On the algorithmic implementation of multiclass kernel-based vector machines,” *Journal of Machine Learning Research*, vol. 2, p. 265–292, 2001.

[163] J. Weston et C. Watkins, “Multi-class support vector machines,” Department of Computer Science, Rapport technique, 1998.

[164] N. L. Roux, M. Schmidt et F. R. Bach, “A stochastic gradient method with an exponential convergence _rate for finite training sets,” dans *Advances in neural information processing systems*, 2012, p. 2663–2671.

[165] S. Shalev-Shwartz et T. Zhang, “Stochastic dual coordinate ascent methods for regularized loss minimization,” *Journal of Machine Learning Research*, vol. 14, n°. Feb, p. 567–599, 2013.

[166] H. Le-Priol, A. Touati et S. Lacoste-Julien, “Adaptive stochastic dual coordinate ascent for conditional random fields,” *NIPS Workshop*, 2017.

[167] A. Osokin, J.-B. Alayrac, I. Lukasewitz, P. K. Dokania et S. Lacoste-Julien, “Minding the gaps for block frank-wolfe optimization of structured svms,” *arXiv preprint arXiv :1605.09346*, 2016.

[168] D. Koller et N. Friedman, *Probabilistic graphical models : principles and techniques*. MIT press, 2009.

[169] S. Chandra, N. Usunier et I. Kokkinos, “Dense and low-rank gaussian crfs using deep embeddings,” dans *Proceedings of the IEEE International Conference on Computer Vision*, 2017, p. 5103–5112.

[170] S. Lacoste-Julien, M. Jaggi, M. Schmidt et P. Pletscher, “Block-coordinate frank-wolfe optimization for structural svms,” dans *Proceedings of the 30th International Conference on Machine Learning (EMNLP*, 2013.

[171] T. Do et T. Artieres, “Neural conditional random fields,” dans *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9. PMLR, 13–15 May 2010, p. 177–184.

[172] S. N. Tran, S. Cherla, A. Garcez et T. Weyde, “Linear-time sequence classification using restricted boltzmann machines,” *arXiv preprint arXiv :1710.02245*, 2018.

[173] F. Soumis, Y. Yaakoubi et S. Lacoste-Julien, “Machine learning feeding mathematical programming for air crew scheduling,” dans *TRISTAN*, 2019.

[174] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, n°. 7587, p. 484, 2016.

[175] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, “Mastering the game of go without human knowledge,” *Nature*, vol. 550, n°. 7676, p. 354, 2017.