Modélisation et conception d'algorithmes pour la planification d'emplois du temps

Mohamed Amine Ouberkouk

Par **Mohamed Amine OUBERKOUK**

Modélisation et conception d'algorithmes pour la planification d'emplois du temps

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC

Soutenue le 27 juin 2023
Spécialité : Informatique : Unité de recherche Heudyasic (UMR-7253)
D2746

Université de Technologie de Compiègne
Laboratoire Heudiasyc

THÈSE
Présentée en vue d'obtenir le grade de
DOCTEUR
par

Mohamed Amine OUBERKOUK

Modélisation et conception d'algorithmes pour la planification d'emplois du temps

---

Soutenue le 27 juin 2023 devant le jury composé de :

- **M. Slim HAMMADI**, professeur, membre rapporteur
  CRIStAL, Ecole centrale de Lille, Villeneuve d'Ascq
- **M. Ammar OULAMARA**, professeur, membre rapporteur
  Université de Lorraine LORIA Nancy, Vandoeuvre les Nancy
- **Mme Aym AUIBADJ**, Aaître Ae Aonférences, Aembre Axaminateur
  LISIC, Aniversité Au Aittoral Aôte A'Opale, Aalais
- **M. Antoine JOUGLET**, professeur, membre examinateur
  Heudiasyc, Université de Technologie de Compiègne
- **M. Aziz MOUKRIM**, professeur, directeur de thèse
  Heudiasyc, Université de Technologie de Compiègne
- **M. Aean-Paul BOUFFLET**, Aaître Ae Aonférences, Ao-encadrant Ae thèse Heudiasyc, Aniversité Ae Aechnologie Ae Aompiègne

Remerciements

Une aventure admirable se termine, à sa fin, je tiens à exprimer mes sincères remerciements pour toute personne ayant participé de près ou de loin à la bonne réussite de cette thèse. Ces années dans le monde de la recherche ont été très passionnantes et pleines d'épreuves, je tiens donc à adresser mes remerciements les plus sincères à tous ceux qui m'ont accompagné tout au long de cette étape importante de ma carrière.

Avant toute chose, je tiens à exprimer ma gratitude et mes sincères remerciements à mes directeurs de thèse, Aziz et Jean-Paul. Leur dévouement, leurs compétences et leur veille à la bonne conduite de ce projet ont été un atout majeur pour la bonne réussite de cette thèse. Je tiens à remercier mes directeurs de thèse pour tout ce qu'ils m'ont apporté sur le plan professionnel et personnel.

Par la suite, je désire adresser mes remerciements à Slim Hammadi et Ammar Oulamara pour avoir investi leur temps dans l'évaluation de mes travaux et à la lecture de mon manuscrit. Vos remarques et vos suggestions ont grandement contribué à ma préparation en vue de la soutenance. J'exprime également ma profonde gratitude envers Rym Guibadj et Antoine Jouglet d'avoir accepté de faire partie du jury de ma thèse. Les échanges fructueux que nous avons eus lors de la soutenance ont permis de valoriser les travaux réalisés durant cette thèse et de fructifier cette étape importante pour la suite de ma carrière en tant que docteur. Je tiens également à exprimer mes amples remerciements à Priscilla Velut, Geneviève Boufflet et Grégory Quentin pour les énormes efforts qu'ils ont fourni tout au long de cette thèse. Je souhaite aussi remercier la DSI et le SME de l'UTC pour leur accompagnement durant ce projet. Je souhaite également exprimer ma profonde reconnaissance envers tous les membres du laboratoire Heudiasyc.

Le soutien de ma famille et mes amis était ma source de motivation et de force tout au long de cette. À ma mère, tes paroles d'encouragement durant toutes ces années de thèse et les efforts que tu as investi depuis ma naissance ont été la principale raison pour que j'arrive là ou je suis aujourd'hui. À mon épouse, ton soutien quotidien, les échanges techniques sur les différents aspects de la thèse, pour les relectures et toute l'aide que tu m'as apporté, je tiens à te remercier d'avoir toujours été à mes cotés et toujours répondre présent. À mon frère, ma soeur, toute ma famille et tous mes amis, je tiens à vous remercier pour votre soutien et tous vos encouragements.

Résumé

Les problèmes de planification font référence à des situations où il faut prendre des décisions sur l'affectation de ressources limitées pour réaliser certaines tâches ou atteindre des objectifs donnés, tout en respectant un ensemble de contraintes. Nous avons étudié dans cette thèse deux problèmes de planification : le problème de planification de cours de l'UTC (University Course TimeTabling Problem of UTC university, UTC-UCTTP) et le problème de planification d'équipes de pompiers de l'institution INFOCA (FireFighters Timetabling Problem, FFTP). Les deux problèmes sont issus de situations réelles. Pour les deux problèmes étudiés, nous avons défini l'ensemble des données et des contraintes suite aux échanges avec le SME de l'UTC et les pompiers de l'institution INFOCA. Un générateur d'instances a été développé pour chaque problème afin de pouvoir générer un benchmark pour tester les méthodes proposées pour résoudre chacun des deux problèmes. Les institutions de formation font face à des problèmes complexes de planification. La conception des plannings devient un enjeu majeur dans un cadre budgétaire contraint. Nous avons étudié dans le premier problème traité, le problème de planification de cours de l'UTC (UTC-UCTTP). Le problème consiste à programmer des activités d'enseignement dans des salles et des timeslots, et les affecter à des enseignants. L'enjeu est d'obtenir des emplois du temps de bonne qualité en respectant les différentes contraintes du problème. Nous avons proposé un modèle ILP, une heuristique AIDCH et une métaheuristique ALNS. Les approches proposées ont été testées à l'aide d'un benchmark de vingt instances que nous avons générées à partir de données réelles. Les résultats obtenus par la méthode ALNS sont satisfaisants dans des temps de traitement raisonnables. Le deuxième problème que nous avons étudié est le problème de planification d'équipes de pompiers FFTP de l'institution INFOCA pour lequel les équipes de pompiers doivent être affectées à des quarts de travail au cours de la période à haut risque de feux de forêt. L'enjeu est d'obtenir une capacité opérationnelle maximale en considérant les contraintes qui permettent d'atteindre cette capacité opérationnelle, en plus des contraintes liées à la réglementation du travail. Nous avons présenté un modèle ILP, une matheuristique ILPH et une métaheuristique ALNS. Les approches proposées ont été testées à l'aide de quatre datasets de taille croissante que nous avons générés à partir de données réelles. L'ALNS a obtenu tous les résultats optimaux atteints en utilisant le modèle ILP sur les plus petites instances. Pour les plus grandes instances, les résultats sont de meilleure qualité par rapport à ceux obtenus par la matheuristique ILPH, et les temps de traitement sont significativement réduits.

Mots-clés : planification de cours d'universités, planification de personnel, planification d'équipes de pompiers, heuristique adaptative, métaheuristique.

À la mémoire de mon défunt père,
À ma chère mère,
À ma chère épouse,
À mon frère et à ma soeur,
À toute ma famille et tous mes amis,
Je dédie ce travail de thèse,
Et dans l'espoir de vous rendre toujours fiers de moi.

Table des Matières

Chapitre 1

Introduction générale

Contexte

Les problèmes de planification font référence à des situations où il faut prendre des décisions sur l'affectation de ressources limitées pour réaliser certaines tâches ou atteindre des objectifs donnés, tout en respectant un ensemble de contraintes.

L'étude des problèmes de planification revêt une grande importance qui attire les chercheurs pour plusieurs raisons. En effet, en résolvant un problème de planification, on peut optimiser l'affectation des ressources et la planification des tâches, ce qui conduit à une utilisation plus efficace des ressources disponibles. Cela permet de réduire les coûts, d'améliorer la productivité et de maximiser l'efficacité opérationnelle globale.

Les ressources sont souvent limitées et doivent être utilisées de manière optimale. C'est le cas dans la majorité des domaines qui s'intéressent aux problèmes de planification. En étudiant les problèmes de planification, on peut trouver des solutions qui permettent de répartir équitablement les ressources, de maximiser leur utilisation et de minimiser les conflits.

Les problèmes de planification se retrouvent dans de nombreux domaines, tels que la logistique, les transports, la gestion de projets, la planification de la production, la planification des horaires, etc. Étudier ces problèmes permet de développer des outils et des méthodes spécifiques à ces domaines, ce qui facilite la résolution de problèmes concrets et contribue à l'amélioration des processus et des performances. L'étude des problèmes de planification est devenue un enjeu majeur. Cela permet d'obtenir des résultats économiques, pratiques et socialement bénéfiques.

Les problèmes de planification peuvent être très complexes, et différentes techniques d'optimisation combinatoire sont utilisées pour les résoudre, telles que les algorithmes génétiques, les algorithmes de recherche locale, la programmation linéaire et la programmation par contraintes.

Nous nous intéressons dans cette étude à deux problèmes de planification : le problème de planification de cours d'universités et le problème de planification d'équipes de pompiers.

CONTRIBUTIONS DE LA THÈSE

Nous avons étudié dans cette thèse deux problèmes de planification : le problème de planification de cours de l'UTC et le problème de planification d'équipes de pompiers de l'institution INFOCA. Les deux problèmes sont issus de situations réelles. Pour les deux problèmes étudiés, nous avons défini l'ensemble des données et des contraintes suite aux échanges avec le SME de l'UTC et les pompiers de l'institution INFOCA. Un générateur d'instances a été développé pour chaque problème afin de pouvoir générer un benchmark pour tester les méthodes proposées pour résoudre chacun des deux problèmes.

Problème de planification de cours de l'UTC (UTC-UCTTP) :

Les institutions de formation font face à des problèmes complexes de planification. La conception des plannings devient un enjeu majeur dans un cadre budgétaire contraint. Les problèmes d'emploi du temps de cours d'universités sont des problèmes de planification sous contraintes de ressources. Il s'agit de construire des emplois du temps (**EDT**), c'est à-dire des plannings de modules d'enseignements composés d'activités pédagogiques (Cours, TD, TP, etc.) en respectant des contraintes. L'objectif est de construire des plannings qui respectent des contraintes dures en minimisant des contraintes souples qui induisent des pénalités qui servent de métrique pour mesurer la qualité des EDTs. L'ensemble de contraintes est spécifique selon l'organisation pédagogique de l'université et des objectifs de qualité d'EDT qu'elle vise à atteindre. De fait, il existe une grande variété dans les contraintes (e.g. intégration de formations par voie d'apprentissage, accueil d'étudiants en situation de handicap, déplacements entre les sites, répartition des étudiants dans les modules en tenant compte des choix des étudiants).

Nous étudions le problème de planification de cours de l'UTC (University Course TimeTabling Problem of UTC university, UTC-UCTTP). Le problème consiste à programmer des activités d'enseignement dans des salles et des timeslots, et les affecter à des enseignants. L'enjeu est d'obtenir des emplois du temps de bonne qualité en respectant les différentes contraintes du problème.

Premièrement, nous proposons un modèle de programmation linéaire en nombres entiers (Integer Linear Program, ILP) pour le problème de planification de cours de l'UTC (UTC-UCTTP). Le modèle ILP est conçu à des fins de modélisation et dans le but d'obtenir des solutions de bonne qualité pour des instances de taille modeste. Cependant, les solveurs ILP peuvent avoir des difficultés à trouver des solutions réalisables dans des délais de traitement raisonnables pour des instances plus conséquentes.

Deuxièmement, nous proposons une méthode heuristique AIDCH (Adaptive Large Neighbourhood Search, AIDCH) pour trouver des solutions réalisables pour les instances du problème, les solutions trouvées peuvent ne pas être de bonne qualité.

Troisièmement, nous proposons une métaheuristique ALNS (Adaptive Large Neighbourhood Search, ALNS) pour étudier une deuxième approche de résolution. Nous menons des expériences préliminaires pour régler les paramètres des composants de l'ALNS. Nous étudions la contribution des composants de l'ALNS.

Enfin, nous montrons que l'approche de résolution ALNS obtient des solutions de bonne qualité dans des temps de traitement raisonnables. L'approche ALNS peut constituer une bonne base pour proposer des emplois du temps adéquats pour l'UTC.

Problème de planification d'équipes de pompiers (FFTP) :

Partout dans le monde, des milliers de kilomètres carrés de forêts sont ravagés par des feux chaque année. Ces feux causent des pertes économiques, naturelles et animales significatives et aussi bien souvent des pertes humaines. La gestion et le confinement des feux deviennent alors un point critique dont les gouvernements européens et le gouvernement australien sont bien conscients. Plusieurs recherches ont été menées par des scientifiques de différentes spécialités en Europe et en Australie qui ont développé des méthodes et des modèles qui peuvent améliorer les processus de gestion et de décision.

Les travaux concernant cette contribution ont été initiés lors d'un détachement de deux mois à l'université RMIT (Royal Melbourne Institut of Technology, RMIT) dans le cadre du projet GEO-SAFE (Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies, GEO-SAFE).

Le projet GEO-SAFE a été coordonné par l'université de Greenwich (Royaume Uni) sur quatre ans (2016-2020). Le projet réunit 17 partenaires dans 7 pays (Australie, Espagne, France, Italie, Pays Bas, Royaume Uni et Suisse) avec un budget estimé à plus de 1.386.000 euros. Le but du projet GEO-SAFE est de créer un environnement de partage d'information et de connaissances afin que deux continents puissent collaborer entre eux en échangeant leurs connaissances, idées et expériences pour progresser dans le domaine et trouver des solutions innovantes pour les différents problèmes qui entourent la thématique.

Nous nous intéressons dans cette deuxième contribution à l'étude du problème de planifications d'équipes de pompiers (FireFighters Timetabling Problem, FFTP). Dans notre problème, une équipe de pompiers peut être vue comme un employé possédant de multiples compétences hautement spécialisées. Ces personnes constituent une unité extrêmement soudée, formée pour avoir des automatismes de groupe pour être efficace dans des situations extrêmes. Le nombre d'équipiers est fixé et n'est pas à minimiser comme dans d'autres problèmes de gestion de personnel. Au contraire, les objectifs sont d'augmenter la capacité opérationnelle tout en considérant un nombre limité d'équipes, de maintenir l'équité entre les équipes et de rendre possible le regroupement des jours de repos lorsque cela est possible.

A notre connaissance, le problème de planification d'équipes de pompiers pour les institutions dont la mission est de lutter contre les feux de forêt n'a pas encore été étudié dans la littérature. En raison du changement climatique, la période à haut risque de feux de forêt s'élargit et les feux de forêt augmentent en nombre et en intensité. La capacité opérationnelle des équipes de pompiers est susceptible de devenir un problème.

Une façon d'y parvenir est de construire de meilleurs plannings des équipes en tenant compte des contraintes réglementaires du travail mais aussi des contraintes de bonnes pratiques pour assurer l'équité.

Premièrement, nous proposons un modèle ILP qui est conçu à des fins de modélisation et dans le but d'obtenir des solutions de bonne qualité pour certaines

instances qui seront utilisées comme solutions de référence. Cependant, les solveurs ILP peuvent avoir des difficultés à trouver des solutions réalisables dans des délais de traitement raisonnables.

Deuxièmement, nous utilisons le modèle ILP comme base pour proposer une matheuristique (Integer Linear Programming Heuristic, ILPH). Nous proposons trois voisinages pour faire travailler le solveur sur des sous-problèmes dans le but d'obtenir des solutions réalisables pour toutes les instances dans un temps de traitement raisonnable. Nous proposons d'explorer l'espace de recherche en utilisant une approche de descente par voisinage variable (Variable Neighborhood Descent, VND) basée sur les trois voisinages.

Troisièmement, nous proposons une métaheuristique ALNS pour étudier une deuxième approche de résolution. Nous menons des expériences préliminaires pour régler les paramètres des composants de l'ALNS. Nous utilisons aussi une version de la matheuristique comme méthode de construction et étudions la contribution des composants de l'ALNS.

Enfin, nous montrons que l'approche de résolution ALNS obtient toutes les solutions optimales qui peuvent être atteintes par le modèle ILP. De meilleures solutions que celles de la matheuristique sont obtenues dans un temps de traitement plus court. L'approche ALNS peut constituer une bonne base pour améliorer la capacité opérationnelle des établissements de pompiers pendant la période à haut risque de feux de forêt.

PRÉSENTATION DE LA THÈSE

Le reste de manuscrit est structuré comme suit.

Le chapitre 2 présente une étude globale sur les problèmes de planification de cours d'universités et les problèmes de planification de personnel.

Le chapitre 3 est consacré à l'étude des problèmes de planification de cours d'universités. Nous présentons les différents problèmes de planifications de cours d'universités ainsi que les différents travaux qui ont étudié ces problèmes dans la littérature.

Le chapitre 4 est consacré à notre étude sur le problème de planification de cours de l'UTC (UTC-UCTTP). Dans ce chapitre, nous présentons la définition du problème et les approches de résolution proposées. Nous présentons les résultats numériques.

Le chapitre 5 est consacré à l'étude des problèmes de planification de personnel. Nous présentons les différents problèmes de planifications de personnel dans plusieurs domaines ainsi que les différents travaux qui ont étudié ces problèmes dans la littérature.

Le chapitre 6 est consacré à notre étude sur le problème de planification d'équipes de pompiers (FFTP). Dans ce chapitre, nous présentons la définition du problème et les approches de résolution proposées. Nous présentons les résultats numériques.

Dans le chapitre 7, nous terminons par une conclusion générale et nous présentons des perspectives de nos travaux de recherches pour les deux problèmes traités.

Chapitre 2

Les problèmes de planification

La planification d'horaires est un type spécifique de planification qui consiste à allouer des activités à des ressources et à des intervalles de temps dans un horizon de planification prédéfini, en respectant des contraintes de précédence, de durée, de capacité, de disjonction et d'exclusion [Ernst et al., 2004b]. Le planning peut être vu comme un calendrier de travail où figurent à la fois le temps (jours et horaires), l'affectation des ressources et les noms des tâches (activités).

Les problèmes de planification font référence à des situations où il faut prendre des décisions sur l'affectation de ressources limitées pour réaliser certaines tâches ou atteindre des objectifs donnés, tout en respectant un ensemble de contraintes.

Plusieurs problèmes de planification ont été étudiés dans la littérature. Nous présentons dans ce qui suit quelques problèmes de planification.

Le problème du voyageur de commerce consiste à trouver le plus court chemin possible pour visiter un ensemble de villes une seule fois et revenir à la ville de départ.

Les problèmes d'ordonnancement visent à déterminer l'ordre d'exécution optimal des tâches dans le but, par exemple, de minimiser le temps total d'exécution, de maximiser l'utilisation des ressources ou de respecter des contraintes temporelles.

Les problèmes d'affectation des ressources impliquent la répartition optimale des ressources (telles que les machines, les travailleurs, les véhicules) pour exécuter un ensemble de tâches tout en minimisant les coûts ou en maximisant l'efficacité.

Les problèmes de planification de la production concernent la planification des opérations de production dans le but de maximiser l'utilisation des ressources, de minimiser les temps d'attente ou de respecter les délais de livraison.

Les problèmes de planification de projet consistent à organiser et à ordonnancer les différentes tâches d'un projet, en tenant compte des dépendances entre les tâches, des contraintes de ressources et des objectifs de délai.

Les problèmes de planification en général englobent un large éventail de situations dans lesquelles il est nécessaire de prendre des décisions et de créer des programmes pour atteindre des objectifs spécifiques, en tenant compte de diverses contraintes et ressources disponibles. Ces problèmes sont présents dans de nombreux domaines et disciplines, tels que la logistique, l'ingénierie, la gestion de projet, la production, les transports, la robotique, l'informatique, etc. Quelques éléments des problèmes de planification peuvent être présentés comme suit :

- Objectifs et contraintes : les problèmes de planification sont généralement associés à des objectifs à atteindre, tels que la minimisation des coûts, la maximisation

des performances, la satisfaction de la demande ou le respect des délais. Ils sont également soumis à des contraintes, telles que des limites de temps, des ressources limitées, des dépendances entre les tâches ou des réglementations spécifiques ;

- Variables de décision : dans les problèmes de planification, il est nécessaire de déterminer les valeurs que prennent les variables de décision, c'est-à-dire les choix qui doivent être faits pour atteindre les objectifs. Ces variables peuvent représenter des tâches à planifier, des ressources à allouer, des ordres d'exécution, des itinéraires, des horaires, etc ;

- Complexité : les problèmes de planification peuvent être très complexes en raison du grand nombre de variables, des contraintes multiples et de la difficulté à trouver une solution optimale. La taille du problème peut augmenter exponentiellement en fonction du nombre d'éléments à planifier, ce qui rend la recherche de solutions exactes difficile, voire impossible pour certains problèmes dans des temps de calcul raisonnables ;

- Méthodes de résolution : différentes approches sont utilisées pour modéliser et traiter les problèmes de planification : les méthodes exactes telles que la programmation mathématique et la programmation par contraintes, mais aussi les méthodes heuristiques et métaheuristiques telles que les algorithmes génétiques, les algorithmes de recherche locale, les algorithmes gloutons, etc ;

- Type de planification (statique ou dynamique) : la planification peut être effectuée à l'avance, avant que les événements ne se produisent, ce qui est appelé la planification statique. Pour ce cas, toutes les données et contraintes du problème sont connues à l'avance. La planification peut aussi être dynamique, c'est-à-dire qu'elle doit être mise à jour en temps réel pour s'adapter aux changements et aux imprévus, les données et les contraintes du problème changent tout au long de la période de planification.

Les problèmes de planification se rencontrent dans de nombreux secteurs d'activité. Ils peuvent inclure la planification des horaires du personnel, la planification des itinéraires de transport, la planification des opérations de production, la planification des projets de construction, la planification des soins de santé, etc. Résoudre ces problèmes de manière efficace permet d'améliorer les performances, de réduire les coûts et d'optimiser l'utilisation des ressources.

Les problèmes de planification sont confrontés à une complexité croissante en raison de la taille des instances à traiter, des contraintes multiples et des interactions entre les variables de décision. Cela nécessite le développement de méthodes efficaces pour résoudre ces problèmes de manière optimale. De nombreux problèmes de planification comportent des objectifs multiples et parfois contradictoires. L'enjeu consiste à trouver un compromis entre ces objectifs, en trouvant des solutions qui sont à la fois réalisables et de bonne qualité. Les environnements réels dans lesquels la planification est appliquée sont souvent dynamiques et soumis à des variations et à des incertitudes. La prise en compte de ces facteurs dans les modèles de planification constitue un défi important pour obtenir des plans robustes et adaptatifs.

Il s'agit d'améliorer l'efficacité et d'optimisation des processus, en minimisant les coûts, en maximisant l'utilisation des ressources et en atteignant les objectifs fixés de manière efficace. En utilisant des modèles et des algorithmes de planification, les décideurs peuvent évaluer différentes stratégies, analyser les conséquences et choisir la meilleure solution possible. La planification efficace permet donc d'améliorer les performances opérationnelles, telles que la réduction des temps d'attente, l'optimisation des flux de travail, la diminution des retards et la maximisation de la productivité globale.

En résumé, les problèmes de planification représentent des défis importants dans de nombreux domaines, nécessitant des approches de résolution spécifiques pour atteindre les objectifs fixés. Leur étude permet de développer des méthodes, des modèles et des algorithmes pour trouver des solutions efficaces et optimales, contribuant ainsi à l'amélioration des processus et des performances dans diverses industries. Ils continueront d'être d'une importance primordiale dans de nombreux domaines. Les avancées dans la modélisation, les algorithmes d'optimisation et l'intégration de l'intelligence artificielle offriront des opportunités pour résoudre des problèmes de planification plus complexes et dynamiques, améliorant ainsi les performances opérationnelles et la prise de décision.

Nous nous intéressons dans cette étude à deux problèmes : la planification de cours d'universités et la planification d'équipes de pompiers.

Les institutions de formation font face à des problèmes complexes de planification. La conception des plannings devient un enjeu majeur dans un cadre budgétaire contraint. Les problèmes d'emploi du temps de cours d'universités sont des problèmes de planification sous contraintes de ressources. Il s'agit de construire des emplois du temps (**EDT**), c'est à-dire des plannings de modules d'enseignements composés d'activités pédagogiques (Cours, TD, TP, etc.) en respectant des contraintes. L'objectif est de construire des plannings qui respectent des contraintes dures en minimisant des contraintes souples qui induisent des pénalités qui servent de métrique pour mesurer la qualité des EDT. L'ensemble de contraintes est spécifique de l'organisation pédagogique de l'université et des objectifs de qualité d'EDT qu'elle vise à atteindre. De fait, il existe une grande variété dans les contraintes (e.g. intégration de formations par voie d'apprentissage, accueil d'étudiants en situation de handicap, déplacements entre les sites, répartition des étudiants dans les modules en tenant compte des choix des étudiants).

Les emplois du temps d'universités font l'objet de compétitions internationales. Dans la compétition ITC-2007 (International Timetabling Competition, ITC) deux problèmes d'EDT de cours sont définis : CB-CTT (Curriculum-Based Course TimeTabling, CB-CTT) et PECT (Post-Enrollement Course Timetabling, PECT).

Le problème CB-CTT (*Curriculum Based Courses TimeTabling Problem*) consiste en la planification hebdomadaire des activités de cours d'université dans un nombre fixé de salles et de créneaux horaires. Les conflits entre les cours sont définis à partir des curricula publiés par l'université. Un curriculum est un groupe de cours tel que tout couple de deux cours du groupe ont des étudiants en commun. Un cours représente une unité d'enseignement qui est suivie par des étudiants. Chaque cours est composé

de plusieurs activités qui doivent être planifiées.

Le CB-CTT correspond au problème de planification de cours de nombreuses universités italiennes mais aussi ce type de problème d'EDT (Emploi Du Temps) se rencontre dans plusieurs universités dans le monde entier. La formulation du problème CB-CTT de l'ITC-2007 a été simplifiée afin de maintenir un certain niveau de généralité. Des contraintes supplémentaires peuvent apparaître dans certains cas réels.

Le problème CB-CTT de l'ITC-2007 est défini par :

Jours, créneaux et horaires : le nombre de jours d'enseignement par semaine est une donnée (généralement 5 ou 6). Chaque jour est divisé en un nombre fixe de créneaux de même durée, qui sont pareils pour tous les jours. Un horaire est une paire composée d'un jour et d'un créneau.

Cours et Enseignants : chaque cours consiste en un nombre déterminé d'activités à programmer sur des périodes distinctes. Il est suivi par un nombre d'étudiants et est enseigné par un enseignant. Pour chaque cours, il y a un nombre minimum de jours pendant lesquels les activités du cours doivent être réparties. De plus, il existe certaines périodes pendant lesquelles le cours ne peut pas être programmé.

Salles : chaque salle a une capacité, exprimée en nombre de sièges disponibles. Toutes les salles sont également appropriées pour tous les cours (si cette salle est assez grande pour contenir tous les étudiants de l'activité du cours). Autrement dit, la seule contrainte de placement d'une activité dans une salle est le respect de la capacité.

Curricula : un curriculum est un groupe de cours tel que tout couple de deux cours du groupe ont des étudiants en commun. Sur la base du curriculum, nous avons les conflits entre les cours et d'autres contraintes souples.

Le problème PECT (*Post Enrollement Course Timetabling*) recouvre un type de problème de planification de cours d'universités. Les étudiants ont la liberté de choisir les cours qu'ils veulent suivre durant un semestre d'enseignement, les emplois du temps sont ensuite construits à partir des choix des étudiants. Nous décrivons le problème PECT qui a été introduit lors de la compétition ITC-2007 [Lewis et al., 2007].

Le problème peut être décrit comme étant un problème de planification où nous disposons d'un nombre d'évènements à planifier dans un nombre de salles et de créneaux tout en respectant un ensemble de contraintes dures et souples. La violation d'une des contraintes dures rend la solution non réalisable. Les contraintes souples sont des contraintes de bonne pratique qui participent à la mesure de la qualité de la solution. La violation d'une contrainte souple donne lieu à une pénalité.

L'objectif est de trouver une solution qui respecte strictement les contraintes dures tout en minimisant les contraintes souples violées.

Les données du problème sont :

- Un ensemble de n événements à planifier dans 45 créneaux (5 jours de 9 heures chacun) ;
- Un ensemble de r salles, chacune avec une capacité, dans lesquelles les événements ont lieu ;
- Un ensemble de caractéristiques de salles qui sont satisfaites par les salles et qui sont requises par les évènements ;
- Un ensemble de s étudiants qui suivent plusieurs combinaisons d'évènements ;
- Un ensemble de créneaux disponibles pour chaque évènement, c'est-à-dire qu'un évènement ne pourra pas être placé dans tous les créneaux de la semaine ;
- Un ensemble *Precedence* qui stipule que certains évènements doivent être placés avant d'autres.

L'objectif est de placer si possible tous les évènements. Il faut affecter à chaque évènement une salle et un créneau en respectant les contraintes dures.

Un planning est accepté si toutes les contraintes dures sont respectées. Néanmoins, des évènements peuvent restés non planifiés. Deux types de plannings sont considérés :

- **Planning valide :** un planning est valide s'il n'y a aucune violation de contraintes dures, mais des évènements peuvent restés non planifiés ;
- **Planning réalisable :** un planning est réalisable s'il n'y a aucune violation de contraintes dures et tous les évènements sont planifiés.

Dans la compétition ITC-2019, le problème consiste, comme dans le cas du PECT de ITC-2007, à construire conjointement le planning de cours (salles, horaires) et à affecter les étudiants dans les cours. Les contraintes dures usuelles permettent de définir l'admissibilité d'une solution (e.g. deux cours ne peuvent pas être planifiés dans une même salle). Les contraintes souples violées usuelles du PECT induisent des pénalités à minimiser pour obtenir des EDT de bonne qualité (e.g. éviter qu'un étudiant soit affecté à deux classes en même temps). Cependant le problème défini est plus proche de situations réelles en intégrant plus de données (e.g. l'horizon de planification est défini sur plusieurs semaines) et des contraintes qui se rapprochent plus de situations réelles (e.g. contraintes de distributions qui permettent d'intégrer les contraintes dures

et souples des enseignants).

Le deuxième problème de planification à lequel nous allons nous intéresser dans ce travail est le problème de planification d'équipes de pompiers, qui rentre dans la catégorie des problèmes de planification de personnel.

La planification du personnel est le processus de construction d'emplois du temps pour son personnel afin qu'une organisation puisse satisfaire la demande pour ses biens ou ses services. Le planning construit doit spécifier pour chaque employé ses jours de travail. En général, le problème de planification du personnel se subdivise en plusieurs sous-problèmes. Dans la classification proposée par [Tien et al., 1982], le processus de planification est vu comme un certain nombre de modules commençant par la détermination des besoins en personnel et se terminant par la spécification du travail à effectuer, sur une certaine période de temps, pour chaque individu appartenant au personnel. Bien que les modules suggèrent une procédure étape par étape, le développement d'une méthode de résolution peut nécessiter seulement quelques modules et, dans de nombreuses implémentations pratiques, plusieurs des modules peuvent être combinés en une seule procédure. De plus, les exigences des différents modules dépendent des applications. Nous présentons dans ce qui suit les différents modules.

Les problèmes de planification de personnel (Personnel Scheduling Problem, PSP) ou d'équipes [Aggarwal, 1982; Ernst et al., 2004b; Tien et al., 1982] traitent de l'allocation de ressources humaines à des plages horaires dans un horizon de planification donné tout en respectant les contraintes de compétence, de priorité, de durée, de capacité, de disjonction et de distribution.

Ces problèmes de planification visent à établir des emplois du temps ainsi que des listes de personnel associées. L'objectif est qu'une organisation puisse répondre aux demandes de biens ou de services. Pour chaque membre du personnel ou équipage, les jours de travail et de repos sont programmés dans un emploi du temps en tenant compte des contraintes réglementaires de l'organisation et des contraintes réglementaires locales, le cas échéant.

Les premiers travaux sur la planification du personnel remontent aux travaux de [Edie, 1954] sur les retards de circulation aux postes de péage.

Depuis lors, des approches de résolutions utilisant des algorithmes de planification ont été appliquées pour résoudre les problèmes de planification d'emploi du temps du personnel dans de nombreux domaines tels que les systèmes de transport (compagnies aériennes, chemins de fer), les systèmes de santé, les services d'urgence (police, ambulances), les centres d'appels et d'autres services (hôtels , restaurants, magasins commerciaux).

Un emploi du temps peut être vu comme une matrice qui inclut à la fois le temps, l'allocation des ressources et les noms des tâches. Le temps peut être des jours et/ou des heures utilisés pour définir un horizon. Les ressources correspondent au personnel pouvant avoir des compétences spécifiques. Les tâches auxquelles le personnel doit être affecté peuvent nécessiter une compétence particulière et peuvent aussi correspondre à des quarts de travail. Les personnes peuvent devoir être regroupées en équipages pour effectuer une tâche. Un équipage peut être vu comme une ressource qui est stable sur un l'horizon.

Les contraintes dures usuellement rencontrées se rapportent à la quantité de demandes pour les tâches. Ces contraintes se rapportent aux contraintes de réglementation du travail et aux contraintes de réglementation locale, le cas échéant. Les demandes sont utilisées pour assurer la couverture des besoins. Les contraintes de la réglementation du travail portent généralement sur des règles de jours de repos, de nombre maximum d'heures travaillées par jour, de jours de compensation à octroyer ou d'affectations successives irréalisables à des quarts de travail.

Les contraintes souples à optimiser concernent le personnel et l'institution. Dans le premier cas il s'agit par exemple d'être juste avec le personnel ou d'assurer l'équité, dans le second cas il s'agit d'améliorer la couverture des besoins si possible à moindre coût.

Un problème de planification de personnel est dit cyclique s'il est périodique, c'est-à-dire qu'une solution (emploi du temps) doit être construite d'abord pour une partie de l'horizon, puis cette solution est à répéter sur tout l'horizon. Un problème est dit acyclique si une solution doit être trouvée sur tout un horizon.

Dans la classification proposée par [Ernst et al., 2004b], le processus de planification est présenté comme un certain nombre de modules. Ces modules commencent par la détermination des besoins en personnel et se terminent par la spécification du travail à effectuer, sur un horizon de planification pour chaque membre du personnel. Les différents modules proposés dans [Ernst et al., 2004b] sont :

- Modélisation de la demande ;
- Planification des jours de repos ;
- Planification des quarts de travail ;
- Construction de lignes de travail ;
- Attribution de la tâche ;
- Affectation du personnel.

Le module de modélisation de la demande consiste à déterminer le nombre de membres du personnel nécessaires au cours d'une période de planification ou d'un horizon de planification. Trois types de demandes sont considérés :

- Listes de tâches individuelles à effectuer ;
- Flexibles, les demandes futures doivent être estimées à l'aide de prévisions ;
- Les quarts de travail, spécification du nombre d'employés qui doivent être en service pendant différents quarts de travail.

Le module de planification des jours de repos (ou congés) est en charge de la gestion des jours de repos et de leur alternance entre les jours de travail pour les différentes lignes de travail.

Le module de planification des quarts de travail aborde la problématique de la sélection. Cette sélection se fera parmi un ensemble potentiellement important d'employés, de quarts à effectuer, ainsi que l'affectation du nombre d'employés à chaque quart, afin de répondre à la demande.

Le module de construction de lignes de travail implique la création de lignes de travail, parfois appelées horaires de travail ou lignes de planification, couvrant l'horizon de planification, pour chaque membre du personnel.

Le module d'attribution de tâches aborde la problématique de l'attribution d'une ou plusieurs tâches à effectuer au cours de chaque quart de travail. Ces tâches peuvent exiger des compétences spécifiques du personnel ou des niveaux d'ancienneté à associer à des lignes de travail particulières.

Le module d'affectation du personnel est chargé de l'affectation individuelle du personnel aux lignes de travail. La dotation est souvent effectuée pendant la construction des lignes de travail.

Bien qu'une décomposition basée sur des modules suggère une procédure étape par étape, le développement d'une méthode de résolution peut ne nécessiter que quelques modules, et dans certaines implémentations pratiques, plusieurs modules peuvent être combinés en une seule procédure. De plus, les exigences des différents modules dépendent des applications.

La planification d'équipage se rencontre dans les systèmes de transport tels que les compagnies aériennes, les chemins de fer, les transports en commun et les bus [Barnhart et al., 2003]. Pour ces problèmes, il existe deux caractéristiques communes. La première est que des contraintes temporelles et spatiales sont impliquées. Chaque tâche est caractérisée par son heure et son emplacement de début, ainsi que par son heure et son emplacement de fin. La seconde est que toutes les tâches à effectuer par les

employés sont déterminées à partir d'un emploi du temps de tâches donné. Les tâches élémentaires sont déterminées sur la base des différentes tâches que l'entreprise doit assurer dans un horizon de planification. Une tâche peut être d'assurer une étape de vol dans les compagnies aériennes ou d'assurer un voyage entre deux segments dans un train.

Pour la planification du personnel dans les centres d'appels, la nature exacte et le nombre de tâches à effectuer ne sont pas connus avant le processus de planification. Ces demandes sont estimées selon un modèle de besoins en main-d'œuvre existant pour l'ensemble de l'horizon de planification. Les besoins en main-d'œuvre dans les applications de centre d'appels peuvent varier d'un jour à l'autre et d'une semaine à l'autre. Les heures de début et la durée des quarts de travail doivent varier afin d'obtenir de bons plannings à faible coût pour couvrir les besoins en main-d'œuvre.

La planification du personnel dans les centres d'appels n'implique pas de caractéristiques géographiques (ou spatiales). C'est la principale différence entre ce problème de planification du personnel et la planification des équipages présenté dans la section précédente.

La planification des infirmières est un des problèmes récurrents rencontré dans les systèmes de santé [Burke et al., 2004]. Chaque hôpital doit régulièrement attribuer des listes de quarts de travail à ses infirmières. Une bonne planification du personnel infirmier a un impact sur la qualité des soins de santé, le recrutement d'infirmières et l'élaboration d'un budget infirmier.

Les emplois du temps peuvent être générés manuellement par les infirmières en charge de l'organisation du service ou par une personnes dédiée dans chaque unité hospitalière. Cependant, l'élaboration d'un planning pour les infirmières a toujours été difficile à cause des différentes contraintes à respecter. Le personnel infirmier des hôpitaux travaille 24 heures sur 24, sept jours sur sept. Dans de nombreux hôpitaux, certaines infirmières occupent des postes prédéfinis à cause de leurs compétences ou leur ancienneté, tandis que d'autres infirmières peuvent être affectées dans d'autres services. Habituellement, les gestionnaires passent beaucoup de temps à élaborer des plannings, surtout lorsqu'il y a beaucoup de demandes de personnel. En raison de ce travail manuel fastidieux et chronophage, le problème de planification des infirmières (Nurse Rostering Problem, NRP) a suscité beaucoup d'intérêt de la part des chercheurs.

Les compétitions internationales de planification des infirmières (International Nurse Rostering Competition, INRC) visent à susciter l'intérêt dans le domaine de la planification des NRP. Elles proposent des modèles de problèmes qui intègrent des contraintes rencontrées dans la vie réelle. Le but des compétitions est de trouver de nouvelles approches pour résoudre le NRP tout en réduisant l'écart qui existe entre les problèmes théoriques et les problèmes réels. Deux compétitions ont été lancés pour le NRP, la première en 2010 (INRC1) et la seconde en 2015 (INRC2).

Nous présenterons dans le chapitre 3 les problèmes de planification de cours d'universités et dans le chapitre 5 les problèmes de planification d'équipes de pompiers ainsi que les méthodes qui ont été appliquées dans la littérature pour les résoudre.

Chapitre 3

Problèmes de planification de cours d'universités

Sommaire

Les problèmes d'emploi du temps de cours d'universités sont des problèmes de planification sous contraintes de ressources. Il s'agit de construire des emplois du temps (**EDT**), c'est-à-dire des plannings de modules d'enseignements composés d'activités pédagogiques (Cours, TD, TP, etc.) en respectant des contraintes.

Les emplois du temps d'universités font l'objet de compétitions internationales. Dans la compétition ITC-2007 (International Timetabling Competition, ITC) deux problèmes sont définis : CB-CTT (Curriculum-Based Course TimeTabling, CB-CTT) [Di Gaspero et al., 2007] et PECT (Post-Enrollement Course Timetabling, PECT) [Lewis et al., 2007].

Le CB-CTT concerne la conception d'emploi du temps de cours sur une semaine standard. Les cursus, ensemble de modules d'enseignement, suivis par des populations d'étudiants connues, sont fixes.

Le PECT concerne aussi la conception d'emploi du temps de cours sur une semaine standard. Cependant, les cursus ne sont pas figés afin d'offrir aux étudiants le maximum de choix possibles dans leurs modules. Les étudiants s'inscrivent aux cours, et la minimisation des conflits d'étudiants fait partie des contraintes souples à minimiser. Ces définitions de problèmes d'emploi du temps d'universités sont des versions simplifiées de problèmes réels.

Dans la compétition ITC-2019, le problème consiste, comme dans le cas du PECT de ITC-2007, à construire conjointement le planning de cours (salles, horaires) et à affecter les étudiants dans les cours. Les contraintes dures usuelles permettent de définir

l'admissibilité d'une solution (e.g. deux cours ne peuvent pas être planifiés dans une même salle). Les contraintes souples violées usuelles du PECT induisent des pénalités à minimiser pour obtenir des EDT de bonne qualité (e.g. éviter qu'un étudiant soit affecté à deux classes en même temps). Cependant le problème définit est plus proche de situation réelles en intégrant plus de données (e.g. l'horizon de planification est défini sur plusieurs semaines) et des contraintes qui se rapprochent plus de situation réelles (e.g. contraintes de distributions qui permettent d'intégrer les contraintes dures et souples des enseignants).

Nous présentons dans ce chapitre les différents types de problèmes de planification de cours d'universités (University Course TimeTabling Problem, UCTTP) qui sont étudiés dans la littérature ainsi que les différentes méthodes de résolution qui ont été appliquées pour les résoudre.

3.1 LE PROBLÈME CB-CTT

Nous présentons dans cette section le problème CB-CTT ainsi que les différents travaux qui l'ont traité dans la littérature.

3.1.1 *Définition du problème CB-CTT*

Le problème CB-CTT (*Curriculum Based Courses TimeTabling Problem*) consiste en la planification hebdomadaire des activités de cours d'université dans un nombre fixé de salles et de créneaux horaires. Les conflits entre les cours sont définis à partir des curricula publié par l'université. Un curriculum est un groupe de cours tel que chaque deux cours du groupe ont des étudiants en commun. Un cours représente une unité d'enseignement qui est suivie par des étudiants. Chaque cours est composé de plusieurs activités qui doivent être planifiées.

Nous présentons la formulation du problème CB-CTT comme elle a été définie dans la compétition ITC-2007 [Di Gaspero et al., 2007]. Cette formulation a été inspirée du problème de l'université d'UDINE en Italie.

Dans [Bonutti et al., 2012], un ensemble de variantes du CB-CTT, appelées UD1, UD2, UD3, UD4 et UD5, ont été proposées. Le problème défini ci-dessous correspond à UD2, qui est la variante la plus étudiée. Nous mentionnons que UD1 correspond à UD2 sans la contrainte souple de la stabilité de la salle et avec un poids différent pour la pénalisation de la compacité du curriculum. Les trois autres variantes ont été introduites dans le but d'inclure de nombreuses contraintes souples du monde réel.

Le problème CB-CTT a comme problème sous-jacent un problème de coloration de graphe, qui est un problème NP-difficile bien connu (voir [Burke et al., 2010a]). Considérons un graphe ayant un sommet pour chaque activité et une arrête pour

chaque paire d'activités qui ne peuvent pas être planifiées simultanément (car soit elles appartiennent au même cours soit à des cours du même curriculum ou qu'elles doivent être enseignées par le même enseignant.). Une couleur va correspondre à un créneau horaire. Le problème principal du CB-CTT est d'attribuer une couleur à chaque sommet, de manière à ce que des couleurs différentes soient attribuées aux sommets adjacents.

De plus, une contrainte est imposée sur le nombre maximal d'utilisations de chaque couleur, défini comme égal au nombre de salles disponibles. Ceci conduit au problème de coloration de graphe borné (*Bounded coloring problem*).

Les entités qui composent le problème CB-CTT sont présenté dans le chapitre 2. Rappelons qu'un curriculum est un groupe de cours tel que chaque deux cours du groupe ont des étudiants en commun. Sur la base du curricula, nous avons les conflits entre les cours et d'autres contraintes souples. La figure 3.1 illustre la structure des curricula. Chaque cours est nommé avec un C suivi des curriculums auxquels il appartient. Par exemple, le cours C125 appartient au curriculum 1, 2 et 5 et les cours C51, C125, C153, C453, C1235 et C2345 constituent le curriculum 5.

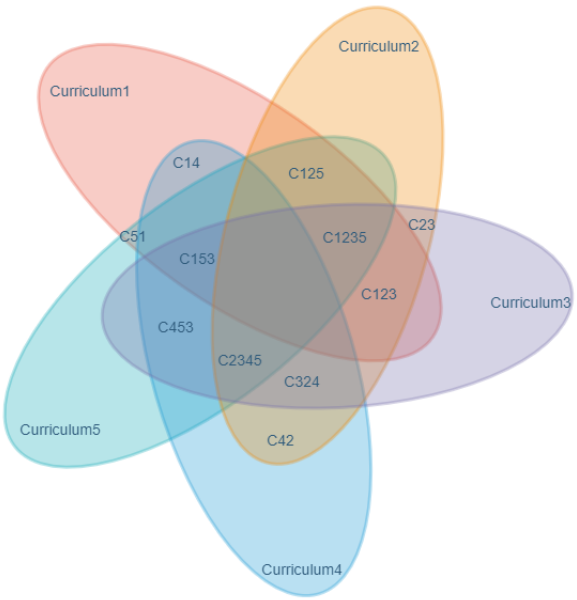

Fig 3.1: Structure des curricula.

Le problème consiste en à affecter un horaire (jour et créneau) et une salle pour toutes les activités de chaque cours.

Le problème est soumis à plusieurs autres contraintes qui peuvent être réparties en deux groupes de contraintes : les contraintes dures et les contraintes souples. Les contraintes dures sont des contraintes qu'il faut respecter strictement. La violation

d'une de ces contraintes rend la solution non réalisable. Les contraintes souples sont des contraintes de bonne pratiques qui participent à la mesure de la qualité d'une solution. La violation d'une contraintes souple induit une pénalité.

3.1.2 *Les contraintes dures du CB-CTT*

Les contraintes dures utilisées dans le problème CB-CTT sont les suivantes :

Activités *(Lectures)* : toutes les activités d'un cours doivent être planifiées à des périodes distinctes. Une violation se produit si une activité n'est pas planifiée.

Occupation d'une salle *(RoomOccupancy)* : deux activités ne peuvent pas avoir lieu dans la même salle au même horaire. Deux activités dans la même salle au même horaire représentent une violation. Toute activité supplémentaire au même horaire et dans la même salle compte pour une violation supplémentaire.

Conflits *(Conflicts)* : les cours d'un même curriculum ou enseignés par le même enseignant doivent tous être programmés à des horaires différents. Deux activités en conflits au même horaire représentent une violation. Trois activités en conflit comptent pour 3 violations : une pour chaque paire.

Disponibilités *(Availabilities)* : si l'enseignant du cours n'est pas disponible pour enseigner ce cours à un horaire donné, aucune activité ne peut être programmé à cet horaire. Chaque activité d'un enseignant non disponible pour ce cours représente une violation.

3.1.3 *Les contraintes souples du CB-CTT*

Les contraintes souples utilisées dans le problèmes CB-CTT sont les suivantes :

Capacité de la salle *(RoomCapacity)* : pour chaque activité, le nombre d'étudiants participant au cours doit être inférieur ou égal au nombre de places de toutes les salles accueillant ses activités. Chaque étudiant en plus de la capacité compte pour 1 point de pénalité.

Nombre minimum de jours de travail *(MinimumWorkingDays)* : les activités de chaque cours doivent être réparties sur le nombre minimum de jours spécifiés pour ce cours. Chaque jour en dessous du minimum compte pour 5 points de pénalité.

La compacité d'un curriculum *(CurriculumCompactness)* : les activités qui appartiennent à un curriculum doivent être adjacentes les unes aux autres, c'est-à-dire planifiés pendant des horaires consécutifs. Pour un curriculum donné, une violation est comptée

chaque fois qu'une activité n'est adjacente à aucune autre activité dans le même jour (un point de pénalité). Chaque activité isolée dans un curriculum compte pour 2 points de pénalité.

Stabilité d'une salle *(RoomStability)* : toutes les activités d'un cours doivent se dérouler si possible dans la même salle. Chaque salle distincte utilisée pour les activités d'un cours, sauf la première, compte pour 1 point de pénalité.

3.1.4 *Présentation du benchmark ITC-2007 relatif au CB-CTT (Track 3)*

Le benchmark ITC-2007 Track 3 qui concerne le problème CB-CTT a été fournis lors de la compétition ITC-2007. Le benchmark contient 21 instances nommées COMP01 à COMP21 de différentes tailles et caractéristiques. Le tableau 3.1 présente une synthèse des caractéristiques des instances. La colonne *Events* représente le nombre d'activités à planifier. La colonne *Periods* représente le nombre de créneaux disponibles. La colonne *Rooms* représente le nombre de salles. La colonne *Couses* représente le nombre de cours. La colonne *Curricula* représente le nombre de curriculum.

Tableau 3.1: Présentation du benchmark ITC-2007 Track 3.

Instance	Events	Periods	Rooms	Courses	Curricula
COMP01	160	30	6	30	14
COMP02	283	25	16	82	70
COMP03	251	25	16	72	68
COMP04	286	25	18	79	57
COMP05	152	86	9	54	139
COMP06	361	25	18	108	70
COMP07	434	25	20	131	77
COMP08	324	25	18	86	61
COMP09	279	25	18	76	75
COMP10	370	25	18	115	67
COMP11	162	46	5	30	13
COMP12	218	36	11	88	150
COMP13	308	25	19	82	66
COMP14	275	25	17	85	60
COMP15	251	25	16	72	68
COMP16	366	25	20	108	71
COMP17	339	25	17	99	70
COMP18	138	36	9	47	52
COMP19	277	25	16	74	66
COMP20	390	25	19	121	78
COMP21	327	25	18	94	78

3.1.5 *Extensions du CB-CTT de l'ITC-2007*

La formulation du CB-CTT proposée dans l'ITC-2007 à été simplifiée afin de respecter un certain niveau de généralité. Plusieurs extensions ont été proposée pour le problème notamment dans [Di Gaspero et al., 2007] et [Bonutti et al., 2012]. Parmi ces extensions nous trouvons :

- prendre en considération la pause déjeuner pour les étudiants : au minimum un créneau libre autour du temps usuel de déjeuner ;
- pénaliser des schémas d'affectation plus que d'autre dans les curricula vis à vis de la contrainte souple (Curriculum Compactness) ;
- ajouter une charge quotidienne maximale d'étudiants pour chaque curriculum ;
- certaines activités doivent (ne doivent pas) être affectées à des horaires consécutifs ;
- certaines salles peuvent ne pas être disponibles à des horaires spécifiques et peuvent ne pas convenir à certaines activités ;
- pénaliser l'affectation des activités aux salles qui sont très grandes par rapport au nombre d'étudiants de l'activité en question ;
- modifications de la manière de calcul de pénalités dans les contraintes souples et les rendre plus complexes ;
- ajouter les préférences des enseignants ;
- prendre en considération le temps de parcours entre deux salles pour deux activités successives.

3.1.6 *Méthodes de résolution exactes*

[Burke et al., 2010b] ont proposé une formulation sous forme de programme linéaire en nombres entiers (ILP) pour le problème CB-CTT tel que défini dans l'ITC-2007. Les auteurs décrivent leur formulation comme une formulation monolithique qui pourra être résolue par n'importe quel solveur ILP jusqu'à optimalité en assumant que le temps d'exécution est suffisant. Par contre, même en allouant à l'ILP des temps d'exécution suffisamment grand (allant jusqu'à plusieurs jours), il n'arrive pas à trouver des solutions optimales pour toutes les instances.

Les auteurs proposent plusieurs bornes inférieures dérivées de la formulation monolithique qui peuvent être produites en des temps d'exécutions raisonnables de l'ILP. Les auteurs ont proposés deux méthodes pour calculer les bornes inférieures.

La première méthode consiste à résoudre la formulation monolithique en ignorant les contraintes souples *RoomCapacity* et *RoomStability* et en ajoutant des contraintes supplémentaires afin de limiter le nombre de salles utilisées à n'importe quel horaire. Les bornes inférieures obtenues avec la première méthode ont été bien meilleures que celles obtenues avec la formulation monolithique brute et en des temps d'exécution nettement moindres.

La première méthode peut être vue comme étant une agrégation de l'ensemble des salles en un ensemble de salles de tailles égale à l'ensemble initial et de capacité égale à la capacité de la plus grande salle.

La deuxième méthode propose une agrégation intermédiaire, qui divise l'ensemble initial de salles en deux ensembles de salles de capacité plus grande ou plus petite à un certain seuil. Cette décomposition permet de réduire le nombre de variables et de contraintes dans la formulation résultante mais en considérant toutes les contraintes souples dans la fonction objectif. Les résultats obtenus ont été meilleurs en moyenne que ceux de la première méthode de calcul de bornes inférieures.

[Burke et al., 2012] ont proposé une méthode de branch-and-cut pour résoudre le CB-CTT jusqu'à optimalité. Cette méthode est inspiré de la formulation monolithique proposée dans [Burke et al., 2010b] pour qui les solutions sont des bornes inférieures au problème CB-CTT d'origine. La nouvelle formulation diffère de la formulation monolithique dans la manière de modéliser la contrainte souple *(CurriculumCompactness)*. Les auteurs observent que cette nouvelle formulation relaxée permet de réduire les pénalités dues aux activités isolées d'un même curriculum à 1 alors que la nombres de ses activités peut être relativement grand. Afin d'aboutir à une formulation valide et complète, les auteurs proposent d'ajouter un nombre exponentiel (en terme de nombre d'horaires par jour) d'inégalités qu'ils appellent *Cuts from Event/Free-Period Patterns*. Les auteurs ont pu résoudre deux instances jusqu'à optimalité et ils ont obtenus des bornes inférieures pour les autres instances.

[Burke et al., 2012] ont aussi introduit trois classes d'inégalités valides afin d'améliorer les performances du modèle relaxé. Ces inégalités sont les suivantes :

Implied Bounds (IB) **:** ces coupes définissent des bornes inférieures/supérieures sur un sous-ensemble de variables. Ces coupes sont déduites à partir du modèle mathématique du problème ;

Days of Instruction (DI) **:** ces coupes prennent en considération les contraintes souples qui pénalisent l'insuffisance de nombre de jours distincts d'enseignement d'un cours.

Clique (CLI) **:** ces coupes indiquent qu'au plus une activité d'un sous-ensemble sélectionné de cours peut avoir lieu à un horaire donné dans n'importe quelle salle.

[Hao et al., 2011] ont proposée une approche de partitionnement qui est basée sur le principe de "diviser pour régner" (*divide-and-conquer principle*) afin de trouver des bornes inférieure pour les instances du problème. A partir du problème initial, un sous-ensemble de contraintes est sélectionné de manière a ce que le problème résultant soit décomposable en de petits sous-problèmes indépendants. Chaque sous-problème est résolu en utilisant le modèle ILP proposée par [Lach et al., 2012]. Une borne inférieure valide pour le problème initial est obtenues en additionnant les coûts des solutions optimales des sous-problèmes. Le partitionnement est effectué en utilisant un algorithme de recherche tabou itérative.

[Cacchiani et al., 2013] ont présenté une nouvelle formulation pour le problème CB-CTT qu'ils appelent *Two Weekly Schedule Types (2WST)*. Cette formulation à été introduite afin de calculer des bornes inférieures pour les instances du problème. La méthode proposée a permis de trouver des résultats intéressants, notamment de nouvelles bornes inférieures en divisant le problème initial en deux sous problèmes formulés comme des ILP. Le premier sous problème traite les assignations des activités aux salles et aux horaires et prend en compte les pénalités dues aux contraintes souples *RoomCapacity* et *RoomStability*. Le deuxième sous problème traite les assignations des activités aux horaires et prend en compte les pénalités dues aux contraintes souples *MinimumWorkingDays* et *CurriculumCompactness*. Les ILP résultants sont résolus en utilisant une méthode par génération de colonnes.

[Achá et al., 2014] ont proposé différents encodages du problème CB-CTT afin que le problème puisse être traité par des solveurs SAT. Ces encodages diffèrent par les sous-ensembles de contraintes dures et souples modélisées et par la manière dont ces contraintes sont définies. Les encodages proposés peuvent fournir des bornes inférieures ou supérieures au problème CB-CTT. Tous les encodages sont basés sur un codage SAT de base, dans lequel toutes les contraintes (dures et souples) du CB-CTT sont définies dans le problème SAT résultant comme étant dures. Parmi ces encodages, on retrouve :

- relaxer la contrainte souple *CurriculumCompactness* et résoudre le problème résultant avec un solveur *Partial MaxSAT* ;
- relaxer les contraintes souples *CurriculumCompactness* et *MinimumWorkingDays* et résoudre le problème résultant avec un solveur *Weighted Partial MaxSAT* ;
- relaxer toutes les contraintes souples et résoudre le problème résultant avec un solveur *Partial MaxSAT* ;
- relaxer toutes les contraintes souples et résoudre le problème *Partial MaxSAT* résultant avec un algorithme de branch-and-bound.

L'encodage qui relaxe toutes les contraintes souples présente les meilleurs résultats. Ce dernier à permis d'obtenir des bornes inférieures valides ainsi que 11 solutions

optimales pour les instances du CB-CTT de l'ITC-2007.

3.1.7 *Méthodes de résolution heuristiques*

3.1.7.1 *Heuristiques basées sur les ILP*

[Lach et al., 2012] ont proposé une modélisation en ILP pour le problème CB-CTT. La modélisation est faite en deux phases séquentielles : une première phase pour l'affectation des cours aux horaires et une deuxième pour l'affectation des salles. Dans la première phase, les contraintes souples *RoomCapacity*, *MinimumWorkingDays* et *CurriculumCompactness* sont prises en considération tandis que la contrainte souple *RoomStability* est considérée dans la deuxième phase.

Les solutions obtenues avec ce modèle peuvent être optimales dans certains cas et sous certaines conditions, mais généralement elles ne le sont pas car la contrainte souple *RoomStability* n'est considérée que dans la deuxième phase et l'affectation des activités aux horaires est faite sans en tenir compte. Aussi, les solutions finales de la première phase sont des bornes inférieures valides pour le problème CB-CTT global.

Les auteurs ont utilisé CPLEX comme solveur ILP et ont pu résoudre jusqu'à optimalité 4 instances du benchmark UDINE. Ils proposent aussi une extension de leur modèle afin de traiter d'autres contraintes souples qui ne sont pas présentes dans la formulation ITC2007.

3.1.7.2 *Métaheuristiques*

[Di Gaspero et al., 2002] proposent un algorithme de recherche tabou ainsi que plusieurs structures de voisinages pour des méthodes de recherche locale. Les structures proposées sont les suivantes :

- ***Neighborhood union*** : dans cette structure, l'union de nombreux voisinages est prise en compte et, à chaque itération, l'algorithme de recherche locale sélectionne un mouvement appartenant à l'un des composants ;
- ***Neighborhood composition*** : dans cette structure, des chaînes de mouvements appartenant à différents voisinages sont considérées ;
- ***Token-ring search*** : dans cette structure, partant d'un état initial et d'un ensemble d'algorithmes basés sur différentes fonctions de voisinages, la recherche dite par 'jeton' exécute circulairement chaque algorithme, en partant toujours de la meilleure solution trouvée par le précédent.

L'espace de recherche est composé des assignations pour lesquelles toutes les activités sont planifiées et attribuées à des horaires distincts, et les contraintes de

disponibilité d'enseignants (*Availabilities*) sont respectées. Les violations des autres contraintes dures sont pénalisées dans la fonction objectif, de même que les contraintes souples.

Deux types de voisinages sont considérés: le premier est défini en changeant l'horaire attribué à une activité d'un cours donné par un nouvel horaire qui satisfait les contraintes de disponibilité ; le second est défini en changeant la salle attribuée à une activité à un horaire donné. Des algorithmes de *Hill Climbing* et de recherche tabou sont comparés sur ces deux types de voisinages, sur leur union, sur leur composition, et combinés dans une recherche par jeton. Les algorithmes sont testés sur quatre instances de l'université Udine (test1 – test4).

Une recherche tabou adaptative est proposée dans [Lü et al., 2010]. L'algorithme comprend trois phases : une phase d'initialisation, une phase d'intensification et une phase de diversification. Dans la phase d'initialisation, un algorithme glouton construit un planning réalisable. L'algorithme part d'un planning vide et sélectionne de manière itérative une activité d'un cours (les cours avec un petit nombre d'horaires disponibles et un grand nombre d'activités non affectées sont prioritaires) et lui attribue un horaire et une salle.

Dès qu'une affectation initiale réalisable est atteinte, les phases d'intensification et de diversification combinées de manière adaptative sont utilisées pour réduire le nombre de violations de contraintes souples. La phase d'intensification consiste en un algorithme de recherche tabou qui exploite deux voisinages : l'un consiste à échanger les horaires et les salles attribués à deux activités de cours différents ; l'autre combine des mouvements définis par l'interchangeabilité de deux chaînes de *Kempe* (voir [Lü et al., 2010]).

Lorsque la recherche s'arrête à un optimum local, la recherche tabou est redémarrée à partir de cet optimum local, mais en utilisant l'autre voisinage. Ce processus est répété jusqu'à ce qu'aucune amélioration ne soit possible. La recherche locale itérée fournit le mécanisme de diversification permettant d'orienter la recherche pour échapper à l'optimum local actuel. Pour détruire la solution optimale locale atteinte, l'algorithme utilise un opérateur de perturbation guidé par des pénalités (voir [Lü et al., 2009]).

[Bellio et al., 2016] ont proposé un algorithme de recuit simulé. Deux voisinages sont utilisés :

- le déplacement d'une activité d'un horaire/salle à un autre horaire et/ou d'une autre salle ;
- l'échange d'horaires et de salles de deux activités de cours distincts.

L'algorithme utilise un paramètre de taux de permutation (*swap rate parameter*) pour contrôler la fréquence de sélection du deuxième voisinage par rapport au premier. L'algorithme consiste en un recuit simulé en une étape, renforcé par deux caractéristiques : un schéma de refroidissement de température basé sur la coupure (*cutoff-based temperature cooling scheme*) et une condition d'arrêt basée sur le nombre maximal d'itérations autorisées. Les auteurs ont aussi proposé une analyse statistique approfondie qui est capable de déterminer un modèle de régression linéaire entre les caractéristiques de l'instance et les paramètres de la méthode de recherche, ce qui permet de définir les paramètres de nouvelles instances sur la base d'une simple inspection des caractéristiques des anciennes instances.

[Müller, 2009] le gagnant de la compétition ITC-2007, à utilisé une approche hybride pour traiter le problème CB-CTT. L'approche a été utilisée pour traiter les trois tracks de la compétition.

L'approche proposée consiste en plusieurs phases : la première phase à pour but de trouver une solution réalisable en utilisant une méthode de recherche itérative avancée (*Iterative Forward Search*). Ensuite, la méthode recherche l'optimum local autour de la solution trouvée dans la phase 1 en utilisant la méthode de *Hill Climbing*. Une fois l'optimum local trouvée, la dernière phase est lancée. Cette phase injecte la solution finale de la phase 2 comme solution de départ d'un algorithme de *Great Deluge*.

3.1.7.3 *Matheuristiques*

[Lindahl et al., 2018] ont proposé une matheuristique pour traiter le problème CB-CTT. L'objectif de ce travail est de combiner les ILP avec des heuristiques, une revue de ces méthodes est présentée dans [Bixby, 2012].

Les auteurs sont les premiers à avoir traiter le problème CB-CTT avec une matheuristique. Ils se sont basés sur le modèle linéaire proposé dans [Lach et al., 2012]. La matheuristique proposée part d'une solution initiale, et qui à chaque itération, elle essaye d'améliorer cette solution en résolvant l'ILP de la phase 1 de [Lach et al., 2012] en fixant une partie des variables de décision. Plusieurs types de voisinages (manière de fixer et de libérer les variables) sont présentés. La phase 2 est lancée ensuite. Les auteurs ont utilisé Gurobi comme solveur.

La matheuristique proposée obtient de meilleurs résultats que les vainqueurs de la compétition [Müller, 2009]. La matheuristique proposée a trouvé des solutions de bonne qualité, cependant, comparée aux métaheuristiques de la littérature [Abdullah et al., 2012] [Kiefer et al., 2017], la méthode reste bonne mais pas aussi performante qu'eux.

3.1.8 *Conclusion CB-CTT*

Nous avons présenté dans cette section le problème CB-CTT de ITC-2007, sa définition ainsi que les différentes méthodes de résolution qui ont été utilisée pour le résoudre dans la littérature.

3.2 LE PROBLÈME PECT

Nous présentons dans cette section le problème PECT ainsi que les différents travaux qui l'ont traité dans la littérature.

3.2.1 *Définition du problème PECT*

Le problème PECT (*Post Enrollement Course Timetabling*) recouvre un type de problèmes de planification de cours d'universités. Les étudiants ont la liberté de choisir les cours qu'ils veulent suivre durant un semestre d'enseignement, les emplois du temps sont ensuite construits à partir des choix des étudiants. Nous décrivons le problème PECT qui a été introduit lors de la compétition ITC-2007 [Lewis et al., 2007].Des évènements sont à planifier dans des salles et des créneaux tout en respectant un ensemble de contraintes dures et souples. Les données du problème sont présentées dans le chapitre 2.

L'objectif est de placer les n évènements. Il faut affecter à chaque évènement une salle et un créneau en respectant les contraintes dures.

Comme la faisabilité des solutions est difficile à obtenir, la description du problème permet la violation de contraintes dures. Nous distinguons deux types de plannings :

- **Planning valide :** un planning est valide s'il n'y a aucune violation de contraintes dures, mais des évènements peuvent restés non planifiés, c'est-à-dire qu'ils ne vont pas apparaître dans le planning ;
- **Planning réalisable :** un planning est réalisable s'il n'y a aucune violation de contraintes dures et tous les évènements sont planifiés, tous les évènements apparaissent dans le planning.

La distances à la faisabilité (*distance to feasibility*) est définie comme étant le nombre d'étudiants qui doivent assister à des évènements qui ne sont pas planifiés. Par exemple, si nous avons 50 étudiants et que parmi eux il y a 10 étudiants qui doivent assister à des évènements qui ne sont pas planifiés, alors la distance à la faisabilité de cette solution est 10.

3.2.2 *Les contraintes dures du PECT*

Les contraintes dures sont des contraintes que toute solution réalisable doit satisfaire. Les différentes contraintes dures utilisées dans le problèmes PECT sont les suivantes :

Conflicts **:** les événements qui ont des étudiants communs ne peuvent pas être planifiés dans le même créneau.

Compatibility **:** un événement ne peut pas être planifié dans une salle qui ne respecte pas l'une des caractéristiques nécessaires à l'événement, ou dans une salle dont la capacité est inférieure au nombre d'étudiants participant à l'événement.

Occupancy **:** pour chaque créneau de la semaine, un seul évènement pourra être planifié dans une salle donnée.

Availability **:** un évènement peut être planifié dans un créneau donné si et seulement si ce dernier à été défini comme étant disponible pour cet évènement.

Precedence **:** les évènements doivent apparaître tels que défini dans l'ensemble *Precedence*, c'est-à-dire qu'un évènement a pourra être placé avant un évènement b sauf si le couple (b,a) n'apparaît pas dans l'ensemble *Precedence*.

Notons que les trois premières contraintes dures (*Conflicts*, *Compatibility* et *Occupancy*) sont les mêmes contraintes qui ont été introduites dans le problème défini dans l'ITC-2002 et les deux autres sont de nouvelles contraintes.

3.2.3 *Les contraintes souples du PECT*

Les contraintes souples utilisées dans le problèmes PECT sont les suivantes :

Late Events **:** un étudiant ne doit pas assister à un événement dans le dernier créneau d'une journée. Les derniers créneaux dans une semaine sont les créneaux 9, 18, 27, 36 et le créneau 45. Pour chaque événement planifié dans le dernier créneau, nous calculons la somme du nombre d'étudiants qui doivent y assister. Chaque étudiant compte pour un point de pénalité.

Consecutive Events **:** un étudiant ne doit pas assister à plus de deux événements consécutifs dans une journée. Le dernier créneau d'une journée et le premier du jour suivant ne sont pas considérés comme consécutifs. Pour chaque jour et pour chaque étudiant, nous calculons la somme des événements consécutifs postérieurs au second. Par exemple, si 3 étudiants doivent assister à 4 événements consécutifs dans une journée, la pénalité est de 3*(4-2)=6.

Isolated Events **:** un étudiant ne doit pas assister à un seul événement au cours de la journée. Pour chaque jour, nous additionnons le nombre d'étudiants devant assister à des événements isolés.

La fonction objectif considérée est constituée de trois termes. Chaque terme représente une des trois contraintes souples décrites précédemment. Chaque violation d'une contrainte souple compte pour un point de pénalité.

En conclusion, la qualité d'une solution est évaluée avec une fonction d'évaluation composée de deux mesures : la distance à la faisabilité (*distance to feasibility*) et la fonction objectif.

L'évaluation est hiérarchique car les solutions avec de petites distances à la faisabilité sont considérées comme meilleures. Si deux solutions ont la même distance à la faisabilité, alors la solution avec la plus petite valeur de fonction objectif est meilleure.

3.2.4 *Présentation du benchmark ITC-2007 relatif au PECT (Track 2)*

Le benchmark ITC-2007 Track 2 qui concerne le problème PECT a été fournis lors de la compétition ITC-2007. Le benchmark contient 24 instances nommées COMP01 à COMP24 de différentes tailles et caractéristiques. Le tableau 3.2 présente une synthèse des caractéristiques des instances. La colonne *Events* représente le nombre d'activités à planifier. La colonne *Rooms* représente le nombre de salles. La colonne *Features* représente le nombre de caractéristiques. La colonne *Students* représente le nombre d'étudiants. La colonne *St/Ev* représente le nombre maximum d'étudiants par activité. La colonne *Ev/St* représente le nombre maximum d'activités par étudiant. La colonne *Ft/R* représente le nombre moyen de caractéristiques par salle. La colonne *Ft/Ev* représente le nombre moyen de caractéristiques par activité.

3.2.5 *Les extensions du problème PECT*

Pour la compétition ITC-2007, les concepteurs du problème PECT ont ignoré des contraintes du monde réel afin de simplifier le problème. Cependant, des contraintes peuvent être ajoutées au problème initial afin d'approcher des situations réelles, notamment des contraintes relatives au positionnement des évènements, des contraintes qui concernent les salles ainsi que des contraintes qui concernes les préférences des intervenants. Nous présentons quelques contraintes supplémentaires au problème PECT décrites dans [Lewis et al., 2007].

Inter-site travel times **:** une université peut être divisée en plusieurs sites. Les étudiants et les enseignants peuvent avoir besoin de temps de trajet pour se déplacer d'un site à un autre. Ainsi, si deux événements i et j ont des étudiants en communs, mais doivent avoir lieu dans des sites différents, alors la contrainte "si l'événement i

Tableau 3.2: Présentation du benchmark ITC-2007 Track 2.

Inst	Events	Rooms	Features	Students	St/Ev	Ev/St	Ft/R	Ft/Ev
COMP01	400	10	10	500	33	25	3	1
COMP02	400	10	10	500	32	24	4	2
COMP03	200	20	10	1000	98	15	3	2
COMP04	200	20	10	1000	82	15	3	2
COMP05	400	20	20	300	19	23	2	1
COMP06	400	20	20	300	20	24	3	2
COMP07	200	20	20	500	43	15	5	3
COMP08	200	20	20	500	39	15	4	3
COMP09	400	10	20	500	34	24	3	1
COMP10	400	10	20	500	32	23	3	2
COMP11	200	10	10	1000	88	15	3	1
COMP12	200	10	10	1000	81	15	4	23
COMP13	400	20	10	300	20	24	2	1
COMP14	400	20	10	300	20	24	3	1
COMP15	200	10	20	500	41	15	2	3
COMP16	200	10	20	500	40	15	5	3
COMP17	100	10	10	500	195	23	4	2
COMP18	200	10	10	500	65	23	4	2
COMP19	300	10	10	1000	55	14	3	1
COMP20	400	10	10	1000	40	15	3	1
COMP21	500	20	20	300	16	23	3	1
COMP22	600	20	20	500	22	25	3	2
COMP23	400	20	30	1000	69	24	5	3
COMP24	400	20	30	1000	41	15	5	3

doit se produire dans l'intervalle de temps x, alors l'événement j ne peut pas se produire dans l'intervalle de temps y le même jour que si (y-x) est supérieur au temps de trajet entre les sites i et j" pourrait être spécifié.

Lunch Break **:** une université peut donner une obligation de donner une pause déjeuner pour ses étudiants.

Relative Timing of Events **:** des contraintes sur le placements des évènements peuvent être envisagées. Des contraintes du genre deux évènements i et j doivent être affectés au même/différent créneau ou au même/différent jour peuvent être considérées par certaines universités.

Events without Rooms **:** certains événements peuvent ne pas nécessiter de salle, ils peuvent avoir lieu à l'extérieur ou impliquer des déplacements vers des sites hors campus par exemple.

Room availability **:** certaines salles peuvent ne pas être disponibles durant certains créneaux. Par exemple, une salle peut être réservée par une autre faculté ou à un autre usage.

Room Hierarchies **:** dans de nombreux établissements, une grande salle peut être divisée en plusieurs salles de classe plus petites à l'aide de séparateurs amovibles. Dans un intervalle de temps, la salle peut être utilisée par un événement avec beaucoup d'étudiants, tandis que dans un autre intervalle de temps, des événements plus petits peuvent être planifiés dans cette salle simultanément.

Filling Rooms **:** une université peut avoir une politique pour placer les évènements dans des salles adéquates par rapport aux nombres d'étudiants de l'évènement et la capacité de la salle. Par exemple, un grand amphi ne peut pas accueillir un événement de 12 étudiants seulement.

Free days **:** il peut être jugé souhaitable dans certaines universités de permettre aux étudiants d'avoir un jour par semaine sans cours afin de laisser du temps pour d'autres activités telles que la recherche, le sport ou les activités des clubs universitaires.

Lecturer Preferences **:** il peut également y avoir un certain nombre d'exigences individuelles de la part des enseignants concernant la répartition de leurs heures d'enseignement. Certains enseignants, par exemple, peuvent préférer faire tout leur enseignement en une seule journée tandis que d'autres peuvent préférer que leurs heures soient réparties également tout au long de la semaine. Ces contraintes se rencontrent dans les institutions d'enseignement de nos jours.

Nous trouvons dans la littérature plusieurs travaux qui portent sur des extensions du problème PECT et leurs applications dans des cas réels de conception d'emplois du temps de cours des universités.

[Méndez-Díaz et al., 2016] ont proposé un modèle linéaire et une heuristique basée sur ce modèle pour traiter une généralisation du problème PECT qui s'applique au problème d'emploi du temps de cours de l'université de Buenos Aires en Argentine. Le problème étudié se distingue par la modélisation hiérarchique des cours. Chaque cours est composée de plusieurs *Commissions*. Chaque étudiant qui suit le cours doit être affecté à une seule *Commission*. Une autre spécificité du problème est la liste de préférence des étudiants. Chaque étudiant à une liste de préférence de cours. Une borne inférieure et une borne supérieure sont fournies avec cette liste et l'étudiant doit être affecté à un nombre de cours entre ces deux bornes. Les auteurs ont proposé un modèle linéaire qui résout le problème en deux phases. La première phase affecte les activités aux salles et aux créneaux et la deuxième phase affecte les étudiants a leurs cours. Les auteurs proposent aussi une heuristique basée sur leur modèle linéaire. La méthode proposée a été testé sur les instances que les auteurs ont obtenu de l'université de Buenos Aires et elle a produit des solutions de bonne qualité.

3.2.6 *Méthodes de résolution exactes*

[Broek et al., 2012] ont proposé deux formulations ILP pour le PECT de l'ITC-2007. La première formulation ne prend en compte que les contraintes dures. Les auteurs affirment que la résolution de cette formulation sans les contraintes souples n'est pas possible dans le temps de calcul autorisé par la compétition ITC-2007 et que le solveur utilisé CPLEX n'arrive même pas à commencer à faire un branchement dans ce temps de calcul autorisé. Les auteurs ont appliqué la génération de colonnes sur une extension de cette formulation afin de construire des solutions pour le problème.

La deuxième formulation proposée par [Broek et al., 2012] est une formulation compacte sous forme de l'ILP. Cette formulation prend en compte toutes les contraintes dures et souples. La fonction objectif se compose de deux parties. La première partie de la fonction objectif vise à ce que le nombre d'étudiants auxquels un événement demandé n'est pas planifié soit minimisé, c'est-à-dire que l'objectif est d'arriver à un planning réalisable où tous les évènements sont planifiés. La deuxième partie se compose des trois termes qui représentent le nombre de violations de contraintes souples. L'objectif le plus important est de trouver une solution avec une distance à la faisabilité égale zéro. Pour cela, les auteurs ont ajouté des poids pour chaque partie de la fonction objectif afin de contrôler l'ILP. Donner un poids important à la première partie de la fonction objectif permet d'aboutir à des solutions avec une meilleure distance à la faisabilité.

[Broek et al., 2012] ont proposée une heuristique basée sur la programmation linéaire afin de traiter le problème PECT de l'ITC-2007. L'heuristique proposée par les auteurs est constitué de deux phases : une phase de construction et une phase d'amélioration. La première phase de construction se base sur la première formulation proposée par les auteurs qui est présentée dans la section précédente. Dans cette première phase on résout une relaxation de cette formulation d'une manière itérative en utilisant la génération de colonnes. Durant chaque itération, un certain nombre d'évènements est assigné à des créneaux et on relance la résolution de l'ILP relaxé à partir de cette dernière solution. En procédant ainsi, l'heuristique garantie l'affectation des évènements tout au long de la semaine. La première phase consomme généralement une petite partie du temps d'exécution alloué.

La deuxième phase démarre avec la dernière solution trouvée dans la première phase et qui sera généralement une solution valide pour laquelle tous les évènements ne sont pas tous placés. Cette solution est injectée comme solution initiale dans l'ILP de la deuxième formulation présentée précédemment. Même si la résolution de l'ILP en entier est impossible dans le temps alloué, les auteurs l'ont utilisée dans un framework heuristique afin de construire des solutions partielles d'une manière itérative. Les résultats trouvés par la première phase sont des solutions valides qui respectent toutes les contraintes dures pour 23 des 24 instances. Les résultats finals de la méthode restent compétitifs avec les finalistes de la compétition.

3.2.7 *Métaheuristiques*

[Jaengchuea et al., 2015] ont proposé un algorithme de recherche tabou qui a été utilisé pour la génération de nouveaux individus dans leur méthode. Les auteurs affirment que la recherche taboue a été très efficace dans leur framework de résolution et a permis la génération de nouveaux individus de bonne qualité. Les auteurs ont utilisé deux structures de voisinages :

- **N1 :** choisir un évènement aléatoirement et l'affecter à un nouveau créneau ;
- **N2 :** choisir deux évènements aléatoirement et intervertir leurs créneaux.

[Lewis, 2012] a présenté une méthode de résolution en trois phases pour le problème PECT. La première phase est une phase constructive qui permet de construire une solution initiale. Cette première phase est suivie de deux phases distinctes de recuit simulé. Le temps d'exécution de chaque phase est défini d'une manière automatique en se basant sur le temps général alloué. L'idée principale de cette méthode est d'organiser les contraintes selon différents niveaux d'importance dans les différentes phases du processus de solution. Les résultats obtenus par la méthode ont été comparable par rapport aux finalistes de la compétition mais restent de moindres qualité.

[Ceschia et al., 2012] ont proposée une méthode de récuit simulé afin de traiter plusieurs variantes du problème PECT présenté dans l'ITC-2007. Les variantées traités sont la version originale du problème de l'ITC-2002, la version standard du problème de l'ITC-2007 ainsi que la version *HARD ONLY* du problème qui ne considère pas les contraintes souples. Les auteurs ont proposé 5 types de pré-traitements afin d'améliorer les performances de leur approche.

La méthode proposée est une méthode de récuit simulé standard qui part d'une solution initiale qu'on améliorer d'une manière itérative. Deux méthodes de construction de solutions initiales ont été proposée. La première méthode est une méthode de construction gloutonne qui part d'un planning vide et qui assigne aléatoirement à chaque itération un évènement à un créneaux et une salle libre. La deuxième méthode est similaire à la première, néanmoins, elle essaye de rendre le nombre d'évènements non planifiés minime. Deux structures voisinages ont été introduites par les auteurs dans leur travail :

- **MoveEvent (ME) :** choisir un évènement aléatoirement et l'affecter à un nouveau créneau. Le mouvement est possible que si le nouveau créneau est admissible pour l'évènement en question et qu'il existe une salle libre adéquate durant ce créneau ;
- **SwapEvents (SE) :** choisir deux évènement aléatoirement et intervertir leurs créneaux. Le mouvement est possible que si les deux créneaux sont admissibles pour les deux évènements et qu'il existe deux salles libres adéquates dans les nouveaux créneaux pour chaque évènement.

Les résultats obtenus par cette méthode ont été meilleures que celle des participants à la compétition pour 9 instances et ils ont pu obtenir la 2 ème place en se basant sur la méthode de classement proposée dans l'ITC-2007.

3.2.8 *Méthodes hybrides*

[Cambazard et al., 2012] ont remporté le track réservé au problème PECT de la compétition ITC-2007. Les auteurs ont proposé trois méthodes pour résoudre le problème : deux méthodes hybrides et une méthode basée sur la programmation par contraintes. La première méthode est une méthode hybride qui combine la recherche locale, la recherche tabou et le recuit simulé. Au début, une recherche locale est effectuée sur des solutions générées aléatoirement pour trouver une solution réalisable. Une liste tabou est maintenue pour empêcher qu'un événement ne soit assigné aux mêmes créneaux pour les k dernières itérations. Plusieurs structures de voisinages ont été utilisées : déplacer un événement vers un créneau vide, échanger deux événements, échanger deux créneaux et le déplacement hongrois [Kuhn, 1955]. La solution réalisable trouvée est améliorée ensuite par une méthode de recuit simulé avec réchauffement. La deuxième méthode proposée est similaire à la première mais avec une relaxation sur les contraintes de salles. Une fois que la meilleure solution avec relaxation est trouvée, une heuristique de correction est appliquée afin de corriger les violations de contraintes de salles. Une recherche locale est lancée à la fin en utilisant les mêmes structures de voisinage que la première méthode afin d'améliorer la solution. Les auteurs ont indiqué que la deuxième méthode était la meilleure pour trouver des solutions réalisables. Une méthode basée sur la programmation par contraintes a également été proposée par auteurs mais elle était moins compétitive par rapport aux méthodes de recherche locale.

[Jaengchuea et al., 2015] ont utilisée une méthode hybride qui combine la recherche locale et la recherche tabou avec les algorithmes génétiques pour traiter le problème PECT. La recherche locale est utilisée dans la phase d'initialisation de l'algorithme génétique afin d'améliorer la qualité des individus. Elle est aussi utilisée avec la recherche tabou à chaque itération avec les opérateurs de croisement et de mutation pour la génération des nouveaux individus. Les solutions trouvées par la méthode ont été très bonnes. Les auteurs obtiennent de meilleures solutions que celles trouvées par les participants à la compétition ITC-2007.

[Goh et al., 2017] ont proposé une méthode heuristique itérative en deux phases qui combine plusieurs méthodes de recherche locale. Dans la première phase, une méthode de recherche tabou est utilisée pour générer des solutions réalisables. Dans la deuxième phase, une variante de recuit simulé *Simulated Annealing with Reheating (SAR)* est utilisée afin d'améliorer la qualité des solutions produites dans la première phase. La variante proposée se distingue par trois caractéristiques : une nouvelle méthode d'évaluation de voisinage, une nouvelle méthode d'estimation des optima locaux, ainsi qu'un nouveau schéma de réchauffement. la méthode SAR permet d'éviter un tunning extensif qui est

généralement requis pour des méthodes de recuit simulé. La méthode proposée par les auteurs a fournis de bons résultats sur le benchmark de l'ITC-2007 et a trouvé plusieurs nouvelles meilleures solutions pour plusieurs instances.

Nous rappelons que [Müller, 2009] le gagnant de la compétition ITC-2007, a proposé une méthode hybride qui a été utilisée simultanément pour traiter les 3 tracks de la compétition. La méthode proposée consiste en trois phases : la première phase à pour but de trouver une solution valide en utilisant une méthode de recherche itérative avancée (*Iterative Forward Search*). Ensuite, la méthode recherche l'optimum local autour de la solution trouvée dans la phase 1 en utilisant la méthode de *Hill Climbing*. Une fois l'optimum local trouvée, la dernière phase est lancée. Cette phase injecte la solution finale de la phase 2 comme solution de départ d'une méthode de *Great Deluge*.

3.2.9 *Méthodes de populations*

[Abdullah et al., 2010] ont proposée une nouvelle formulation multi-objectif pour le problème PECT. Cette formulation part de la formulation présentée dans l'ITC-2007 et l'enrichi avec une nouvelle contrainte souple qui fera l'objet du nouveau objectif afin d'enrichir le problème initial, le rendre plus complexe et d'aboutir à une formulation qui représente au mieux les cas réels des universités. La nouvelle contrainte souple introduite ici tente de minimiser le nombre total de créneaux d'attente entre les différents évènements d'une journée pour chaque étudiant. Les auteurs ont proposée une approche basée sur les algorithmes génétiques avec une taille variable de la population et un temps de vie pour chaque individu qui est évalué par rapport à sa naissance, la qualité de l'individu ainsi que les caractéristiques de la population.

Dans leur travail les auteurs représentent chaque chromosome comme un planning réalisable afin de simplifier les opérations de croisement et de mutation. Le croisement est effectué en choisissant aléatoirement des évènements avec une méthode de roulette russe tout en gardant le planning réalisable. La méthode de croisement vérifie que les nouveaux créneaux sont libres et qu'il n'y ait pas de conflits entre les évènements avec cette nouvelles affectation. La mutation sélectionne aléatoirement une structure de voisinage et en se basant sur un taux de mutation. Les deux voisinages présentés sont :

- **NH1 :** choisir deux évènements aléatoirement et échanger leurs créneaux ;
- **NH2 :** choisir un évènement aléatoirement et le déplacer vers un nouveau créneaux compatible.

Les auteurs ont utilisée MATLAB comme langage de programmation et ont testé leur solution sur les benchmarks de [Socha et al., 2002]. Les résultats ont été meilleurs que ceux de leur anciens travaux mais ils restent inférieurs à ceux de la littérature.

[Nothegger et al., 2012] ont proposée une méthode basée sur l'optimisation par colonies de fourmis. Les fourmis artificielles construisent successivement les solutions en se basant sur les phéromones. La méthode proposée de utilise deux matrices de phéromones afin d'améliorer la convergence de la méthode. La première matrice représente la probabilité d'assignation de l'évènement i au créneau j. La deuxième matrice représente la probabilité d'assignation de l'évènement i à la salle k. A chaque itération, les matrices de phéromones sont mises à jours. Comme chaque fourmis construit sa solution indépendamment des autres, les autres ont parallélisé ce processus. La méthode proposée a été testé sur les benchmarks de l'ITC-2007 et a fournis de bons résultats.

3.2.10 *Conclusion PECT*

Nous avons présenté dans cette section le problème PECT de ITC-2007, sa définition ainsi que les différentes méthodes de résolution qui ont été utilisée pour le résoudre dans la littérature.

3.3 LE PROBLÈME DE ITC-2019

S'appuyant sur le succès des précédentes compétitions de planification, l'ITC-2019 a pour objectif de motiver la recherche autour des problématiques complexes de planification de cours d'universités issues de situations pratique.

3.3.1 *Définition du problème ITC-2019*

Le problème de planification de cours d'universités définie dans l'ITC-2019 [Müller et al., 2018] consiste à trouver une affectation de salles, horaires et aussi d'étudiants à un ensemble de cours. Le problème est soumis à deux groupes de contraintes : des contraintes dures et des contraintes souples.

Il est important de faire remarquer que pour chaque instance de la compétition il existe une solution réalisable qui respecte toutes les contraintes dures et que les inscriptions des étudiants sont connues pour chaque instance (pour celles qui comportent des étudiants à affecter dans les cours).

Chaque instance du problème a pour paramètre le nombre de semaines considérées, le nombre de jours dans une semaine ainsi que le nombre de créneau dans une journée. Une salle est définie par un identifiant et une capacité. Il peut exister entre deux salles un temps de trajet (symétrique) à respecter.

Un cours est une structure hiérarchisée de classes. Il peut être constitué d'une ou plusieurs configurations. Chaque configuration est décomposée en sous-parties, et

chaque sous-partie est composée d'une ou plusieurs classes.

Un étudiant affecté à un cours doit suivre une unique configuration et exactement une classe dans chaque sous-partie qui compose la configuration. Pour certaines classes, une classe parent est définie : tout étudiant participant à cette classe doit obligatoirement participer également à la classe parent. Une classe peut se réunir plusieurs fois par semaine, sur toutes ou partie des semaines. Lorsqu'elle se réunit, la classe débute toujours à la même heure, dure le même nombre de créneaux et est toujours dans la même salle. Pour chaque classe, on dispose de la pénalité associée au choix des salles possibles pour cette classe. Chaque classe dispose également d'une limite d'étudiants qu'elle peut accueillir.

Les étudiants sont inscrits aux cours et pour un cours un étudiant doit être inscrit à une configuration. Une pénalité survient si un étudiant ne peut pas suivre un cours. Pour toutes les instances de la compétition, il existe au moins une solution réalisable.

La solution d'un problème doit respecter des contraintes relatives aux étudiants, des contraintes d'allocations de salles et des contraintes de distribution (espacement, regroupement, simultanéité, ...).

Les contraintes relatives aux étudiants concernent leur affectation aux classes :

1. Chaque étudiant doit suivre exactement une configuration de chaque cours pour lequel il est inscrit ;
2. Si un étudiant est dans la configuration k, il doit suivre exactement une classe de chaque sous-partie; sinon il ne suit aucune des classes ;
3. Une classe ne peut pas accepter plus d'étudiants que l'effectif prévu ;
4. Un étudiant qui suit la classe a doit également suivre la classe parent a' (si elle existe).

Les contraintes d'allocations de salles sont :

1. Une salle est choisie pour chaque classe ;
2. La salle assignée à la classe a est celle assignée à un créneau de cette classe ;
3. Une salle est allouée pour la classe a lors du créneau t si et seulement si la classe a a lieu lors du créneau t ;
4. Une salle ne peut être allouée pour deux créneaux t et t' de deux classes a et a' que si ces deux créneaux ne se chevauchent pas.

Les contraintes de distribution sont :

1. **SameStart(**a,a'**)** : les classes a et a' doivent être planifiées à la même heure, indépendamment du jour et de la semaine ;

2. **SameTime(**a,a'**)** : les classes a et a' doivent être planifiées telles que le créneau horaire de la classe de plus courte durée soit inclus dans le créneau horaire de la classe de plus longue durée, indépendamment du jour et de la semaine ;

3. **DifferentTime(**a,a'**)** : les classes a et a' doivent être planifiées à des horaires différents, indépendamment du jour et de la semaine, leurs créneaux horaires ne doivent pas s'intersecter ;

4. **SameDays(**a, a'**)** : les classes a et a' doivent être planifiées aux mêmes jours, indépendamment de la semaine. Si la classe a est planifiée sur un nombre de jours inférieur à a', les jours de a doivent être un sous-ensemble des jours de a' ;

5. **DifferentDays(**a,a'**)** : les classes a et a' doivent être planifiées dans des jours différents, indépendamment de la semaine ;

6. **SameRoom(**a,a'**)** : les classes a et a' doivent être planifiées dans la même salle ;

7. **DifferentRoom(**a,a'**)** : les classes a et a' doivent être planifiées dans des salles différentes ;

8. **Overlap(**a,a'**)** : les classes a et a' doivent se chevaucher. Il existe au moins un jour d'une semaine sur l'horizon de planification où les deux classes sont planifiées au même moment ;

9. **NotOverlap(**a,a'**)** : les classes a et a' ne doivent jamais se chevaucher. Elles ne partagent pas de semaine ou de jour sur l'horizon de planification. Elles ne sont jamais planifiées au même moment ;

10. **SameAttendees(**a,a'**)** : les classes a et a' sont planifiées dans des salles et à des créneaux tels qu'il soit possible de se rendre aux deux classes. Les deux classes ne doivent pas se chevaucher et le temps entre la fin d'un créneau d'une classe et le début du créneau de la suivante doit être supérieur au temps de trajet entre les salles où sont planifiées les classes ;

11. **Precedence(**a,a'**)** : la première occurrence de la classe a doit être planifiée avant la première occurrence le la classe a', ce qui correspond aux situations suivantes :

    - **Semaine** : la première occurrence de la classe a se déroule pour la première fois lors d'une semaine qui précède la planification de la première occurrence de la classe a' ;
    - **Semaine/jour** : les classes se déroulent pour la première fois lors de la même semaine, et la première occurrence de la classe a se déroule pour la première fois lors d'un jour qui précède la planification de la première occurrence de la classe a' ;

- **Semaine/jour/créneau** : les deux classes se déroulent pour la première fois lors du même jour de la même semaine, et la première occurrence de la classe a se termine avant que la première occurrence de la classe a' soit planifiée.

12. **WorkDay(**a,a',S**)** : deux classes a et a', lorsqu'elles ont lieu le même jour d'une même semaine, ne sont pas planifiées sur plus de S créneaux entre le début de la classe planifiée la plus tôt et la fin de la classe planifiée la plus tard ;
13. **MaxDays(**a,D**)** : la classe n'est pas planifiée plus de D jours différents, indépendamment de la semaine.
14. **MaxDayLoad(**a,S**)** : la classe n'est pas planifiée sur plus de S créneaux, chaque jour de chaque semaine ;
15. **MaxBreaks(**a,R,S**)** : la classe ne doit pas induire plus de R pauses de taille S chaque jour de chaque semaine. Une pause a lieu lorsque deux créneaux d'une journée sont espacés de plus de S créneaux-unité ;
16. **MaxBlock(**a,M,S**)** : la classe ne doit pas former plus de M blocs, chaque jour de chaque semaine. Un bloc est formé par les classes qui se suivent à moins de S créneaux-unité d'intervalle.

3.3.2 *Méthodes de résolution*

[Lemos et al., 2021] ont présenté un algorithme qui utilise un encodage MaxSAT pour résoudre le problème défini dans ITC-2019. Les auteurs ont développé un algorithme qui combine des techniques de recherche locale et un solveur MaxSAT.

Les auteurs ont proposé deux types de prétraitements :

- Identification des sous-problèmes indépendants des cours. Afin de réduire l'espace de recherche du problème, une instance est décomposée en des sous-instances autonomes qui peuvent être résolues indépendamment sans perdre aucune solution ;
- Fusion des étudiants avec le même plan d'inscription au cours. Afin de réduire le nombre de variables et de contraintes similaires, des groupes d'étudiants partageant le même plan scolaire sont créés.

[Holm et al., 2019] ont proposé une formulation ILP pour modéliser le problème de ITC-2019. Les auteurs ont aussi proposé une matheuristique pour résoudre le problème. Avant de commencer la résolution, les auteurs réalisent un ensemble de pré-traitements afin d'enlever les informations non nécessaire pour résoudre le problème. Ensuite, à partir d'une solution initiale, les auteurs lancent une matheuristique *Fix-And-Optimize* pour résoudre le problème. Les auteurs ont remporté la compétition ITC-2019.

[Gashi et al., 2021] ont proposé une méthode de récuit simulé pour résoudre le problème de ITC-2019. L'algorithme proposé recherche à la fois les régions réalisables et irréalisables de l'espace des solutions. Une solution irréalisable est traitée en utilisant une combinaison de pénalisation incrémentale et de recherche restreinte sur des contraintes dures spécifiques. Les auteurs proposent aussi des fonctions d'évaluation et de température qui sont adaptés au problème ITC-2019.

[Er-rhaimini, 2019] ont proposé une méthode de résolution heuristique qui s'appelle *Forest Growth Optimisation*. Les auteurs commencent par trouver une solution initiale avec une heuristique constructive qui utilise des listes de priorité. Ensuite, les auteurs essayent d'améliorer la solution obtenue en effectuant des désaffectation/réaffectation des différentes classes.

[Rappos et al., 2022] ont proposé une formulation ILP pour modéliser le problème de ITC-2019. Les auteurs proposent une méthode de Brand-And-Cut pour résoudre le problème. Une fois une solution réalisable trouvée, la méthode proposée fixe itérativement des variables ont utilisant des *cuts* et essaye de trouver une solution optimale locale.

3.4 COMPARAISON DES PROBLÈMES ET CONCLUSIONS

Nous avons présenté dans ce chapitre les différents problèmes de la littérature autour du problème UCTTP. Pour chaque problème, nous avons décrit la formulation, les instances ainsi que les méthodes de résolution qui ont été appliquées dans la littérature pour le résoudre.

Le tableau 3.3 présente une comparaison entre les différents problèmes de planification de cours d'universités présents dans la littérature ainsi que le problème de l'UTC.

Tableau 3.3: Comparaison de problèmes UCTTP

Contrainte	CB-CTT ITC-2007	PECT ITC-2007	ITC-2019	UTC
Etudiants Connus	OUI	OUI	OUI	NON (seulement effectif)
Cursus Fixes	OUI	NON	NON	NON
Choix à la carte	NON	OUI	OUI	OUI
Sections multiples	NON	NON	NON	OUI
Affectation	NON (implicite)	OUI	OUI	NON
Enseignants	OUI	NON	NON	OUI
Créneaux	Fixes	Fixes	Timeslots prédifinis	Timeslots ouverts
Disponibilité des salles	NON	NON	OUI	OUI
Salles multiples	NON	NON	NON	OUI
Evenement sans salles	NON	NON	NON	OUI
Salles spécifiques	NON	OUI	OUI	OUI
Salles partagées	NON	NON	NON	OUI
Déplacement entre salles	NON	NON	OUI	OUI
Tout Placer ?	OUI	NON	OUI	OUI

Les problèmes sont différents entre eux dans la structure, la hiérarchie des cours, les contraintes ainsi que dans plusieurs aspects spécifiques du problème.

Nous remarquons dans le tableau 3.3 que le problème de l'UTC ne s'occupe pas de l'affectation des étudiants car le choix des étudiants n'est pas encore connu, seul les effectifs prévisionnels sont connus.

Un autre aspect qui est spécifique au problème de l'UTC c'est les sections multiples. En effet, la plus petite entité d'enseignement qui est la classe dans ITC-2019 peut être composée de une ou plusieurs sections dans le problème de l'UTC. Ces sections ont lieu séparément dans la semaine et un étudiant affecté à la classe doit être capable de suivre toutes les sections de la classe.

Nous remarquons aussi une différence dans la définition des créneaux. Pour les problèmes CB-CTT et PECT de ITC-2007 les créneaux sont fixes et même durée. Pour l'ITC-2019, nous disposons pour chaque classe d'un ensemble de choix de créneaux et la classe doit être affectée à l'un d'eux. Pour le problème de l'UTC, nous ne disposons que de la durée de la section, et la section doit être affectée à un créneau dans la semaine de cette durée là.

Le problème de l'UTC se distingue aussi avec plusieurs aspects liés aux salles qui ne sont pas présents dans le problème ITC-2019 tel que le concept des salles multiples où une classe peut avoir besoin de plusieurs salles pour avoir lieu, le concept des évènements sans salles où des classes peuvent avoir lieu sans avoir besoin d'une salles et enfin le concept de salles partagées où deux classes peuvent avoir lieu dans la même salle.

Le problème de l'UTC est aussi spécifique avec l'introduction du concept de grilles d'UVs et d'un ensembles de contraintes de compatibilité et d'incompatibilité variées entre les UVs de la grilles. Les contraintes enseignants sont un plus aussi dans le problèmes de l'UTC. De nouvelles contraintes de distribution qui ne sont pas présente dans ITC-2019 sont introduite dans le problème de l'UTC.

Nous introduisons dans le chapitre suivant la définition du problème de planification de cours de l'UTC ainsi que ses différents composants et contraintes.

Chapitre 4

Problème d'emplois du temps de cours de l'UTC

Sommaire

Nous étudions dans ce chapitre le problème de planification de cours de l'UTC (University Course TimeTabling Problem of UTC university, UTC-UCTTP). Le problème consiste à programmer des activités d'enseignement dans des salles et des timeslots, et les affecter à des enseignants. L'enjeu est d'obtenir des emplois du temps de bonne qualité en respectant les différentes contraintes du problème. Les apports de ce travail se résument comme suit :

- Premièrement, nous proposons un modèle de programmation linéaire en nombres entiers (Integer Linear Program, ILP) pour le problème de planification de cours de l'UTC (UTC-UCTTP). L'ILP est conçu à des fins de modélisation et dans le but d'obtenir des solutions de bonne qualité pour certaines instances. Cependant, les solveurs ILP peuvent avoir des difficultés à trouver des solutions réalisables dans des délais de traitement raisonnables ;

- Deuxièmement, nous proposons une méthode heuristique AIDCH (Adaptive Large Neighbourhood Search, AIDCH) pour trouver des solutions réalisables pour les instances du problème. Cependant, les solutions trouvées peuvent ne pas être de bonne qualité ;

- Troisièmement, nous proposons une métaheuristique ALNS (Adaptive Large Neighbourhood Search, ALNS) pour améliorer la qualité des solutions. Nous menons des expériences préliminaires pour régler les paramètres des composants de l'ALNS. Nous étudions la contribution des composants de l'ALNS ;

- Enfin, nous montrons que l'approche de résolution ALNS obtient des solutions de bonne qualité dans des temps de traitement raisonnables. L'approche ALNS peut constituer une bonne base pour proposer des emplois du temps adéquats pour l'UTC.

Le chapitre est organisé comme suit. La section 4.1 présente le système d'enseignement à l'UTC. La section 4.2 présente le processus actuel de conception des emplois du temps à l'UTC. La section 4.3 présente les techniques de conception actuelles du SME. La section 4.4 présente les contraintes et les paramètres du problème de planification de cours de l'UTC (UTC-UCTTP) que nous adressons. La section 4.5 présente l'ILP que nous utilisons pour fournir un modèle formel. L'heuristique AIDCH est présentée dans la section 4.6. La méthode ALNS que nous proposons pour résoudre le problème UTC-UCTTP est décrite dans la section 4.7 et les composants sont détaillés. Les expériences de tests sont rapportées et commentées dans la section 4.8. L'efficacité des composants de l'ALNS est présentée. La conclusion et les perspectives futures se trouvent dans la section 4.9.

4.1 SYSTÈME D'ENSEIGNEMENT À L'UTC

L'enseignement de la formation ingénieur à l'UTC est divisé en Unités de Valeur (UV) et Unités d'Enseignement (UE) pour la formation Master. Pour le reste du chapitre, nous allons gardé la terminologie UV pour désigner les UVs et les UEs. Chaque UV correspond à la quantité de travail nécessaire (en général de 100 à 150 heures) pour atteindre en un semestre un objectif donné. Cet objectif pourra être l'acquisition de connaissances dans un domaine précis, l'apprentissage d'une méthode ou d'un langage, la découverte d'un aspect de la vie professionnelle ou la réalisation d'un projet au sein de l'UTC ou à l'extérieur.

Une UV regroupe plusieurs types d'activité d'enseignement telles que le cours magistral, les travaux dirigés (TD) et les travaux pratiques (TP). A chaque activité d'enseignement correspond un ou plusieurs groupes. Un groupe représente un ensemble d'étudiants qui se réunissent généralement une fois par semaine, voir une semaine sur deux, pour suivre une activité d'une UV.

Les UVs de tronc commun et de branche sont classées dans l'une des catégories suivantes :

- Connaissances Scientifiques (CS) ;

- Techniques et Méthodes (TM) ;

- Technologie et Sciences de l'Homme (TSH) ;

- Stages et Périodes de travail à l'extérieur (SP).

L'obtention du diplôme d'ingénieur nécessite l'acquisition d'un nombre minimum de crédits ECTS dans chacune des catégories : c'est le "profil minimum de formation". A l'échelle européenne, un crédit ECTS (European Credit Transfer System) correspond environ à 25 h de travail. Ainsi, pour valider une UV à 6 crédits, il faut environ 6*25 = 150 h de travail sur le semestre.

Il existe deux cycles de formation à l'UTC : le premier cycle ou le tronc commun (TC) et le cycle ingénieur (Branche). L'UTC propose à ses étudiants 5 branches d'enseignement :

- Génie Biologique (GB) ;
- Génie Informatique (GI) ;
- Génie des Procédés (GP) ;
- Génie Urbain (GU) ;
- Ingénierie Mécanique (IM).

Chaque branche propose entre 3 à 8 filières. Si l'étudiant a un projet de carrière qui ne s'intègre pas dans une filière de sa branche, il peut construire ce qu'on appelle une filière libre.

En plus de la formation ingénieur, l'UTC propose aussi une formation de Master ainsi qu'une formation d'école doctorale.

Le système d'enseignement à l'UTC est un système semestriel. L'année est divisée en deux semestres : le semestre d'automne et le semestre de printemps. Chaque semestre est constitué de 17 semaines d'enseignement. A chaque début de semestre, l'étudiant doit choisir la liste de ses UVs en s'inscrivant sur le site correspondant.

4.2 PROCESSUS ACTUEL DE CONCEPTION DES EMPLOIS DU TEMPS (EDT) À L'UTC

Le processus de gestion d'EDT à l'UTC est un processus cyclique. Une année universitaire est divisée en deux semestres. A chaque semestre correspond un cycle de conception d'EDT pour le semestre suivant. Les deux cycles sont légèrement différents, nous présenterons les différences entre ces deux cycles ultérieurement.

La rentrée d'automne s'effectue au mois de septembre, ce qui correspond à la fin du cycle de conception d'EDT pour ce semestre. Nous disposons des EDT provisoires ainsi que des choix des étudiants. Le travail des rentrées consiste à répondre au mieux à la demande des étudiants tout en tenant compte des ressources humaines (enseignants) et matérielles. La majorité du travail se fait sur les UV trop pleines en ajoutant des groupes de TD/TP, en augmentant le nombre d'étudiants par groupe ou en désinscrivant des étudiants. Chaque étudiant qui est désinscrit à une UV doit, dans la mesure du possible, être réinscrit dans son second voire troisième choix d'UV. La difficulté du processus de réinscription se situe dans le fait que les seconds choix peuvent être eux aussi complets ou incompatibles avec le reste de l'emploi du temps de l'étudiant.

Une UV est dite en sous-effectif si elle n'atteint pas le nombre minimum d'étudiants pour l'ouvrir. Les UVs qui restent en sous-effectif sont fermées. Dans ce cas, il faut inscrire les étudiants qui étaient inscrits dans l'UV en sous-effectif dans une UV qui convient pédagogiquement et qui soit compatible avec le reste de leurs UVs. La notion de convenance pédagogique est définie par les responsables pédagogiques.

Ensuite, la phase d'affectation commence. On entend par phase d'affectation la phase d'inscription automatique de tous les étudiants aux différents groupes de TD, TP voire de cours quand il y en a plusieurs. Chaque étudiant reçoit ensuite son emploi du temps individuel et chaque responsable d'UV reçoit la liste des différents groupes d'activités de son UV.

Nous présentons le cycle d'automne qui correspond à la conception d'EDT du semestre de printemps.

Le cycle d'automne démarre durant le mois d'octobre. Durant ce mois, la création des EDT provisoires de printemps est entamée. On commence par la mise à jour de l'offre de formation en collaboration avec les responsables pédagogiques. Cette mise à jour consiste en la création ou suppression d'UVs, l'actualisation de la liste des UV qui apparaîtront dans les différents EDT et l'actualisation des formats des UV. Ensuite, une prévision des effectifs des activités et des nombres de groupes est faite. Pour établir cette prévision, on a recours aux informations relatives aux inscriptions des étudiants dans le semestre d'automne de l'année dernière, la transition des étudiants entre le semestre d'automne et le semestre de printemps de l'année dernière ainsi que les inscriptions des étudiants pour le semestre d'automne actuel. Ce travail est réalisé en effectuant des échanges avec les responsables pédagogiques. Le but de ces échanges est de définir un certain nombre de contraintes d'inscription pour les étudiants afin de réussir un placement sur une semaine type. Ces contraintes vont servir pour définir les règles de choix d'UVs pour les étudiants.

Comme il est impossible de rendre toutes les UVs compatibles dans un EDT d'une semaine type, les discussions avec les responsables pédagogiques visent à définir des ensembles pédagogiques d'UVs par catégorie d'étudiants. Ceci va induire la mise en place des contraintes d'inscription aux UVs pour les étudiants.

Durant le mois de novembre, la phase de recueil de données auprès des responsables des UVs est engagée afin de permettre la validation ou la modification des données relatives à chaque UV. Chaque responsable d'UV est informé du nombre de groupes prévus pour son UV, et doit identifier en retour la liste des intervenants pour les activités. Aussi, chaque intervenant doit indiquer la liste de ses souhaits et préférences en terme d'horaires. Des préférences de type "Je ne veux pas enseigner dans tel créneau horaire" pour les internes et "Je voudrais intervenir dans tel créneau horaire" pour les externes sont contrôlables par le système actuel.

Durant cette phase s'effectue aussi l'édition du calendrier des semaines du semestre : alternances (semaine A et semaine B) et changement de jours (lundi qui devient un jeudi par exemple). L'alternance permet de programmer les TPs par exemple, pour un groupe de TD de 24 étudiants, un intervenant va faire le même TP, une semaine pour 12 étudiants et la semaine suivante pour les 12 autres étudiants. Ce mécanisme est appelé alternance semaine A/B. Les changements de jour permettent de remplacer des journées d'enseignements qui ont été perdus à cause des jours fériés par exemple ou

d'évènements particuliers. Le but est d'arriver en fin de semestre à un planning où tous les jours perdus sont rattrapés, c'est-à-dire, qu'il y ait le même nombre de lundi, mardi, etc sur l'horizon des 17 semaines d'enseignement que comporte un semestre. Ainsi, les activités d'enseignement planifiées sont équitablement suivies par les étudiants. A la fin du mois, les EDT provisoires commencent à prendre forme.

Durant le mois de décembre s'effectue la mise en place effective des EDT provisoires. On commence par garder les EDT de l'année dernière (affectation des groupes aux horaires) et on commence à effectuer des modifications selon les changements qui ont été identifiés dans les phases précédentes.

En début janvier, les EDT provisoires sont stabilisés. Ils sont transmis aux responsables des UVs pour consultation afin d'identifier les anomalies et procéder aux modifications nécessaires. Le site d'inscription aux UVs est préparé. Cela consiste à mettre à disposition les EDT provisoires et à mettre en place les vérifications des contraintes d'inscriptions recueillies précédemment.

Durant le mois de février, le site d'inscription est mis à disposition des étudiants . Cette opération s'effectue en général la première semaine de février. Une fois terminée, nous disposerons des inscriptions effectives des étudiants aux UVs selon les EDT provisoires, puis de nouveau, le travail de rentrée de traitement des UV en sur-effectif est lancé. Après ceci le cycle de printemps commence et ainsi de suite.

4.3 TECHNIQUES DE CONCEPTION ACTUELLES DU SME

Cette section a pour but de résumer les éléments qui peuvent aider à concevoir un algorithme de conception d'EDT qui s'inspire de la manière actuelle de conception du SME pour la construction des EDTs.

Le processus de conception de l'EDT provisoire comporte les points suivants :

1. Récupérer l'EDT du semestre précédent ;
2. Traiter les données en mettant à jour le nombre de groupes de chaque UV ;
3. Commencer à placer les groupes qui ne sont pas placés dans l'EDT dans l'ordre suivant :
   (a) Les cours de (TC) car ils sont contraints ;
   (b) Les TDs/TPs de (TC) à cause des contraintes de salles enseignant ;
   (c) Les groupes de l'apprentissage et les gros cours de début de branche, beaucoup d'UVs mutualisées FISE/FISA donc on ne peut en placer en parallèle avec l'apprentissage (regarder aussi les débuts de filière/branche à cause de la mutualisation) ;
   (d) Les cours de début de branche ;
   (e) Les TDs/TPs de début de branche à cause des contraintes de salles et des contraintes enseignant ;
   (f) Les groupes de filière à cause des salles informatiques et les contraintes enseignants ;

(g) Les TDs/TPs de début de branche ;
(h) Les TDs/TPs de TC ;
(i) Les groupes Hutech et Master qui peuvent être placés avant les groupes de TC à cause des contraintes de salles ;
(j) Revérifier la cohérence de l'EDT ;
(k) Faire une affectation des étudiants avec cet EDT avec les inscriptions de l'année dernière et observer le rendu ;
(l) Des modifications si besoin en cas de conflit.

4. A chaque placement, vérifier :
   (a) Les contraintes de salles,
   (b) Les contraintes de compatibilité dans toutes les grilles ou apparaît l'UV,
   (c) Les contraintes enseignants,
   (d) La répartition des étudiants.
5. Si l'EDT n'est pas réalisable, il faut négocier des contraintes d'incompatibilités avec les responsables pédagogiques.

4.4 DESCRIPTION DU PROBLÈME UTC-UCTTP

Le problème UTC-UCTTP est un problème d'emplois du temps de cours d'universités qui concerne la planification des activités pédagogiques à l'UTC. Il s'agit de trouver une affectation pour chaque groupe en terme de créneau, de salle et d'enseignant tout en respectant les différentes contraintes liées au problème. Nous présentons dans ce qui suit les différentes entités du problème UTC-UCTTP.

4.4.1 *Les UVs*

Les UVs possèdent la structure hiérarchique suivante :

- une UV possède une ou plusieurs configurations ;
- chaque configuration possède une ou plusieurs activités ;
- chaque activité possède une ou plusieurs classes (groupes) ;
- chaque classe possède une ou plusieurs sections.

La figure montre 4.1 montre la structure hiérarchique des UVs à l'UTC.

Un étudiant qui est inscrit dans une UV, doit pouvoir être affecté aux sections associées pour une configuration, donc à une classe pour chaque activité qui fait partie de la configuration.

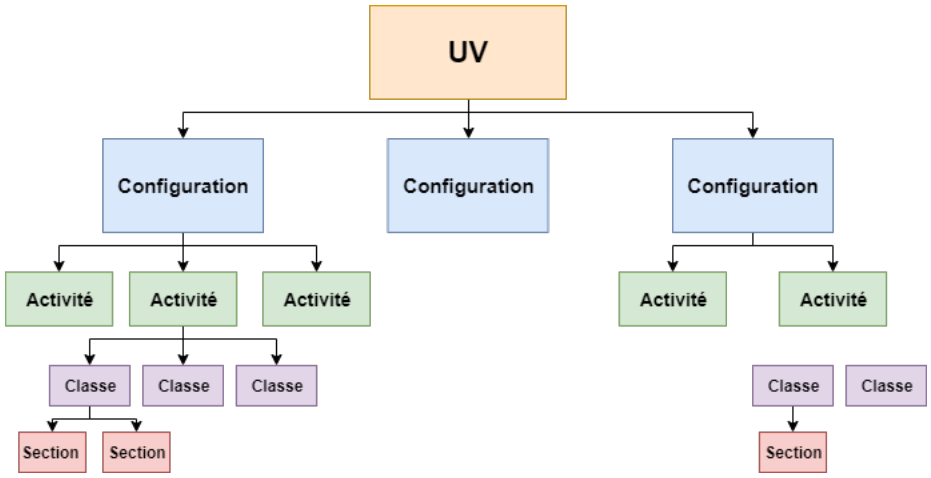

Fig 4.1: Structure des UVs à l'UTC.

Chaque classe est lié à un horizon de planification et possède un statut. Le statut peut être soit **Actif** pour les groupes normaux ou **OnHand** pour les groupes de réserve. Un groupe de réserve représente un groupe qu'on planifie dans l'EDT et qui peut être soit utilisé en cas de surreffectif de l'UV soit enlevé si les groupes actifs suffisent.

Chaque section possède une durée qui est exprimée en terme de nombre de quarts d'heure.

Chaque configuration possède un nombre maximum d'étudiants qui peuvent être inscrits, donc potentiellement affectables à la configuration. La somme des nombres d'étudiants des configurations doit être égale à chacune des sommes des nombres d'étudiants des classes et ceux pour chaque activité. Par exemple, si la somme des nombres d'étudiants est égal à 96 alors pour une activité de type TD, la somme des nombres d'étudiants des classes (groupes) doit être égale à 96.

Il existe des UVs qui sont complémentaires entre elles qu'on appelle demi-UVs. Trois types de demi-UVs sont à considérer :

- Type 1 : mêmes étudiants, mêmes créneaux, mêmes salles, pas forcément mêmes enseignants (MT90, MT91). ;
- Type 2 : pas mêmes étudiants, mêmes créneaux, pas forcément mêmes salles, pas forcément mêmes enseignants (MA90, MA91). ;
- Type 3 : pas forcément mêmes enseignants, pas forcément mêmes salles, mêmes étudiants, pas forcément mêmes durée (TF70, TF71).

4.4.2 *Les Horizons et les Créneaux*

Un horizon représente les semaines de planification des activités qui lui seront affectées. Un horizon est représenté par une chaîne de caractères binaire qui contient 24 caractères

représentant les semaines de planification. Par exemple, si une activité à comme horizon 11110000000000000000000, alors l'activité doit être planifiée les quatre premières semaines du semestre.

Un créneau est représenté par sa durée en terme de nombre de quarts d'heure, son heure de début, les jours possibles du créneau qui sont présentés sous forme d'une chaîne de sept caractères (0100000 représente le mardi par exemple) ainsi que la pénalité associée au créneau.

Il existe huit durées de créneaux possibles de 1 heure à 5 heures (1 heure, 1 heure et 30 minutes, 2 heures, 2 heures et 30 minutes, 3 heures, 3 heures et 30 minutes, 4 heures et 5 heure).

4.4.3 *Les salles*

L'UTC est structurée en plusieurs sites. Chaque site possède un ensemble de bâtiments. Chaque bâtiment possède plusieurs étages. A chaque étage se trouve plusieurs salles.

À chaque salle, on associe une capacité qui représente le nombre d'étudiants que la salle peut contenir en tenant compte de ses équipements et une pénalité si la salle est utilisée.

Nous définissons une matrice de transport entre sites, ainsi qu'une matrice de transport entre bâtiments afin de pouvoir calculer la durée de transport entre chaque couple de salles.

On introduit aussi la notion d'ensembles de salles afin de pouvoir regrouper les salles qui possèdent les mêmes caractéristiques.

Afin de pouvoir lier les UVs et les salles, on introduit le principe de demande de salle. Chaque activité doit avoir une demande de salle afin de pouvoir spécifier les caractéristiques de la salle souhaitée.

Pour chaque demande de salle pour une activité, nous aurons un RoomRequest. Nous avons donc un ET logique entre les RoomRequest. Nous aurons une salle principale obligatoire et plusieurs salles secondaires si besoin.

Pour chaque possibilité de salle, nous aurons plusieurs RoomRequestChoice, on doit choisir un RoomRequestChoice à satisfaire. Nous avons donc un OU logique entre les RoomRequestChoice.

Un RoomRequestChoice va contenir plusieurs SetOfRooms (ensembles de salles) représentant les caractéristiques de la salle demandée. Toutes les caractéristiques doivent être satisfaites. Nous avons donc un ET logique.

Si une salle précise est requise, on spécifie directement la salle demandée. Si on veut demander une salle pour certaines semaines spécifiques, on rajoute les semaines correspondantes comme information dans la demande.

4.4.4 *Les enseignants*

Nous disposons d'un ensemble d'enseignants. Pour chaque enseignant, nous disposons de la liste des activités dans lesquelles il intervient, avec le nombre de classes qu'il assure, ainsi que la liste de ses souhaits et contraintes concernant les créneaux d'enseignement.

Une assignation d'un enseignant à une activité peut avoir deux types :

- FULL pour dire que l'enseignant est l'enseignant principal de la classe et que c'est lui qui assure toutes (ou presque) les séances du semestre ;
- PARTIAL pour dire que l'enseignant est l'un des enseignants secondaires de la classe et qu'il intervient seulement dans quelques séances du semestre.

Pour chaque enseignant, on donne la liste des contraintes relatives aux créneaux. Chaque contrainte concerne une activité dans laquelle intervient l'enseignant. Pour chaque contrainte on spécifie les jours concernés, l'heure de début et l'heure de fin ainsi que le type de la contrainte.

Les enseignants peuvent exprimer des préférences de salles pour les activités dans lesquelles ils interviennent. Ces préférences peuvent être exprimées en terme de salle, ensemble de salles, site, bâtiment ou en terme d'étage. Chaque préférence possède un type.

Une pénalité est associée à chaque contrainte/préférence si besoin. Il existe quatre types de contraintes/préférences :

- Required : pour exprimer une préférence requise ;
- Prohibited : pour exprimer une préférence interdite ;
- Recommended : pour exprimer une préférence souhaitée ;
- NotRecommended : pour exprimer une préférence non souhaitée.

4.4.5 *Les grilles*

Une grille d'UVs représente un regroupement d'UVs du même type, cycle ou branche. Une grille est divisée en plusieurs ensembles d'UVs sur lesquelles sont imposées des contraintes de compatibilité. Nous présentons dans ce qui suit les différentes contraintes compatibilité :

1. Toutes les UVs de l'ensemble U doivent impérativement être compatibles ;
2. Si possible toutes les UVs de l'ensemble U doivent être compatibles ;
3. Exception de compatibilité, pas de compatibilité requise pour les UVs de l'ensemble U ;
4. Toutes les combinaisons de n UVs de l'ensemble U doivent impérativement être compatibles ;
5. Si possible, toutes les combinaisons de n UVs de l'ensemble U doivent être compatibles ;
6. Toutes les combinaisons de n UVs de l'ensemble U doivent impérativement être compatibles avec toutes les combinaisons de m UVs de l'ensemble V ;

7. Si possible, toutes les combinaisons de n UVs de l'ensemble U doivent être compatibles avec toutes les combinaisons de m UVs de l'ensemble V ;

Notons que toutes les contraintes de grilles, peuvent être ramenées à des combinaisons de contraintes 1 et 2 de la liste citée si-dessus.

4.4.6 *Les contraintes de distribution*

Les contraintes de distributions sont des contraintes qui sont utilisées pour contrôler la disposition des sections à placer dans l'EDT. Une contrainte de distribution peut concerner soit plusieurs classes ou bien plusieurs activités. Une contrainte de distribution peut être soit dure soit souple. Dans le cas des contraintes de distribution souples, une pénalité est associée à la contrainte.

Les différents types de contraintes de distributions sont :

- **SameTimeSlot** : deux classes a et a' dans cette contrainte doivent partager le même créneau le même jour de semaine indépendamment des semaines ;
- **SuccessiveTimeSlot** : deux classes a et a' dans cette contrainte doivent avoir lieu dans des créneaux successifs la même journée ;
- **SameStart** : deux classes a et a' dans cette contrainte doivent commencer à la même heure, indépendamment du jour de la semaine indépendamment des semaines;
- **SameTime** : deux classes a et a' dans cette contrainte doivent se dérouler au même moment, indépendamment du jour de la semaine et indépendamment des semaines. Cela signifie que le créneau horaire de la classe la plus courte doit être intégralement contenu dans celui de la classe la plus longue ;
- **DifferentTime** : deux classes a et a' dans cette contrainte doivent se dérouler sur des horaires différents, indépendamment du jour de la semaine et indépendamment des semaines. Cela signifie que leurs créneaux horaires ne doivent pas se chevaucher ;
- **SameDays** : deux classes a et a' dans cette contrainte doivent se dérouler sur les mêmes jours ;
- **DifferentDays** : deux classes a et a' dans cette contrainte ne doivent pas partager de jour ;
- **SameRoom** : deux classes a et a' dans cette contrainte doivent avoir lieu dans la même salle ;
- **DifferentRoom** : deux classes a et a' dans cette contrainte doivent avoir lieu dans des salles différentes ;
- **Overlap** : deux classes a et a' dans cette contrainte doivent se chevaucher ;

- **NotOverlap** : deux classes a et a' dans cette contrainte ne doivent pas se chevaucher ;
- **Precedence** : deux classes a et a' dans cette contrainte doivent avoir lieu l'une avant l'autre dans la semaine (la première classe avant la deuxième), si elles sont planifiées la même semaine ;
- **SameAttendees** : deux classes a et a' dans cette contrainte doivent avoir lieu dans des salles et à des créneaux tels qu'il soit possible de se rendre aux deux classes. Cela signifie que les deux classes ne doivent pas se chevaucher, et que le temps entre la fin d'une classe et le début de la suivante soit supérieur au temps de trajet entre les salles où se déroulent les classes ;
- **MustShareRoom** : deux classes a et a' dans cette contrainte doivent avoir lieu dans la même salle, dans le même créneau, la même journée. La contrainte est équivalente à poser les contraintes SameRoom et SameTimeSlot simultanément sur les deux sections ;
- **CanShareRoom** : deux classes a et a' dans cette contrainte peuvent avoir lieu dans la même salle, dans le même créneau, la même journée. La contrainte est équivalente à ne pas poser la contrainte d'interdiction de réservation multiples de salles pour le couple de sections.

4.4.7 *Les contraintes cumulatives spécifiques*

Deux types de contraintes sont prises en compte dans le problème UTC-UCTTP :

- **Les contraintes logiciel** : nous disposons d'un ensemble de logiciels qui sont utilisés par certaines activités. Pour chaque logiciel, nous disposons de la liste des activités qui l'utilisent et du nombre de licences. La contrainte impose que la somme du nombre d'étudiants des activités qui ont lieu en des timeslots qui se chevauchent ne doit pas dépasser le nombre de licences, tout au long de la période de planification ;
- **Maximum nombre d'étudiants par étages** : pour certains étages, et à chaque instant de la période de planification, la somme du nombre d'étudiants présent dans ses étages ne doit pas dépasser une certaine limite. C'est une contrainte de sécurité pour l'évacuation en cas d'urgences.

4.4.8 *La distance par rapport à un EDT de référence*

Nous définissons une fonction de distance qui permet de calculer la distance de l'EDT actuel par rapport à un EDT de référence qui est généralement l'EDT du semestre précédent. Le but est de perturber le moins les EDTs par rapport aux EDTs des semestres précédents afin que les enseignants puissent avoir une certaine stabilité pour pouvoir garder la même planification pour le reste de leur activités de recherche et

d'administration par exemple. Chaque écart dans les timeslots compte pour 2 et chaque écart dans les jours compte pour 1. Par exemple, si un cours était planifié le lundi à 8h du matin dans l'EDT de référence et qu'il est planifié le mardi à 14h15 dans l'EDT en cours de construction, alors la distance en terme de timeslots est égale à 2 et la distance en terme de jours est égale à 1, donc la distance totale est égale à 3.

4.5 MODÈLE ILP POUR LE PROBLÈME UTC-UCTTP

Nous présentons dans cette section une modélisation ILP pour le problème UTC-UCTTP. Nous présentons les données, les variables, les contraintes et la fonction objectif.

4.5.1 *Les données*

Nous présentons dans ce qui suit l'ensemble des données du problème :

- $\mathcal{W}$: ensemble de semaines ;
- Δ : ensemble de jours ;
- $\mathcal{SH}$: ensemble d'horizons ;
- $\mathcal{W}_h$: ensemble des semaines actives de l'horizon h ;
- $\mathtt{slots}$: nombre de slots de quarts d'heure dans une journée, *i.e.* 96 pour toutes les instances de l'UTC ;
- $\mathcal{ST}$: ensemble de sites ;
- $\mathcal{B}_{st}$: ensemble de bâtiments du site st ;
- $\mathcal{F}_b$: ensemble d'étages du bâtiment b ;
- $\mathcal{R}_f$: ensemble de salles de l'étage f ;
- $\mathcal{R}$: ensemble de salles ;
- $\mathcal{SR}$: ensemble des ensembles de salles ;
- $\mathcal{C}$: ensemble des UVs ;
- $\mathcal{K}_c$: ensemble des configurations de l'UV c ;
- $\mathcal{P}_{c,k}$: ensemble des activités de la configuration k de l'UV c ;
- $\mathcal{CL}_{c,k,p}$: ensemble des classes possibles pour l'activité p de la configuration k de l'UV c ;
- $\mathcal{A}_{c,k,p,cl}$: ensemble des sections possibles pour la classe cl de l'activité p de la configuration k de l'UV c ;

- $\mathcal{SW}$: ensemble des logiciels ;
- $\mathcal{A}_s$: ensemble des sections utilisant le logiciel s ;
- $\mathcal{T}$: ensemble de tous les timeslots t, où un timeslot t consiste en un slot de début, un slot de fin, un ensemble de jours dans lesquelles la section se réunit et un ensemble de semaines.
- $\mathcal{T}_a$: ensemble des timeslots possibles pour la section a ;
- $Days(t)$: ensemble de jours du timeslot t ;
- $Weeks(t)$: ensemble de semaines du timeslot t ;
- $\mathcal{Q}_a$: ensemble des demandes de salles de la section a ;
- $\mathcal{R}_{a,q} \subset \mathcal{R}$: sous-ensemble de salles disponibles pour la section a pour la demande de salle q ;
- Tr : $|\mathcal{R}| \times |\mathcal{R}|$ matrice de temps de déplacement entre tous les couples de salles ;
- $\mathcal{R}_{mf} \subset \mathcal{R}$: sous-ensemble de salles r pour la contrainte "Maximum nombre d'étudiants par étages" ;
- $\mathcal{DC}$: ensemble des contraintes de distribution d ($d \in \mathcal{HDC}$ si d est une contrainte dure, $d \in \mathcal{SDC}$ si d est une contrainte souple). Une contrainte de distribution d est définie comme un triplet $(type, a, a')$, où $type$ représente le type de la contrainte de distribution et a et a' les deux sections pour lesquelles la contrainte de distribution est définie ;
- $\mathcal{GC}$: ensemble des contraintes de grilles g ($g \in \mathcal{HGC}$ si g est une contrainte dure, $g \in \mathcal{SGC}$ si g est une contrainte souple) ;
- $\mathcal{L}$: ensemble d'enseignants ;
- $\mathcal{P}_l$: ensemble des activités assurées par l'enseignant l ;
- $sec(l, p)$: nombre de sections qui vont être assurées par l'enseignant l pour l'activité p ;
- $\mathcal{T}_l$: ensemble des timeslots souhaités par l'enseignant l ;
- $\bar{\mathcal{T}}_l$: ensemble des timeslots non souhaités par l'enseignant l ;
- $\mathcal{R}_{l,p}$: ensemble des salles souhaitées par l'enseignant l pour l'activité p ;
- $\bar{\mathcal{R}}_{l,p}$: ensemble des salles non souhaitées par l'enseignant l pour l'activité p ;
- $t_{l,p}$: ensemble des timeslots requis par l'enseignant l pour l'activité p ;
- $\bar{t}_{l,p}$: ensemble des timeslots interdits par l'enseignant l pour l'activité p ;

- $r_{l,p}$: ensemble des salles requises par l'enseignant l pour l'activité p ;
- $\bar{r}_{l,p}$: ensemble des salles interdites par l'enseignant l pour l'activité p ;
- $\mathcal{DAYS}$: ensemble de tous les jours dans l'horizon de planification ;
- $\mathcal{TAU}$: ensemble de tous les slots dans une journée ;
- $\mathcal{HC}$: ensemble des demi-UVs $hc = (type, c, c')$ où $type$ définit le type de lien entre les deux demi-UVs ($type \in \{1, 2, 3\}$).
- n_s : le nombre de sections.
- n_l : le nombre d'enseignants.
- n_r : le nombre de salles.
- n_t : le nombre de timeslots.

4.5.2 *Les attributs*

Nous présentons dans ce qui suit l'ensemble des attributs :

- st(t) : slot de début du timeslot t, $st(t) \in \{1, 96\}$ pour toutes les instances de l'UTC ;
- len(t) : durée en terme de slots du timeslot t ;
- end(t) : fin du timeslot t, $end(t) = st(t) + len(t)$;
- $days_\delta(t)$: booléen, 1 si le jour δ est compris dans le timeslot t ;
- $weeks_w(t)$: booléen, 1 si la semaine w est compris dans le timeslot t ;
- $isOn_{w,\delta}(t)$: booléen, 1 si le jour δ dans la semaine w est compris dans le timeslot t ;
- fw(t) : première semaine du timeslot t ;
- fd(t) : premier jour du timeslot t ;
- req(a) : nombre de demandes de salles pour la section a ;
- stud(a) : nombre maximal d'étudiants pour la section a ;
- $limit_{mf}$: nombre d'étudiants pour la contrainte "Maximum nombre d'étudiants par étages" ;
- $limit_{mf}$: nombre de licences pour le logiciel s ;
- ψ_d : pénalité induite par la violation de la contrainte de distribution d ;

- ψ_g : pénalité induite par la violation de la contrainte de grille g ;
- ψ_t : pénalité induite par l'assignation du timeslot t à la section a ;
- γ_l : pénalité induite par l'assignation d'un timeslot non souhaité à l'enseignant l ;
- θ_l : pénalité induite par l'assignation d'une salle non souhaitée à l'enseignant l ;

4.5.3 *Les sous-ensembles*

Nous présentons dans ce qui suit l'ensemble des sous-ensembles :

- $\mathcal{A}_c = \bigcup_{k \in \mathcal{K}_c,\ p \in \mathcal{P}_{c,k},\ cl \in \mathcal{CL}_{c,k,p}} \mathcal{A}_{c,k,p,c}$: ensemble de toutes les sections pour l'UV c ;
- $\mathcal{A} = \bigcup_{c \in \mathcal{C}} \mathcal{A}_c$: ensemble de toutes les sections ;
- $\mathcal{A}_r = \{a \in \mathcal{A} \mid r \in \mathcal{R}_a\}$: ensemble de toutes les sections pour la salle r ;
- $\mathcal{A}_d$: ensemble des sections pour lesquelles la contrainte de distribution d est posée ;
- $\mathcal{A}_p$: ensemble de toutes les sections de l'activité p.

4.5.4 *Les variables de décision*

Nous présentons dans ce qui suit l'ensemble des variables de décision :

- $\forall a \in \mathcal{A},\ \forall q \in \mathcal{Q},\ \forall r \in \mathcal{R}_{a,q},\ y_a^{r,q}$: 1 si la section a est assignée à la salle r pour la demande de salle q
- $\forall a \in \mathcal{A},\ \forall t \in \mathcal{T}_a,\ \forall q \in \mathcal{Q},\ \forall r \in \mathcal{R}_{a,q},\ y_{a,t}^{r,q}$: 1 si la section a est assignée au timeslot t et à la salle r pour la demande de salle q ;
- $\forall a \in \mathcal{A},\ \forall t \in \mathcal{T}_a,\ \forall r \in \mathcal{R}_a,\ y_{a,t}^{r}$: 1 si la section a est assignée au timeslot t et à la salle r;
- $\forall r \in \mathcal{R},\ \forall t \in \mathcal{T},\ y_t^r$: 1 si la salle r est assignée au timeslot t ;
- $\forall r \in \mathcal{R},\ \forall a \in \mathcal{A},\ y_a^r$: 1 si la salle r est assignée à la section a ;
- $\forall a \in \mathcal{A},\ \forall t \in \mathcal{T}_a,\ z_{a,t}$: 1 si la section a est assignée au timeslot t ;
- $\forall l \in \mathcal{L},\ \forall a \in \mathcal{A},\ x_{a,l}$: 1 si la section a est assurée par l'enseignant l ;

4.5.5 *Les variables auxiliaires*

Nous présentons dans ce qui suit l'ensemble des variables auxiliaires :

- $\forall d \in \mathcal{SDC},\ p_d$: 1 si la contrainte de distribution d est violée ;
- $\forall g \in \mathcal{SGC},\ q_g$: 1 si la contrainte de grilles g est violée ;

4.5.6 *Les contraintes*

Nous présentons dans ce qui suit l'ensemble des contraintes du problème UTC-UCTTP :

4.5.6.1 *Salles et Timeslots*

Une salle est assignée pour chaque section pour chaque demande de salles :

$$\sum_{r \in \mathcal{R}_{a,q}} y_a^{r,q} = 1 \qquad \forall a \in \mathcal{A},\ \forall q \in \mathcal{Q} \tag{4.1}$$

Une salle r est assignée à la section a durant le timeslot t si la section a est assignée à la salle r pour la demande de salle q :

$$\sum_{t \in \mathcal{T}_a} y_{a,t}^{r,q} = y_a^{r,q} \qquad \forall a \in \mathcal{A},\ \forall q \in \mathcal{Q},\ \forall r \in \mathcal{R}_{a,q} \tag{4.2}$$

Une salle est assignée à la section a durant le timeslot t pour chaque demande de salles q si la section a est assignée au timeslot t :

$$\Big(\sum_{q \in \mathcal{Q}_a} \sum_{r \in \mathcal{R}_{a,q}} y_{a,t}^{r,q}\Big) - req(a) + 1 \leqslant z_{a,t} \leqslant \Big(\sum_{q \in \mathcal{Q}_a} \sum_{r \in \mathcal{R}_{a,q}} y_{a,t}^{r,q}\Big) \qquad \forall a \in \mathcal{A},\ \forall t \in \mathcal{T}_a \tag{4.3}$$

Contrainte d'assignation de salles :

$$\sum_{a \in \mathcal{A}} \sum_{q \in \mathcal{Q}_\dashv} y_{a,t}^{r,q} = y_t^r \qquad \forall r \in \mathcal{R},\ \forall t \in \mathcal{T} \tag{4.4}$$

Un seul timeslot par section (conflits de timeslots) :

$$\sum_{t \in \mathcal{T}_a} z_{a,t} = 1 \qquad \forall a \in \mathcal{A} \tag{4.5}$$

Interdiction de plusieurs réservations de la salle r au même moment (conflits de salle) :

$$y^r_t + y^r_{t'} \leqslant 1 \qquad \forall r \in \mathcal{R}\ \forall t \in \mathcal{T},\ \forall t' \in \mathcal{T}, t'.\mathrm{overlap}(t) \tag{4.6}$$

Excepté pour les contraintes **MustShareRoom** et **CanShareRoom** pour lesquelles une même salle peut être utilisée en même temps au même moment.

4.5.6.2 *Contraintes cumulatives*

Maximum nombre d'étudiants par étages :

$$\sum_{a \in \mathcal{A}} \sum_{r \in \mathcal{R}_{mf}} \sum_{\substack{t \in \mathcal{T}_a, \\ \tau.\mathrm{isInside}(t)}} (y^r_{a,t}).\mathrm{stud}(a) \leqslant \mathrm{limit}_{mf} \qquad \forall d \in \mathcal{DAYS},\ \forall \tau \in \mathcal{TAU} \tag{4.7}$$

Contraintes logiciels :

$$\sum_{a \in \mathcal{A}_s} \sum_{r \in \mathcal{R}} \sum_{\substack{t \in \mathcal{T}_a, \\ \tau.\mathrm{isInside}(t)}} (y^r_{a,t}).\mathrm{stud}(a) \leqslant \mathrm{limit}_s \qquad \forall s \in \mathcal{SW},\ \forall d \in \mathcal{DAYS},\ \forall \tau \in \mathcal{TAU} \tag{4.8}$$

4.5.6.3 *Les contraintes de distribution*

SameTimeSlot

Version dure

$$z_{a,t} = z_{a',t} \qquad \forall t \in \mathcal{T} \tag{4.9}$$

Version souple

$$z_{a,t} + p_d = z_{a',t} + p'_d \qquad \forall t \in \mathcal{T}, p_d, p'_d \in \{0,1\} \tag{4.10}$$

SuccessiveTimeSlot

Version dure

$$z_{a,t} \leqslant \sum_{t' \in \mathcal{T}_{a'}, t'.\mathrm{isSuccessive}(t)} z_{a',t'} \qquad \forall t \in \mathcal{T} \tag{4.11}$$

Version souple

$$z_{a,t} \leqslant \sum_{t' \in \mathcal{T}_{a'}, t'.\mathrm{isSuccessive}(t)} z_{a',t'} + p_d \qquad \forall t \in \mathcal{T},\ p_d \in \{0,1\} \tag{4.12}$$

SameStart

Version dure

$$\sum_{t\in\mathcal{T}_a, st(t)=\tau} z_{a,t} + \sum_{t'\in\mathcal{T}_{a'}, st(t')\neq\tau} z_{a',t'} \leqslant 1 \qquad \forall \tau \in \mathcal{TAU} \tag{4.13}$$

Version souple

$$\sum_{t\in\mathcal{T}_a, st(t)=\tau} z_{a,t} + \sum_{t'\in\mathcal{T}_{a'}, st(t')\neq\tau} z_{a',t'} - 1 \leqslant p_d,\ \forall \tau \in \mathcal{TAU},\ p_d \in \{0,1\} \tag{4.14}$$

SameTime

Version dure

$$z_{a,t} + \sum_{\substack{t'\in\mathcal{T}_{a'},\\ \neg((st(t)\leqslant st(t')\wedge end(t')\leqslant end(t))\\ \vee(st(t')\leqslant st(t)\wedge end(t)\leqslant end(t')))}} z_{a',t'} \leqslant 1 \qquad \forall t \in \mathcal{T}_a \tag{4.15}$$

Version souple

$$z_{a,t} + \sum_{\substack{t'\in\mathcal{T}_{a'},\\ \neg((st(t)\leqslant st(t')\wedge end(t')\leqslant end(t))\\ \vee(st(t')\leqslant st(t)\wedge end(t)\leqslant end(t')))}} z_{a',t'} - 1 \leqslant p_d \qquad \forall t \in \mathcal{T}_a,\ p_d \in \{0,1\} \tag{4.16}$$

DifferentTime

Version dure

$$z_{a,t} + \sum_{\substack{t'\in\mathcal{T}_{a'},\\ \neg((end(t)\leqslant st(t'))\vee(end(t')\leqslant st(t)))}} z_{a',t'} \leqslant 1 \qquad \forall t \in \mathcal{T}_a \tag{4.17}$$

Version souple

$$z_{a,t} + \sum_{\substack{t'\in\mathcal{T}_{a'},\\ \neg((end(t)\leqslant st(t'))\vee(end(t')\leqslant st(t)))}} z_{a',t'} - 1 \leqslant p_d \qquad \forall t \in \mathcal{T}_a,\ p_d \in \{0,1\} \tag{4.18}$$

SameDays

Version dure

$$z_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \\ Days(t) \not\subseteq Days(t') \wedge Days(t') \not\subseteq Days(t)}} z_{a',t'} \leqslant 1 \qquad \forall t \in \mathcal{T}_a \tag{4.19}$$

Version souple

$$z_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \\ Days(t) \not\subseteq Days(t') \wedge Days(t') \not\subseteq Days(t)}} z_{a',t'} - 1 \leqslant p_d \qquad \forall t \in \mathcal{T}_a,\ p_d \in \{0,1\} \tag{4.20}$$

DifferentDays

Version dure

$$z_{a,t} + \sum_{t' \in \mathcal{T}_{a'}, Days(t) \cap Days(t') \neq \emptyset} z_{a',t'} \leqslant 1 \qquad \forall t \in \mathcal{T}_a \tag{4.21}$$

Version souple

$$z_{a,t} + \sum_{t' \in \mathcal{T}_{a'}, Days(t) \cap Days(t') \neq \emptyset} z_{a',t'} - 1 \leqslant p_d \qquad \forall t \in \mathcal{T}_a,\ p_d \in \{0,1\} \tag{4.22}$$

SameRoom

Version dure

$$y_a^{r,q} + \sum_{r' \in \mathcal{R}_{a'} \setminus \{r\}} y_{a'}^{r',q} \leqslant 1 \qquad \forall r \in \mathcal{R}_a \forall q \in \mathcal{Q}_a \tag{4.23}$$

Version souple

$$y_a^{r,q} + \sum_{r' \in \mathcal{R}_{a'} \setminus \{r\}} y_{a'}^{r',q} - 1 \leqslant p_d \qquad \forall r \in \mathcal{R}_a \forall q \in \mathcal{Q}_a, p_d \in \{0,1\} \tag{4.24}$$

DifferentRooms

Version dure

$$y_a^{r,q} + \sum_{q' \in \mathcal{Q}_{a'}} y_{a'}^{r,q'} \leqslant 1 \qquad \forall r \in \mathcal{R}_a \forall q \in \mathcal{Q}_a \tag{4.25}$$

Version souple

$$y_a^{r,q} + \sum_{q' \in Q_{a'}} y_{a'}^{r,q'} - 1 \leqslant p_d \qquad \forall r \in \mathcal{R}_a \forall q \in Q_a, p_d \in \{0,1\} \tag{4.26}$$

Overlap

Version dure

$$y_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \\ \neg((st(t') \leqslant end(t)) \wedge (st(t) \leqslant end(t')) \\ \wedge (Days(t) \cap Days(t') \neq \emptyset) \\ \wedge (Weeks(t) \cap Weeks(t') \neq \emptyset))}} y_{a',t'} \leqslant 1 \qquad \forall t \in \mathcal{T}_a \tag{4.27}$$

Version souple

$$y_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \\ \neg((st(t') \leqslant end(t)) \wedge (st(t) \leqslant end(t')) \\ \wedge (Days(t) \cap Days(t') \neq \emptyset) \\ \wedge (Weeks(t) \cap Weeks(t') \neq \emptyset))}} y_{a',t'} - 1 \leqslant p_d \qquad \forall t \in \mathcal{T}_a, p_d \in \{0,1\} \tag{4.28}$$

NotOverlap

Version dure

$$y_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \\ ((st(t') \leqslant end(t)) \wedge (st(t) \leqslant end(t')) \\ \wedge (Days(t) \cap Days(t') \neq \emptyset) \\ \wedge (Weeks(t) \cap Weeks(t') \neq \emptyset))}} y_{a',t'} \leqslant 1 \qquad \forall t \in \mathcal{T}_a \tag{4.29}$$

Version souple

$$y_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \\ ((st(t') \leqslant end(t)) \wedge (st(t) \leqslant end(t')) \\ \wedge (Days(t) \cap Days(t') \neq \emptyset) \\ \wedge (Weeks(t) \cap Weeks(t') \neq \emptyset))}} y_{a',t'} - 1 \leqslant p_d \qquad \forall t \in \mathcal{T}_a, p_d \in \{0,1\} \tag{4.30}$$

Precedence

Version dure

$$z_{a,t} \leqslant \sum_{t' \in \mathcal{T}_{a'}, t'.isAfter(t)} z_{a',t'} \qquad \forall t \in \mathcal{T} \tag{4.31}$$

Version souple

$$z_{a,t} \leqslant \sum_{t' \in \mathcal{T}_{a'}, t'.\text{isAfter}(t)} z_{a',t'} + p_d \qquad \forall t \in \mathcal{T}, p_d \in \{0,1\} \tag{4.32}$$

Version souple
SameAttendees

Version dure

$$y^r_{a,t} + \sum_{t' \in \mathcal{T}_{a'}, t.\text{overlap}(t')} z_{a',t'} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \neg t.\text{overlap}(t'), \\ r' \in \mathcal{R}_{a'}, t.\text{overlap}(t',r',r)}} y^{r'}_{a',t'} \leqslant 1 \qquad \forall t \in \mathcal{T}_a,\ \forall r \in \mathcal{R}_a \tag{4.33}$$

Version souple

$$y^r_{a,t} + \sum_{t' \in \mathcal{T}_{a'}, t.\text{overlap}(t')} z_{a',t'} + \sum_{\substack{t' \in \mathcal{T}_{a'}, \\ \neg t.\text{overlap}(t'), \\ r' \in \mathcal{R}_{a'}, \\ t.\text{overlap}(t',r',r)}} y^{r'}_{a',t'} - 1 \leqslant p_d \qquad \forall t \in \mathcal{T}_a,\ \forall r \in \mathcal{R}_a, p_d \in \{0,1\} \tag{4.34}$$

4.5.6.4 *Les contraintes de grilles*

Nous considérons la contrainte : toutes les combinaisons de n UVs doivent être compatibles. Pour cette contrainte, on considère d'abord toutes les combinaisons de configurations.

On construit des ensembles S_i, où chaque ensemble contient une combinaison d'une section de chaque classe de chaque activité de chaque configuration de chaque UV de la grille. On note le nombre de combinaisons possibles npc, donc $i \in I$, avec $I = \{1, 2, ..., npc\}$.

Pour chaque ensemble S_i, on introduit une variable $\alpha_i, i \in I$ qui va être mise à 1 si toutes les combinaisons des sections S_i sont compatibles.

Pour chaque ensemble S_i et pour chaque combinaison de sections $a, a' \in S_i$, on pose une variante de la contrainte **SameAttendies** qui sera définie ultérieurement : $VarSameAttendies(a, a', \beta_{a,a'})$.

Nous introduisons deux variables auxiliaires :

- La variable $\beta_{a,a'} = 1$ si un étudiant peut assister aux deux sections a et a'.

- La variable $\beta_{a,a',t,r} = 1$ si un étudiant peut assister aux deux sections a et a', tels que la section a est programmée dans le timeslot t dans la salle r.

On lie les variables α_i et $\beta_{a,a'}$ avec les contraintes :

$$\sum_{a,a' \in S_i} (1 - \beta_{a,a'}) \geqslant 1 - \alpha_i \qquad \forall i \in I \tag{4.35}$$

$$\alpha_i \leqslant \beta_{a,a'} \qquad \forall i \in I,\ \forall a, a' \in S_i \tag{4.36}$$

Ensuite on impose qu'au moin une des variables α_i est égale à 1 avec la contrainte :

$$\sum_{i \in I} \alpha_i \geqslant 1 \tag{4.37}$$

On définit la contrainte $VarSameAttendies$ comme suit :

$$(1 - \beta_{a,a',t,r}) \leqslant \frac{1}{2}(y^r_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'},\\ t.overlap(t')}} z_{a',t'} + \sum_{\substack{t' \in \mathcal{T}_{a'},\\ \neg t.overlap(t'),\\ r' \in \mathcal{R}_{a'},\\ t.overlap(t',r',r)}} y^{r'}_{a',t'}) \qquad \forall t \in \mathcal{T}_a, \forall r \in \mathcal{R}_a \tag{4.38}$$

$$y^r_{a,t} + \sum_{\substack{t' \in \mathcal{T}_{a'},\\ t.overlap(t')}} z_{a',t'} + \sum_{\substack{t' \in \mathcal{T}_{a'},\\ \neg t.overlap(t'),\\ r' \in \mathcal{R}_{a'},\\ t.overlap(t',r',r)}} y^{r'}_{a',t'} - 1 \leqslant M(1 - \beta_{a,a',t,r}) \qquad \forall t \in \mathcal{T}_a, \forall r \in \mathcal{R}_a \tag{4.39}$$

Ensuite on lie les variables $\beta_{a,a',t,r}$ et $\beta_{a,a'}$ comme suit :

$$\sum_{\substack{t \in \mathcal{T}_a \\ r \in \mathcal{R}_a}} (1 - \beta_{a,a',t,r}) \geqslant 1 - \beta_{a,a'} \qquad \forall i \in I,\ \forall a, a' \in S_i \tag{4.40}$$

$$\beta_{a,a'} \leqslant \beta_{a,a',t,r} \qquad \forall i \in I,\ \forall t \in \mathcal{T}_a,\ \forall r \in \mathcal{R}_a\ \forall a, a' \in S_i \tag{4.41}$$

Pour la version souple des contraintes de grilles, on remplace la contrainte 4.37 par la contrainte suivante :

$$1 - q_g \leqslant \sum_{i \in I} \alpha_i \leqslant M(1 - q_g) \tag{4.42}$$

4.5.6.5 *Les demi-UVs*

Si une contrainte de demi-Uvs $hc = (type, c, c')$ est définie entre les UVs c et c', alors nous posons les contraintes de distribution suivantes entre les combinaisons de sections valables entre les deux UVs selon le type de le contrainte demi-UVs :

- **Type = 1** : mêmes étudiants, même timeslots et mêmes salles. On utilise les contraintes de distribution : SameAttendies, SameStart, SameTime, SameDays et SameRoom ;
- **Type = 2** : même timeslots. On utilise les contraintes de distribution : SameStart, SameTime et SameDays ;
- **Type = 3** : mêmes étudiants. On utilise la contrainte de distribution : SameAttendies.

4.5.6.6 *Les contraintes enseignant*

Une section est assurée par un enseignant (Conflit d'enseignants) :

$$\sum_{l \in \mathcal{L}} x_{a,l} = 1 \qquad \forall a \in \mathcal{A} \tag{4.43}$$

Assigner le nombre correct de sections pour chaque enseignant (Couverture d'enseignants) :

$$\sum_{a \in \mathcal{A}_p} x_{a,l} = sec(l,p) \qquad \forall l \in \mathcal{L}\ \forall p \in \mathcal{P}_l \tag{4.44}$$

Les mêmes sections d'une classe cl sont assurées par le même enseignant l :

$$x_{a,l} = x_{a',l} \qquad \forall cl \in \mathcal{CLS},\ \forall a, a' \in \mathcal{A}_{cl},\ \forall l \in \mathcal{L} \tag{4.45}$$

Les timeslots requis par un enseignant l pour une activité p (s'ils existent) $t_{l,p} = t$:

$$x_{a,l} \leqslant z_{a,t} \qquad \forall a \in \mathcal{A}_p \tag{4.46}$$

Les timeslots souhaités par un enseignant l pour une activité p :

$$x_{a,l} - z_{a,t} \leqslant pt \wedge z_{a,t} - x_{a,l} \leqslant pt \qquad \forall a \in \mathcal{A}_p,\ \forall t \in \mathcal{T}_l \tag{4.47}$$

Les timeslots non souhaités par un enseignant l pour une activité p :

$$x_{a,l} + z_{a,t} - 1 \leqslant pt \qquad \forall a \in \mathcal{A}_p,\ \forall t \in \bar{\mathcal{T}}_l \tag{4.48}$$

Les salles requises par un enseignant l pour une activité p (si elles existent) $r_{l,p} = r$:

$$x_{a,l} \leqslant y_a^r \qquad \forall a \in \mathcal{A}_p \tag{4.49}$$

Les salles interdites par un enseignant l pour une activité p (si elles existent) $\bar{r}_{l,p} = r$:

$$x_{a,l} + y^r_a \leqslant 1 \qquad \forall a \in \mathcal{A}_p \tag{4.50}$$

Les salles souhaitées par un enseignant l pour une activité p :

$$x_{a,l} - y^r_a \leqslant pr \qquad \forall a \in \mathcal{A}_p,\ \forall r \in \mathcal{R}_{l,p} \tag{4.51}$$

Les salles non souhaitées par un enseignant l pour une activité p :

$$x_{a,l} + y^r_a - 1 \leqslant pr \qquad \forall a \in \mathcal{A}_p, \forall r \in \bar{\mathcal{R}}_{l,p} \tag{4.52}$$

4.5.6.7 *La fonction objectif*

Le terme 4.53a est lié à une pénalité induite lorsqu'une section a est assignée à un timeslot t.

Pour chaque contrainte de distribution souple d est définie une variable de pénalité p_d. Les contraintes sont ajoutées à la fonction objectif dans le terme 4.53d.

Pour chaque contrainte de grille souple g est définie une variable de pénalité q_d. Les contraintes sont ajoutées à la fonction objectif dans le terme 4.53e.

Pour chaque enseignant est définie une variable de pénalité de temps pt_l et une variable de pénalité de salle pr_l. Les contraintes sont ajoutées à la fonction objectif dans les termes 4.53b et 4.53c respectivement.

La distance par rapport à un EDT de référence est pénalisé dans l'équation 4.53f.

Nous obtenons donc la fonction objectif à minimiser définie comme suit :

$$\sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}_a} \psi_t z_{a,t} \tag{4.53a}$$

$$+ \sum_{l \in \mathcal{L}} \gamma_l pt_l \tag{4.53b}$$

$$+ \sum_{l \in \mathcal{L}} \theta_l pr_l \tag{4.53c}$$

$$+ \sum_{d \in \mathcal{SDC}} \psi_d p_d \tag{4.53d}$$

$$+ \sum_{g \in \mathcal{SGC}} \psi_g q_g \tag{4.53e}$$

$$+ \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}_a} \psi_{dist}\ z_{a,t}\ dist(z_{a,t}) \tag{4.53f}$$

4.5.6.8 *Les Fonctions*

La fonction Overlap prend différents paramètres en entrée et renvoie un résultat booléen si le timeslot et les paramètres d'entrée se chevauchent.

- $t.overlap(t')$ renvoie vrai si les timeslots t et t' ont au moins une semaine et un jour en commun et que les slots horaires se chevauchent;

- $t.overlap(t', r', r)$ renvoie vrai si les timeslots t et t' ont au moins une semaine et un jour en commun et que les slots horaires se chevauchent ou qu'il n'est pas possible d'aller d'une salle à l'autre dans le temps qui sépare les deux ;

- $\tau.isInside(t)$ renvoie vrai si le slot τ est à l'intérieur du timeslot t.

La fonction distance $dist(z_{a,t})$ renvoie un entier qui représente la distance entre le créneau actuel de la section a et le créneau de la section a dans l'EDT de référence. Cette distance est calculée par rapport à l'écart en terme d'heures et de jours.

La fonction de succession de timeslots $t'.isSuccessive(t)$ renvoie 1 si le timeslot t' est successeur immédiat du timeslot t. Par exemple, t se termine à 10:00 et t' commence à 10:15.

La fonction de précédence de timeslots $t'.isAfter(t)$ renvoie 1 si le timeslot t' est considéré comme successeur du timeslot t dans la même semaine de l'horizon de planification.

4.6 HEURISTIQUE AIDCH POUR LE PROBLÈME UTC-UCTTP

Les solveurs peuvent rencontrer des difficultés pour obtenir des solutions suffisamment bonnes dans des délais de traitement courts en exécutant le modèle ILP.

Pour palier ce problème, nous proposons une heuristique Adaptive Iterative Destruction Construction Heuristic (AIDCH) pour calculer des solutions de bonne qualité pour le problème UTC-UCTTP.

Nous présentons dans l'algorithme 1 sa structure générale avant de donner un aperçu des composants.

Algorithm 1: AIDCH for UTC-UCTTP

Input : An instance of UTC-UCTTP
Output : S_{best} best solution found
Parameters: D_{limit} limit for diversification degree,
Variables : $MaxIter$ maximum number of iterations without any improvement prior stopping the AIDCH
D_{max} current diversification degree
S_{cur} solution under work

```
1  iter := 0 /* number of iterations */
2  MaxIter := ε.n_s
3  D_max := 3
4  D_limit := ⌈(n_s.n_l)/n_r^2⌉
5  S_best, S_cur:= BuildFeasibleSchedule()
6  while iter < MaxIter do
7      k := rand(1,D_max)
8      S_cur := AdaptiveRandomDestruction(S_cur,k)
9      S_cur := ApplyRandoConstruction(S_cur) /* insert all removed sections in S_cur */
10     if Obj(S_cur) < Obj(S_best) then
11         S_best := S_cur
12         iter := 0
13         D_max := 3
14     else
15         iter++
16         D_max := Min(D_max+1,D_limit)
```

Une solution initiale S_{old} est calculée à l'aide d'une heuristique $BuildFeasibleSchedule()$. Cette première solution est construite pour respecter toutes les contraintes dures.

Le principe de cette méthode de construction d'une solution initiale est de remplir des listes de priorités avec toutes les sections de l'instance. Ses listes ont été définies en se basant sur l'ordre de traitement des UVs utilisé par le SME dans leur processus de conception actuel. Pour chaque liste l_i, on associe un facteur de priorité f_{l_i}. Plus la liste est prioritaire , plus le facteur est petit. Une fois les listes remplies, nous définissons pour chaque section a affecté à une liste l_i un score d'insertion défini comme suit :

$$InsertScore_a = f_{l_i} * Poss(a)$$

Le score est calculé en multipliant le facteur de priorité de la liste de priorité l de la section a par le nombre $Poss(a)$ qui représente le nombre de possibilités de placement de la section a. Les listes sont triées selon l'ordre croissant des scores d'insertion des sections. A chaque itération, on procède au placement d'une section en tête de l'une des listes de priorités. La section choisie est celle qui possède le score d'insertion le plus petit. Le placement est effectué en respectant toutes les contraintes dures. A l'issue de l'exécution de cet algorithme de construction, on arrive à constituer une solution réalisable.

Ensuite, pour chaque itération de l'heuristique AIDCH, nous supprimons k sections aléatoirement de la solution courante. Nous définissons D_{max} comme le degré de diversification. Cette valeur est initialisée à trois puis incrémentée après chaque itération sans amélioration jusqu'à D_{limit}. Nous choisissons de fixer $D_{limit} = \left\lceil \frac{n_s.n_l}{n_r^2} \right\rceil$. Si une amélioration est trouvée, nous réinitialisons D_{max} à trois pour explorer entièrement le voisinage de la nouvelle solution. Ce mécanisme de diversification adaptative permet d'élargir la recherche autour d'une solution dans le but d'obtenir une meilleure solution.

Nous implémentons un algorithme adaptatif basé sur la meilleure insertion possible (Best Insertion Algorithm, BIA) qui évalue également les affectations réalisables à l'aide du BIDC. L'algorithme BIA sera défini dans la section suivante.

Lorsqu'un nombre maximum d'itérations $MaxIter$ sans aucune amélioration est atteint, l'algorithme AIDCH s'arrête, puis renvoie la meilleure solution trouvée jusqu'ici dans les itérations S_{best}.

Nous choisissons de définir $MaxIter$ à $\varepsilon.n_s$, n_s étant le nombre de sections, où ε doit être réglé pour obtenir un bon compromis entre des solutions de bonne qualité et des temps de traitement raisonnables.

4.7 MÉTAHEURISTIQUE ALNS POUR LE PROBLÈME UTC-UCTTP

Les approches de résolution basées sur les métaheuristiques ont été largement rapportées dans la littérature pour traiter une grande variété de problèmes d'optimisation, pour une étude complète nous invitons le lecteur à se reporter [Hussain et al., 2019].

Nous proposons une métaheuristique Adaptive Large Neighborhood Search (ALNS) pour calculer des solutions de bonne qualité pour le problème UTC-UCTTP. Pour une revue plus générale et récente sur la métaheuristique ALNS et ses applications, nous invitons le lecteur à se référer à [Mara et al., 2022]. Nous présentons dans l'algorithme 2 sa structure générale avant de donner un aperçu des composants.

Algorithm 2: ALNS for UTC-UCTTP

Input : An instance of UTC-UCTTP
Output : S_{best} best solution found
Parameters: D_{limit} limit for diversification degree,
Variables : $MaxIter$ maximum number of iterations without any improvement prior stopping the ALNS
$AcceptIter$ number of iterations without any improvement prior accepting a degradation,
D_{max} current diversification degree, M_d destruction method,
S_{old} current solution to be improved, S_{cur} solution under work

```
i, i' := 0 /* i, i' number of iterations */
MaxIter := ε.n_s
AcceptIter := ⌈(n_s.n_l)/n_r^2⌉
D_max := 3
D_limit := ⌈n_s/n_r⌉
S_best, S_old := BuildFeasibleSchedule()
while i < MaxIter do
    M_d := ChooseDestructionMethod()
    k := rand(1,D_max)
    S_cur := AdaptiveDestruction(S_old, k, M_d)
    S_cur := ApplyConstruction(S_cur)
    if Obj(S_cur) < Obj(S_best) then
        S_best, S_old := S_cur
        i, i' := 0
        D_max := 3
    else
        if (i' ⩾ AcceptIter and AcceptDegradation(S_cur, S_old)) then
            S_old := S_cur
            i' := 0
        i++
        i'++
        D_max := Min(D_max+1, D_limit)
    UpdateDestructionsScores()
```

Une solution initiale S_{old} est calculée à l'aide de l'heuristique $BuildFeasibleSchedule()$ présentée dans la section précédente. Cette première solution est construite pour respecter toutes les contraintes dures.

Une méthode de destruction M_d est choisie au hasard parmi **D**estruction **A**léatoire (Random Destruction, RD), **D**estruction **I**ntelligente (Smart Destruction, SD), **D**estruction orientée **S**alle (Room Destruction, RMD) et **D**estruction orientée **E**nseignant (Teacher Destruction, TCD) en utilisant la procédure de choix $ChooseDestructionMethod()$.

La méthode de **D**estruction **I**ntelligente utilise le **C**ritère de **M**eilleure **D**estruction ou **I**nsertion (Best Insertion Destruction Criterion, BIDC) pour évaluer l'impact de la désaffectation des sections.

Nous supprimons $k \leqslant D_{max}$ sections à chaque itération. Nous définissons D_{max} comme le degré de diversification. Cette valeur est initialisée à trois puis incrémentée après chaque itération sans amélioration jusqu'à D_{limit}. Nous choisissons de fixer $D_{limit} = \left\lceil \frac{n_s . n_l}{n_r^2} \right\rceil$ qui représente le nombre moyen de sections et d'enseignants pouvant être affectés par salle. Si une amélioration est trouvée, nous réinitialisons D_{max} à trois pour explorer entièrement le voisinage de la nouvelle solution. Ce mécanisme de diversification adaptative permet d'élargir la recherche autour d'une solution dans le but d'obtenir une meilleure solution.

En appliquant la procédure $AdaptiveDestruction(S_{old}, k, M_d)$ nous obtenons la solution courante S_{cur}. Les sections désaffectées sont insérées provisoirement dans les listes de priorités correspondantes qui ont été définies lors de la phase de construction de la solution initiale afin d'établir un meilleur ordre d'insertion. En effet, le fait de bien choisir l'ordre d'insertion permet à l'ALNS d'éviter de se trouver dans des situations où elle n'arrive pas à trouver une affectation réalisable pour une section.

La méthode de construction vise à compléter et à améliorer la solution actuelle. Nous implémentons un algorithme adaptatif basé sur la meilleure insertion possible (Best Insertion Algorithm, BIA) qui évalue également les affectations réalisables à l'aide du BIDC.

Nous insérons ensuite toutes les sections qui ont été désaffectées tout en respectant les contraintes dures en utilisant $ApplyConstruction(S_{cur})$.

Pour éviter d'être bloqué dans des optima locaux, l'approche ALNS nécessite une procédure d'acceptation qui rend possible la sélection d'une solution de faible qualité dans le but d'explorer d'autres parties de l'espace de recherche. Nous utilisons une approche basée sur le principe d'enregistrement à enregistrement (Record-to-record) (Dueck, 1993). À condition qu'un certain nombre d'itérations $AcceptIter$ sans aucune amélioration soient effectuées, et à condition qu'une solution de qualité inférieure soit acceptée par $AcceptDegradation(S_{cur}, S_{old})$, nous rendons possible de continuer à explorer l'espace de recherche en utilisant cette solution de qualité inférieure. Nous choisissons de définir $AcceptIter$ à n_s, l'idée générale est d'augmenter le temps de traitement à mesure que le nombre de sections n_s augmente.

Le succès des méthodes de destruction peut varier en fonction de l'instance. Un choix adaptatif conduit généralement à de meilleurs résultats plutôt que de fixer les choix une

fois pour toutes. Chaque méthode de destruction a un score qui représente sa part dans une roue de roulette. La procédure $UpdateDestructionsScores()$ met à jour les scores.

Lorsqu'un nombre maximum d'itérations $MaxIter$ sans aucune amélioration est atteint, l'algorithme ALNS s'arrête, puis renvoie la meilleure solution trouvée S_{best}. Nous choisissons de définir $MaxIter$ à $\varepsilon.n_s$, où ε doit être réglé pour obtenir un bon compromis entre des solutions de bonne qualité et des temps de traitement raisonnables.

Des mécanismes adaptatifs de destruction, de construction et des procédures d'acceptation sont utilisées. Plusieurs paramètres doivent être réglés pour obtenir une bonne efficacité de ces mécanismes adaptatifs.

Critère de Meilleure Destruction ou Insertion, BIDC

Nous utilisons le BIDC dans la méthode **D**estruction **I**ntelligente et dans l'algorithme adaptatif basé sur la meilleure insertion possible BIA. Soit (a, t, r, l) un quadruplet pour une section, un timeslot, une salle et un enseignant. Soit une solution à améliorer S_{old} ou une solution courante S_{cur}, nous évaluons une désaffectation ou une affectation d'un quadruplet en calculant le BIDC comme suit :

$$TP^{\alpha}.\, SDP^{\beta}.\, SGP^{\gamma}.\, LP^{\theta}.\, DP^{\omega}$$

Le BIDC est composé d'un terme pour chaque contrainte souple : TP est pour les pénalités relatives aux timeslots, SDP est pour les pénalités relatives aux contraintes de distribution, SGP est pour les pénalités relatives aux contraintes de grilles, LP est pour les pénalités relatives aux contraintes enseignants et DP est pour les pénalités relatives à la distance par rapport à un EDT de référence.

La valeur BIDC prend la valeur $+\infty$ lorsqu'une contrainte stricte est violée.

Les valeurs des paramètres $(\alpha, \beta, \gamma, \theta, \omega)$ sont gérées par une stratégie adaptative qui porte sur le BIA, cela permet d'adapter l'importance relative des termes au cours du déroulement de l'algorithme.

Méthodes de destruction et mécanisme adaptatif pour choisir une méthode de destruction

La procédure $AdaptiveDestruction(S_{old}, k, M_d)$ permet de désaffecter certaines sections de S_{old}, solution à améliorer. Étant donné une valeur k pour le nombre de sections à désaffecter, une des quatre méthodes de destruction suivantes s'applique :

Destruction aléatoire (RD) : les sections sont sélectionnées aléatoirement ;

Destruction intelligente (SD) : les sections sont sélectionnées à l'aide du BIDC ;

Destruction orientée Salle (RMD) : les sections sont sélectionnées parmi celles affectées à une salle donnée. La salle est choisie aléatoirement à l'intérieur de cette méthode de destruction. Si le nombre de sections de la salle sélectionnée est inférieur au nombre de sections à désaffecter, une autre salle est choisie ensuite pour compléter le nombre de sections à désaffecter, et ainsi de suite ;

Destruction orientée Enseignant (TCD) : les sections sont sélectionnées parmi celles affectées à un enseignant donné. L'enseignant est choisi aléatoirement à l'intérieur de

cette méthode de destruction. Si le nombre de sections de l'enseignant sélectionné est inférieur au nombre de sections à désaffecter, un autre enseignant est choisi ensuite pour compléter le nombre de sections à désaffecter, et ainsi de suite.

Pour la méthode **D**estruction **I**ntelligente, nous considérons tous les quadruplets (a, t, r, l) des sections affectées, et nous évaluons leurs scores BIDC.

Nous utilisons le jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega)$ qui a produit la meilleure solution à l'itération précédente de BIA (voir **Méthode de construction**). Les scores sont ensuite utilisés comme parts dans une roulette qui sert à sélectionner les sections k à désaffecter. Notre objectif est de sélectionner les sections affectées avec le BIDC le plus élevé dans le but d'affecter ces sections à de meilleurs salles, timeslots et enseignants.

La sélection d'une méthode de destruction est gérée à l'aide d'un mécanisme adaptatif. Soit $DeSc_{ij}$ le **S**core de **D**estruction de la méthode $j \in \{RD, SD, RMD, TCD\}$ à l'itération i. Après chaque itération i, les scores de destruction sont mis à jour par $UpdateDestructionsScores()$ comme suit :

$DeSc_{ij} = (1 + \lambda).DeSc_{(i-1)j}$ si $Obj(S_{cur}) < Obj(S_{best})$;

$DeSc_{ij} = (1 + (1/2)\lambda).DeSc_{(i-1)j}$ si $Obj(S_{cur}) < Obj(S_{old})$ et $Obj(S_{cur}) \geqslant Obj(S_{best})$;

$DeSc_{ij} = (1 - (1/2)\lambda).DeSc_{(i-1)j}$ si $Obj(S_{cur}) \geqslant Obj(S_{old})$ et $Obj(S_{cur}) \geqslant Obj(S_{best})$;

$DeSc_{ij} = DeSc_{(i-1)j}$ si la méthode de destruction j n'est pas utilisé.

L'objectif est de privilégier la méthode de destruction qui obtient le meilleur résultat au cours de l'algorithme. Cependant, cela ne peut pas être fait au détriment définitif de l'un contre l'autre puisque l'efficacité relative peut dépendre de l'instance et peut également changer au cours de l'algorithme. Le paramètre λ est utilisé pour lisser le renforcement et doit être ajusté pour obtenir un mécanisme adaptatif efficace.

Méthode de construction

Le $\mathtt{ApplyConstruction}(S_{cur})$ est utilisé pour compléter la solution courante S_{cur} en affectant toutes les sections désaffectées. Étant donné S_{cur}, la méthode de construction utilise l'algorithme BIA qui est basé sur un mécanisme adaptatif pour gérer les paramètres $(\alpha, \beta, \gamma, \theta, \omega)$.

L'algorithme BIA (voir Algorithme 3) utilise S_{cur} la solution partielle en cours, un jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega)$ et une section a comme entrées, et il essaie d'insérer la section a dans S_{cur}.

Le BIA utilise le **C**ritère de **M**eilleure **D**estruction ou **I**nsertion (BIDC) pour évaluer toutes les insertions réalisables (a, t, r). La meilleure insertion est effectuée. Étant donné un jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega)$, une exécution de BIA renvoie S_{best}, la meilleure solution trouvée qui peut ensuite éventuellement être utilisée comme nouvelle S_{cur} pour la prochaine itération de l'ALNS.

Algorithm 3: BIA for UTC-UCTTP

```
Input  : S_cur a partial solution under work
         (α, β, γ, θ, ω) parameter set
         a section to be assigned
Output : S_best solution found over the BIA algorithm
Variables : (t,r,l)* best triplet
1 S_curbest := S_cur (t,r,l)* := (∅, ∅, ∅)
  /* Using BIDC to assess, find the best triplet, if any */
2 foreach t ∈ T_a do
3     foreach r ∈ R_a do
4         foreach l ∈ L do
5             ComputeBIDC(a,t,r,l)
6             UpdateBestTriplet (t,r,l)*
7 Insert(S_cur, (a,t,r,l)*)
8 S_best := S_cur
```

Nous utilisons un mécanisme adaptatif pour gérer la construction d'une nouvelle S_{cur} en utilisant le BIA. Nous exécutons séparément quatre BIA avec différentes valeurs du jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega)$. Soit α_{i-1}, β_{i-1}, γ_{i-1}, θ_{i-1} et ω_{i-1} les meilleures valeurs de paramètres obtenues lors d'une précédente itération de l'ALNS. Les valeurs α, β, γ, θ et ω sont choisies aléatoirement dans l'espace à cinq dimensions ayant pour centre $(\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}, \theta_{i-1}, \omega_{i-1})$ et la longueur de côté ϕ.

La meilleure solution S_{best} obtenue parmi les quatre exécutions est conservée et le jeu de paramètres qui la produit est stocké pour être utilisé pour la prochaine itération. Ce meilleur jeu de paramètres est utilisé par **D**estruction **I**ntelligente et l'algorithme BIA lorsqu'il est choisi. Ce mécanisme adaptatif permet d'accélérer la convergence de l'algorithme ALNS vers une bonne solution.

Le paramètre de longueur de côté ϕ doit être ajusté pour obtenir une bonne performance du mécanisme adaptatif pour la méthode de destruction intelligente et pour la méthode de construction BIA.

Stratégie d'acceptation

La stratégie d'acceptation que nous avons mise en œuvre dans $AcceptDegradation(S_{cur}, S_{old})$ est basée sur l'approche enregistrement à enregistrement (Record-to-recod) introduite dans [Dueck, 1993]. À condition que $AcceptIter$ itérations aient été effectuées sans améliorer la qualité de S_{old}, cette solution peut être remplacée par S_{cur} une solution de moins bonne qualité. Nous utilisons un taux d'acceptation τ à ajuster qui joue le rôle d'un paramètre de déviation. La solution S_{cur} prend la place de S_{old} si $(S_{cur} - S_{old})/S_{old} \leqslant \tau$.

4.8 EXPÉRIMENTATIONS, TESTS ET RÉSULTATS

Dans nos expérimentations, nos objectifs étaient : (i) fournir une synthèse sur le réglage des paramètres de la métaheuristique ALNS permettant d'obtenir les meilleurs résultats ; (ii) montrer l'efficacité des mécanismes adaptatifs que nous avons mis en place pour les méthodes de destruction et de construction ; (iii) évaluer l'apport des méthodes de destruction ; (iv) comparer les performances entre le modèle ILP (quand cela est possible), l'heuristique AIDCH et la métaheuristique ALNS ; (v) évaluer la qualité des solutions calculées par AIDCH et ALNS sur les différentes instances.

Les tests ont été effectués en utilisant C++ compilé avec gcc version 7.5.0, en utilisant STL, à l'aide d'un solveur CPLEX 12.10 [IBM, 2020] avec un seul thread et le paramètre ILPEmphasis défini sur faisabilité, sur une machine avec un processeur Intel(R) Xeon(R) X7542 à 2,6 GHz et 64 Go de RAM.

Les instances

Nous avons testé les méthodes AIDCH et ALNS sur un benchmark composé de 20 instances. La première instance (Instance_00) est une instance réelle qui a été construite suite aux différents échanges avec le SME de l'UTC. Les 15 instances suivantes (de Instance_01 à Instance_15) sont des instances que nous avons générées à partir de la première instance en utilisant un générateur d'instances. Chaque instance contient 1337 sections, 139 salles et 379 enseignants. Le générateur d'instances permet de générer des contraintes de compatibilité, des contraintes enseignants et des contraintes de distribution afin de varier la difficulté des instances. Quatre instances dites "instances jouets" ont été générées afin de pouvoir valider le ILP et vérifier la convergence de l'ALNS vers l'optimalité pour ces petites instances.

Réglage des paramètres

L'ALNS que nous proposons utilise des paramètres qui doivent être réglés pour obtenir une bonne efficacité des mécanismes de destruction, de construction, d'adaptation et des procédures d'acceptation. Nous récapitulons ces paramètres dans le tableau 6.1.

Nous avons réalisé des expériences préliminaires afin de calibrer les paramètres. Cinq instances sont sélectionnées aléatoirement pour le réglage. L'ALNS étant une méthode de recherche probabiliste, les expériences ont été répétées dix fois avec une graine aléatoire différente.

Tableau 4.1: Parametres ϕ, ε, λ et τ à régler pour l'ALNS pour le UTC-UCTTP.

ϕ	longueur de coté, mécanisme adaptatif pour le jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega)$
ε	$MaxIter = \varepsilon.n_c$, nombre maximum d'itérations sans aucune amélioration
λ	mécanisme adaptatif de gestion des méthodes de destruction
τ	taux d'acceptation

Nous évaluons le réglage des paramètres à l'aide de la métrique **E**rreur **R**elative en **P**ourcentage (Relative Percentage Error, RPE) calculée comme $RPE = 100 \cdot \frac{Z_{min} - Z_{best}}{Z_{best}}$, où Z_{best} désigne le meilleur résultat que nous avons obtenu sur toutes les exécutions effectuées pour une instance, et Z_{min} désigne le meilleur résultat que nous avons obtenu parmi les dix exécutions effectuées. Les valeurs **RPE** sont moyennées et rapportées en pourcentage dans les figures.

Nous avons commencé par régler le premier paramètre ϕ, puis nous avons réglé chaque paramètre l'un après l'autre compte tenu de l'ordre du tableau 6.1. Les valeurs initiales que nous utilisons pour régler d'abord ϕ sont $\varepsilon = 5$, $\lambda = 1$ et $\tau = 0.01$. Ensuite, pour régler un paramètre on retient le meilleur réglage des autres paramètres trouvés avant de procéder à son réglage.

Les valeurs de ϕ, la longueur du côté pour le mécanisme de construction adaptative, peuvent affecter significativement les performances de l'ALNS. Ce mécanisme permet d'adapter les valeurs des paramètres $(\alpha, \beta, \gamma, \theta, \omega)$ de sorte que l'importance relative des termes du BIDC peut changer au cours de l'algorithme.

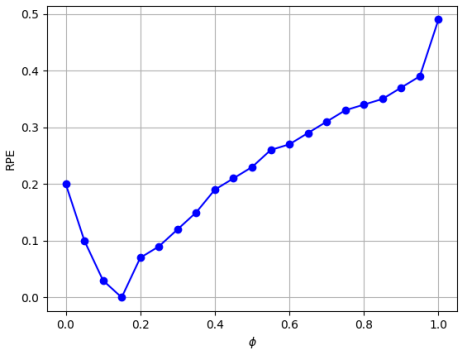

Fig 4.2: Calibrage de ϕ, impact sur le RPE moyen.

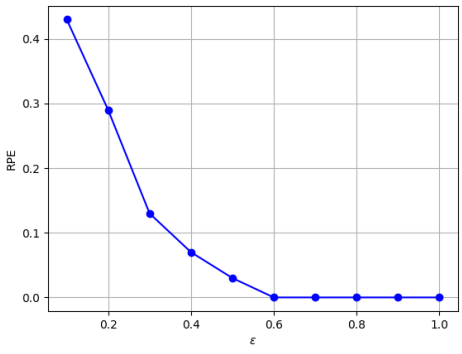

Fig 4.3: Calibrage de ε, impact sur le RPE moyen.

Dans la figure 4.2, nous montrons l'impact sur le RPE moyen en faisant varier ϕ de 0,0 à 1,0 avec un pas de 0,025. Lorsque ϕ est inférieur à 0,15, les quatre recherches parallèles indépendantes utilisent des valeurs de $(\alpha, \beta, \gamma, \theta, \omega)$ trop proches de la valeur initiale. Lorsque ϕ est supérieur à 0,15, le RPE moyen est dégradé, comme le montrent

les valeurs moyennes du RPE. Nous avons choisi de fixer ϕ à 0,15 pour lequel le RPE moyen minimum est obtenu. Comme le mécanisme est aussi utilisé par la méthode AIDCH, nous avons décidé de garder la même valeur pour cette méthode aussi.

Le nombre maximum d'itérations sans aucune amélioration est $MaxIter = \varepsilon.n_s$ où n_s est le nombre de sections. Comme on pouvait s'y attendre, le temps de traitement augmente avec la valeur ε. Comme on pouvait s'y attendre avec ce réglage, le RPE moyen diminue à mesure que ε augmente. Pour régler ε, nous le faisons varier dans l'intervalle $]0, 1]$ avec un pas de 0.1. Comme on peut le voir sur la figure 4.3, le minimum est atteint lorsque ε est égal à 0.6, alors il devient constant. Nous avons choisi de fixer ε à 0.6 pour ne pas trop consommer de temps de traitement. Comme le paramètre est aussi utilisé par la méthode AIDCH, nous avons décidé de garder la même valeur pour cette méthode aussi.

Le **S**core de **D**estruction des méthodes est géré par le paramètre λ. Nous avons fait varier $\lambda \in [0, 20]$ avec un pas de 0,5. Lorsque la valeur $\lambda = 0$ le mécanisme de mise à jour du score est désactivé, et on n'obtient pas le RPE moyen minimum mais la valeur est proche de zéro. Les valeurs RPE diminuent jusqu'à $\lambda = 1.0$, puis elles sont constantes pour les valeurs λ dans $[1, 4]$. Ensuite, les valeurs RPE augmentent à mesure que la valeur λ augmente.

Le mécanisme de mise à jour des scores permet d'obtenir de meilleurs résultats en gérant la sélection des méthodes de destruction.

Nous avons choisi de fixer λ à =1.0, cela suffit à rendre le mécanisme efficace.

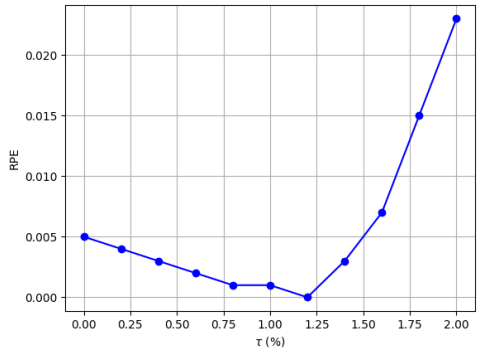

Fig 4.4: Calibrage de τ, impact sur le RPE moyen.

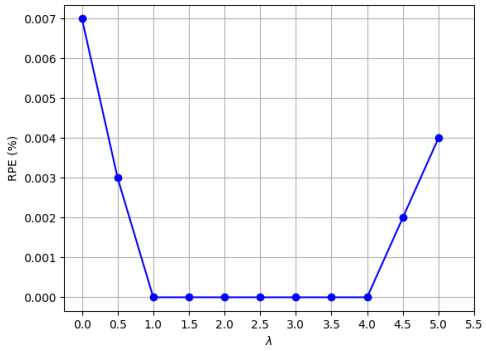

Fig 4.5: Calibrage de λ, impact sur le RPE moyen.

Une solution peut être remplacée par une solution de moindre qualité lorsque $(S_{cur} - S_{old})/S_{old} \leqslant \tau$. Dans la figure 4.4, nous montrons l'évolution du RPE moyen en faisant varier τ de 0 % à 2 % avec un pas de 0,2 %. Lorsque $\tau = 0\,\%$, le mécanisme d'acceptation d'enregistrement à enregistrement est désactivé. Comme on peut le voir, on obtient une bonne valeur RPE moyenne avec $\tau = 0\%$. Les valeurs RPE diminuent jusqu'à la valeur $\tau = 1.2\%$, ensuite elles augmentent lorsque τ augmente. Comme on pouvait s'y attendre, le mécanisme d'acceptation basé sur l'algorithme d'enregistrement à enregistrement joue son rôle. Nous avons choisi de fixer $\tau = 1.2\%$.

Les valeurs de calibrage finales sont $\phi = 0.15$, $\varepsilon = 0.6$, $\lambda = 1.0$ et $\tau = 0.012$. Elles ont été utilisées dans la suite pour nos expérimentations sur toutes les instances du benchmark, et elles ont été choisies pour obtenir un bon compromis entre la qualité de solution et le temps de traitement.

Évaluation des mécanismes adaptatifs de l'ALNS, évaluation des méthodes de destruction

Nous évaluons ici l'efficacité du mécanisme adaptatif BIA et l'efficacité du mécanisme de diversification pour la destruction.

Nous évaluons également l'efficacité des méthodes de destruction en désactivant chacune d'entre elles une par une. Pour ces expériences, nous utilisons toutes les instances du benchmark et nous enregistrons la meilleure solution trouvée sur 15 000 itérations. Nous avons effectué dix exécutions sur chaque instance du benchmark.

Le jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega)$ est utilisé pour calculer le BIDC. Le mécanisme de construction adaptative vise à guider la recherche en calculant le meilleur compromis entre les différentes valeurs de ces paramètres sur des itérations consécutives. Nous avons mené des expériences avec le mécanisme de construction adaptative et sans le mécanisme de construction adaptative. Dans ce dernier cas, les paramètres $(\alpha, \beta, \gamma, \theta, \omega)$ sont choisis aléatoirement dans $[0, 1]$. Dans la figure 4.6, nous montrons le RPE moyen par rapport aux itérations pour ces deux versions.

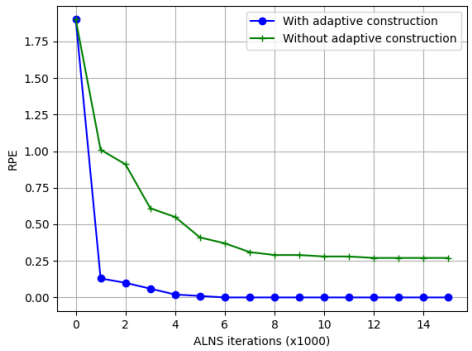

Fig 4.6: Impact de la construction adaptative avec le BIA.

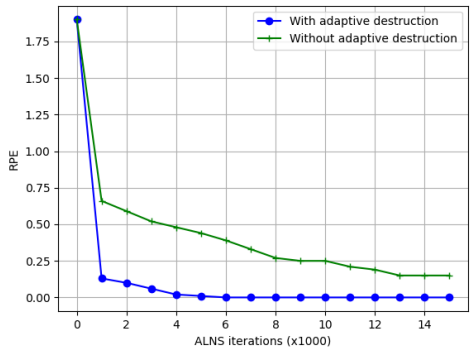

Fig 4.7: Impact de la destruction adaptative.

Comme on peut le voir sur la figure 4.6, les valeurs RPE sont détériorés lorsque les paramètres $(\alpha, \beta, \gamma, \theta, \omega)$ sont choisis aléatoirement. Le mécanisme de construction adaptative avec ϕ fixé à 0.15 accélère significativement la convergence vers de bonnes solutions.

Le mécanisme de diversification adaptative gère la valeur de $D_{max} \in \{3, \cdots, D_{limit}\}$ (voir Algorithme 2). Ce mécanisme permet d'explorer le voisinage de la nouvelle solution dès qu'une amélioration est trouvée et permet également d'explorer un voisinage plus éloigné dès lors que la recherche est piégée dans un optimum local. Nous avons désactivé le mécanisme de diversification adaptative en fixant le degré de diversification

D_{max} à 3. Dans la figure 4.7, nous montrons le RPE moyen par rapport aux itérations pour ces deux versions. Le mécanisme de diversification adaptative obtient toujours de meilleurs résultats comme en témoignent les valeurs moyennes du RPE. Le mécanisme de diversification adaptative permet de converger plus rapidement vers les bonnes solutions.

Dans le tableau 4.2, nous montrons les résultats pour les méthodes de destruction. À des fins de comparaison, nous utilisons l'erreur moyenne relative en pourcentage calculée comme $ARPE = 100 \cdot \frac{Z_{avg} - Z_{best}}{Z_{best}}$ où Z_{best} désigne le meilleur résultat que nous avons obtenu sur toutes les exécutions que nous avons effectuées pour une instance, et Z_{avg} représente la moyenne des résultats. Cela nous permet de discuter des résultats pour montrer si l'ALNS est stable au fil des exécutions. Dans le tableau 4.2 sous la rubrique "ALNS", nous montrons les résultats avec toutes les méthodes de destruction activées. Sous les rubriques "No RD", "No SD", "No RMD" et "No TCD", nous montrons les résultats en désactivant chacun d'eux un par un. Par souci de compacité, nous avons calculé les valeurs ARPE moyennes par groupe de 4 instances. Le temps de traitement moyen est en secondes. Les meilleurs résultats sont indiqués en caractères gras.

Tableau 4.2: Évaluation des méthodes de destruction.

	ALNS		No RD		No SD		No RMD		No TCD	
	ARPE	t(s)	ARPE	t(s)	ARPE	t(s)	ARPE	t(s)	ARPE	t(s)
I_00_to_03	**0**	**1,780**	**0**	1,801	0.66	2,315	0.06	1,883	0.03	1,871
I_04_to_07	**0.03**	**2,073**	0.05	2,101	0.97	2,713	0.09	2,191	0.07	2,186
I_08_to_11	**0.06**	**2,101**	0.09	2,133	1.31	2,819	0.16	2,206	0.12	2,193
I_12_to_15	**0.11**	**2,621**	0.15	2,653	1.89	3,373	0.24	2,771	0.21	2,753

L'ALNS sans RD (rubrique "No RD") obtient des résultats pour l'ARPE inférieurs à ceux de l'ALNS standard, et les délais de traitement sont plus longs. La Destruction Aléatoire RD permet d'éviter d'être coincé dans un optimum local en réalisant des perturbations. L'ALNS sans SD (rubrique "No SD") obtient des résultats pour le critère ARPE inférieurs à ceux de l'ALNS standard, et les temps de traitement sont plus importants. Nous remarquons que la qualité des solutions est détériorée d'une manière importante lors de l'absence de la méthode SD. Comme on peut le voir dans le tableau 4.2 en comparant les colonnes "t(s)", la SD accélère la convergence de l'algorithme ALNS et permet d'obtenir de meilleures solutions. L'ALNS sans RMD (rubrique "No RMD") et l'ALNS sans TCD (rubrique "No TCD") obtiennent des résultats pour l'ARPE inférieurs à ceux de l'ALNS standard mais proche, cependant les délais de traitement sont plus longs. Une légère infériorité en terme de qualité de solutions est remarquée lors de l'absence de la méthode RMD par rapport à TCD. Ceci est dû au fait que la libération de certaines affectations par rapport à une salle donnée permettent de donner plus de possibilités pour d'autres sections ce que permet à la méthode ALNS d'explorer d'autres zones de l'espace de recherche et accéder la convergence. Toutes les méthodes de destruction RD, SD, RMD et TCD sont bénéfiques pour explorer le voisinage d'une solution soit en obtenant de meilleurs résultats, soit en raccourcissant le temps de traitement.

Comparaison entre ILP, AIDCH et ALNS pour les instances jouets

Afin de pouvoir valider le modèle ILP et vérifier la qualité des solutions produites par les méthodes AIDCH et ALNS, nous avons généré quatre instances jouets de petite taille afin de pouvoir y arriver. Les instances jouets contiennent respectivement 5,10,15 et 20 sections. Nous utilisons le critère de gap relative (Relative Pourcentage gap, RPG) calculé comme $\text{RPG} = 100 \cdot \frac{Z_{min} - Z_{ILP}}{Z_{ILP}}$ où Z_{ILP} désigne la valeur atteinte par l'ILP pour une instance (si possible), et Z_{min} désigne le meilleur résultat obtenu parmi les exécutions effectuées. Cela nous permet de montrer à quel point les meilleures solutions calculées par les méthodes AIDCH et ALNS sont éloignées des valeurs optimales. Le temps maximal alloué à l'ILP est de 3600 secondes. Les meilleures valeurs sont rapportées en gras dans le tableau.

Tableau 4.3: Comparaison des méthodes AIDCH et ALNS avec l'ILP sur les instances jouets.

	ILP			AIDCH				ALNS			
Instance	Z_{min}	gap	t(s)	Z_{min}	RPG	ARPE	t(s)	Z_{min}	RPG	ARPE	t(s)
Toy_01	**0**	**0**	169	**0**	**0**	**0**	**101**	**0**	**0**	**0**	133
Toy_02	**40**	**0**	532	**40**	**0**	**0**	**138**	**40**	**0**	**0**	159
Toy_03	**120**	**0**	1,325	**120**	**0**	0.02	**211**	**120**	**0**	**0**	233
Toy_04	220	**0.1**	3,600	260	18.18	0.05	**251**	**200**	**-9.09**	**0**	288

Nous remarquons dans le tableau 4.3 que l'ILP arrive à trouver une solution optimale pour les trois première instances, cependant, il n'arrive pas à trouver l'optimalité pour l'instance Toy_04 avec un gap de 0.1. Nous remarquons que le temps d'exécution de l'ILP augmente avec l'augmentation de la taille de l'instance, et pour l'instance Toy_04 l'ILP n'arrive pas à trouver une solution optimale dans un temps de 3600 secondes. La méthode AIDCH arrive à trouver les mêmes solutions que l'ILP pour les trois premières instances dans des temps d'exécutions raisonnables. Pour l'instance Toy_04, la méthode AIDCH trouve une solution de plus mauvaise qualité que l'ILP avec un RPG égal à 18.18. La méthode ALNS arrive à trouver toutes les solutions optimales trouvées par l'ILP et trouve une meilleure solution que celle trouvée par l'ILP dans le temps imparti pour l'instance Toy_04 avec un RPG égal à -9.09. Ces tests nous ont permis de valider l'ILP et la convergence de nos méthodes AIDCH et ALNS vers les solutions optimales. Pour les 16 instances restantes du benchmark, l'ILP n'arrive pas à trouver de solution. Nous allons discuter dans ce qui suit les résultats des méthodes AIDCH et ALNS sur les 16 instances du benchmark.

Comparaison des résultats entre AIDCH et ALNS

Afin de pouvoir mesurer la stabilité des méthodes AIDCH et ALNS, nous exécutons dix fois les deux méthodes sur chaque instance. Nous utilisons le critères Z_{min} et ARPE définis précédemment comme critères d'évaluation.

Dans le tableau 4.4, nous montrons les résultats pour toutes les instances du benchmark. Les meilleurs résultats sont indiqués en caractères gras.

Pour chaque instance, sous les rubriques “AIDCH” et “ALNS”, nous affichons le meilleur résultat obtenu parmi les exécutions effectuées (la meilleure valeur Obj), l’ARPE et le temps de calcul en secondes (sous les colonnes “Z_{min}”, “ARPE” et “t(s)”, respectivement).

Tableau 4.4: Résultats des méthodes AIDCH et ALNS sur les instances du benchmark.

	AIDCH			ALNS		
	Z_{min}	ARPE	t(s)	Z_{min}	ARPE	t(s)
Instance_00	8,412	0,21	1,314	0	0	1,345
Instance_01	12,452	0,25	1,255	0	0,06	1,752
Instance_02	13,244	0,33	1,548	0	0,09	1,963
Instance_03	12,888	0,41	1,648	0	0,11	2,001
Instance_04	14,120	0,39	1,459	120	0,1	1,863
Instance_05	14,880	0,51	1,436	180	0,13	1,932
Instance_06	13,560	0,48	1,846	260	0,19	2,239
Instance_07	13,980	0,52	1,756	420	0,17	2,145
Instance_08	12,744	0,59	1,422	0	0,18	1,842
Instance_09	13,120	0,81	1,613	0	0,21	2,136
Instance_10	14,250	0,89	1,345	340	0,26	1,856
Instance_11	14,788	0,77	1,957	220	0,14	2,415
Instance_12	13,880	0,65	1,823	180	0,18	2,213
Instance_13	13,650	0,71	1,712	0	0,22	2,286
Instance_14	15,230	0,63	1,998	520	0,25	2,971
Instance_15	15,126	0,84	1,951	480	0,23	2,896
Avg	13,520	0.56	1,630	170	0.15	2,115

Dans le tableau 4.4, nous montrons que les deux méthodes AIDCH et ALNS sont arrivées à trouver des solutions réalisables dans le temps imparti. La qualité de la solution des deux méthodes ne peut pas être évaluée à l’aide du critère RPG défini dans la section précédente car l’ILP ne peut pas obtenir de solutions dans le délai imparti. La méthode AIDCH arrive à trouver des solutions dans des temps de calcul légèrement meilleurs que ceux de l’ALNS avec une moyenne de 1630 secondes pour AIDCH et une moyenne de 2115. Cependant, la qualité des solutions est largement meilleure pour la méthode ALNS avec une moyenne de Z_{min} égale à 170 pour ALNS et 13520 pour AIDCH. Cela est dû au fait que la méthode AIDCH se trouve coincée dans des optima locaux et n’arrive plus à explorer d’autre zones de l’espace de recherche. L’intégration des composants adaptatifs, du critère d’acceptation de solution dégradante enregistrement à enregistrement, l’utilisation des listes de priorités lors de la construction et l’utilisation de plusieurs voisinages de destruction ont permis à la méthode ALNS d’avoir de meilleures performances que la méthode AIDCH. La méthode ALNS reste stable comme en témoignent les valeurs d’ARPE avec une moyenne de 0.15. La stabilité de la méthode AIDCH reste moins bonne avec des valeurs d’ARPE d’une moyenne de 0.56.

Les résultats que nous avons obtenus montrent que l'ALNS a obtenu des résultats optimaux sur les instances jouets. Pour les plus grandes instances, l'ALNS a obtenu des résultats de bien meilleure qualité que AIDCH dans des temps de traitement raisonnables.

4.9 CONCLUSION ET PERSPECTIVES

Dans ce chapitre, nous avons abordé le problème réel de planification de cours de l'UTC (UTC-UCTTP) pour lequel les activités d'enseignement doivent être programmées dans des salles et des timeslots, et doivent être affectées à des enseignants. L'enjeu est d'obtenir des emplois du temps de bonne qualité en respectant les différentes contraintes du problème.

Nous avons présenté un modèle ILP, une heuristique AIDCH et métaheuristique ALNS pour aborder le problème UTC-UCTTP. Les approches proposées ont été testées à l'aide d'un benchmark de vingt instances que nous avons générés à partir de données réelles.

L'approche ILP a obtenu des résultats optimaux pour les instances jouets, mais a rencontré des difficultés pour obtenir des solutions réalisables pour les instances plus grandes dans un délai de traitement raisonnable.

L'heuristique AIDCH a pu trouver des solutions réalisables dans des temps de traitement raisonnables mais la qualité des solutions avait une bonne marge d'amélioration. Pour obtenir de meilleurs résultats, nous avons ensuite proposé une approche de solution métaheuristique ALNS qui utilise l'algorithme de meilleure insertion comme méthode de construction. Des expériences ont été menées pour ajuster les paramètres afin d'obtenir un bon compromis entre la qualité de la solution et le temps de traitement. Des expériences supplémentaires ont été menées pour évaluer l'efficacité des principaux composants de l'approche ALNS, et tous ces composants sont nécessaires pour obtenir de bonnes solutions dans un bon temps de traitement. L'ALNS a obtenu tous les résultats optimaux obtenus en utilisant l'ILP sur les instances jouets. Pour les plus grosses instances, les résultats sont de meilleure qualité par rapport à ceux obtenus par l'heuristique AIDCH.

Une future direction de recherche serait de considérer l'ILP dans la phase de construction de solutions en proposant une méthode matheuristique et l'exploiter.

Chapitre 5

Problèmes de planification de personnel

Les problèmes de planification de personnel (Personnel Scheduling Problem, PSP) ou d'équipes [Aggarwal, 1982; Ernst et al., 2004b; Tien et al., 1982] sont des problèmes de planification qui impliquent l'allocation de ressources humaines à des plages horaires dans un horizon de planification donné tout en respectant les contraintes réglementaires de compétence, de priorité, de durée, de capacité, de disjonction et de distribution (espacement, regroupement).

Un problème de planification de personnel est dit cyclique s'il est périodique, c'est-à-dire qu'une solution (emplois du temps) doit être trouvée d'abord pour une partie de l'horizon, puis cette solution est à répéter sur tout l'horizon. Un problème est dit acyclique si une solution doit être trouvée sur tout un horizon sans répétition.

les revues et les articles de synthèse de la littérature couvrent une large variété de problèmes, de nombreuses références sur la planification du personnel peuvent être trouvées dans [Ernst et al., 2004a; Bergh et al., 2013]. Les travaux sont classés par type de problème, domaine d'application et méthode de résolution.

La planification du personnel est connue sous le nom de planification d'équipage dans le domaine des systèmes de transport tels que les compagnies aériennes, les chemins de fer, les transports en commun et les bus [Barnhart et al., 2003]. Pour ces problèmes, il existe deux caractéristiques communes. La première est que des contraintes temporelles et spatiales sont impliquées. Chaque tâche est caractérisée par son heure et son emplacement de début, ainsi que par son heure et son emplacement de fin. La seconde est que toutes les tâches à effectuer par les employés sont déterminées à partir d'un emplois du temps de tâches donné. Les tâches élémentaires sont déterminées sur la base des différentes tâches que l'entreprise doit assurer dans un horizon de planification. Une tâche peut être d'assurer une étape de vol dans les compagnies aériennes ou d'assurer un voyage entre deux segments dans un train.

Pour la planification du personnel dans les centres d'appels, la nature exacte et le nombre de tâches à effectuer ne sont pas connus avant le processus de planification. Ces demandes sont estimées selon un modèle de besoins en main-d'œuvre existant

pour l'ensemble de l'horizon de planification. Les besoins en main-d'œuvre dans les applications de centre d'appels peuvent varier d'un jour à l'autre et d'une semaine à l'autre. Les heures de début et la durée des quarts de travail doivent varier afin d'obtenir de bons plannings à faible coût pour couvrir les besoins en main-d'œuvre.

La planification du personnel dans les centres d'appels n'implique pas de caractéristiques géographiques (ou spatiales). C'est la principale différence entre ce problème de planification du personnel et la planification des équipages présenté dans la section précédente.

La planification des infirmières est un des problèmes principaux de la planification dans les systèmes de santé [Burke et al., 2004]. Chaque hôpital doit régulièrement et à plusieurs reprises attribuer des listes de quarts de travail à ses infirmières. Une bonne planification du personnel infirmier a un impact sur la qualité des soins de santé, le recrutement d'infirmières et l'élaboration d'un budget infirmier.

Les emplois du temps peuvent être générés manuellement par les infirmières en charge de l'organisation du service ou par un agent spécialisé pour chaque unité hospitalière. Cependant, l'élaboration d'un planning pour les infirmières a toujours été difficile à cause des différentes contraintes à respecter. Le personnel infirmier des hôpitaux travaille 24 heures sur 24, sept jours sur sept. Dans de nombreux hôpitaux, certaines infirmières sont autorisées à postuler à des postes prédéfinis dans tout l'hôpital à cause de leurs compétences ou leur ancienneté, tandis que d'autres infirmières sont affectées ou planifiées dans un service. Habituellement, les gestionnaires passent beaucoup de temps à élaborer des plannings, surtout lorsqu'il y a beaucoup de demandes de personnel. En raison de ce travail manuel fastidieux et chronophage, le problème de planification des infirmières (Nurse Rostering Problem, NRP) a suscité beaucoup d'intérêt de la part des chercheurs.

Dans les services médicaux d'un hôpital, le personnel est classé en différents niveaux de compétences relativement à des tâches. Par exemple, pour le problème de planification des infirmières, chaque infirmière possède des compétences spécifiques et elle ne peut être affectée à un quart de travail qui nécessite une compétence donnée que si elle la possède. Les emplois du temps doivent fournir des infirmières qualifiées pour couvrir la demande résultant du nombre de patients dans les services. L'emploi du temps doit respecter les règles de travail tout en distinguant le personnel permanent et occasionnel. Cet emploi du temps doit garantir que les quarts de travail de nuit et de fin de semaine sont répartis équitablement, en tenant compte des congés et des jours de repos, et en tenant compte des préférences des employés. Dans la plupart des cas, les problèmes de planification d'infirmières sont des problèmes de planification fortement contraints.

Le NRP consiste à produire une affectation de tâches périodique (hebdomadaire, bimensuelle ou mensuelle) pour les infirmières. L'ensemble des contraintes concerne les règlements, les politiques de gestion du personnel, les préférences et autres exigences

qui peuvent être spécifiques à l'hôpital [Burke et al., 2004]. L'ensemble des contraintes peut varier d'un hôpital à l'autre ainsi que les objectifs de planification. Il existe une variété de modèles de NRP et une large gamme d'approches de résolution ont été développées pour ces modèles.

Les Compétitions Internationales de Planification des Infirmières (International Nurse Rostering Competition, INRC) visent à susciter l'intérêt dans le domaine de la planification des NRP. Elles proposent usuellement aux candidats des modèles qui intègrent des contraintes rencontrées dans la vie réelle. Le but des compétitions est de trouver de nouvelles approches pour résoudre le NRP tout en réduisant l'écart qui existe entre les problèmes théoriques et les problèmes réels. Deux compétitions ont été lancés pour le NRP, la première en 2010 (INRC1) et la seconde en 2015 (INRC2).

Le problème considéré pour l'INRC1 [Haspeslagh et al., 2014] est l'affectation des infirmières à des quarts de travail dans un horizon de planification fixe, avec de nombreux types de contraintes dures et souples. Trois types d'instances ont été proposés. Les administrateurs ont également fourni un protocole pour évaluer la puissance de calcul des machines. Ceci permet d'évaluer les performances des solveurs équitablement afin que tous les participants aient un temps de calcul selon les performances de leur machine. Des solutions optimales ainsi que de nouvelles meilleures solutions ont été trouvées après la compétition.

Dans la deuxième compétition de planification des infirmières INRC2 [Ceschia et al., 2015], les administrateurs ont proposé un ensemble plus restreint de contraintes. Cependant, la formulation du problème est plus complexe car certaines informations, appelées historiques, se propagent entre deux semaines consécutives. Les informations qui proviennent des semaines précédentes doivent être prises en compte. L'historique comprend des données frontalières, telles que le dernier quart de travail de chaque infirmière, et des compteurs de données cumulatives, tels que le nombre total de quarts de nuit travaillés. La valeur des compteurs doit être comparée aux seuils globaux, mais uniquement à la fin de la période de planification. Tous les compteurs sont vérifiés au cours de la dernière semaine par rapport à leurs limites. Dans la littérature sur le problème NRP de l'INRC2 plusieurs approches hybrides et de nouvelles méthodes pour résoudre le NRP ont été proposées.

5.1 LES TRAVAUX DE LA LITTÉRATURE

En fonction de la charge de travail dans les institutions, la demande en personnel varie, ce qui implique un changement dans les emplois du temps du personnel. Les responsables doivent produire de nouveaux emplois du temps chaque fois que les contraintes évoluent ce qui rend la tâche difficile et fastidieuse. Depuis plus de 50 ans, les chercheurs de la communauté scientifique essayent de trouver des méthodes automatiques de génération d'emplois du temps en s'inspirant de plusieurs disciplines

comme la recherche opérationnelle et l'intelligence artificielle.

Dans un large éventail de situations telles que les soins de santé, les services de protection, les chemins de fer ou les entrepôts, les employés ou les équipages doivent travailler dans différents quarts de travail pour répondre aux demandes. De nombreuses versions différentes des problèmes de planification du personnel ont été décrites dans la littérature. Dans la synthèse de la littérature, des centaines d'articles traitant de différents types de problèmes sont classés ([Afshar-Nadjafi, 2021; Heil et al., 2020; De Bruecker et al., 2015; Qin et al., 2015; Bergh et al., 2013; Brucker et al., 2011; Ernst et al., 2004b; Ernst et al., 2004a; Burke et al., 2004]).

La littérature distingue habituellement les problèmes cycliques et les problèmes acycliques et on considère ici les problèmes acycliques. Certains d'entre eux ont reçu le plus d'attention en raison des ensembles de données accessibles au public qui peuvent être utilisés pour comparer les méthodes de résolution ([Curtois, 2014; Smet et al., 2014; Fages et al., 2014; Lapègue et al., 2013; Krishnamoorthy et al., 2012]).

Par exemple, le problème de planification de minimisation des tâches du personnel (Shift Minimization Personnel Task Scheduling Problem, SMPTSP) consiste à affecter des tâches à des employés polyvalents [Hojati, 2018a]. Les tâches doivent être affectées à des équipes déjà prédéfinies dans le but de minimiser le nombre total d'employés affectés. L'objectif d'équité et la planification des pauses peuvent également être considérés comme pour le problème de conception de quarts et de planification des tâches du personnel avec objectif d'équité (Shift Design and Personnel Task Scheduling Problem with Equity objective, SDPTSP-E) défini dans [Lapègue et al., 2013]. Des problèmes spécifiques sont également étudiés car ils correspondent à des besoins particuliers.

Le tableau 5.1 présente un aperçu synthétique de travaux récents sur les problèmes acycliques.

Les premières approches utilisées pour résoudre les problèmes d'emplois du temps étaient les heuristiques constructives. Les heuristiques constructives sont des méthodes qui s'inspirent de la construction manuelle des emplois du temps. Ensuite, les chercheurs ont commencé à utiliser les méthodes issues de la recherche opérationnelle telles que la programmation linéaire et la programmation dynamique. Comme la taille des instances était trop importante, et que ces méthodes étaient très gourmandes en temps d'exécution, cela a conduit à l'utilisation des méthodes approchées comme les méta-heuristiques et les méthodes issues de l'intelligence artificielle qui permettent d'obtenir des solutions réalisables, de bonne qualité dans un temps raisonnable.

Nous allons présenter dans ce qui suit, sans être exhaustifs, les méthodes les plus utilisées pour résoudre le problème de planification de personnel.

Tableau 5.1: Travaux récents sur les problèmes de planification acycliques de personnel et d'équipes.

No.	Literature	Problem characteristics							Optimization method
		Skill	Shift	Task	P/C	H	Data	Obj	
1	[Guerriero et al., 2022]	✓	✓		P	1w	RD	MO/SO	MO EA: MIP
2	[Zucchi et al., 2021]	✓	✓		P	2w	RD	SO	EA: MIP
3	[Chandrasekharan et al., 2021]	✓	✓	✓	P	1d	LB	SO	MA
4	[Kletzander et al., 2020]	✓	✓	✓	P	2-52w	LB	SO	MH: SA
5	[Porto et al., 2019]	✓	✓		P	1w	RD	SO	EA: MIP
6	[Tadumadze et al., 2019]			✓	P	1d	RD	SO	EA: MIP H
7	[Hoffmann et al., 2019]		✓	✓	C	1d	RD	SO	EA: ILP
8	[Demirović et al., 2019]		✓		P	1-52w	LB	SO	MO MaxSat
9	[Hojati, 2018b]	✓	✓	✓	P	1d	LB	SO	H:G
10	[Pour et al., 2018]		✓	✓	C	10-40d	RD	SO	EA: CP MIP
our			✓		C	135d	RD	SO	EA: ILP MA MH: ALNS

Notes :

- Skill, Shift, Task, P/C(Person, Crew), H(Horizon, **w**eek/**d**ay).
- Données: RD(Real Data), LB(Literature Benchmark).
- Objectif: SO(Single-Objective), MO(Multi-Objective).
- Méthode de résolution: EA(Exact Algorithm), H(Heuristics), MA(Matheuristic), MH(Metaheuristics), MO(Model).
- G(Greedy), CP(Constraint Programming), ILP(Integer Linear Programming), MIP(Mixed Integer Program), ALNS(Adaptive Large Neighborhood Search).

5.1.1 *Méthodes exactes*

Les méthodes exactes sont des méthodes de recherche opérationnelle qui effectuent une recherche exhaustive sur l'espace des solutions dans le but de trouver une solution optimale au problème. L'inconvénient majeur des méthodes exactes est qu'elles requièrent des temps d'exécution éventuellement grands selon la taille des instances.

Parmi les méthodes de résolution reposant sur la description mathématique du problème, on trouve la programmation linéaire (Linear Programming, LP) et la programmation linéaire en nombres entiers (Integer Linear Programming, ILP). Ces méthodes consistent à définir des variables de décision, une fonction objectif à maximiser ou à minimiser en fonction des variables de décision. Ces dernières servent à définir l'objectif à atteindre après résolution du problème. Ces méthodes définissent aussi un certain nombre de contraintes sur les variables de décisions formulées comme des équations mathématiques afin de limiter l'espace de recherche. On parle d'ILP quand les variables de décision ne prennent que des valeurs entières, ce qui est le cas généralement pour les problèmes de planification de personnel.

Un algorithme Branch-and-Cut est proposé dans [Santos et al., 2016]. L'approche proposée utilise un mécanisme d'amélioration à double borne qui s'inspire de la méthode de séparation en cliques. Les auteurs ont proposé un modèle mathématique pour résoudre le problème présenté dans INRC1. Les auteurs ont aussi proposé un graphe de conflit implicite généré par les inégalités valides. Ils ont aussi proposé un algorithme de séparation ajusté qui consiste en deux modules : un module pour

séparer toutes les cliques violées dans le sous-graphe de conflit induit par les variables fractionnaires et un module qui prolonge les cliques générées en considérant le graphe de conflits d'origine. Les auteurs proposent des heuristiques afin d'améliorer les bornes des sous-problèmes primaux.

[Legrain et al., 2019] a pris la deuxième place à INRC2 avec son solveur et a proposé une formulation en ILP du problème. Les auteurs ont utilisé une approche de type Branch-and-Price pour résoudre le problème. Dans le modèle proposé chaque colonne de l'ILP correspond à une rotation, c'est-à-dire une séquence de jours de travail consécutifs pour une infirmière, et non à un horaire complet. Les auteurs ont pu obtenir de bons résultats en intégrant l'algorithme Branch-and-Price avec un mécanisme de recherche locale dont les solutions initiales sont obtenues avec une heuristique adéquate.

[Guerriero et al., 2022] ont proposé des modèles pour répondre aux problèmes de planification du personnel en tenant compte des nouveaux critères impliqués par la situation de pandémie de Covid-19. Le modèle initial vise à optimiser les jours de travail sur site et à distance, en considérant plusieurs contraintes comme une limitation des capacités du bureau. Les auteurs ont ensuite proposé des modèles ILP dérivés pour étudier des scénarios. Pour optimiser les critères associés à un scénario, le modèle ILP implémenté utilise une somme de termes pondérés. Les résultats de calcul sont obtenus en utilisant des données réelles d'un département de l'Université de Calabre (Italie).

[Zucchi et al., 2021] se sont également intéréssés aux problèmes de planification du personnel pendant la pandémie de Covid-19 pour une entreprise qui fournit des produits pharmaceutiques aux hôpitaux. L'objectif est de minimiser la somme des écarts par rapport au nombre contractuel d'heures de travail de chaque travailleur. Les auteurs ont proposé une formulation ILP avec des contraintes visant à limiter un risque de contagion estimé à partir d'un réseau de relations entre salariés.

[Porto et al., 2019] ont évalué les avantages potentiels de l'intégration de la flexibilité du travail dans la planification du personnel dans le contexte d'un magasin de vente en détail. L'objectif est de minimiser les niveaux de sur-effectif et de sous-effectif en tenant compte d'une main-d'œuvre polyvalente qui a un contrat flexible. Les auteurs ont proposé un modèle ILP et étudié des scénarios pour évaluer plusieurs stratégies de gestion des ressources humaines.

[Hoffmann et al., 2019] ont étudié un problème de planification des équipes ferroviaires avec des taux de présence. L'objectif est de trouver un horaire de déplacement à coût minimum satisfaisant aux conditions d'exploitation et tenant compte des contraintes de régulation du travail. Les auteurs ont proposé un modèle ILP et des inégalités valides. Les instances de petite taille sont résolues à optimalité, les inégalités valides se sont avérées efficaces pour accélérer le processus de résolution et améliorer les bornes.

[Chandrasekharan et al., 2021] ont proposé une matheuristique constructive (Constructive Mathematical Heuristic, CMH) pour traiter le SMPTSP. La matheuristique

traites des sous-problèmes en utilisant la programmation linéaire en nombres entiers. Le sous-problèmes sont obtenus en décomposant le problème initial et sont résolus jusqu'à l'optimalité. Une décomposition orienté employés et une décomposition orienté temps sont proposées. Une matheuristique constructive en est dérivée. La CMH basée sur le temps s'est avéré efficace, des solutions de bonne qualité sur tous les ensembles de données sont obtenues, et dans certains cas, de nouvelles solutions optimales sont atteintes.

La programmation par satisfaction de contraintes était l'une des premières méthodes utilisées pour la résolution des problèmes de planification de personnel. Dans les situations où il y a un grand nombre de contraintes à satisfaire, il peut être plus approprié de modéliser le NRP par exemple comme un problème de satisfaction de contrainte (Constraint Satisfaction Problem, CSP). Formellement, un CSP peut être défini comme un triplet (V, D, C) où V est un ensemble de n variables v_i, D est un ensemble de n domaines D_i, tels que chaque D_i est l'ensemble fini de valeurs possibles pour chaque v_i et C est un ensemble fini de contraintes dont chacune agit sur un sous-ensemble de variables en V restreignant les combinaisons possibles de valeurs que ces variables peuvent prendre.

[Heus, 1996] ont proposé une approche basée sur les techniques de programmation par contraintes pour résoudre le NRP. La formulation du problème comme problème de satisfaction de contraintes (CSP) favorise une gestion transparente des contraintes et des affectations journalières des infirmières, ce qui facilite la prise en compte des préférences individuelles et qui permet un management plus participatif des plannings.

[Okada et al., 1988] ont utilisé la programmation logique pour planifier les affectations de quarts favorables au jour le jour, en se référant à l'information accumulée dans la base de données de l'hôpital. En Prolog, diverses exigences peuvent être exprimées avec une relative facilité, et le processus de la méthode manuelle peut être incorporé dans le système d'une manière naturelle. Les résultats obtenus ont montré la validité de l'approche.

5.1.2 *Heuristiques*

Les méthodes exactes sont en capacité de trouver la solution optimale à un problème donné. Néanmoins, l'application de ces méthodes devient rapidement inenvisageable car le temps d'exécution croit considérablement avec l'augmentation de la taille du problème. Il est alors préférable de trouver des solutions, même approchées, en un temps raisonnable. Ceci devient un enjeu majeur dans le cas des systèmes de planification délicats où les contraintes du problème changent régulièrement, et où l'arrêt du système pour chercher la solution optimale pendant un temps important est impossible. Les méthodes heuristiques, répondent à ce besoin en permettant de trouver

des solutions de bonne qualité en des temps relativement courts.

Une heuristique s'appuie généralement sur les caractéristiques du problème et exploite au mieux la structure du problème dans le but de trouver une solution raisonnable en un temps réduit. Une heuristique doit respecter les contraintes du problème considéré. Les performances d'une heuristique sont liées à la qualité de la solution produite ainsi qu'au temps de calcul nécessaire pour l'obtenir. Selon la stratégie de recherche de solution, on distingue deux types d'heuristiques : constructives et de voisinage. Dans ce qui suit, nous présentons les méthodes heuristiques qui ont été appliquées pour résoudre les problèmes de planification de personnel.

5.1.2.1 *Heuristiques constructives*

Historiquement, la génération des plannings du personnel était faite par les responsables de chaque service. De nombreuses approches heuristiques consistent à automatiser ces résolutions manuelles. Il s'agit des méthodes constructives, elle ont été largement étudiées et documentées dans la littérature.

Les méthodes constructives partent d'une solution initiale vide S_0, et insèrent à chaque étape k une composante x_k dans la solution partielle courante $S_{k-1} = (x_0, ..., x_{k-1})$ tout en respectant les contraintes du problème. L'objectif est d'aboutir enfin à une solution admissible de la forme $S = (x_0, ..., x_n)$.

Les deux avantages de ces méthodes sont leur facilité de mise en oeuvre et leur rapidité d'exécution. Par contre, la faible qualité des solutions trouvées est en général leur grand défaut. En effet, ces méthodes ne tiennent pas compte de l'effet du choix de la composante à insérer sur les choix futur et sur la qualité de la solution finale.

[Smith et al., 1977] ont proposé un algorithme de planification qui simule la méthode manuelle en trois phases. La première phase consiste en la production d'un résumé du statut de personnel hebdomadaire de chaque unité de soins infirmiers. Après les vacances et les jours fériés, sont considérées. Cela vise à encourager la réaffectation du personnel aux unités qui connaissent des lacunes en matière de personnel avant d'essayer de produire des plannings détaillés. L'algorithme effectue durant la deuxième phase une génération de plannings provisoires indiquant l'existence de manque de personnel chaque jour, et identifier les cas où certaines contraintes ont été violées. La troisième phase du processus consiste en l'ajustement manuel des plannings obtenus.

[Bell et al., 1986] ont développé un système d'aide à la décision. Une heuristique construit une trame de base qui répond aux contraintes de couverture et qui satisfait les niveaux de compétences requis. Une fois la trame principale définie, les plannings des prochaines semaines sont dérivés avec des modifications pour répondre aux exigences. Le planning calculé est contrôlé par le décideur qui peut effectuer des modifications si

nécessaire.

[Kostreva et al., 1991] ont proposé un algorithme en deux phases pour résoudre le problème de planification des infirmières. Tous les plannings réalisables sont calculés dans un premier temps. Les plannings respectent les exigences minimales en matière de personnel et chaque planning individuel remplit toutes les contraintes strictes de travail. Dans la deuxième phase, on calcule le meilleur score d'aversion possible. Un score d'aversion est une métrique basée sur les préférences des infirmières qui permet d'évaluer les plannings. Toutes les catégories de compétences sont programmées indépendamment, ce qui revient à une décomposition en des problèmes partiels.

[Hojati, 2018b] ont proposé une méthode de solution heuristique gloutonne pour le SMPTSP. Un problème réduit est résolu de manière itérative en sélectionnant la meilleure affectation possible de tâches à un travailleur. À chaque itération, le travailleur avec la valeur d'objectif maximale est choisi. Pour les très grandes instances, la méthode heuristique gloutonne proposée fonctionne très bien par rapport aux autres approches de solution qui nécessitent un solveur ILP commercial.

Dans l'étude réalisée par [Burke et al., 2004], d'autres heuristiques constructives commencent par remplir l'emploi du temps case par case de façon à respecter toutes les contraintes strictes. Quand ce n'est plus possible, l'heuristique permute les plannings des infirmières et évalue l'impact de la permutation sur la qualité du planning. Ce type d'heuristique s'appelle des heuristiques de permutation (Shuffling Heuristic, SH). D'autres heuristiques calculent toutes les permutations possibles pour tout le personnel, puis elles les trient par ordre décroissant de la fonction objectif. Ce processus est répété autant de fois que possible.

5.1.2.2 *Heuristiques de voisinage*

Une méthode typique de voisinage est un processus itératif fondé sur deux éléments essentiels : un voisinage et une procédure exploitant le voisinage. Une méthode de voisinage débute avec une solution initiale (générée aléatoirement ou par l'application d'une heuristique par construction), et réalise ensuite un processus itératif qui consiste à remplacer la configuration courante (solution courante) par l'une de ses voisines en tenant compte de la fonction coût. Ce processus s'arrête et retourne la meilleure configuration trouvée quand des conditions d'arrêts sont satisfaites (un nombre d'itérations ou un objectif à réaliser).

La méthode VND (Variable Neighborhood Descent, VND) [Hansen et al., 2019] consiste de choisir une solution initiale x, de trouver une direction de descente la plus raide à partir de x, dans un voisinage $N(x)$, et de se déplacer vers le minimum de $f(x)$ dans $N(x)$ le long cette direction. S'il n'y a pas de sens de descente, l'heuristique s'arrête, sinon elle réitère.

La méthode VNS (Variable Neighborhood Search, VNS) [Hansen et al., 2019] est une heuristique de recherche locale qui exploite systématiquement l'idée de changement de voisinage, à la fois en descente vers les minima locaux et aussi de pouvoir s'en échapper afin d'explorer d'autres zones de l'espace de recherche. La méthode VNS exploite les constats suivants :

- Un minimum local par rapport à une structure de voisinage n'est pas forcément pour un autre ;
- Un minimum global est un minimum local par rapport à tous les structures de voisinage ;
- Pour de nombreux problèmes, les minima locaux par rapport à un ou plusieurs voisinages sont relativement proches les uns des autres.

La stratégie de génération des solutions voisines est l'élément clé des méthodes de voisinage, c'est ce facteur qui contrôle la performance de l'algorithme. Pour les problèmes d'emplois du temps, les voisins d'une solution peuvent être par exemple les emplois du temps avec deux éléments permutés.

[Burke et al., 2003] ont proposé un algorithme de recherche locale VNS avec plusieurs méthodes de génération de voisinage. Une solution qui est générée avec une heuristique constructive [Burke et al., 2001]. Ensuite l'algorithme choisit aléatoirement une stratégie de voisinage et commence la recherche locale. Un mécanisme de basculement entre stratégies de voisinages a été mis en place pour permettre à l'algorithme d'échapper aux optima locaux et permettre une certaine diversification.

À l'issue de la première compétition internationale de planification des infirmières, [Valouxis et al., 2012], vainqueurs de la compétition, ont proposé un algorithme hybride qui résout le problème en deux phases. Un programme linéaire est hybridé avec une recherche locale afin d'améliorer la qualité de la solution produite par le programme linéaire. Dans un premier temps, le solveur essai de produire des plannings réalisables qui respectent les contraintes stricte. Ces plannings sont construits en deux phases. La première phase affecte des infirmières aux jours de travail, la seconde phase effectue ensuite l'affectation des infirmières aux quarts de travail. Après chaque itération (Les deux affectations définies précédemment constituent), des heuristiques de recherche locale sont appliquées afin d'améliorer la qualité de la solution produite et booster les performances de l'algorithme. Les voisinages utilisés sont des voisinages de permutation. Cette hybridation a permis d'avoir les meilleurs résultats sur les instances proposées.

5.1.3 *Métaheuristiques*

C'est la dépendance des heuristiques aux problèmes spécifiques qui a incité les chercheurs à s'intéresser à des méthodes génériques qui peuvent être généralisées à de

multiples problèmes. Usuellement, on distingue deux familles de métaheuristiques : les métaheuristiques manipulant une seule solution et les métaheuristiques évolutives manipulant une population de solutions.

5.1.3.1 *Les métaheuristiques basées sur une seule solution*

Les métaheuristiques manipulant une seule solution sont des algorithmes qui font évoluer une seule solution sur l'espace de recherche à chaque itération. L'idée est d'explorer l'espace de recherche en faisant évoluer une solution courante. La notion de voisinage est alors primordiale. Parmi les méthodes les plus connues on peut citer la recherche tabou (Tabu Search, TS) [Glover et al., 1998] et le recuit simulé (Simulated Annealing, SA) [Kirkpatrick et al., 1983].

[Burke et al., 1998] ont présenté une approche TS hybride qui a été développée pour un système commerciale de planification des infirmières. Dans cette approche, un planning initial réalisable est obtenu en utilisant trois stratégies possibles : le planning actuel dans le système, le planning précédent qui a été utilisé dans la période de planification précédente ou utiliser l'initialisation aléatoire. Ensuite, la TS est lancée. Cette recherche est hybridée avec des heuristiques de diversification afin de mieux explorer l'espace de recherche. Avant de fournir le planning final, des modifications sont apportées au planning par les responsables afin de rendre le planning plus adéquats à leurs préférences.

[Hadwan et al., 2010] ont proposé un algorithme en trois phases pour le problème de planification des infirmières. Durant la première phase, des patterns sont construits et c'est selon ces patterns que les solutions initiales seront construites dans la deuxième phase grâce à une heuristique de construction. Ensuite, le SA est déclenché dans la troisième phase afin de trouver une solution de meilleure qualité à partir de la solution initiale.

[Kletzander et al., 2020] ont proposé un framework pour résoudre le problème général de planification des employés en utilisant le recuit simulé. Différents problèmes issus de la littérature qui couvrent différents types de demandes et de contraintes sont étudiés. Toutes les violations de contraintes strictes sont pénalisées en utilisant un générateur de pondération de contraintes strictes afin d'ajuster les pondérations pour chaque ensemble de données. Les auteurs ont implémenté un ensemble de mouvements dans le cadre général. L'approche a obtenu de bons résultats par rapport aux algorithmes dédiés.

5.1.3.2 *Les métaheuristiques basées sur une population de solutions*

Les métaheuristiques manipulant une population de solutions s'inspirent des principes d'évolution naturelle. L'une des méthodes évolutives qui a donné de bon résultats sur les problèmes d'optimisation est les algorithmes génétiques (Genetic Algorithm, GA).

[Aickelin et al., 2004] ont développé un GA pour résoudre le NRP. Au lieu de travailler directement avec des populations de solutions potentielles et de manipuler les contraintes en utilisant des fonctions de pénalité ou de réparation, les auteurs proposent une approche indirecte dans laquelle la tâche d'équilibrage optimisation et satisfaction de contraintes est partagée entre une heuristique gloutonne et le GA. Les individus sont représentés par des permutations des infirmières disponibles et l'heuristique est utilisée pour établir les horaires en affectant les infirmières à leurs quarts de travail dans l'ordre donné.

[Burke et al., 2001] ont proposé un algorithme mémétique (Memetic Algorithm, MA) qui incorpore la recherche tabou dans un GA, en utilisant une approche VND pour chaque individu.

[Rajeswari et al., 2017], les auteurs ont utilisé un modèle de programmation mathématique multi-objectif et ont proposé une méthode d'adaptation de l'approche d'optimisation multi-objective d'essaims d'abeilles. Cette approche a été utilisée avec succès pour des problèmes de planification. L'approche proposée est une intégration d'une recherche locale déterministe, un environnement de système de particules multi-agents et un processus de prise de décision basé sur les essaims d'abeilles. Le solveur a donné de très bons résultats sur les instances de la première compétition internationale de planification des infirmières INRC1.

[Awadallah et al., 2017] ont proposé un algorithme de recherche harmonique (Harmonic Search Algorithm, HSA) hybridé avec une heuristique de recherche locale afin de renforcer la capacité d'exploitation de l'espace de recherche du HSA. Aussi, l'opérateur de considération de la mémoire harmonique est modifié en remplaçant la sélection aléatoire avec la meilleure particule rencontrée afin d'améliorer la convergence de l'algorithme. L'algorithme a été testé sur des instances de l'INRC1 et a obtenu de très bons résultats, pour plusieurs instances des solutions de meilleure qualité ont été trouvée.

5.1.4 *Hyper-heuristiques*

Les métaheuristiques et leurs hybridations avec les heuristiques ont été utilisées avec succès pour résoudre des problèmes réels de planification de personnel. La réutilisabilité des métaheuristiques n'est pas souvent possible car elles sont conçues pour traiter un problème spécifique. Les performances des métaheuristiques dépendent fortement de leur paramétrage qui nécessite l'intervention d'experts.

Les hyper-heuristiques (Hyper-Heuristic, HH) ont été proposées pour pallier ces inconvénients. L'idée est de développer des méthodes applicables sur une plus large classe d'instances de problèmes, tout en gardant une bonne performance en sortie et

une facilité d'implémention.

Telle que définie dans [Burke et al., 2013], une hyper-heuristique est une méthode qui, en s'aidant de d'extraction de connaissance et d'apprentissage, choisit, combine ou génère des heuristiques faciles à implémenter dans le but de résoudre efficacement une instance ou une classe d'instances d'un problème. Les heuristiques manipulées sont appelés "heuristiques de bas niveau". Le processus qui manipule les heuristiques de bas niveau est appelé "stratégie de haut niveau".

[Asta et al., 2016] ont proposé une hyper-heuristique pour résoudre le NRP. L'approche proposée s'effectue en quatre phases. Durant les trois premières phases, des techniques de machine-learning sont appliqué afin de paramétrer l'algorithme de la quatrième phase qui est une hyper-heuristique dédiée. L'heuristique de haut niveau utilisée est une heuristique de choix aléatoire. Les auteurs ont utilisé une variété d'heuristiques de bas niveau : des heuristiques de mutation, des heuristiques de croisement ainsi que des heuristiques de recherche locale. De nouvelles solutions optimales pour plusieurs benchmarks ont été trouvées.

[Kheiri et al., 2016] ont proposé une hyper-heuristique de sélection pour résoudre le NRP de la compétition INRC2. La méthode proposée est une hyper-heuristique dont l'heuristique de haut niveau choisit une séquence d'heuristiques de bas niveau au lieu d'une seule à chaque itération globale. Le choix de la séquence optimale est guidé avec un mécanisme d'apprentissage durant la recherche à l'aide des chaines de markov cachées (Hidden Makov Model, HMM). Les auteurs ont trouvés de bons résultats et ils ont pu obtenir la troisième place de la compétition INRC2.

[Václavík et al., 2016] ont remarqué que la majorité des algorithmes proposés dans la littérature pour le NRP en particulier, et pour tous les problèmes de planification de manière général prennent beaucoup de temps dans l'évaluation des solutions. Dans certains cas, l'évaluation des solutions consomme jusqu'à 80% du temps d'exécution, alors que la majorité des solutions évaluées durant la phase de recherche sont de qualité médiocre. Les auteurs ont proposé un mécanisme de classification basé sur des techniques de reconnaissance des formes qui permet de décider rapidement et selon la structure de la solution si est est potentiellement intéressante à évaluer. Le classificateur est construit à l'aide d'un réseau de neurones. Le classificateur proposé peut être vu comme un filtre qui élimine au préalable les solutions de mauvaises qualités sans les évaluer. Les solutions restantes sont évaluées normalement à l'aide de la fonction coût. Le test de la méthode proposée a donné une amélioration du temps d'exécution de plusieurs algorithmes proposés dans la littérature.

5.2 CONCLUSION

Nous avons présenté dans ce chapitre différents problèmes de planification de personnel qui existent dans la littérature ainsi que des méthodes qui ont été appliquées pour les résoudre.

Le tableau 5.1 montre que de nombreuses méthodes de résolution différentes ont été appliquées, allant des méthodes exactes aux matheuristiques, heuristiques, métaheuristiques et hybridation d'approches de résolution.

Les méthodes exactes peuvent trouver une solution optimale pour les instances de petite et moyenne taille, mais rencontrent souvent des difficultés à obtenir une solution pour les instances plus grandes dans un temps de traitement raisonnable. Les matheuristiques peuvent aider à résoudre ce problème selon les caractéristiques du problème. Dans certains cas, des approches heuristiques ou métaheuristiques sont nécessaires pour obtenir des solutions de bonne qualité.

Il existe une grande variété de PSP, allant de problèmes types basés sur des problèmes réels, pour lesquels des ensembles de données (benchmarks) sont accessibles au public, à des problèmes spécifiques qui sont adressés pour répondre à un besoin. Le problème que nous traitons correspond à ce dernier cas.

Nous nous intéressons dans le chapitre suivant à l'étude du problème de planifications d'équipes de pompiers (FireFighters Timetabling Problem, FFTP). Dans notre problème, une équipe de pompiers peut être vue comme un employé composé de personnes possédant de multiples compétences hautement spécialisées. Ces personnes constituent une unité extrêmement soudée, formée pour avoir des automatismes de groupe pour être efficace dans des situations extrêmes. Le nombre d'équipiers est fixé et n'est pas à minimiser comme dans d'autres problèmes de gestion de personnel. Au contraire, les objectifs sont d'augmenter la capacité opérationnelle tout en considérant un nombre limité d'équipes, de maintenir l'équité entre les équipes et de rendre possible le regroupement des jours de repos lorsque cela est possible.

Chapitre 6

Problème de planification d'équipes de pompiers

Sommaire

6.1 Introduction

Dans cette étude, nous considérons le problème de planification d'équipes de pompiers (FFTP) de l'institution INFOCA en Andalousie (Espagne) dont la mission est de lutter contre les feux de forêt. Nous partons d'un certain nombre d'équipes de pompiers, de types de quarts de travail et de demandes quotidiennes pour ces quarts de travail sur un horizon de planification fixe. L'objectif est de construire un planning complet pour chaque équipe de pompiers pour la période à haut risque de feux de forêt. Cette période est une période annuelle où des feux de forêt se produisent fréquemment. La période considérée pour l'institution INFOCA va du 1 er juin jusqu'au 15 octobre. Cette période a été définie après plusieurs années d'expérience afin d'augmenter la capacité opérationnelle des équipes de pompiers qui sont fortement sollicités pour faire face aux feux de forêt. Malheureusement, l'horizon de planification d'aujourd'hui est sur le point d'être élargi en raison du réchauffement climatique, motivant davantage le besoin d'une solution de planification efficace. Les membres d'une équipe ont l'habitude de faire face à des feux de forêt dans des conditions extrêmes. La confiance mutuelle est la clé de voûte d'une équipe de pompiers, chaque membre d'une équipe a pour responsabilité première la vie des autres membres. Les membres d'une équipe de pompiers sont

stables sur la période de planification, c'est-à-dire que les équipes ne changent pas. Les pompiers d'une même équipe s'entraînent ensemble pour renforcer sa cohésion.

Les équipes de pompiers peuvent être affectées à six différents types de quarts de travail (par exemple travail en hélicoptère, travail en quart de nuit ou travail à la demande). Il existe différents créneaux horaires et durées. De plus, il faut répartir équitablement les jours de repos et les jours de compensation supplémentaires, accordés lorsqu'un certain nombre d'heures ont été travaillées. La capacité opérationnelle globale doit être assurée tout en respectant strictement les exigences minimales et les contraintes réglementaires imposées par l'établissement. Elles portent sur les successions interdites de quart de travail, la charge de travail maximale sur la période de planification, les jours de compensation à accorder et le nombre maximal de jours de travail consécutifs. Les contraintes de bonnes pratiques doivent également être prises en compte pour rendre le planning adéquat pour les équipes. Elles portent sur le regroupement des affectations pour un même type de quarts pour des jours consécutifs (ou de quarts commençant à la même heure) ou l'attribution de jours de compensation après les jours de repos afin de pouvoir donner aux équipes des périodes de repos élargies car ils n'ont pas droit de prendre de congés durant cette période. Par souci d'équité, la charge de travail devrait également être équilibrée sur la période de planification, tant en nombre d'affectations sur les types de quarts de travail qu'en termes d'écarts de temps de travail entre les équipes puisque tous les types de quarts de travail n'ont pas la même durée.

Les exigences minimales garantissent une capacité globale minimale pendant la période à haut risque de feux de forêt. Cependant, s'il existe une marge d'amélioration tout en respectant l'ensemble des contraintes strictes exprimées, l'établissement souhaite équilibrer les affectations supplémentaires sur les types de quarts. Pour une journée, il serait préférable d'équilibrer la capacité opérationnelle sur des types de quart différents plutôt que d'affecter toutes les équipes mobilisables au-delà des exigences minimales sur un seul type de quart.

A notre connaissance, le problème de planification d'équipes de pompiers pour les institutions dont la mission est de lutter contre les feux de forêt n'a pas encore été étudiée dans la littérature. En raison du changement climatique, la période à haut risque de feux de forêt s'élargie et les feux de forêt augmentent en nombre et en intensité. La capacité opérationnelle des équipes de pompiers est susceptible de devenir un problème. Une façon d'y parvenir est de construire de meilleurs plannings des équipes en tenant compte des contraintes réglementaires du travail mais aussi des contraintes de bonnes pratiques pour assurer l'équité. Les apports de ce travail se résument comme suit :

- Premièrement, nous proposons un modèle de programmation linéaire en nombres entiers (Integer Linear Program, ILP) pour le problème de planification d'équipes de pompiers que nous abordons. L'ILP est conçu à des fins de modélisation et dans le but d'obtenir des solutions de bonne qualité pour certaines instances qui seront utilisées comme solutions de référence. Cependant, les solveurs ILP peuvent avoir des difficultés à trouver des solutions réalisables dans des délais de traitement raisonnables ;

- Deuxièmement, nous utilisons l'ILP comme base pour proposer une matheuristique (Integer Linear Programming Heuristic, ILPH). Nous proposons trois voisinages pour faire travailler le solveur sur des sous-problèmes dans le but d'obtenir des solutions réalisables pour toutes les instances dans un temps de traitement raisonnable. Nous proposons d'explorer l'espace de recherche en utilisant une approche de descente par voisinage variable (Variable Neighborhood Descent, VND) basée sur les trois voisinages ;

- Troisièmement, nous proposons une métaheuristique ALNS (Adaptive Large Neighbourhood Search, ALNS) pour étudier une deuxième approche de résolution. Nous menons des expériences préliminaires pour régler les paramètres des composants de l'ALNS. Nous utilisons aussi une version de la matheuristique comme méthode de construction. Nous étudions la contribution des composants de l'ALNS ;

- Enfin, nous montrons que l'approche de résolution ALNS obtient toutes les solutions optimales qui peuvent être atteintes par l'ILP. De meilleures solutions que celles de la matheuristique sont obtenues dans un temps de traitement plus court. L'approche ALNS peut constituer une bonne base pour améliorer la capacité opérationnelle des établissements de pompiers pendant la période à haut risque de feux de forêt.

Le chapitre est organisé comme suit. La section 6.2 présente les contraintes et les paramètres du problème de planification d'équipes de pompiers (FFTP) que nous adressons. La section 6.3 présente l'ILP que nous utilisons pour fournir un modèle formel, et la section 6.4 décrit la matheuristique ILPH que nous proposons à partir du modèle ILP. La méthode ALNS que nous proposons pour résoudre le problème FFTP est décrite dans la section 6.5 et les composants sont détaillés. Les expériences de tests sont rapportées et commentées dans la section 6.6. L'efficacité des composants de l'ALNS est présentée. La conclusion et les perspectives futures se trouvent dans la section 6.7.

6.2 DÉFINITION DU PROBLÈME FFTP

Dans cette section, nous présentons un aperçu global du problème de planification des équipes de pompiers du monde réel que nous abordons. Nous donnons l'ensemble des quarts de travail quotidiens à considérer, nous introduisons les contraintes dures à respecter et les contraintes souples permettant d'évaluer la qualité d'une solution. La notation utilisée par l'institution INFOCA pour les types de quarts de travail est la suivante :

(T12) de 8h à 16h à la caserne de pompiers, quart de jour régulier ;

(T16) de 15h à 22h à la caserne de pompiers, quart de jour régulier ;

(H) de 8h à 16h à la caserne de pompiers, quart de jour régulier, affecté à un hélicoptère ;

(N) de 22h à 8h à la caserne des pompiers, quart de nuit régulier ;

(G7) de 7h à 15h à la caserne des pompiers, stand-by pour faire face instantanément à toute demande urgente ;

(G24) de garde 24h/24h, l'équipe reste chez elle mais peut être mobilisée pour faire face à toute situation d'urgence ;

(A3) de 8h à 18h à la caserne des pompiers (ou ailleurs) à des fins de formation ;

(R) jour de repos ;

(C) jour de compensation supplémentaire accordé lorsqu'un certain nombre d'heures ont été travaillées.

Pour le problème d'emploi du temps d'équipes de pompiers considéré, les contraintes dures liées à la réglementation du travail et à la réglementation locale de l'institution INFOCA sont les suivantes :

(H1) un quart de travail par jour : une équipe de pompiers ne peut être affectée qu'à un seul quart de travail par jour ;

(H2) exigences minimales : chaque quart de travail quotidien a une demande minimale d'équipes de pompiers ;

(H3) successions de quarts interdites : certaines affectations de quarts sur des journées consécutives sont interdites ;

(H4) charge de travail maximale : sur l'horizon de planification, une charge de travail maximale pour chaque équipe ne doit pas être dépassée ;

(H5) compensation : les jours de compensation sont accordés en fonction des heures travaillées, ils doivent être utilisés ;

(H6) maximum de jours de travail consécutifs : il y a un nombre maximum de jours de travail consécutifs pour chaque équipe de pompiers.

Certaines affectations consécutives sont interdites pour une équipe (H3), par exemple une affectation de nuit N se termine à 8h et ne peut être suivie d'une affectation hélicoptère H qui commence à 8h, cette affectation consécutive interdite est notée (N, H).

Les contraintes souples sont des contraintes de bonne pratique qu'il convient de satisfaire au mieux. La violation de toute contrainte souple induit une pénalité. Une somme pondérée des pénalités mesure la qualité de la solution produite. Pour le problème de planification d'équipes de pompiers étudié, les contraintes souples sont les suivantes :

(S1) groupement de quarts de travail : les affectations d'une équipe à un même quart de travail doivent être regroupées. Chaque changement d'affectation entre deux jours consécutifs est pénalisé ;

(S2) même heure de début : les heures de début doivent être les mêmes indépendemment des quarts de travail sur des journées de travail consécutives. Chaque changement d'heure de début de quart de travail entre deux jours consécutifs est pénalisé ;

(S3) affectation des jours de compensation : les affectations des jours de compensation doivent avoir lieu juste après les jours de repos, l'objectif est de permettre aux pompiers d'avoir des périodes de repos élargies pendant la période de planification. Toute attribution de compensation non juste après les jours de repos est pénalisée ;

(S4) équité sur la période de planification : par souci d'équité, la charge de travail devrait être équilibrée entre les équipes sur la période de planification. Le déséquilibre de la charge de travail entre les équipes doit être minimisé ;

(S5) préférences : chaque affectation d'équipe à un poste non souhaité est pénalisée ;

(S6) équilibre des quarts de travail supplémentaires quotidiens : l'affectation d'équipes supplémentaires aux différents types de quarts de travail doit être équilibrée chaque jour. Le déséquilibre des affectations supplémentaires à différents quarts de travail doit être minimisé chaque jour.

Sous réserve que l'exigence minimale (H2) soit respectée, il s'agit au-delà de (S6) d'assurer un équilibre entre les affectations. Si on peut affecter trois équipes supplémentaires dans une journée, il est préférable d'affecter une équipe à trois types de quarts différents plutôt que d'affecter les trois équipes à un même type de quart.

6.3 MODÈLE ILP POUR LE PROBLÈME FFTP

Dans cette section, nous présentons le modèle ILP pour minimiser les critères que nous avons détaillés dans la section 6.2.

Le modèle ILP nous permet d'étudier le problème auquel nous sommes confronté. Nous pouvons éventuellement obtenir des solutions optimales pour les plus petites instances à des fins de comparaison avec l'approche ALNS que nous proposons.

Ce modèle est une amélioration d'un premier modèle présenté dans [Ouberkouk et al., 2021], moins de variables sont utilisées et nous obtenons de meilleurs résultats sur les plus petites instances. Notre ILP proposé présente une meilleure formulation pour les contraintes 6.11, 6.12 et 6.13. Pour l'ILP présenté en [Ouberkouk et al., 2021], le nombre de chaque variable α, β et γ est égal à $n_c \cdot n_s \cdot n_s \cdot n_d$ alors que dans notre ILP, le nombre de chacune d'elles est égal à $n_c \cdot n_d$.

Nous présentons les données et les paramètres avant les variables de décision, nous présentons ensuite le modèle.

Les données et paramètres sont les suivants :

$\mathcal{D}$ ensemble de jours de la période de planification, un jour $d \in [1, \cdots, n_d]$, taille n_d ;

$\mathcal{D}^-$ ensemble de jours de la période de planification, à l'exception du dernier jour n_d ;

$\mathcal{S}$ ensemble de types de quarts, un quart $s \in \{T12, T16, H, N, G7, G24, A3, R, C\}$, taille n_s ;

$\mathcal{S}_w$ ensemble de types de quarts de travail, un quart $s \in \{T12, T16, H, N, G7, G24, A3\}$, taille n_w ;

$\mathcal{C}$ ensemble d'équipes de pompiers, taille n_c ;

F ensemble de couples d'affectations consécutives interdites, par ex. $(N, H) \in F$;

r_s demande minimale quotidienne pour un quart de travail $s \in \mathcal{S}_w$;

l_s durée du quart de travail s (en heures) ;

L charge de travail maximale pour tout équipe au cours de la période de planification ;

t_s heure de début du quart de travail s ;

w_{oc} pondération de la capacité opérationnelle ;

w_{sg} pondération de violation de regroupement de quarts de travail (S1) ;

w_{sst} pondération de changement de la même heure de début (S2) ;

w_{ca} pondération de violation des affectations de compensation (S3) ;

w_p pondération de la violation des préférences (S5) ;

p_{csd} si l'équipe c ne préfère pas travailler le quart s le jour d nous fixons $p_{csd} = w_p$, zéro sinon (S5) ;

MAX_d nombre maximal de jours de travail consécutifs pour une équipe (H6) ;

WHC nombre d'heures travaillées donnant un jour de compensation.

Les variables booléennes primaires sont X_{csd}, si l'équipe c travaille le quart s le jour d alors $X_{csd} = 1$, zéro sinon. Les variables booléennes secondaires utilisées dans le modèle sont les suivantes :

$\alpha_{cd} = 1$ si l'équipe c travaille sur le quart s le jour d et travaille sur un quart différent s' le jour $d + 1$, zéro sinon ;

$\beta_{cd} = 1$ si l'équipe c travaille sur le quart s le jour d et travaille sur un quart différent s' le jour $d + 1$ avec $t_s \neq t_{s'}$, zéro sinon ;

$\gamma_{cd} = 1$ si l'équipe c travaille sur le quart s le jour d avec $s \neq R$ et est affectée au quart $s' = C$ le jour $d + 1$, zéro sinon.

Si une violation de changement de quart se produit $\alpha_{cd} = 1$ (S1, groupement de quarts de travail), si une violation de modification du temps de travail se produit $\beta_{cd} = 1$ (S2, même heure de début) et si une violation d'affectation de compensation se produit $\gamma_{cd} = 1$ (S3, affectation des jours de compensation).

Les variables entières utilisées dans le modèle sont les suivantes :

λ_d différence quotidienne entre le nombre maximum d'équipes affectables (n_c) et celles affectées ;

δ_c nombre total de quarts travaillés pour l'équipe c au cours de la période de planification ;

θ_c temps de travail total de l'équipe c sur la période de planification ;

ρ_{cd} nombre d'heures travaillées de l'équipe c du premier jour au jour d ;

$\phi_{cc'}$ différence de nombre d'affectations de quarts entre les équipes c et c' (S4) ;

$\varphi_{cc'}$ différence de temps de travail entre les équipes c et c' (S4) ;

$\psi_{ss'}$ déséquilibre des affectations entre les quarts s et s' (S6).

L'objectif est de maximiser la capacité opérationnelle sur la période de planification tout en minimisant les violations des contraintes souples. Nous proposons l'ILP suivant pour résoudre ce problème :

Min

$$w_{oc} \cdot \sum_{d \in \mathcal{D}} \lambda_d \tag{6.1a}$$

$$+ \sum_{c \in \mathcal{C}} \sum_{d \in \mathcal{D}} (w_{sg} \cdot \alpha_{cd} + w_{sst} \cdot \beta_{cd} + w_{ca} \cdot \gamma_{cd}) \tag{6.1b}$$

$$+ \sum_{c \in \mathcal{C}} \sum_{c' \in \mathcal{C}} (\phi_{cc'} + \varphi_{cc'}) \tag{6.1c}$$

$$+ \sum_{c \in \mathcal{C}} \sum_{s \in \mathcal{S}_w} \sum_{d \in \mathcal{D}} p_{csd} \cdot X_{csd} \tag{6.1d}$$

$$+ \sum_{s \in \mathcal{S}_w} \sum_{s' \in \mathcal{S}_w} \psi_{ss'} \tag{6.1e}$$

sous les contraintes :

$$\sum_{s \in \mathcal{S}} X_{csd} = 1 \quad \forall c \in \mathcal{C},\ \forall d \in \mathcal{D} \tag{6.2}$$

$$\sum_{c \in \mathcal{C}} X_{csd} \geqslant r_s \quad \forall d \in \mathcal{D},\ \forall s \in \mathcal{S}_w \tag{6.3}$$

$$X_{csd} + X_{cs'(d+1)} \leqslant 1 \quad \forall (s, s') \in F,\ \forall c \in \mathcal{C},\ \forall d \in \mathcal{D}^- \tag{6.4}$$

$$\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}} l_s \cdot X_{csd} \leqslant L \quad \forall c \in \mathcal{C} \tag{6.5}$$

$$\sum_{s\in\mathcal{S}_w}\sum_{d'\in\mathcal{D},d'\leqslant d} l_s \cdot X_{csd} = \rho_{cd} \quad \forall c \in \mathcal{C},\ \forall d \in \mathcal{D} \tag{6.6}$$

$$WHC \cdot \sum_{d'\in\mathcal{D},d'\leqslant d} X_{csd} \leqslant \rho_{cd} \quad s = C,\ \forall c \in \mathcal{C},\ \forall d \in \mathcal{D} \tag{6.7}$$

$$\sum_{d\in\mathcal{D}} X_{csd} = \left\lfloor \frac{\rho_{c(l_d)}}{WHC} \right\rfloor + 1 \quad s = C, \forall c \in \mathcal{C} \tag{6.8}$$

$$\sum_{s\in\mathcal{S}_w}\sum_{d'\leqslant 1+MAX_d,\, d+d'\leqslant n_d} X_{csd} \leqslant MAX_d \quad \forall c \in \mathcal{C},\ \forall d \in \mathcal{D} \tag{6.9}$$

$$\sum_{c\in\mathcal{C}}\sum_{s\in\mathcal{S}_w} X_{csd} = n_c - \lambda_d \quad \forall d \in \mathcal{D} \tag{6.10}$$

$$X_{csd} + X_{cs'd+1} \leqslant 1 + \alpha_{cd} \quad \forall c \in \mathcal{C}, \forall d \in \mathcal{D}^-, \forall s, s' \in \mathcal{S}_w, s \neq s' \tag{6.11}$$

$$X_{csd} + X_{cs'd+1} \leqslant 1 + \beta_{cd} \quad \forall c \in \mathcal{C}, \forall d \in \mathcal{D}^-, \forall s, s' \in \mathcal{S}_w, s \neq s' \text{ with } t_s \neq t_{s'} \tag{6.12}$$

$$X_{csd} + X_{cs'd+1} \leqslant 1 + \gamma_{cd} \quad \forall c \in \mathcal{C},\ \forall d \in \mathcal{D}^-\ \forall s \in \mathcal{S}_w, s' = C \tag{6.13}$$

$$\sum_{s\in\mathcal{S}_w}\sum_{d\in\mathcal{D}} X_{csd} = \delta_c \quad \forall c \in \mathcal{C} \tag{6.14}$$

$$\sum_{s\in\mathcal{S}_w}\sum_{d\in\mathcal{D}} l_s \cdot X_{csd} = \theta_c \quad \forall c \in \mathcal{C} \tag{6.15}$$

$$\delta_c - \delta_{c'} \leqslant \phi_{cc'} \quad \forall c, c' \in \mathcal{C},\ c \neq c' \tag{6.16}$$

$$\theta_c - \theta_{c'} \leqslant \varphi_{cc'} \quad \forall c, c' \in \mathcal{C},\ c \neq c' \tag{6.17}$$

$$\left(\sum_{c\in\mathcal{C}} X_{csd} - r_s\right) - \left(\sum_{c\in\mathcal{C}} X_{cs'd} - r_{s'}\right) \leqslant \psi_{ss'} \quad \forall d \in \mathcal{D},\ \forall s, s' \in \mathcal{S}_w \tag{6.18}$$

$$X_{csd},\ \alpha'_{cd},\ \beta'_{cd},\ \gamma'_{cd} \in \{0, 1\} \tag{6.19}$$

$$\delta_c, \theta_c, \rho_{cd}, \phi_{cc'}, \varphi_{cc'}, \psi_{ss'} \in \mathbb{N} \tag{6.20}$$

Les cinq termes de la fonction objectif visent à maximiser la capacité opérationnelle tout en minimisant les violations des contraintes souples. Le premier terme (6.1a) vise à maximiser la capacité opérationnelle. La somme pondérée (6.1b) évalue les violations de contraintes souples : regroupement de quarts (S1), même heure de début (S2) et affectations de jours de compensation (S3). La contrainte souple d'équité sur la période de planification (S4) porte sur le nombre de différences d'affectation des quarts et sur les différences de temps de travail entre les équipes. Ils sont considérés dans le terme (6.1c). Les préférences des pompiers (S5) sont considérées dans le terme (6.1d). L'équilibre des quarts de travail supplémentaires quotidiens (S6) est considéré à l'aide du terme (6.1e).

Les contraintes dures **un quart de travail par jour** (H1) sont imposées par les Contraintes (6.2). Les contraintes dures **exigences minimales** (H2) sont imposées par les Contraintes (6.3). Les contraintes dures **successions de quarts interdites** (H3) sont imposées par les Contraintes (6.4). Les contraintes dures **charge de travail maximale** (H4) sont imposées par les Contraintes (6.5). Les contraintes dures **compensation** (H5) sont imposées par les Contraintes (6.6)-(6.8). Le nombre d'heures travaillées de l'équipe c depuis le premier jour de la période de planification jusqu'au jour d est évalué par les Contraintes (6.6). Un jour de compensation est accordé lorsque WHC heures travaillées sont effectuées, le nombre de jours de compensation accordés est imposé par les Contraintes (6.7). Tous les jours de compensation doivent être attribués sur l'horizon de planification, ceci est appliqué par les Contraintes (6.8). Les contraintes dures **maximum de jours de travail consécutifs** (H6) sont imposées par les Contraintes (6.9). Une équipe est affectée à au plus MAX_d quarts de travail consécutifs (les jours de repos et de compensation ne sont pas pris en compte).

Les écarts quotidiens entre le nombre maximum d'équipes affectables (n_c) et celles affectées sont à minimiser pour optimiser la capacité opérationnelle globale, les valeurs λ_d sont évaluées par les Contraintes (6.10).

Considérons une équipe c, et deux jours d et $d+1$. Si l'équipe est affecté à deux quarts différents ($s \neq s'$) une violation **groupement de quarts de travail** (S1) se produit, nous fixons α_{cd} à un (Contraintes (6.11)). Si l'équipe est affecté à deux quarts différents ($s \neq s'$) et que les heures de début de ces quarts sont différents ($t_s \neq t_{s'}$), une violation se produit pour la contrainte **même heure de début** (S2), on fixe β_{cd} à un (Contraintes (6.12)). Si l'équipe est affecté à un quart de travail le jour d, et si cette équipe est affecté à un jour de compensation ($s' = C$) le jour $d+1$ une violation **affectation de compensation** (S3) se produit, nous fixons γ_{cd} à un (Contraintes (6.13)). Chaque affectation de jour de compensation se fera juste après une journée de repos (Contraintes de bonnes pratiques imposées par l'établissement).

Considérons une équipe c. Le nombre total de quarts travaillés sur la période de planification δ_c est évalué par les contraintes (6.14). Le temps de travail total sur la période de planification θ_c est évalué par les contraintes (6.15). On obtient le nombre de différences d'affectation de quarts $\phi_{cc'}$ par les contraintes (6.16). Étant donné que $\phi_{cc'}$ est un entier naturel, une différence négative implique que $\phi_{cc'}$ sera égal à zéro, donc pour tout couple d'équipes, seules les différences positives sont comptées. Le même raisonnement s'applique pour les contraintes (6.17) qui évaluent la différence de temps de travail $\varphi_{cc'}$. Ces variables $\phi_{cc'}$ et $\varphi_{cc'}$ sont utilisées pour l'évaluation des violations

de contraintes souples **équité sur la période de planification** (S_4). Rappelons que les violations de contraintes souples **préférences** (S_5) sont évaluées par les contraintes (6.1d). Considérons un jour d et deux équipes s et s'. Les contraintes (6.18) visent à satisfaire la contrainte **équilibre des quarts de travail supplémentaires** (S6).

Dans ce qui suit, on note $Obj(S)$ la fonction qui calcule la qualité d'une solution S telle que présentée dans les équations (6.1a)-(6.1e).

6.4 MATHEURISTIQUE ILPH POUR LE PROBLÈME FFTP

Les solveurs ILP peuvent rencontrer des difficultés pour exécuter le modèle ILP proposé à mesure que la taille des instances augmente, et des solutions réalisables peuvent être difficiles à obtenir dans des délais de traitement raisonnables. L'utilisation de solveurs ILP dans un contexte heuristique pour produire de bonnes solutions a été étudiée pour certains problèmes de planification de personnel comme le problème de planification des infirmières NRP (Santos et al., 2016), la planification des équipes de maintenance (Pour et al., 2018) et la maintenance des avions (De Bruecker et al., 2018). L'idée générale est de faire travailler les solveurs sur des sous-problèmes afin d'effectuer des recherches locales dans le but d'obtenir de bonnes solutions réalisables dans un temps de traitement plus court.

La matheuristique ILPH que nous proposons est en deux phases successives : une phase de construction pour créer un sous-problème et une phase de recherche locale qui utilise l'ILP que nous avons proposé dans la section 6.3. Étant donné une solution actuelle réalisable S_{cur}, un sous-problème est obtenu en fixant un sous-ensemble de variables liées aux allocations des équipes de pompiers (fixation dure), puis le sous-problème est résolu de manière optimale à moins qu'un délai donné ne soit atteint.

Un sous-problème permet d'explorer un certain voisinage d'une solution et nous proposons trois voisinages : Fix Day, Fix Shift et Fix Crews.

Les voisinages que nous proposons créent des sous-problèmes en fixant un ensemble donné de variables d'une solution courante S_{cur}. Ensuite, la matheuristique ILPH, est utilisée. L'idée générale est d'explorer l'espace de recherche en utilisant une approche de descente de voisinage de variable (VND) basée sur ces trois voisinages.

Voisinage Fix Day

Les sous-problèmes sont générés en fixant toutes les allocations des équipes de pompiers sur $|\mathcal{D}| - Size_w$ jours dans la période de planification où $Size_w$ est la taille de la fenêtre glissante. Ainsi, l'ILP peut être utilisé pour optimiser les allocations sur $Size_w$ jours qui est un paramètre du voisinage.

A la première itération ($i = 0$), un sous-problème est créé à partir d'une solution S_{cur} en fixant les allocations de tous les équipes sauf ceux des jours de $day_A = 1$ à $day_B = Size_w$. Le sous-problème est ensuite résolu à l'aide de l'ILP et la solution S_{cur} est mise à jour si la solution est améliorée. Le glissement de la fenêtre $Size_w$ est géré en mettant à jour le day_A de début et le day_B de fin de la fenêtre comme suit :

$$day_A = 1 + i.Slide_w \qquad day_B = Size_w + i.Slide_w$$

où i est le numéro de l'itération courante et $Slide_w$ est le deuxième paramètre qui définit de combien de jours la fenêtre glisse entre deux sous-problèmes consécutifs. Lorsque $dayA$ ou day_B est supérieur à n_d, la valeur est fixée à l_d. L'idée est de faire glisser la fenêtre et de résoudre à l'optimalité chaque sous-problème tour à tour, si possible. La valeur du paramètre $Slide_w$ permet de déterminer $n_d/Slide_w$ le nombre de sous-problèmes à résoudre c'est-à-dire aussi le nombre d'itérations. Étant donné une itération i, nous calculons une solution S_i, on met à jour S_{cur} si cette solution est meilleure. Ainsi la solution retournée S_{cur} est le résultat d'améliorations successives, s'il y en a.

Plus la valeur de $Slide_w$ est petite, plus le nombre de sous-problèmes explorés dans le voisinage est grand. La taille de chaque sous-problème est déterminée par $Size_w$. De petites valeurs peuvent créer des sous-problèmes qui ne contiennent pas de meilleure solution. Des valeurs élevées peuvent créer des sous-problèmes pour lesquels le solveur peut avoir des difficultés à atteindre une solution optimale dans un temps de traitement court. A l'intérieur de la fenêtre, où toutes les allocations des équipes sont libres, tous les termes de la fonction objectif sont également considérés et aucun n'est privilégié.

Voisinage Fix Shift

Les sous-problèmes sont créés en fixant toutes les allocations des équipes qui concernent tous les types de quarts de travail sauf un. Les allocations des équipes qui concernent un type de quart sont libres. Cela tend à améliorer, par exemple, le groupement des quarts de travail (S1) et à équilibrer les quarts de travail supplémentaires (S6). Nous trions d'abord les types de quarts par leurs valeurs de pénalité décroissantes dans la solution courante. Étant donné une liste triée, à la première itération, seules les affectations des équipes relatives au premier type de quart sont libres, et ainsi de suite.

Il y a n_s itérations puisque le nombre de sous-problèmes différents générés dans ce voisinage est égal au nombre de types de quarts.

Voisinage Fixe Crew

Les sous-problèmes sont créés en fixant toutes les allocations d'équipes qui concernent tous les équipes sauf une. Les allocations aux quarts qui concernent une équipe sont libres. Cela tend par exemple à améliorer les affectations de jours de compensation (S3) (Voir terme (6.1b)). Nous trions d'abord les équipes par leur valeur de pénalité décroissante dans la solution actuelle. Étant donné une liste triée, à la première itération, seules les allocations des équipes qui concernent la première équipe sont libres, et ainsi de suite.

Il y a n_c itérations puisque le nombre de sous-problèmes différents générés dans ce voisinage est égal au nombre d'équipes.

Matheuristique ILPH

L'algorithme 4 montre la matheuristique ILPH. Le voisinage Fix Day ne privilégie aucun terme de la fonction objectif tandis que le voisinage Fix Shift est orienté type de quart et le voisinage Fix Crew est orienté équipe. Ainsi, nous décidons de chercher d'abord une meilleure solution en utilisant le premier voisinage et si la solution est améliorée on explore ensuite successivement les deux autres.

Algorithm 4: ILPH for FFTP

Input : S_{cur} a current solution, T_{ILPH} time limit
Output : S_{best} best solution found
Variables : $Size_w$ size of the sliding window,
$Slide_w$ by how many days the window slides

$S_{best} := S_{cur}$ /* S_{cur} stored as best solution */
$Slide_w := 2$
while ($Slide_w \leqslant n_d/2$) *and* (T_{ILPH} *not reached*) **do**
  $Size_w := 2.Slide_w$
  $S_{cur} := Explore\_Fixday\_Neighborhood(S_{cur}, Size_w, Slide_w)$
  if $Obj(S_{cur}) < Obj(S_{best})$ **then**
    $S_{best} := Explore\_Fixshift\_Neighborhood(S_{cur})$
    $S_{best} := Explore\_Fixcrews\_Neighborhood(S_{best})$
  $Slide_w := Slide_w + 1$
return S_{best}

Nous explorons d'abord les petites fenêtres, puis nous explorons les plus grandes. Notre objectif est d'améliorer progressivement la solution afin d'obtenir des évaluations de solutions qui aideront à élaguer la recherche au fur et à mesure que les fenêtres augmentent.

Cela rend également possible le chevauchement des fenêtres sur les itérations de l'ILPH. Par souci de simplicité, nous choisissons de lier les deux paramètres comme suit : $Size_w = 2.Slide_w$ et nous fixons la valeur initiale de $Slide_w$ à deux. La boucle augmente la valeur de $Slide_w$ de 1. Sous réserve qu'un délai global T_{ILPH} soit atteint ou que la dernière valeur de $Slide_w$ est supérieure à $n_d/2$, l'ILPH s'arrête.

L'ILPH que nous proposons sera évaluée comme une matheuristique pour faire face au problème FFTP. De plus, certaines composantes de l'ILPH peuvent également être considérées comme une composante de voisinage qui peuvent également être intégrés dans l'approche ALNS que nous proposons plus loin dans le chapitre, cela sera également étudié.

6.5 MÉTAHEURISTIQUE ALNS POUR LE PROBLÈME FFTP

Les solveurs peuvent rencontrer des difficultés pour obtenir des solutions suffisamment bonnes dans des délais de traitement courts en exécutant le modèle ILP et la matheuristique ILPH. Les approches de résolution basées sur les métaheuristiques ont été largement utilisées dans la littérature pour traiter une grande variété de problèmes d'optimisation, pour une étude complète nous invitons le lecteur à se reporter [Hussain et al., 2019].

Nous proposons une métaheuristique Adaptive Large Neighborhood Search (ALNS) pour calculer des solutions de bonne qualité pour le problème de planification d'équipes de pompiers (FFTP). Pour une revue plus générale et récente sur la métaheuristique ALNS et ses applications, nous invitons le lecteur à se référer à [Mara et al., 2022]. Nous présentons dans l'Algorithme 5 sa structure générale avant de donner un aperçu des composants.

Algorithm 5: ALNS for FFTP

Input : An instance of FFTP
Output : S_{best} best solution found
Parameters: D_{limit} limit for diversification degree,
Variables : $MaxIter$ maximum number of iterations without any improvement prior stopping the ALNS
$AcceptIter$ number of iterations without any improvement prior accepting a degradation,
D_{max} current diversification degree, M_d, M_c destruction and construction methods,
S_{old} current solution to be improved, S_{cur} solution under work

```
1  i, i' := 0 /* i, i' number of iterations */
2  MaxIter := ε.n_c
3  AcceptIter := n_c
4  D_max := 3
5  D_limit := ⌈n_c / n_s⌉
6  S_best, S_old := BuildFeasibleSchedule()
7  while i < MaxIter do
8      M_d := ChooseDestructionMethod()
9      k := rand(1,D_max)
10     S_cur := AdaptiveDestruction(S_old, k, M_d)
11     M_c := ChooseConstructionMethod()
12     S_cur := ApplyConstruction(S_cur, M_c) /* insert as many crews as possible in S_cur */
13     if Obj(S_cur) < Obj(S_best) then
14         S_best, S_old := S_cur
15         i, i' := 0
16         D_max := 3
17     else
18         if (i' ⩾ AcceptIter and AcceptDegradation(S_cur, S_old)) then
19             S_old := S_cur
20             i' := 0
21         i++
22         i'++
23         D_max := Min(D_max+1, D_limit)
24     UpdateDestructionsScores()
```

Une solution initiale S_{old} est calculée à l'aide d'une simple heuristique $BuildFeasibleSchedule()$. Cette première solution est construite pour respecter toutes les contraintes dures.

Une méthode de destruction M_d est choisie au hasard parmi **D**estruction **A**léatoire (Random Destruction, RD) et **D**estruction **I**ntelligente (Smart Destruction, SD) en utilisant la procédure de choix $ChooseDestructionMethod()$.

La méthode de **D**estruction **I**ntelligente utilise le **C**ritère de **M**eilleur **D**estruction ou **I**nsertion (Best Insertion Destruction Criterion, BIDC) pour évaluer l'impact de la désaffectation des équipes.

Nous supprimons $k \leqslant D_{max}$ équipes à chaque itération. Nous définissons D_{max} comme le degré de diversification. Cette valeur est initialisée à trois puis incrémentée après chaque itération sans amélioration jusqu'à D_{limit}. Nous choisissons de fixer $D_{limit} = \lceil n_c/n_s \rceil$ qui représente le nombre moyen d'équipes pouvant être affectées à un type de quart pour une journée. Si une amélioration est trouvée, nous réinitialisons D_{max} à trois pour explorer entièrement le voisinage de la nouvelle solution. Ce mécanisme de diversification adaptative permet d'élargir la recherche autour d'une solution dans le but d'obtenir une meilleure solution.

En appliquant la procédure $AdaptiveDestruction(S_{old}, k, M_d)$ nous obtenons la solution courante S_{cur}. Certaines équipes ont été désaffectées, et d'autres peuvent ne pas encore être attribuées. Nous pouvons donc utiliser ces équipes pour trouver une meilleure solution.

Une méthode de construction M_c est sélectionnée aléatoirement en utilisant $\mathtt{ChooseConstructionMethod}()$. Les méthodes de construction visent à compléter et à améliorer la solution actuelle. Nous implémentons un algorithme adaptatif basé sur la meilleure insertion possible (Best Insertion Algorithm, BIA) qui évalue également les affectations réalisables à l'aide du BIDC. Nous utilisons aussi comme méthode de construction une version de l'ILPH (voir Section 6.4).

Nous insérons ensuite autant d'équipes que possible tout en respectant les contraintes dures en utilisant $\mathtt{ApplyConstruction}(S_{cur}, M_c)$.

Pour éviter d'être bloqué dans des optima locaux, l'approche ALNS nécessite une procédure d'acceptation qui rend possible la sélection d'une solution de faible qualité dans le but d'explorer d'autres parties de l'espace de recherche. Nous utilisons une approche basée sur le principe d'enregistrement à enregistrement (Record-to-record) (Dueck, 1993). À condition qu'un certain nombre d'itérations $\mathtt{AcceptIter}$ sans aucune amélioration soient effectuées, et à condition qu'une solution de qualité inférieure soit acceptée par $\mathtt{AcceptDegradation}(S_{cur}, S_{old})$, nous rendons possible de continuer à explorer l'espace de recherche en utilisant cette solution de qualité inférieure. Nous choisissons de définir $\mathtt{AcceptIter}$ à n_c, l'idée générale est d'augmenter le temps de traitement à mesure que le nombre d'équipes n_c augmente.

Le succès des méthodes de destruction peut varier en fonction de l'instance. Un choix adaptatif conduit généralement à de meilleurs résultats plutôt que de fixer les choix une fois pour toutes. Chaque méthode de destruction a un score qui représente sa part dans une roue de roulette. La procédure $\mathtt{UpdateDestructionsScores}()$ met à jour les scores.

Lorsqu'un nombre maximum d'itérations $\mathtt{MaxIter}$ sans aucune amélioration est atteint, l'algorithme ALNS s'arrête, puis renvoie la meilleure solution trouvée S_{best}. Nous choisissons de définir $\mathtt{MaxIter}$ à $\varepsilon.n_c$, où ε doit être réglé pour obtenir un bon compromis entre des solutions de bonne qualité et des temps de traitement raisonnables.

Des mécanismes adaptatifs de destruction, de construction et des procédures d'acceptation sont utilisées. Plusieurs paramètres doivent être réglés pour obtenir une bonne efficacité de ces mécanismes adaptatifs.

Critère de Meilleure Destruction ou Insertion, BIDC

Nous utilisons le BIDC dans la méthode **D**estruction **I**ntelligente et dans l'algorithme adaptatif basé sur la meilleure insertion possible BIA. Soit (d, s, c) un triplet pour un jour, un quart et une équipe. Soit une solution à améliorer S_{old} ou une solution courante S_{cur}, nous évaluons une désaffectation ou une affectation d'un triplet en calculant le BIDC comme suit :

$$(SG^{\alpha}.\ SST^{\beta}.\ CA^{\gamma}.\ PF^{\theta}.\ P^{\omega}.\ EB^{\mu})$$

Le BIDC est composé d'un terme pour chaque contrainte souple : SG est pour le groupement de quarts (S1), SST est pour la même heure de début (S2), CA est pour l'affectation des compensations (S3), PF est pour l'équité sur la période de planification (S4), P est pour les préférences (S5) et EB est pour l'équilibre des quarts de travail supplémentaires quotidiens (S6).

La valeur BIDC prend la valeur $+\infty$ lorsqu'une contrainte stricte est violée.

Les valeurs des paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ sont gérées par une stratégie adaptative qui porte sur le BIA, cela permet d'adapter l'importance relative des termes au cours du déroulement de l'algorithme.

Méthodes de destruction et mécanisme adaptatif pour choisir une méthode de destruction

La procédure $AdaptiveDestruction(S_{old}, k, M_d)$ permet de désaffecter certaines équipes de S_{old}, solution à améliorer. Étant donné une valeur k pour le nombre d'équipes à désaffecter, une des deux méthodes de destruction suivantes s'applique :

Destruction aléatoire (RD) : les équipes sont sélectionnées aléatoirement ;

Destruction intelligente (SD) : les équipes sont sélectionnées à l'aide du BIDC.

Pour la méthode **D**estruction **I**ntelligente, nous considérons tous les triplets (d, s, c) des équipes affectées, et nous évaluons leurs scores BIDC.

Nous utilisons le jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ qui a produit la meilleure solution à l'itération précédente de BIA (voir **Méthodes de construction**). Les scores sont ensuite utilisés comme parts dans une roulette qui sert à sélectionner les équipes k à désaffecter. Notre objectif est de sélectionner les équipes assignées avec le BIDC le plus élevé dans le but d'affecter ces équipes à de meilleurs jours et quarts de travail.

La sélection d'une méthode de destruction est gérée à l'aide d'un mécanisme adaptatif. Soit $DeSc_{ij}$ le **S**core de **D**estruction de la méthode $j \in \{RD, SD\}$ à l'itération i. Après chaque itération i, les scores de destruction sont mis à jour par $UpdateDestructionsScores()$ comme suit :

$DeSc_{ij} = (1 + \lambda).DeSc_{(i-1)j}$ si $Obj(S_{cur}) < Obj(S_{best})$;

$DeSc_{ij} = (1 + (1/2)\lambda).DeSc_{(i-1)j}$ si $Obj(S_{cur}) < Obj(S_{old})$ et $Obj(S_{cur}) \geqslant Obj(S_{best})$;

$DeSc_{ij} = (1 - (1/2)\lambda).DeSc_{(i-1)j}$ si $Obj(S_{cur}) \geqslant Obj(S_{old})$ et $Obj(S_{cur}) \geqslant Obj(S_{best})$;

$DeSc_{ij} = DeSc_{(i-1)j}$ si la méthode de destruction j n'est pas utilisé.

L'objectif est de privilégier la méthode de destruction qui obtient le meilleur résultat au cours de l'algorithme. Cependant, cela ne peut pas être fait au détriment définitif de l'un contre l'autre puisque l'efficacité relative peut dépendre de l'instance et peut également changer au cours de l'algorithme. Le paramètre λ est utilisé pour lisser le renforcement et doit être ajusté pour obtenir un mécanisme adaptatif efficace.

Méthodes de construction

Le $ApplyConstruction(S_{cur}, M_c)$ est utilisé pour compléter la solution courante S_{cur} en affectant autant d'équipes que possible. Étant donné S_{cur}, une des deux méthodes constructions suivantes s'appliquent :

ILPH : en utilisant uniquement le voisinage Fix Day ;

BIA : en utilisant un mécanisme adaptatif pour gérer les paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$.

La matheuristique ILPH est chronophage, nous avons donc choisi de n'utiliser que le voisinage Fix Day pour implémenter une méthode de construction. De plus, nous avons fixé la limite de temps T_{ILPH} à une minute afin d'éviter de perdre inutilement du temps de traitement dans la résolution de l'ILP. Le paramètre $Size_w$ doit être réglé pour obtenir un bon compromis entre la qualité de la solution et le temps de traitement. Rappelons que les deux paramètres $Size_w$ et $Slide_w$ sont liés comme $Size_w = 2 . Slide_w$.

L'algorithme BIA (voir Algorithme 6) utilise S_{cur} la solution partielle en cours et un jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ comme entrées, et il essaie d'insérer autant d'équipes non affectées c que possible.

Le BIA utilise le **C**ritère de **M**eilleure **D**estruction ou **I**nsertion (BIDC) pour évaluer toutes les insertions réalisables (d, s, c). Le meilleur triplet est retenu, le cas échéant. La meilleure insertion est effectuée, puis la qualité de la solution est évaluée. Lorsque plus aucune insertion valide n'est possible, le BIA s'arrête. Étant donné un jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$, une exécution de BIA renvoie $S_{curbest}$, la meilleure solution trouvée qui peut ensuite éventuellement être utilisée comme nouvelle S_{cur} pour la prochaine itération de l'ALNS.

Algorithm 6: BIA for FFTP

```
Input      : S_cur a partial solution under work
             (α, β, γ, θ, ω, μ) parameter set
Output     : S_curbest best solution found over the BIA iterations
Variables  : (d,s,c)* best triplet, Bsuccess boolean
1  S_curbest := S_cur  /* store reference solution for BIA */
2  Bsuccess := true
3  while Bsuccess do
4      (d,s,c)* := (∅, ∅, ∅)
       /* Using BIDC to assess, find the best triplet, if any */
5      foreach d ∈ 𝒟 do
6          foreach s ∈ 𝒮 do
7              foreach c ∈ UnassignedCrews(d, s) do
8                  ComputeBIDC(d,s,c)
9                  UpdateBestTriplet (d,s,c)*
10     Bsuccess := Insert(S_cur, (d,s,c)*)  /* if no feasible insertion return false */
       /* Comparing S_cur and S_curbest, all terms of the objective function are assessed */
11     if Obj(S_cur) < Obj(S_curbest) then
12         S_curbest := S_cur
```

Nous utilisons un mécanisme adaptatif pour gérer la construction d'une nouvelle S_{cur} en utilisant le BIA. Nous exécutons séparément quatre BIA avec différentes valeurs du jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$. Soit α_{i-1}, β_{i-1}, γ_{i-1}, θ_{i-1}, ω_{i-1} et μ_{i-1} les meilleures valeurs de paramètres obtenues lors d'une précédente itération de l'ALNS. Les valeurs α, β, γ, θ, ω et μ sont choisies aléatoirement dans l'espace à six dimensions ayant pour centre $(\alpha_{i-1}, \beta_{i-1}, \gamma_{i-1}, \theta_{i-1}, \omega_{i-1}, \mu_{i-1})$ et la longueur de côté ϕ.

La meilleure solution $S_{curbest}$ obtenue parmi les quatre exécutions est conservée et le jeu de paramètres qui la produit est stocké pour être utilisé pour la prochaine itération. Ce meilleur jeu de paramètres est utilisé par **D**estruction **I**ntelligente et l'algorithme BIA lorsqu'il est choisi. Ce mécanisme adaptatif permet d'accélérer la convergence de l'algorithme ALNS vers une bonne solution.

Le paramètre de longueur de côté ϕ doit être ajusté pour obtenir une bonne performance du mécanisme adaptatif pour la méthode de destruction intelligente et pour la méthode de construction BIA.

Enfin, $ChooseConstructionMethod()$ utilise un paramètre ψ pour choisir soit le BIA (avec une probabilité ψ) soit la version de l'ILPH (avec une probabilité $(1-\psi)$. Le paramètre ψ doit être ajusté pour obtenir un bon compromis entre l'utilisation du BIA ou l'utilisation de la méthode de construction basée sur l'ILPH.

Stratégie d'acceptation

La stratégie d'acceptation que nous avons mise en œuvre dans $AcceptDegradation(S_{cur}, S_{old})$ est basée sur l'approche enregistrement à enregistrement (Record-to-recod) introduite dans [Dueck, 1993]. À condition que $AcceptIter$ itérations aient été effectuées sans améliorer la qualité de S_{old}, cette solution peut être remplacée par S_{cur} une solution de moins bonne qualité. Nous utilisons un taux d'acceptation τ à ajuster qui joue le rôle d'un paramètre de déviation. La solution S_{cur} prend la place de S_{old} si $(S_{cur} - S_{old})/S_{old} \leqslant \tau$.

6.6 EXPÉRIMENTATIONS, TESTS ET RÉSULTATS

Dans nos expérimentations, nos objectifs étaient : (i) fournir une synthèse sur le réglage des paramètres de la métaheuristique ALNS permettant d'obtenir les meilleurs résultats ; (ii) montrer l'efficacité des mécanismes adaptatifs que nous avons mis en place pour la destruction et les méthodes de construction ; (iii) évaluer l'apport des méthodes de destruction ; (iv) comparer les performances entre le modèle ILP, la matheuristique ILPH et deux versions de l'approche ALNS ; (v) évaluer la qualité des solutions calculées par l'ILP, l'ILPH et l'ALNS sur les instances des différents datasets.

Les tests ont été effectués en utilisant C++ compilé avec gcc version 7.5.0, en utilisant STL, à l'aide d'un solveur CPLEX 12.10 [IBM, 2020] avec un seul thread et le paramètre ILPEmphasis défini sur faisabilité, sur une machine avec un processeur Intel(R) Xeon(R) X7542 à 2,6 GHz et 64 Go de RAM.

Aperçu des datasets et poids des termes de la fonction objectif

Nous avons testé les versions ILP, ILPH et ALNS sur un benchmark composé de quatre datasets de sept instances que nous avons généré à partir de données réelles de l'institution INFOCA. Les instances sont classées et étiquetées en fonction du nombre d'équipes n_c et du nombre total quotidien de demandes de quarts de travail (c'est-à-dire $\sum r_s$). Ainsi, les instances sont notées cXXrYY(a/b), la notation (a/b) est utilisée si n_c et $\sum r_s$ sont égaux pour deux instances distinctes qui sont différentes dans les distributions des exigences minimales.

Selon les exigences de l'institution INFOCA et pour exprimer l'importance relative des différentes contraintes, nous fixons w_{oc} à 2, w_{sg} à 1, w_{sst} à 1 , w_{ca} à 1 et w_p à 2. Ces poids sont utilisés pour calculer $Obj(S)$, la valeur de la fonction objectif (voir aussi les équations (6.1a)-(6.1e)).

Réglage des paramètres

L'ALNS que nous proposons utilise des paramètres qui doivent être réglés pour obtenir une bonne efficacité des mécanismes de destruction, de construction, d'adaptation et des procédures d'acceptation. Nous récapitulons ces paramètres dans le Tableau 6.1.

Tableau 6.1: Parametres ϕ, ε, λ, τ, ψ et $Size_w$ à régler pour l'ALNS pour le FFTP.

ϕ	longueur de coté, mécanisme adaptatif pour le jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$
ε	$MaxIter = \varepsilon.n_c$, nombre maximum d'itérations sans aucune amélioration
λ	mécanisme adaptatif de gestion des méthodes de destruction
τ	taux d'acceptation
ψ	probabilité utilisée pour choisir parmi les méthodes de construction
$Slide_w$	de combien de jours la fenêtre glisse (ILPH)

Nous avons réalisé des expériences préliminaires afin de calibrer les paramètres. Deux instances sont sélectionnées aléatoirement dans chaque dataset pour le réglage. L'ALNS étant une méthode de recherche probabiliste, les expériences ont été répétées dix fois avec une graine aléatoire différente.

Nous évaluons le réglage des paramètres à l'aide de la métrique **E**rreur **R**elative en **P**ourcentage (Relative Percentage Error, RPE) calculée comme $RPE = 100 \cdot \frac{Z_{min} - Z_{best}}{Z_{best}}$, où Z_{best} désigne le meilleur résultat que nous avons obtenu sur toutes les exécutions effectuées pour une instance, et Z_{min} désigne le meilleur résultat que nous avons obtenu parmi les dix exécutions effectuées. Les valeurs **RPE** sont moyennées et rapportées en pourcentage dans les figures.

Nous avons commencé par régler le premier paramètre ϕ, puis nous avons réglé chaque paramètre l'un après l'autre compte tenu de l'ordre du Tableau 6.1. Les valeurs initiales que nous utilisons pour régler d'abord ϕ sont $\varepsilon = 5$, $\lambda = 1$, $\tau = 0.01$, $\psi = 0.5$ et $Slide_w = 3$. Ensuite, pour régler un paramètre on retient le meilleur réglage des autres paramètres trouvés avant de procéder à son réglage.

Les valeurs de ϕ, la longueur du côté pour le mécanisme de construction adaptative, peuvent affecter significativement les performances de l'ALNS. Ce mécanisme permet d'adapter les valeurs des paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ de sorte que l'importance relative des termes du BIDC peut changer au cours de l'algorithme.

Dans la figure 6.1, nous montrons l'impact sur le RPE moyen en faisant varier ϕ de 0,0 à 1,0 avec un pas de 0,025. Lorsque ϕ est inférieur à 0,1, les quatre recherches parallèles indépendantes utilisent des valeurs de $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ trop proches de la valeur initiale. Lorsque ϕ est supérieur à 0,1, le RPE moyen est dégradé, comme le montrent les valeurs moyennes du RPE. Nous avons choisi de fixer ϕ à 0,1 pour lequel le RPE moyen minimum est obtenu.

Le nombre maximum d'itérations sans aucune amélioration est $MaxIter = \varepsilon.n_c$ où n_c est le nombre d'équipes. Comme on pouvait s'y attendre, le temps de traitement

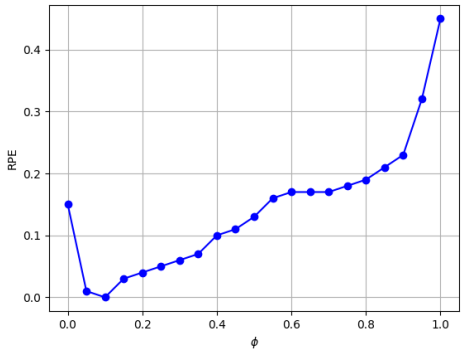

Fig 6.1: Calibrage de ϕ, impact sur le RPE moyen.

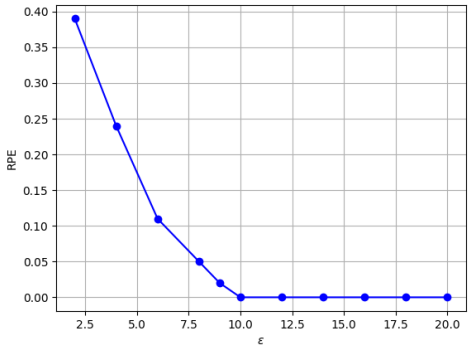

Fig 6.2: Calibrage de ε, impact sur le RPE moyen.

augmente avec la valeur ε. Comme on pouvait s'y attendre avec ce réglage, le RPE moyen diminue à mesure que ε augmente. Pour régler ε, nous le faisons varier dans $\{2, \cdots, 20\}$. Comme on peut le voir sur la figure 6.2, le minimum est atteint lorsque ε est égal à dix, alors il devient constant. Nous avons choisi de fixer ε à dix pour ne pas trop consommer de temps de traitement.

Le **S**core de **D**estruction des méthodes est géré par le paramètre λ. Nous avons fait varier $\lambda \in [0, 20]$ avec un pas de 0,5. Lorsque la valeur $\lambda = 0$ le mécanisme de mise à jour du score est désactivé, et on n'obtient pas le RPE moyen minimum mais la valeur est proche de zéro. Les valeurs RPE diminuent jusqu'à $\lambda = 1.0$, puis elles sont constantes pour les valeurs λ dans $[1, 3]$. Ensuite, les valeurs RPE augmentent à mesure que la valeur λ augmente.

Le mécanisme de mise à jour des scores permet d'obtenir de meilleurs résultats en gérant la sélection des méthodes de destruction.

Nous avons choisi de fixer λ à =1.0, cela suffit à rendre le mécanisme efficace.

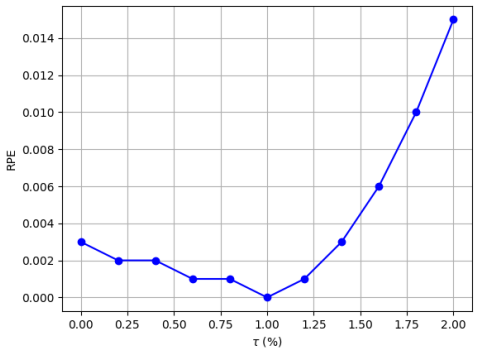

Fig 6.3: Calibrage de τ, impact sur le RPE moyen.

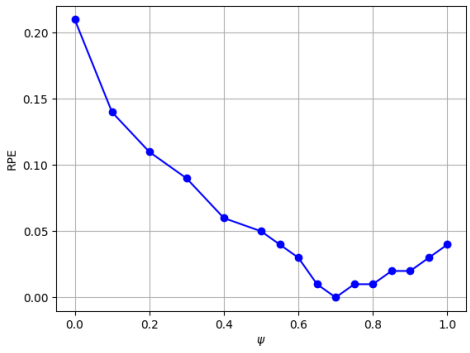

Fig 6.4: Calibrage de ψ, impact sur le RPE moyen.

Une solution peut être remplacée par une solution de moindre qualité lorsque $(S_{cur} - S_{old})/S_{old} \leqslant \tau$. Dans la Figure 6.3, nous montrons l'évolution du RPE moyen en faisant varier τ de 0 % à 2 % avec un pas de 0,2 %. Lorsque $\tau = 0\,\%$, le mécanisme d'acceptation d'enregistrement à enregistrement est désactivé. Comme on peut le voir, on obtient une bonne valeur RPE moyenne avec $\tau = 0\%$. Les valeurs RPE diminuent jusqu'à la valeur $\tau = 1\%$, ensuite elles augmentent lorsque τ augmente. Comme on pouvait s'y attendre, le mécanisme d'acceptation basé sur l'algorithme d'enregistrement à enregistrement joue son rôle. Nous avons choisi de fixer $\tau = 1\%$.

Le BIA est choisi avec une probabilité ψ et l'ILPH est choisi avec une probabilité $1 - \psi$. Dans la Figure 6.4, nous montrons l'évolution du RPE moyen en faisant varier ψ de 0,0 à 1,0 avec un pas de 0,1. Lorsque $\psi = 0.0$ l'ALNS n'utilise que l'ILPH comme méthode de construction, ce n'est pas efficace comme le montre la valeur moyenne du RPE. Les valeurs RPE diminuent jusqu'à $\psi = 0,7$, ensuite elles augmentent lorsque ψ augmente. L'ILPH contribue à obtenir de meilleurs résultats ce qui peut être observé sur la figure 6.4. Nous avons choisi $\psi = 0.7$, ce qui est un bon compromis entre les méthodes de construction BIA et ILPH.

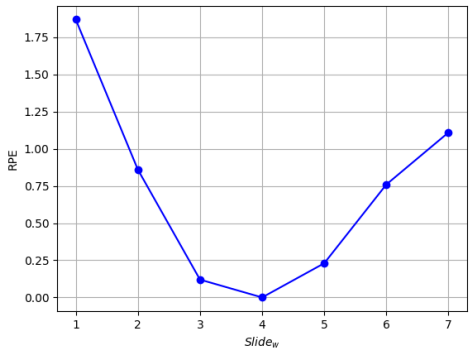

Fig 6.5: Calibrage de $Slide_w$, impact sur le RPE moyen.

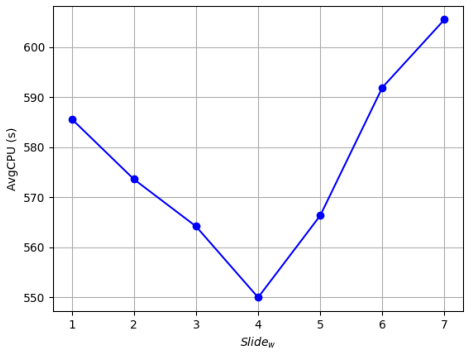

Fig 6.6: Calibrage de $Slide_w$, impact sur le temps d'exécution.

Nous utilisons une version de l'ILPH comme méthode de construction avec uniquement le voisinage Fix Day.

Nous avons effectué des expériences en faisant varier $Slide_w$ parmi les valeurs $\{1, 7\}$, et avec une minute pour le temps limite T_{ILPH}.

Dans la Figure 6.5, nous montrons que lorsque $Slide_w$ varie entre un et quatre la valeur RPE diminue, et sur la figure 6.6 nous montrons que le temps de traitement diminue.

Ensuite, le RPE moyen et les valeurs de temps de traitement augmentent à mesure que $Slide_w$ augmente. Lorsque $Slide_w < 4$ les fenêtres sont trop petites cela ne permet pas au voisinage Fix Day de trouver de meilleures solutions que celles trouvées par le BIA. Nous perdons du temps de traitement en exécutant la méthode de construction basée sur ILPH sans aucun avantage. Lorsque $Slide_w > 4$, la méthode de construction basée sur ILPH nécessite plus de temps de traitement pour obtenir des solutions. Cela

impacte également négativement la convergence de l'algorithme vers de bonnes solutions puisque nous fixons une limite de temps afin d'éviter de rester coincé dans une résolution ILP dure. Nous avons choisi de définir $Slide_w$ à quatre.

Les valeurs de calibrage finales sont $\phi = 0.1$, $\varepsilon = 10$, $\lambda = 1.0$, $\tau = 0.01$, $\psi = 0.7$ et $Slide_w = 4$. Elles ont été utilisées dans la suite pour nos expérimentations sur toutes les instances du benchmark, et elles ont été choisies pour obtenir un bon compromis entre la qualité de solution et le temps de traitement.

Évaluation des mécanismes adaptatifs de l'ALNS, évaluation des méthodes de destruction

Nous évaluons ici l'efficacité du mécanisme adaptatif BIA et l'efficacité du mécanisme de diversification pour la destruction.

Nous évaluons également l'efficacité des méthodes de destruction en désactivant chacune d'entre elles une par une. Pour ces expériences, nous utilisons toutes les instances des quatre datasets et nous enregistrons la meilleure solution trouvée sur 15 000 itérations. Nous avons effectué dix exécutions sur chaque instance du benchmark.

Le jeu de paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ est utilisé pour calculer le BIDC. Le mécanisme de construction adaptative vise à guider la recherche en calculant le meilleur compromis entre les différentes valeurs de ces paramètres sur des itérations consécutives. Nous avons mené des expériences avec le mécanisme de construction adaptative et sans le mécanisme de construction adaptative. Dans ce dernier cas, les paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ sont choisis aléatoirement dans $[0, 1]$. Dans la figure 6.7, nous montrons le RPE moyen par rapport aux itérations pour ces deux versions.

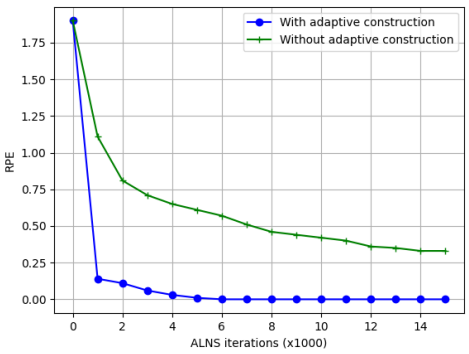

Fig 6.7: Impact de la construction adaptative avec le BIA.

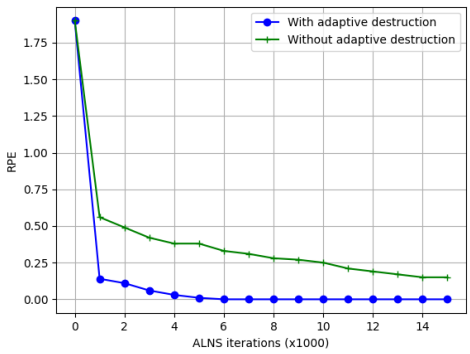

Fig 6.8: Impact de la destruction adaptative.

Comme on peut le voir sur la figure 6.7, les valeurs RPE sont détériorés lorsque les paramètres $(\alpha, \beta, \gamma, \theta, \omega, \mu)$ sont choisis aléatoirement. Le mécanisme de construction adaptative avec ϕ fixé à 0.1 accélère significativement la convergence vers de bonnes solutions.

Le mécanisme de diversification adaptative gère la valeur de $D_{max} \in \{3, \cdots, D_{limit}\}$ (voir Algorithme 5). Ce mécanisme permet d'explorer le voisinage de la nouvelle solution dès qu'une amélioration est trouvée et permet également d'explorer un voisinage plus éloigné dès lors que la recherche est piégée dans un optimum local. Nous avons désactivé le mécanisme de diversification adaptative en fixant le degré de diversification D_{max} à 3. Dans la figure 6.8, nous montrons le RPE moyen par rapport aux itérations pour ces deux versions. Le mécanisme de diversification adaptative obtient toujours de meilleurs résultats comme en témoignent les valeurs moyennes du RPE. Le mécanisme de diversification adaptative permet de converger plus rapidement vers les bonnes solutions.

Dans le Tableau 6.2, nous montrons les résultats pour les méthodes de destruction. À des fins de comparaison, nous utilisons l'erreur moyenne relative en pourcentage calculée comme $ARPE = 100 \cdot \frac{Z_{avg} - Z_{best}}{Z_{best}}$ où Z_{best} désigne le meilleur résultat que nous avons obtenu sur toutes les exécutions que nous avons effectuées pour une instance, et Z_{avg} représente la moyenne des résultats. Cela nous permet de discuter des résultats pour montrer si l'ALNS est stable au fil des exécutions. Nous utilisons également le critère de gap relative (Relative Pourcentage Gap, RPG) calculé comme $RPG = 100 \cdot \frac{Z_{min} - Z_{ILP}}{Z_{ILP}}$ où Z_{ILP} désigne la valeur atteinte par l'ILP pour une instance (si possible), et Z_{min} désigne le meilleur résultat obtenu parmi les exécutions effectuées. Cela nous permet de montrer à quel point les meilleures solutions calculées par l'ALNS sont éloignées des valeurs optimales. Dans le Tableau 6.2 sous la rubrique "ALNS", nous montrons les résultats avec les méthodes de destruction RD (Random Destruction) et SD (Smart Destruction) activées. Sous les rubriques "No RD" et "No SD", nous montrons les résultats en désactivant chacun d'eux un par un. Par souci de compacité, nous avons calculé les valeurs ARPE et RPG moyennes par dataset. Le temps de traitement moyen est en secondes. Les meilleurs résultats sont indiqués en caractères gras et **nc** est utilisée pour exprimer **n**on **c**alculable.

Tableau 6.2: Évaluation des méthodes de destruction.

	ALNS			No RD			No SD		
	ARPE	RPG	t(s)	ARPE	RPG	t(s)	ARPE	RPG	t(s)
c18	**0**	**0**	**323.42**	**0**	**0**	331.45	0.56	1.71	542.31
c30	**0.06**	**0**	**494.14**	0.09	**0**	511.26	0.89	3.93	788.23
c50	**0.20**	nc	**597.28**	0.25	nc	639.43	1.14	nc	923.67
c70	**0.43**	nc	**842.57**	0.51	nc	877.54	1.42	nc	1,269.03

L'ALNS sans RD (rubrique "No RD") obtient des résultats pour l'ARPE inférieurs à ceux de l'ALNS qui utilise RD et SD, et les délais de traitement sont plus longs. La Destruction Aléatoire RD permet d'éviter d'être coincé dans un optimum local en réalisant des perturbations. L'ALNS sans SD (rubrique "No SD") obtient des résultats pour les critères ARPE et RPG inférieurs à ceux de l'ALNS qui utilise RD et SD, et les temps de traitement sont plus importants. Comme on peut le voir dans le Tableau 6.2 en comparant les colonnes "t(s)", la SD accélère la convergence de l'algorithme ALNS et permet d'obtenir de meilleures solutions.

Les valeurs RPG peuvent être utilisées à des fins de comparaison pour les datasets c18 et c30 pour lesquels des solutions optimales peuvent être obtenues. La méthode SD est efficace, nous obtenons des solutions optimales pour ces datasets. Comme on peut l'observer pour les datasets c50 et c70, la méthode RD est nécessaire car de meilleurs résultats sont obtenus dans des temps de traitement plus courts. Les méthodes RD et SD sont bénéfiques pour explorer le voisinage d'une solution soit en obtenant de meilleurs résultats, soit en raccourcissant le temps de traitement.

Comparaison de l'ILP, de l'ILPH et de l'ALNS

À des fins de comparaison, nous avons décidé de mener des expériences avec deux versions de l'ALNS, sans et avec la méthode constructive basée sur ILPH qui utilise le voisinage Fix Day. Nous exécutons dix fois les deux versions de l'ALNS sur chaque instance. Toutes les approches de solution proposées sont exécutées dans un délai fixé à 3 600 s. Nous utilisons les critères RPG et ARPE définis précédemment comme critères d'évaluation.

Dans le Tableau 6.3, nous montrons les résultats pour les datasets c18 et c30 pour lequel l'ILP a réussi à obtenir une solution. Dans le Tableau 6.4, nous montrons les résultats pour les datasets c50 et c70 pour laquelle l'ILPH a réussi à obtenir une solution lorsque l'ILP n'a pas réussi à trouver une solution dans le délai imparti. Dans ces tableaux, **ns** représente le fait qu'il n y a pas de solution, **nc** représente **n**on **c**alculable,

et **-** indique que le délai est atteint. La dernière ligne des deux tableaux rapporte quelques valeurs moyennes. Les meilleurs résultats sont indiqués en caractères gras.

Tableau 6.3: Pour les datasets c18 et c30, les résultats de ILP, ILPH, ALNS sans ILPH et ALNS.

	ILP			ILPH			ALNS (no ILPH)				ALNS			
	Obj	gap	t(s)	Obj	gap	t(s)	Z_{min}	RPG	ARPE	t(s)	Z_{min}	RPG	ARPE	t(s)
c18r09a	**1,325**	0	1,023	1,432	8.08	-	**1,325**	0	0	271	**1,325**	0	0	281
c18r10a	**1,359**	0	1,001	1,456	7.14	-	**1,359**	0	0	287	**1,359**	0	0	292
c18r10b	**1,344**	0	1,112	1,421	5.73	-	**1,344**	0	0	301	**1,344**	0	0	311
c18r11a	**1,378**	0	1,234	1,456	5.66	-	**1,378**	0	0	309	**1,378**	0	0	323
c18r11b	**1,420**	0	1,787	1,532	7.89	-	**1,420**	0	0	321	**1,420**	0	0	341
c18r12a	**1,422**	0	1,455	1,498	5.34	-	**1,422**	0	0	341	**1,422**	0	0	355
c18r12b	**1,440**	0	1,564	1,562	8.47	-	**1,440**	0	0	344	**1,440**	0	0	361
c30r15a	**1,754**	0	2,892	1,931	10.09	-	**1,754**	0	0	451	**1,754**	0	0	455
c30r16a	**1,780**	0	2,911	1,945	9.27	-	**1,780**	0	0.04	463	**1,780**	0	0.03	469
c30r17a	**1,810**	0	3,002	1,911	5.58	-	**1,810**	0	0.10	472	**1,810**	0	0.11	483
c30r18a	**1,832**	0	3,014	1,934	5.57	-	**1,832**	0	0.03	481	**1,832**	0	0.04	495
c30r19a	**1,880**	0	3,111	1,990	5.85	-	**1,880**	0	0.05	489	**1,880**	0	0.12	501
c30r20a	1,932	1.79	-	1,978	4.21	-	**1,898**	-1.75	0.13	513	**1,898**	-1.75	0.09	522
c30r21a	**1,913**	0	3,213	2,005	4.81	-	**1,913**	0	0.07	502	**1,913**	0	0.05	534
Avg		0.13	2,208		6.69	3,600		-0.13	0.03	396		-0.13	0.03	408

Pour chaque instance, sous les rubriques “ILP” et “ILPH”, nous montrons la valeur de la fonction objectif, l’écart et le temps de traitement (en secondes) que nous avons obtenus pour l’ILP et la matheuristique ILPH (sous “Obj ”, “gap” et “t(s)”, respectivement).

Pour chaque instance, sous les rubriques “ALNS (no ILPH)” et “ALNS”, nous affichons le meilleur résultat obtenu parmi les exécutions effectuées (la meilleure valeur Obj), le RPG, l’ARPE et le temps de calcul en secondes (sous les colonnes “Z_{min}”, “RPG”, “ARPE” et “t(s)”, respectivement).

Dans le Tableau 6.3, nous montrons que l’ILP atteint des solutions optimales pour toutes les instances sauf pour l’instance c30r20a pour laquelle l’écart est de 1,79%. Aucune des solutions optimales n’est obtenue par la matheuristique ILPH, et le gap moyen est de 6,69%. Les deux versions de l’ALNS ont réussi à obtenir toutes les solutions optimales. Une meilleure solution est trouvée pour l’instance c30r20a lorsque l’ILP ne peut pas atteindre une solution optimale dans le délai d’une heure.

Les valeurs RPG et ARPE sont soit égales à zéro soit presque nulles sauf pour l’instance c30r20a. Pour l’instance c30r20a, les deux versions ALNS ont réussi à atteindre une meilleure solution, on obtient donc la valeur négative -1.75% pour RPG. Lorsque nous avons utilisé les solutions trouvées par les deux versions de l’ALNS pour cette instance comme solutions initiales pour le modèle ILP, nous avons constaté qu’elles étaient optimales. Pour le dataset c30, les deux versions ALNS atteignent toutes les solutions optimales.

Pour les datasets c18 et c30, les deux versions ALNS sont stables lors du calcul des solutions, comme le montrent les valeurs ARPE. Les temps de traitement des deux versions ALNS sont environ cinq fois plus courts que ceux de l’ILP en moyenne, et pour toutes les instances nous avons obtenu des solutions optimales.

Dans le Tableau 6.4, nous montrons que l’ILP a échoué à atteindre une solution dans le délai imparti, et que la matheuristique ILPH a réussi à obtenir des solutions réalisables. La qualité de la solution des deux versions ALNS ne peut pas être évaluée à l’aide du critère RPG car l’ILP ne peut pas obtenir de solutions optimales dans le délai imparti. À l’exception de l’instance c50r31a, les deux versions ALNS ont obtenu les mêmes valeurs Z_{min}. Les temps de calcul sont à peu près les mêmes (environ 715 s et 720 s en moyenne) et entraînent des dépenses supplémentaires raisonnables par rapport aux datasets c18 et c30 avec des instances plus petites (environ 400 s en moyenne). Les deux versions ALNS restent stables comme en témoignent les valeurs d’ARPE qui sont proches (0,34 et 0,32 en moyenne). Pour l’instance c50r31a, l’ALNS qui utilise la méthode constructive basée sur ILPH a réussi à obtenir une meilleure solution. La méthode constructive basée sur ILPH contribue à obtenir de meilleurs résultats comme le montre la figure 6.4 que nous avons présentée lors du réglage de ψ.

Les résultats que nous avons obtenus montrent que l’ALNS a obtenu des résultats optimaux sur les plus petites instances. Pour les plus grandes instances, l’ALNS a obtenu des résultats de meilleure qualité dans des temps de traitement plus courts par rapport à ceux obtenus par la matheuristique ILPH utilisée seule.

Comparaison avec les travaux précédents

Tableau 6.4: Pour les datasets c50 et c70, les résultats de ILP, ILPH, ALNS sans ILPH et ALNS

	ILP			ILPH			ALNS (no ILPH)				ALNS			
	Obj	gap	t(s)	Obj	gap	t(s)	Z_{min}	RPG	ARPE	t(s)	Z_{min}	RPG	ARPE	t(s)
c50r22a	ns	nc	-	3,876	nc	-	**3,623**	nc	0.22	534	**3,623**	nc	0.17	566
c50r23a	ns	nc	-	3,845	nc	-	**3,680**	nc	0.14	545	**3,680**	nc	0.15	579
c50r26a	ns	nc	-	3,854	nc	-	**3,710**	nc	0.15	567	**3,710**	nc	0.23	543
c50r28a	ns	nc	-	3,911	nc	-	**3,740**	nc	0.33	573	**3,740**	nc	0.12	567
c50r31a	ns	nc	-	3,967	nc	-	3,834	nc	0.27	588	**3,830**	nc	0.23	591
c50r33a	ns	nc	-	4,034	nc	-	**3,878**	nc	0.31	579	**3,878**	nc	0.29	563
c50r35a	ns	nc	-	4,211	nc	-	**3,987**	nc	0.21	781	**3,987**	nc	0.22	772
c70r31a	ns	nc	-	4,931	nc	-	**4,609**	nc	0.39	793	**4,609**	nc	0.39	832
c70r33a	ns	nc	-	4,976	nc	-	**4,680**	nc	0.43	814	**4,680**	nc	0.41	801
c70r37a	ns	nc	-	5,011	nc	-	**4,713**	nc	0.47	803	**4,713**	nc	0.43	844
c70r40a	ns	nc	-	4,988	nc	-	**4,770**	nc	0.52	841	**4,770**	nc	0.50	866
c70r44a	ns	nc	-	5,113	nc	-	**4,834**	nc	0.37	857	**4,834**	nc	0.41	811
c70r47a	ns	nc	-	5,134	nc	-	**4,936**	nc	0.58	863	**4,936**	nc	0.51	867
c70r50a	ns	nc	-	5,178	nc	-	**5,002**	nc	0.34	879	**5,002**	nc	0.35	877
Avg									0.34	715			0.32	720

Par souci de compacité, les résultats obtenus dans [Ouberkouk et al., 2021] n'ont pas été présentés pour chaque instance dans le Tableau 6.3 et le Tableau 6.4. Dans le Tableau 6.5 et le Tableau 6.6, nous comparons les approches de résolution par dataset.

L'ILP que nous avons présenté dans la section 6.3 améliore les formulations des contraintes qui portent sur les variables α, β et γ afin de réduire le nombre de ces variables.

Dans le Tableau 6.5, pour chaque dataset, la colonne "#Variables" montre la somme du nombre de variables α, β et γ, la colonne "Avg Gap" montre l'écart (gap) moyen, la colonne "Avg t(s)" montre le temps de traitement moyen en secondes, et la colonne "#OS" indique le nombre de solutions optimales trouvées.

Comme on peut le voir dans les colonnes '#Variables", le nombre de variables est réduit. Ainsi, nous avons obtenu six nouvelles solutions optimales pour le dataset c30 alors qu'aucune solution optimale n'a été trouvée par l'ILP précédent. De plus, les délais de traitement sont réduits.

Dans le Tableau 6.6, nous montrons une comparaison entre l'heuristique adaptative de construction de destruction itérative (Adaptive Iterative Desctruction Construction Heuristic, AIDCH) présentée dans [Ouberkouk et al., 2021] et l'ALNS que nous avons proposée. La colonne "Avg Z_{min}" montre la moyenne des meilleurs résultats obtenus parmi les exécutions effectuées, les colonnes "Avg RPG" et "Avg ARPE" montrent le RPG moyen et l'ARPE moyen lorsqu'ils peuvent être calculé, et la colonne "Avg t(s)" indique le temps de traitement moyen en secondes.

Tableau 6.5: Ancien ILP vs Nouveau ILP.

	Earlier ILP				New ILP			
Dataset	#Variables	Avg Gap	Avg t(s)	#OS	#Variables	Avg Gap	Avg t(s)	#OS
c18	129,360	0	1,488	7/7	2,640	0	1,311	7/7
c30	216,090	2.67	3,600	0/7	4,410	0.25	3,106	6/7
c50	360,150	nc	3,600	nc	7,350	nc	3,600	nc
c70	504,210	nc	3,600	nc	10,290	nc	3,600	nc

Tableau 6.6: AIDCH vs ALNS.

	AIDCH				ALNS			
Dataset	Avg Z_{min}	Avg RPG	Avg ARPE	Avg t(s)	Avg Z_{min}	Avg RPG	Avg ARPE	Avg t(s)
c18	1,384	0	0	374	1,384	0	0	323
c30	1,873	5.97	0.16	602	1,838	0	0.06	494
c50	3,878	nc	0.43	791	3,778	nc	0.20	597
c70	5,185	nc	0.70	1,038	4,792	nc	0.43	843

Des résultats équivalents (c18) ou meilleurs (c30, c50 et c70) sont obtenus par l'ALNS, comme on le voit par les valeurs Avg Z_{min}. L'ALNS est plus stable comme le montrent les valeurs ARPE moyennes. De plus, les délais de traitement sont réduits. L'ALNS fonctionne mieux que l'approche de la solution AIDCH pour toutes les métriques que nous avons utilisées.

L'AIDCH n'utilise pas de multiples méthodes de destruction/construction avec les mécanismes adaptatifs associés. L'AIDCH ne met pas en œuvre un mécanisme d'acceptation de solution de faible qualité. De plus, les améliorations de l'ILP permettent de proposer une version de l'ILPH que nous avons utilisée comme méthode de construction pour l'ALNS.

6.7 CONCLUSION ET PERSPECTIVES

Dans ce chapitre, nous avons abordé le problème réel de planification d'équipes de pompiers (FFTP) de l'institution INFOCA pour lequel les équipes de pompiers doivent être programmées au cours de la période à haut risque de feux de forêt. L'enjeu est d'obtenir une capacité opérationnelle maximale en considérant les contraintes qui permettent d'atteindre cette capacité opérationnelle, en plus des contraintes liées à la réglementation du travail.

Nous avons présenté un modèle ILP, une matheuristique ILPH et une métaheuristique ALNS pour aborder le FFTP d'INFOCA. Les approches proposées ont été testées à l'aide de quatre datasets de taille croissante que nous avons générés à partir de données réelles.

L'approche ILP a obtenu des résultats optimaux pour les deux datasets avec de petites instances, mais a rencontré des difficultés pour obtenir des solutions réalisables pour les instances plus grandes dans un délai de traitement raisonnable.

Le modèle ILP nous a permis de concevoir la matheuristique ILPH qui permet d'obtenir des solutions pour toutes les instances mais de moindre qualité par rapport aux solutions optimales que nous avons obtenues, et au prix d'un temps de traitement important. Pour obtenir de meilleurs résultats dans des temps de traitement plus courts, nous avons ensuite proposé une approche de solution métaheuristique ALNS qui utilise à la fois l'algorithme de meilleure insertion et une version de la matheuristique ILPH comme méthodes constructives. Des expériences ont été menées pour ajuster les paramètres afin d'obtenir un bon compromis entre la qualité de la solution et le temps de traitement. Des expériences supplémentaires ont été menées pour évaluer l'efficacité des principaux composants de l'approche ALNS, et tous ces composants sont nécessaires pour obtenir de bonnes solutions dans un bon temps de traitement. L'ALNS a obtenu tous les résultats optimaux obtenus en utilisant l'ILP sur les plus petites instances. Pour les plus grosses instances, les résultats sont de meilleure qualité par rapport à ceux obtenus par la matheuristique ILPH, de plus les temps de traitement sont significativement réduits.

Une future direction de recherche serait de considérer un problème plus large et également de considérer des objectifs et des contraintes supplémentaires en considérant le planning des opérations de maintenance préventive des ressources matérielles, celles-ci peuvent impacter la disponibilité de certaines ressources donc le nombre de certains types de quarts par jour.

Chapitre 7

Conclusion générale et perspectives

Nous nous sommes intéressés dans cette étude à deux problèmes de planification : le problème de planification de cours de l'UTC et le problème de planification d'équipes de pompiers de l'institution INFOCA. Les deux problèmes sont issus de situations réelles. Pour les deux problèmes étudiés, nous avons défini l'ensemble des données et des contraintes suite aux échanges avec le SME de l'UTC et les pompiers de l'institution INFOCA. Un générateur d'instances a été développé pour chaque problème afin de pouvoir générer un benchmark pour tester les méthodes proposées pour résoudre chacun des deux problèmes.

Nous avons étudié dans le premier problème traité le problème de planification de cours de l'UTC (UTC-UCTTP). Nous avons présenté un modèle ILP, une heuristique AIDCH et une métaheuristique ALNS pour aborder le problème UTC-UCTTP. Les approches proposées ont été testées à l'aide d'un benchmark de vingt instances que nous avons générées à partir de données réelles. Les résultats obtenus par la méthode ALNS ont été satisfaisants avec des temps de traitement raisonnables.

Une future direction de recherche sur le problème serait de considérer le modèle ILP dans la phase de construction de solutions en proposant une méthode matheuristique et l'exploiter.

Une autre direction de recherche est d'utiliser les méthodes proposées pour traiter le problème UTC-UCTTP comme des outils d'aide à la décision lors de la phase de conception des EDTs par le SME.

Un exemple d'application est l'étude de la compatibilité entre UVs dans une grille. L'étude est à réaliser avant de commencer la conception des EDTs. L'hypothèse de départ est qu'aucun emploi du temps, complet ou partiel n'est connu. Considérant une grille d'emploi du temps composé d'UVs, l'objectif est d'étudier les compatibilités entre les UVs de cette grille. L'idée générale est de rechercher les contraintes de compatibilités qui peuvent, potentiellement, être respectées sans tenir compte de contraintes de ressources comme les salles ou les enseignants.

Un autre exemple est l'étude de l'ajout d'une nouvelle UV dans une grille d'UVs. Considérant une grille existante d'UVs et en ayant les contraintes de compatibilités que l'on souhaite assurer pour ces UVs, l'objectif est de déterminer s'il est possible d'ajouter une nouvelle UV dans cette grille. Pour un responsable pédagogique, il s'agit de demander s'il est possible d'ajouter une nouvelle UV à une grille existante et sous quelles conditions. Cette étude de faisabilité structurelle se traduit en terme de gestion des

incompatibilités entre l'UV que l'on souhaite ajouter et les UVs qui composent actuellement la grille. Il s'agit de tenir compte des contraintes de compatibilité entre les UVs actuelles de la grille et la nouvelle UV pour déterminer s'il existe une solution réalisable ou non, relativement à la possibilité de construire un emploi du temps individuel pour un étudiant effectuant des choix d'UVs. Dans la négative, il faudra être en mesure de maximiser la réalisation des compatibilités (ou minimiser les violations de ces contraintes).

Le deuxième problème que nous avons étudié est le problème de planification d'équipes de pompiers FFTP de l'institution INFOCA pour lequel les équipes de pompiers doivent être programmées au cours de la période à haut risque de feux de forêt. L'enjeu est d'obtenir une capacité opérationnelle maximale en considérant les contraintes qui permettent d'atteindre cette capacité opérationnelle, en plus des contraintes liées à la réglementation du travail.

Nous avons présenté un modèle ILP, une matheuristique ILPH et une métaheuristique ALNS pour aborder le FFTP d'INFOCA. Les approches proposées ont été testées à l'aide de quatre datasets de taille croissante que nous avons générés à partir de données réelles. L'ALNS a obtenu tous les résultats optimaux obtenus en utilisant le modèle ILP sur les plus petites instances. Pour les plus grandes instances, les résultats sont de meilleure qualité par rapport à ceux obtenus par la matheuristique ILPH, de plus les temps de traitement sont significativement réduits.

Une future direction de recherche serait de considérer un problème plus large en intégrant des objectifs et des contraintes supplémentaires, par exemple en considérant le planning des opérations de maintenance préventive des ressources matérielles, celles-ci peuvent impacter la disponibilité de certaines ressources comme le nombre de certains types de quarts par jour.

Bibliographie

Abdullah, Salwani and Hamza Turabieh (2012). "On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems". In: *information sciences* 191, pp. 146–168 (cited on p. 27).

Abdullah, Salwani, Hamza Turabieh, Barry McCollum, and Paul McMullan (2010). "A multi-objective post enrolment course timetabling problems: a new case study". In: *IEEE congress on evolutionary computation*. IEEE, pp. 1–7 (cited on p. 36).

Achá, Roberto Asín and Robert Nieuwenhuis (2014). "Curriculum-based course timetabling with SAT and MaxSAT". In: *Annals of Operations Research* 218.1, pp. 71–91 (cited on p. 24).

Afshar-Nadjafi, Behrouz (2021). "Multi-skilling in scheduling problems: A review on models, methods and applications". In: *Computers & Industrial Engineering* 151, p. 107004. DOI: 10.1016/j.cie.2020.107004 (cited on p. 86).

Aggarwal, Sumer C (1982). "A focussed review of scheduling in services". In: *European Journal of Operational Research* 9.2, pp. 114–121 (cited on pp. 12, 83).

Aickelin, Uwe and Kathryn A Dowsland (2004). "An indirect genetic algorithm for a nurse-scheduling problem". In: *Computers & operations research* 31.5, pp. 761–778 (cited on p. 94).

Asta, Shahriar, Ender Özcan, and Tim Curtois (2016). "A tensor based hyper-heuristic for nurse rostering". In: *Knowledge-based systems* 98, pp. 185–199 (cited on p. 95).

Awadallah, Mohammed A, Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, Asaju La'aro Bolaji, and Mahmud Alkoffash (2017). "Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem". In: *Neural Computing and Applications* 28.3, pp. 463–482 (cited on p. 94).

Barnhart, Cynthia, Amy M Cohn, Ellis L Johnson, Diego Klabjan, George L Nemhauser, and Pamela H Vance (2003). "Airline crew scheduling". In: *Handbook of transportation science*. Springer, pp. 517–560 (cited on pp. 14, 83).

Bell, Peter, Genevieve Hay, and Y Liang (1986). "A visual interactive decision support system for workforce (nurse) scheduling". In: *INFOR: Information Systems and Operational Research* 24.2, pp. 134–145 (cited on p. 90).

Bellio, Ruggero, Sara Ceschia, Luca Di Gaspero, Andrea Schaerf, and Tommaso Urli (2016). "Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem". In: *Computers & Operations Research* 65, pp. 83–92 (cited on p. 26).

Bergh, Jorne Van den, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck (2013). "Personnel scheduling: A literature review". In: *European journal of operational research* 226.3, pp. 367–385 (cited on pp. 83, 86).

Bixby, Robert E (2012). "A brief history of linear and mixed-integer programming computation". In: *Documenta Mathematica*, pp. 107–121 (cited on p. 27).

Bonutti, Alex, Fabio De Cesco, Luca Di Gaspero, and Andrea Schaerf (2012). "Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results". In: *Annals of Operations Research* 194.1, pp. 59–70 (cited on pp. 18, 22).

Broek, JJJ Van den and Cor AJ Hurkens (2012). "An IP-based heuristic for the post enrolment course timetabling problem of the ITC2007". In: *Annals of Operations Research* 194.1, pp. 439–454 (cited on p. 33).

Brucker, Peter, Rong Qu, and Edmund Burke (2011). "Personnel scheduling: Models and complexity". In: *European Journal of Operational Research* 210.3, pp. 467–473 (cited on p. 86).

Burke, Edmund, Peter Cowling, Patrick De Causmaecker, and Greet Vanden Berghe (2001). "A memetic approach to the nurse rostering problem". In: *Applied intelligence* 15.3, pp. 199–214 (cited on pp. 92, 94).

Burke, Edmund, Patrick De Causmaecker, and Greet Vanden Berghe (1998). "A hybrid tabu search algorithm for the nurse rostering problem". In: *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, pp. 187–194 (cited on p. 93).

Burke, Edmund, Patrick De Causmaecker, Sanja Petrovic, and Greet Vanden Berghe (2003). "Variable neighborhood search for nurse rostering problems". In: *Metaheuristics: computer decision-making*. Springer, pp. 153–172 (cited on p. 92).

Burke, Edmund K, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem (2004). "The state of the art of nurse rostering". In: *Journal of scheduling* 7.6, pp. 441–499 (cited on pp. 15, 84–86, 91).

Burke, Edmund K, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu (2013). "Hyper-heuristics: A survey of the state of the art". In: *Journal of the Operational Research Society* 64.12, pp. 1695–1724 (cited on p. 95).

Burke, Edmund K, Jakub Mareček, Andrew J Parkes, and Hana Rudová (2010a). "A supernodal formulation of vertex colouring with applications in course timetabling". In: *Annals of Operations Research* 179.1, pp. 105–130 (cited on p. 18).

— (2010b). "Decomposition, reformulation, and diving in university course timetabling". In: *Computers & Operations Research* 37.3, pp. 582–597 (cited on pp. 22, 23).

— (2012). "A branch-and-cut procedure for the udine course timetabling problem". In: *Annals of Operations Research* 194.1, pp. 71–87 (cited on p. 23).

Cacchiani, Valentina, Alberto Caprara, Roberto Roberti, and Paolo Toth (2013). "A new lower bound for curriculum-based course timetabling". In: *Computers & Operations Research* 40.10, pp. 2466–2477 (cited on p. 24).

Cambazard, Hadrien, Emmanuel Hebrard, Barry O'Sullivan, and Alexandre Papadopoulos (2012). "Local search and constraint programming for the post enrolment-based course timetabling problem". In: *Annals of Operations Research* 194.1, pp. 111–135 (cited on p. 35).

Ceschia, Sara, Nguyen Thi Thanh Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf (2015). "Second International Nurse Rostering Competition (INRC-II)—Problem Description and Rules—". In: *arXiv preprint arXiv:1501.04177* (cited on p. 85).

Ceschia, Sara, Luca Di Gaspero, and Andrea Schaerf (2012). "Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem". In: *Computers & Operations Research* 39.7, pp. 1615–1624 (cited on p. 34).

Chandrasekharan, Reshma Chirayil, Pieter Smet, and Tony Wauters (2021). "An automatic constructive matheuristic for the shift minimization personnel task scheduling problem". In: *Journal of Heuristics* 27.1-2, pp. 205–227. DOI: 10.1007/s10732-020-09439-9 (cited on pp. 87, 88).

Curtois, Tim (2014). *Employee Shift Scheduling Benchmark Data Sets*. URL: http://www.schedulingbenchmarks.org (cited on p. 86).

De Bruecker, Philippe, Jeroen Beliën, Jorne Van den Bergh, and Erik Demeulemeester (2018). "A three-stage mixed integer programming approach for optimizing the skill mix and training schedules for aircraft maintenance". In: *European Journal of Operational Research* 267.2, pp. 439–452 (cited on p. 106).

De Bruecker, Philippe, Jorne Van den Bergh, Jeroen Beliën, and Erik Demeulemeester (2015). "Workforce planning incorporating skills: State of the art". In: *European Journal of Operational Research* 243.1, pp. 1–16. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2014.10.038. URL: https://www.sciencedirect.com/science/article/pii/S0377221714008601 (cited on p. 86).

Demirović, Emir, Nysret Musliu, and Felix Winter (2019). "Modeling and solving staff scheduling with partial weighted maxSAT". In: *Annals of Operations Research* 275.1, pp. 79–99 (cited on p. 87).

Di Gaspero, Luca, Barry McCollum, and Andrea Schaerf (2007). *The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3)*. Tech. rep. Citeseer (cited on pp. 17, 18, 22).

Di Gaspero, Luca and Andrea Schaerf (2002). "Multi-neighbourhood local search with application to course timetabling". In: *International Conference on the Practice and Theory of Automated Timetabling*. Springer, pp. 262–275 (cited on p. 25).

Dueck, Gunter (1993). "New optimization heuristics: The great deluge algorithm and the record-to-record travel". In: *Journal of Computational physics* 104.1, pp. 86–92 (cited on pp. 70, 74, 110, 113).

Edie, Leslie C (1954). "Traffic delays at toll booths". In: *Journal of the operations research society of America* 2.2, pp. 107–138 (cited on p. 12).

Er-rhaimini, Karim (2019). "Forest growth optimization for solving timetabling problems". In: *Proceedings of the International Timetabling Competition* (cited on p. 41).

Ernst, Andreas T, Houyuan Jiang, Mohan Krishnamoorthy, Bowie Owens, and David Sier (2004a). "An annotated bibliography of personnel scheduling and rostering". In: *Annals of Operations Research* 127.1, pp. 21–144 (cited on pp. 83, 86).

Ernst, Andreas T, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier (2004b). "Staff scheduling and rostering: A review of applications, methods and models". In: *European journal of operational research* 153.1, pp. 3–27 (cited on pp. 7, 12, 13, 83, 86).

Fages, Jean-Guillaume and Tanguy Lapègue (2014). "Filtering AtMostNValue with difference constraints: Application to the shift minimisation personnel task scheduling problem". In: *Artificial Intelligence* 212, pp. 116–133 (cited on p. 86).

Gashi, Edon, Kadri Sylejmani, and Adrian Ymeri (2021). "Simulated annealing with penalization for university course timetabling". In: *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling PATAT*. Vol. 2, pp. 361–366 (cited on p. 41).

Glover, Fred and Manuel Laguna (1998). "Tabu search". In: *Handbook of combinatorial optimization*. Springer, pp. 2093–2229 (cited on p. 93).

Goh, Say Leng, Graham Kendall, and Nasser R Sabar (2017). "Improved local search approaches to solve the post enrolment course timetabling problem". In: *European Journal of Operational Research* 261.1, pp. 17–29 (cited on p. 35).

Guerriero, Francesca and Rosita Guido (2022). "Modeling a flexible staff scheduling problem in the Era of Covid-19". In: *Optimization Letters* 16.4, pp. 1259–1279 (cited on pp. 87, 88).

Hadwan, Mohammed and Masri Ayob (2010). "A constructive shift patterns approach with simulated annealing for nurse rostering problem". In: *2010 International Symposium on Information Technology*. Vol. 1. IEEE, pp. 1–6 (cited on p. 93).

Hansen, Pierre, Nenad Mladenović, Jack Brimberg, and José A Moreno Pérez (2019). "Variable neighborhood search". In: *Handbook of metaheuristics*. Springer, pp. 57–97 (cited on pp. 91, 92).

Hao, Jin-Kao and Una Benlic (2011). "Lower bounds for the ITC-2007 curriculum-based course timetabling problem". In: *European Journal of Operational Research* 212.3, pp. 464–472 (cited on p. 24).

Haspeslagh, Stefaan, Patrick De Causmaecker, Andrea Schaerf, and Martin Stølevik (2014). "The first international nurse rostering competition 2010". In: *Annals of Operations Research* 218.1, pp. 221–236 (cited on p. 85).

Heil, Julia, Kirsten Hoffmann, and Udo Buscher (2020). "Railway crew scheduling: Models, methods and applications". In: *European journal of operational research* 283.2, pp. 405–425. DOI: 10.1016/j.ejor.2019.06.016 (cited on p. 86).

Heus, Kamel (1996). "Gestion des plannings infirmiers: Application des techniques de programmation par contraintes". PhD thesis. Grenoble 1 (cited on p. 89).

Hoffmann, Kirsten and Udo Buscher (2019). "Valid inequalities for the arc flow formulation of the railway crew scheduling problem with attendance rates". In: *Computers & Industrial Engineering* 127, pp. 1143–1152. DOI: 10.1016/j.cie.2018.05.031 (cited on pp. 87, 88).

Hojati, Mehran (2018a). "A greedy heuristic for shift minimization personnel task scheduling problem". In: *Computers & Operations Research* 100, pp. 66–76 (cited on p. 86).

— (2018b). "A greedy heuristic for shift minimization personnel task scheduling problem". In: *Computers & Operations Research* 100, pp. 66–76. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2018.07.010. URL: https://www.sciencedirect.com/science/article/pii/S0305054818301941 (cited on pp. 87, 91).

Holm, Dennis S, Rasmus Ø Mikkelsen, Matias Sørensen, and Thomas R Stidsen (2019). "A mip based approach for international timetabling competation 2019". In: *Proceedings of the International Timetabling Competition* 2020 (cited on p. 40).

Hussain, Kashif, Mohd Najib Mohd Salleh, Shi Cheng, and Yuhui Shi (2019). "Metaheuristic research: a comprehensive survey". In: *Artificial intelligence review* 52.4, pp. 2191–2233 (cited on pp. 69, 108).

IBM (2020). *CPLEX User's Manual*. URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.10.0/COS_KC_home.html (cited on pp. 74, 113).

Jaengchuea, Sawaphat and Dome Lohpetch (2015). "A hybrid genetic algorithm with local search and tabu search approaches for solving the post enrolment based course timetabling problem: outperforming guided search genetic algorithm". In: *2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)*. IEEE, pp. 29–34 (cited on pp. 34, 35).

Kheiri, Ahmed, Ender Ozcan, Rhydian Lewis, and Jonathan Thompson (2016). "A sequence-based selection hyper-heuristic: A case study in nurse rostering". In: (cited on p. 95).

Kiefer, Alexander, Richard F Hartl, and Alexander Schnell (2017). "Adaptive large neighborhood search for the curriculum-based course timetabling problem". In: *Annals of Operations Research* 252.2, pp. 255–282 (cited on p. 27).

Kirkpatrick, Scott, C Daniel Gelatt, and Mario P Vecchi (1983). "Optimization by simulated annealing". In: *science* 220.4598, pp. 671–680 (cited on p. 93).

Kletzander, Lucas and Nysret Musliu (2020). "Solving the general employee scheduling problem". In: *Computers & Operations Research* 113, p. 104794. DOI: 10.1016/j.cor.2019.104794 (cited on pp. 87, 93).

Kostreva, Michael M and Karen SB Jennings (1991). "Nurse scheduling on a microcomputer". In: *Computers & operations research* 18.8, pp. 731–739 (cited on p. 91).

Krishnamoorthy, Mohan, Andreas T Ernst, and Davaatseren Baatar (2012). "Algorithms for large scale shift minimisation personnel task scheduling problems". In: *European Journal of Operational Research* 219.1, pp. 34–48 (cited on p. 86).

Kuhn, Harold W (1955). "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2, pp. 83–97 (cited on p. 35).

Lach, Gerald and Marco E Lübbecke (2012). "Curriculum based course timetabling: new solutions to Udine benchmark instances". In: *Annals of Operations Research* 194.1, pp. 255–272 (cited on pp. 24, 25, 27).

Lapègue, Tanguy, Odile Bellenguez-Morineau, and Damien Prot (2013). "A constraint-based approach for the shift design personnel task scheduling problem with equity". In: *Computers & Operations Research* 40.10, pp. 2450–2465 (cited on p. 86).

Legrain, Antoine, Jérémy Omer, and Samuel Rosat (2019). "A rotation-based branch-and-price approach for the nurse scheduling problem". In: *Mathematical Programming Computation*, pp. 1–34 (cited on p. 88).

Lemos, Alexandre, Pedro T Monteiro, and Inês Lynce (2021). "ITC 2019: University Course Timetabling with MaxSAT". In: *Practice and Theory of Automated Timetabling* 1 (cited on p. 40).

Lewis, Rhyd (2012). "A time-dependent metaheuristic algorithm for post enrolment-based course timetabling". In: *Annals of Operations Research* 194.1, pp. 273–289 (cited on p. 34).

Lewis, Rhydian, Ben Paechter, and Barry McCollum (2007). "Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition". In: (cited on pp. 10, 17, 28, 30).

Lindahl, Michael, Matias Sørensen, and Thomas R Stidsen (2018). "A fix-and-optimize matheuristic for university timetabling". In: *Journal of Heuristics* 24.4, pp. 645–665 (cited on p. 27).

Lü, Zhipeng and Jin-Kao Hao (2009). "A critical element-guided perturbation strategy for iterated local search". In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, pp. 1–12 (cited on p. 26).

— (2010). "Adaptive tabu search for course timetabling". In: *European Journal of Operational Research* 200.1, pp. 235–244 (cited on p. 26).

Mara, Setyo Tri Windras, Rachmadi Norcahyo, Panca Jodiawan, Luluk Lusiantoro, and Achmad Pratama Rifai (2022). "A survey of adaptive large neighborhood search algorithms and applications". In: *Computers & Operations Research*, p. 105903 (cited on pp. 69, 108).

Méndez-Díaz, Isabel, Paula Zabala, and Juan José Miranda-Bront (2016). "An ILP based heuristic for a generalization of the post-enrollment course timetabling problem". In: *Computers & Operations Research* 76, pp. 195–207 (cited on p. 32).

Müller, Tomáš (2009). "ITC2007 solver description: a hybrid approach". In: *Annals of Operations Research* 172.1, p. 429 (cited on pp. 27, 36).

Müller, Tomáš, Hana Rudová, Zuzana Müllerová, et al. (2018). "University course timetabling and international timetabling competition 2019". In: *Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2018)*. Vol. 1, pp. 5–31 (cited on p. 37).

Nothegger, Clemens, Alfred Mayer, Andreas Chwatal, and Günther R Raidl (2012). "Solving the post enrolment course timetabling problem by ant colony optimization". In: *Annals of Operations Research* 194.1, pp. 325–339 (cited on p. 37).

Okada, Mihoko and Masahiko Okada (1988). "Prolog-based system for nursing staff scheduling implemented on a personal computer". In: *Computers and Biomedical Research* 21.1, pp. 53–63 (cited on p. 89).

Ouberkouk, Mohamed-Amine, Jean-Paul Boufflet, and Aziz Mourkim (2021). "Adaptive iterative destruction construction heuristic for the firefighters timetabling problem".

In: *8th International Conference on Metaheuristics and Nature Inspired Computing* (cited on pp. 101, 121).

Porto, Andrés Felipe, César Augusto Henao, Héctor A. López-Ospina, and Esneyder Rafael González (2019). "Hybrid flexibility strategy on personnel scheduling: Retail case study". In: *Comput. Ind. Eng.* 133, pp. 220–230. DOI: 10.1016/j.cie.2019.04.049 (cited on pp. 87, 88).

Pour, Shahrzad M, John H Drake, Lena Secher Ejlertsen, Kourosh Marjani Rasmussen, and Edmund K Burke (2018). "A hybrid constraint programming/mixed integer programming framework for the preventive signaling maintenance crew scheduling problem". In: *European Journal of Operational Research* 269.1, pp. 341–352 (cited on pp. 87, 106).

Qin, Ruwen, David A. Nembhard, and Walter L. Barnes II (2015). "Workforce flexibility in operations management". In: *Surveys in Operations Research and Management Science* 20.1, pp. 19–33. ISSN: 1876-7354. DOI: https://doi.org/10.1016/j.sorms.2015.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1876735415000033 (cited on p. 86).

Rajeswari, M, J Amudhavel, Sujatha Pothula, and P Dhavachelvan (2017). "Directed bee colony optimization algorithm to solve the nurse rostering problem". In: *Computational intelligence and neuroscience* 2017 (cited on p. 94).

Rappos, Efstratios, Eric Thiémard, Stephan Robert, and Jean-François Hêche (2022). "A mixed-integer programming approach for solving university course timetabling problems". In: *Journal of Scheduling*, pp. 1–14 (cited on p. 41).

Santos, Haroldo G, Túlio AM Toffolo, Rafael AM Gomes, and Sabir Ribas (2016). "Integer programming techniques for the nurse rostering problem". In: *Annals of Operations Research* 239.1, pp. 225–251 (cited on pp. 87, 106).

Smet, Pieter, Tony Wauters, Mihail Mihaylov, and Greet Vanden Berghe (2014). "The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights". In: *Omega* 46, pp. 64–73 (cited on p. 86).

Smith, L Douglas and A Wiggins (1977). "A computer-based nurse scheduling system". In: *Computers & Operations Research* 4.3, pp. 195–212 (cited on p. 90).

Socha, Krzysztof, Joshua Knowles, and Michael Sampels (2002). "A max-min ant system for the university course timetabling problem". In: *International Workshop on Ant Algorithms*. Springer, pp. 1–13 (cited on p. 36).

Tadumadze, Giorgi, Nils Boysen, Simon Emde, and Felix Weidinger (2019). "Integrated truck and workforce scheduling to accelerate the unloading of trucks". In: *European Journal of Operational Research* 278.1, pp. 343–362. DOI: 10.1016/j.ejor.2019.04.024 (cited on p. 87).

Tien, James M and Angelica Kamiyama (1982). "On manpower scheduling algorithms". In: *SIAM review* 24.3, pp. 275–287 (cited on pp. 12, 83).

Václavík, Roman, Přemysl Šcha, and Zdeněk Hanzálek (2016). "Roster evaluation based on classifiers for the nurse rostering problem". In: *Journal of Heuristics* 22.5, pp. 667–697 (cited on p. 95).

Valouxis, Christos, Christos Gogos, George Goulas, Panayiotis Alefragis, and Efthymios Housos (2012). "A systematic two phase approach for the nurse rostering problem". In: *European Journal of Operational Research* 219.2, pp. 425–433 (cited on p. 92).

Zucchi, Giorgio, Manuel Iori, and Anand Subramanian (2021). "Personnel scheduling during Covid-19 pandemic". In: *Optimization Letters* 15.4, pp. 1385–1396 (cited on pp. 87, 88).

Liste des tableaux

Liste des figures

List of Abbreviations

AIDCH	**Adaptive Iterative Construction Destruction Heuristic**
ALNS	**Adaptive Large Neighborhood Search**
ARPE	**Average Relative Pourcentage Error**
BIA	**Best Insertion Algorithm**
BIDC	**Best Insertion Destruction Criterion**
CB-CTT	**Curriculum Based Course TimeTabling**
CMH	**Constructive Mathematical Heuristic**
CSP	**Constraint Satisfaction Problem**
FFTP	**FireFighters Timetabling Problem**
GA	**Genetic Algorithm**
HH	**Hyper-Heuristic**
HMM	**Hidden Markov Model**
HSA	**Harmonic Search Algorithm**
ILP	**Integer Linear Programming**
ILPH	**Integer Linear Programming Heuristic**
INRC	**International Nurse Rostering Competition**
ITC	**International Timetabling Competition**
LP	**Linear Programming**
MA	**Memetic Algorithm**
NRP	**Nurse Rostering Problem**
PECT	**Post Enrollement Course Timetabling**
PSP	**Personnel Scheduling Problem**

RD	**Random Destruction**
RMD	**RooM Destruction**
RPE	**Relative Pourcentage Error**
RPG	**Relative Pourcentage Gap**
SA	**Simulated Annealing**
SD	**Smart Destruction**
SDPTSP-E	**Shift Designand Personnel Task Scheduling Problem with Equity objective**
SH	**Shuffling Heuristic**
SMPTSP	**Shift Minimization Personnel Task Scheduling Problem**
TCD	**TeaCher Destruction**
TS	**Tabu Search**
UCTTP	**University Course TimeTabling Problem**
UTC-UCTTP	**University Course TimeTabling Problem of UTC**
VND	**Variable Neighborhood Decent**
VNS	**Variable Neighborhood Search**