Université Paris 6 Pierre et Marie Curie
UFR d'Informatique

Rapport de stage en vue de l'obtention du diplôme
Master Intelligence Artificielle et Décision
de l'université Paris 6

Optimisation de tournées de véhicules pour l'exploitation de Réseau Telecom

Arnaud Malapert
<malapert.arnaud@gmail.com>

Tuteur France Telecom : Cedric Chamayou & Matthieu Chardy
Tuteur universitaire : Safia Kedad-Sidhoum

France Télécom
Fonction Groupe Technologie & Innovation
France Télécom R&D
38-40, rue du Général-Leclerc - 92794 Issy Moulineaux Cedex 9
Téléphone : 01 45 29 44 44
Téléphone international : + 33 1 45 29 44 44

SA au capital de 8 926 860 308 - 380 129 866 RCS Paris

Paris, 6 septembre 2006

REMERCIEMENTS

Ce stage a été réalisé dans le cadre du Master 2 IAD (Intelligence Artificielle et Décision) de l'université Pierre et Marie Curie au sein du centre de recherche de France Télécom.

Je remercie la division R&D de m'avoir accueilli au sein du laboratoire « Architecture Optimisation Coût » (CORE/AOC) durant ce stage ainsi que Bertrand Decocq responsable de l'Unité de Recherche Developpement CORE/M2V/AOC.

Je remercie l'université Pierre et Marie Curie de m'avoir permis de réaliser ce stage et plus particulièrement Safia Kedad-Sidhoum, mon encadrante universitaire, pour sa disponibilité et l'attention qu'elle a porté à mon travail.

Je remercie mon encadrant Cedric Chamayou de m'avoir permis de réaliser ce stage ainsi que de son aide et de ses conseils avisés.

Je remercie aussi Matthieu Chardy d'avoir été, avec Cedric Chamayou, mon co-encadrant, ainsi que Michael Fanini, son stagiaire, avec qui j'ai travaillé dans de très bonnes conditions et une très bonne ambiance.

Je remercie toutes les personnes des équipes TECH/LEI et CORE/AOC de m'avoir accueilli si chaleureusement.

Enfin, je tiens à remercier Marie Toumanoff de m'avoir aider pour la correction de ce rapport de stage.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES TABLEAUX

INTRODUCTION

L'optimisation des modèles économiques et de la chaîne logistique sont des enjeux industriels importants du XXIème siècle : dans un contexte de concurrence accrue, disposer d'un chaîne logistique efficiente permettant une réduction des coûts et satisfaction client importante est un atout. C'est pourquoi de nombreux industriels se dotent (par achat ou développement interne) d'outils d'aide à la décision intégrant des méthodes de recherche opérationnelle pour couvrir des problèmes de logistique : problèmes de transport, problèmes d'ordonnancement, problèmes de tournées de véhicules ...
A ce sujet, on pourra noter que la communauté française de recherche opérationnelle permet aux industriels de soumettre une problématique de recherche à l'ensemble de la communauté scientifique via le challenge ROADEF : cette année France Telecom a proposé une problématique de planification-ordonnancement d'interventions pour la maintenance de ses réseaux coeur.

Dans le cadre des télécommunications, le déploiement des réseaux et services ainsi que leur maintenance nécessite, pour France Telecom, l'implication de plusieurs milliers de techniciens dans la réalisation de plusieurs centaines de milliers d'interventions par an. La problématique générique qui se pose est de définir une planification de ces interventions et une affectation de ses techniciens à ces interventions qui permettent de minimiser les coûts opérationnels et maximiser la satisfaction de ses clients. Cette problématique peut se formaliser comme un problème de tournées de véhicules.

Durant ce stage, nous nous sommes intéressés à plusieurs problématiques de tournées. Nous avons d'abord adressé une problématique issue de la réalité terrain et plus précisément rencontrée dans le cadre de la gestion des Unités d'Intervention Clients, problématique à partir de laquelle nous avons défini et validé plusieurs modèles linéaires en nombres entiers et testé des méthodes de résolution exactes (via solveur commercial).

Dans un deuxième temps ; nous avons focalisé sur une problématique plus simple en termes de modèle (correspondant en fait à un sous-ensemble de produits déployées par l'opérateur) sur laquelle nous avons développé des heuristiques de résolution. Ces approches ont été conçues et développées avec le souci de pouvoir être adaptées et étendues à la problématique plus générale traitée en première partie de stage.

Ce manuscrit se termine par les perspectives offertes par le travail réalisé et l'expérience acquise au cours de ce stage.

CHAPITRE 1

CONTEXTE GÉNÉRAL DU STAGE

1.1 Le groupe France Telecom et sa division de Recherche et Développement

Le groupe France Telecom représente un des plus importants opérateurs de télécommunications mondiaux, que ce soit en terme de services (téléphonie fixe et mobile, internet, ...) ou de couverture (présence sur tous les continents). Le groupe possède une division de Recherche et Développement très importante pour s'adapter au mieux au milieu changeant des télécommunications et répondre à ses besoins d'innovation.

C'est dans cette division, au sein de l'équipe CORE/M2V/AOC sous la responsabilité de Cedric Chamayou que s'est déroulé le présent stage.

1.2 CORE/M2V/AOC

1.2.1 Le Centre de Recherche Développement « Coeur de réseau » (CORE)

Présentation générale

Le CRD "Coeur de Réseau" (CORE) a la responsabilité de :
- définir, dans une vision d'opérateur intégré, l'évolution de l'architecture des réseaux, en particulier pour la convergence des réseaux coeur, le multiservices et le haut débit, la VoIP, la sécurité et la QOS des réseaux support,
- assurer les développements du coeur de réseau fixe et mobile en maintenant une cohérence et un urbanisme favorisant l'intégration des services, pour la voix et les données (transport, collecte et longue distance),
- identifier les ruptures potentielles des nouvelles technologies en coeur de réseau, notamment du point de vue économique.

Les enjeux

Les enjeux financiers associés aux réseaux de France Télécom sont très significatifs, à la fois en patrimoine et en coûts opérationnels. Les échelles de temps pour l'évolution des réseaux sont longues. Le CRD CORE doit donc faire en sorte que les réseaux de France Télécom évoluent afin d'être les fournisseurs des produits intégrés du Groupe, par une anticipation des points bloquants et une défense permanente de leur compétitivité. La maîtrise des réseaux est au coeur des enjeux de l'opérateur intégré : devant une demande et une offre de services de plus en plus complexes, France Télécom vise à la convergence des services fixes, mobiles et Internet dans trois environnements : personnel, domestique et entreprise.

Le trafic voix circuit est en baisse, au profit du trafic VoIP. Les services de données explosent avec la mise à disposition d'accès ADSL haut débit. De nouveaux services utilisant le réseau de transport IP émergent (VoD, TV/ADSL, peer to peer, etc). Les aspects sécurité (protection du réseau coeur contre

des attaques, authentification des clients, confidentialité/intégrité des données) et qualité de service deviennent primordiaux.

Les technologies disponibles industriellement foisonnent tant au niveau de l'accès (xDSL, radio, FTTP,..), que du transport (brassage optique, SDH, WDM, ATM, IP,..), du support des services (NGN, IMS,..) avec des coûts différents (Capex (investissements corporels ou incorporels) et Opex (coûts d'exploitation nécessaires au fonctionnement d'une entreprise)) et des architectures induites différentes.

Les contraintes réglementaires imposent l'ouverture de certains points du réseau, avec des tarifs réglementés basés sur des coûts théoriques d'une architecture et technologie récente.

Une concurrence à tous les niveaux (retail et wholesale) impose de garantir à chaque niveau, la performance économique avec la qualité de service requise. Dans ce contexte, l'enjeu pour France Télécom est de maîtriser ses réseaux, en France et à l'étranger, et d'en gérer la complexité pour :

- concevoir des principes solides d'architecture de réseau pour tenir compte de l'ensemble des paramètres de façon pérenne,
- améliorer la réactivité des développements de nouveaux services (time to market), en fournissant aux Unités d'Affaires et filiales les "API" réseau nécessaires à leur développement et leur exécution,
- assurer la sécurité des réseaux,
- optimiser le dimensionnement des réseaux pour transporter de façon optimale économiquement et qualitativement le trafic.

L'organisation

Le CRD "Coeur de Réseau" de la Division Recherche & Développement, sous la responsabilité de Roberto Kung, réunit plus de 500 personnes sur 3 sites différents et se compose de :

- 6 laboratoires de R&D structurés et tournés principalement vers la fourniture des architectures réseau et des équipements associés pour l'opérateur intégré,
- 4 fonctions domaines pour la gestion des portefeuilles de projets,
- 3 pôles de recherche,
- 3 fonctions de pilotage sur la stratégie, les opérations et l'international,
- la responsabilité technique de 2 Initiatives Transverses de Croissance (ITC) et de 2 chantiers.

1.2.2 Le laboratoire « Multimedia networks for conversational fixed/mobile services : Voice, Video » (M2V)

Le laboratoire M2V (Multimedia networks for conversational fixed/mobile services : Voice, Video) dirigé par Alain Henry, a pour mission de définir, sélectionner, évaluer, valider et intégrer les systèmes réseaux nécessaire aux services conversationnels fixes et mobiles voix et vidéo-téléphonie. A ce titre, le laboratoire développe pour le groupe France Télécom les expertises nécessaires à l'évolution des réseaux fixes et mobiles vers l'IP pour ces services. Il contribue à la construction des futures architectures réseaux convergentes de l'opérateur intégré basées sur les concepts du NGN et de l'IMS, en s'assurant plus particulièrement de la robustesse et la sécurité des équipements ainsi que de l'optimisation économique des réseaux.

1.2.3 L'unité de Recherche Développement « Architecture Optimisation Coût » (AOC)

La mission de l'URD AOC est l'optimisation des coûts liés aux architectures. Elle comporte deux axes principaux :

- des études technico-économiques pour aider la branche réseaux à évaluer l'impact des différentes architectures de service.
- L'anticipation sur les réseaux à venir et recherche sur ces sujets.

AOC s'occupe entre autres de la réductions des coûts associés à l'entretien du réseau (OPEX) ou de problème de localisation d'éléments du réseau (CAPEX).

1.3 Description du stage

1.3.1 Le « sujet » du stage

La maîtrise de la chaîne logistique constitue un élément déterminant pour le succès d'un fournisseur de services de télécommunications. Un pilotage réactif et priorisé des opérations de livraison de services ainsi qu'une gestion optimisée des ressources disponibles assurant ces opérations sont nécessaires pour l'amélioration de la chaîne logistique et celle de la productivité. Ces exemples de problématiques concrets auxquels un opérateur de télécommunication est confronté s'inscrivent dans le cadre général des problématiques de management dynamique des ressources et de leur optimisation. C'est dans cette optique que France Telecom s'intéresse au probléme de la gestion des interventions pour l'exploitation des réseaux (réparation, maintenance). Cependant, ce stage a été réalisé en collaboration avec un autre CRD TECH qui s'intéresse à la même problématique mais dans le cadre de la gestion des interventions des unités d'intervention chez le client.

La division de Recherche et Développement de France Télécom souhaite mettre au point des modèles et des approches de résolution pour cette problématique. Ce stage s'inscrit dans la perspective de résolution de cette problématique : il a pour double objectif la mise au point de techniques de résolution (méthodes exactes et heuristiques) ainsi que le développement d'une maquette logiciel dédiée à la résolution de cette problématique.

1.3.2 Objectifs détaillés du stage

Les objectifs détaillés sont :
- Comprendre le contexte télécom du stage et plus particulièrement l'exploitation des réseaux l'activité de déploiement de services chez le client (pour lequel les données sont plus accessibles).
- Synthétiser les informations reccueillies pour aboutir à une modélisation aussi fine que possible de la problématique de planification des interventions pour le déploiement de services chez le client,
- Situer cette problématique dans l'ensemble des problématiques de planification, affectation, ordonnancement et/ou tournées de véhicules, traitées dans la littérature (état de l'art).
- Développer un outil d'aide à la décision pour la planification des interventions basé sur des méthodes issues de la recherche opérationnelle (utilisation d'un solveur commercial, développement d'heuristiques, ...).

1.3.3 Le déroulement du stage

Un stage en binôme

Le début du stage a été réalisé en binôme avec Michael FANINI, étudiant en Master 2 IMOD à l'université d'Avignon.

Planning du stage

Nous présentons dans cette partie le déroulement du stage et de la mission réalisée :
- Avril : découverte du laboratoire et compréhension de la problèmatique (lecture du travail déjà effectué, visite sur le terrain,...).
- Mai : formalisation de la problématique et établissement d'une bibliographie,
- Juin : Modélisations mathématiques + Implémentation sous CPLEX 10.0 réalisée en binôme,
- Juillet : débuggage et retour sur le modèle + campagne de tests intermédiaire.
- Aout : méthode heuristiques + implémentation +tests

CHAPITRE 2

PROBLÉMATIQUE

2.1 Présentation de la problématique pour France Telecom

2.1.1 Présentation du problème industriel

Dans le cadre des réseaux et télécommunications, France Telecom déploie différents types de réseaux chez ses clients à travers les unités d'intervention client (UIC). Les réseaux déployés peuvent être à destination de différents types de clients. On distingue les technologies pour les entreprises et celles pour la grande diffusion. Notons que ces technologies se découpent selon des types distincts :

- Les produits Multi Marché (en rapport avec la boucle locale) : ADSL, RTC, Dégroupage et Numéris.
- Les produits Affaires (ne se rapportant à la boucle locale) : LL, TDSL, T2 et Interlan.
- Les produits PABX-RP : PABX et Réseaux Locaux.

Les produits Multi Marché sont destinés au grand public. Les produits PABX-RP et Affaires sont quant à eux à destination des entreprises. Il est important de noter que les entreprises ont, bien entendu, accès aux technologies Multi Marché mais ce n'est pas le cas pour le grand public, au niveau des technologies entreprise.

Entité PABX-RP

Dans le cadre de notre étude, nous nous intéressons à un type de technologie entreprise : les PABX et Réseaux Locaux. L'entité PABX-RP a pour rôle de planifier, gérer et réaliser les interventions d'installation ou de modification chez le client (pas de maintenance). Ces interventions concernent un ensemble de technologies :

- Des PABX de marques différentes (Alcatel, Siemens, ...) et de types différents(anciens PABX, grandes installations, nouveaux PABX et périphérie PABX).
- Des réseaux locaux de types différents (RLE, Wifi, ...).

Les interventions d'installation consistent à venir installer entièrement une technologie chez un client. Les interventions de modification, par contre, se réalisent à partir d'une technologie existante et se déclinent en plusieurs types :

- Les adjonctions de postes.
- Les adjonctions de cartes.
- Les déplacements de matériels.
- Les récupérations de matériels.

Globalement, la gestion des interventions à réaliser suit le schéma de la figure 2.1 page 13.

Rôle du vendeur

Tout d'abord, le client prend contact avec le vendeur qui rassemble un certain nombre d'informations :

- Les coordonnées du client.
- La tâche à réaliser.
- La date d'intervention souhaitée par le client.

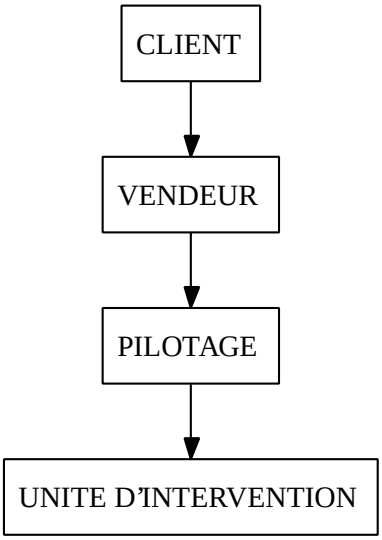

FIG. 2.1 – Gestion des interventions.

- La date contractuelle pour les gros clients ou les grosses interventions (cette date contractuelle n'existe donc pas toujours).
- Le devis en terme de prix du contrat et en nombre d'heures d'intervention (celui-ci n'est pas toujours réalisé par le vendeur).

Ensuite, ces informations sont introduites dans le système d'information de France Telecom.

Rôle du Pilote

Le pilote récupère alors les interventions à réaliser (environ dix par jour), en provenance des vendeurs. Le pilotage est composé de trois responsables d'opérations (RO) qui s'occupent de gérer, superviser et d'assurer le bon déroulement des interventions. Les commandes clients sont récupérées via 2 applications :

- Une pour les interventions dont le devis (temps d'intervention et prix du contrat) est déjà réalisé par le vendeur.
- Une autre pour le reste des interventions.

Chaque jour, les RO ont donc un ensemble de dossiers à traiter afin que les interventions puissent être réalisées par les techniciens. Un dossier est traité par un RO et un seul et correspond à une et une seule activité à réaliser (Exemple : rajout de 10 lignes téléphoniques sur un PABX Alcatel e-diat S/M/L). De manière générale, le traitement d'un dossier se décompose en deux voire trois parties :

- L'établissement du devis si cela n'a pas été fait par le vendeur.
- L'étude du dossier.
- La planification des ressources.

L'étude d'un dossier par un RO consiste à vérifier l'ensemble des informations contenues dans le dossier : vérification des informations et coordonnées du client, prise en compte des besoins nécessaires (en terme de matériel, de technologies, ...). Cette étude prend un temps variable, de l'ordre d'une heure dans le meilleur des cas. Une fois l'étude réalisée, il faut planifier l'assignation des ressources. Cela comprend :

- La commande du matériel nécessaire,
- L'affectation d'un ou deux techniciens à l'intervention.

La commande du matériel se fait via le magasin. Généralement, les commandes sont livrées avec un délai d'une semaine mais il y a des variations. L'affectation des techniciens se fait via une feuille excel. les RO planifient les journées des techniciens en deux parties : le matin et l'après-midi. Les techniciens peuvent donc rendre visite au plus à deux clients différents par jour. De manière générale, les interventions d'installation prennent beaucoup de temps (plusieurs jours) alors que les interventions de modification sont généralement bien plus courtes (de l'ordre de 2h, 4h ou 6h). Ce faisant, un technicien, affecté à

une intervention de production chez un client, s'y rendra plusieurs jours de suite ; alors qu'un technicien affecté à plusieurs interventions de modification, changera globalement de lieu à chaque demi-journée.

Rôle du technicien

Les techniciens récupèrent les interventions à réaliser, en provenance du pilotage. Ils ont chacun une certaine sphère de compétences (cf. documents PABX) correspondant aux technologies et produits qu'ils maîtrisent. On associe à chacune de leurs compétences un chiffre indiquant leur niveau :

- 0 pour ne connaît pas,
- 1 pour débutant,
- 2 pour connaît le produit,
- 3 pour expert.

Outre leurs compétences, les techniciens sont affectés à une zone géographique, toute intervention en dehors de leur zone est exclue.
Les techniciens réalisent donc un ensemble d'interventions chez le client en rapport avec leurs compétences techniques et leur zone géographique tout en respectant les termes de leurs contrats (RTT, congés, ...). Une fois les interventions réalisées, les techniciens le mentionnent au pilotage pour que les RO clôturent le dossier et le fassent passer à la facturation.

2.1.2 Les contraintes opérationnelles

Les contraintes de ce problème dépendent donc des interventions (jobs) et des techniciens (ressources). Nous allons présenter les contraintes que nous devons respecter.

Contraintes sur les interventions

On considère trois types d'interventions différentes, les pré-visites, les interventions courtes et les interventions longues. Les pré-visites ont toutes la même durée et ont lieu lorsque les informations sur une tâche future sont incomplètes. Elles servent à acquérir les informations qui permettront de planifier l'intervention. Les interventions simples ont un temps de réalisation inférieur à une journée, elles doivent être réaliser sans interruption. Les interventions longues s'étendent sur plusieurs jours. Les interventions simples et longues peuvent nécessiter deux techniciens simultanément. On autorise les interruptions sur les jobs longues.
Nous présentons une liste exhaustive des contraintes appliquées aux interventions :

- Les techniciens affectés à une intervention doivent l'exécuter entièrement.
- Certaines interventions sont exécutées par un seul technicien, alors que d'autres doivent être entièrement ou partiellement réalisées par deux techniciens.
- Chaque intervention doit être exécutée dans une fenêtre de temps allant d'une journée à l'ensemble de l'horizon de planification.
- Si une intervention est interrompue, on majore le temps passé à la réalisation d'autres jobs pendant ces interruptions.
- Si une intervention est réalisée sur plusieurs journées, on impose que ces journées soient consécutives.
- Pour certaines interventions, on doit aller chercher du matériel à un dépôt avant de commencer.

Contraintes sur les techniciens

Une ressource est un technicien. On peut remarquer que le nombre maximal de tournées réalisables en une journée est majoré par le nombre de techniciens disponibles ce jour là. Les contraintes sur les ressources permettent de créer des tournées réalisables. Nous présentons les contraintes retenues :

- La durée d'une tournée est limitée en temps, et correspond à la durée de la journée de travail. On peut éventuellement assouplir la contrainte sur la durée d'une journée de travail si on rajoute une contrainte sur la durée hebdomadaire ou sur l'horizon de planification.
- Chaque technicien a un planning défini à priori et les tournées doivent être compatibles avec ce planning.

2.1.3 Objectifs de L'UIC

Le but de notre problématique est donc de réaliser la planification des interventions des techniciens en respectant un ensemble de contraintes et soumise à différents objectifs. Nos objectifs peuvent être de deux ordres :
- La minimisation des coûts.
- La maximisation de la satisfaction client.

Problèmes de réduction des coûts

L'objectif de minimisation des coûts peut se décliner en différents objectifs :
- Minimiser les distances ou les temps de trajet des techniciens.
- Respecter les dates contractuelles pour minimiser les pénalités.
- Minimiser l'hétérogénéité des plans de charge des techniciens.

Problèmes de satisfaction client

L'objectif de maximisation de la satisfaction client peut avoir différents objectifs sous-jacents :
- Réaliser les dates contractuelles et mieux si possible,
- Réaliser au mieux les dates souhaitées des clients,
- Maximiser la qualité des interventions (fonction des niveaux de compétences des techniciens par exemple).

2.1.4 Les problèmes opérationnels

Nous allons maintenant décrire les problèmes opérationnels auxquels nous sommes confrontés. Concernant les vendeurs, on peut rencontrer les problèmes suivants :
- devis non réalisé ou incomplet,
- devis erroné (par exemple, le temps d'intervention peut être mal estimé)

Concernant les RO, on peut rencontrer les problèmes suivants :
- temps de traitement du dossier variable (informations manquantes ou erronées sur le devis)
- délai de livraison du matériel variable

Pour pallier aux différents problèmes énoncés précédemment nous imposons un certain nombre de règles qui modifie la problématique.
- Des tâches de prévisite si le devis n'est pas réalisé par le vendeur ou si les informations recueillies par ce dernier paraissent incomplètes. Ces tâches nous permettrons d'avoir de bonnes estimations en terme de temps pour les interventions à réaliser. Une fois ces prévisites réalisées (si possible au plus tôt) les vrais tâches d'interventions seront réintroduites dans notre problème avec les bonnes informations associées.
- Des interventions longues (productions et grosses modifications) qui prennent plus d'un jour. Des interventions courtes (petites modifications) qui prennent quelques heures et des interventions de prévisites qui sont elles aussi de courtes durées et ne nécessitent pas de compétences spécifiques.
- Le travail des RO sera considéré de manière fixe en terme de temps de traitement. Cet intervalle de temps comprendra l'étude du dossier, l'allocation des ressources, et la commande et la réception du matériel si nécessaire. On pourra éventuellement considérer deux temps différents s'il y a du matériel à commander ou pas.
- La journée de travail est actuellement diviser en demi-journée. Nous abandonnons cette division pour ne prendre en compte que les durées associées aux déplacements et aux jobs. Nous espérons ainsi assouplir la planification en permettant à un technicien de réaliser plusieurs (≥ 2) interventions dans la journée.

2.2 Positionnement de la problématique de l'UIC en terme de recherche opérationnelle

Afin de motiver notre approche et les modèles développés, nous positionnerons notre problématique par rapport à différents problèmes classiques traités en recherche opérationnelle.

Nous pouvons distinguer trois parties dans notre problème ; exécuter les tâches dans un ordre compatible avec les contraintes temporelles (problème d'ordonnancement) ; affecter à chaque job une ressource qui l'exécutera (problème d'ordonnancement avec contraintes de ressource ou problème d'affectation) ; affecter à chaque ressource, une séquence de jobs qui minimise la distance totale parcourue. Nous allons maintenant présenter brièvement chaque type de problème.

Problème d'ordonnancement Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement) et de contraintes portant sur la disponibilité des ressources requises. Un ordonnancement constitue une solution au problème d'ordonnancement. Il est défini par le planning d'exécution des tâches (« calendrier ») et d'allocation des ressources et vise à satisfaire un ou plusieurs objectifs (date de fin du dernier job, retard absolu,...). Dans certains problème d'ordonnancement, des algorithmes polynomiaux existent, mais ce n'est pas le cas pour tous.

Problème d'affectation Un problème d'affectation consiste à associer un job à une ressource de manière unique en fonction d'un objectif. Il peut être ramené à un problème de couplage dans un graphe (un couplage est un ensemble d'arêtes deux à deux non-adjacentes). On sait trouver en temps polynomial un couplage de taille (ou même de poids) maximum.

Problème de tournées de véhicules Les problèmes de tournée de véhicule (Vehicle Routing Problem VRP) correspondent à une classe de problèmes dont le but est de trouver un ensemble de routes (tournées) pour une flotte de véhicules basée dans un ou plusieurs dépôts, afin de satisfaire les demandes d'un ensemble de clients. Le but du VRP est de satisfaire la demande de tous les clients en minimisant le coût de transport avec des tournées débutant et terminant à un dépôt. Il existe divers variantes du VRP avec contraintes de capacité, fenêtres de temps,...
La version basique est NP-difficile et peut se voir comme une généralisation du problème du voyageur de commerce.

Remarque 1 *Notre objectif et certaines contraintes incluent des éléments spatiaux, telle que la distance totale parcourue lors des tournées. Cette particularité est typique des problèmes de tournées de véhicules (Vehicle Routing Problem , VRP) qui améliorent la prise en compte de ces critères par rapport à un modèle orienté ordonnancement.*

En vertu de cette conclusion, nous présenterons un état de l'art sur les problèmes de tournées de véhicules au chapitre 3, puis nous présenterons les modèles linéaires associés à notre problèmeau chapitre 4.

CHAPITRE 3

ÉTAT DE L'ART DES PROBLÈMES DE TOURNÉES DE VÉHICULES

Ce chapitre présente une revue des problèmes de tournées sur les noeuds. Nous présenterons d'abord les problèmes les plus connus, puis nous nous intéresserons aux problématiques proches de celle abordée pendant ce stage.

3.1 Introduction

Le Problème de Tournées de Véhicules (VRP, Vehicle Routing Problem) est un des problèmes d'optimisation combinatoire les plus étudié. Il pose le problème suivant, visiter des clients à partir d'un dépôt et au moyen d'une flotte de véhicules, avec un coût minimal. De nombreuses variantes existent, dont certaines sont détaillées dans les sections suivantes. Historiquement, le VRP est une version étendue du Problème du Voyageur de Commerce (TSP, Traveling Salesman Problem), qui consiste à visiter l'ensemble des clients avec un seul véhicule.

Dans ce chapitre, nous présenterons succinctement le TSP en section 3.2, puis le Vehicle Routing Problem en section 3.3, puis nous détaillerons ensuite certaines extensions mono période du Vehicle Routing Problem, le Capacited VRP en section 3.4 et le VRP avec fenêtres de temps en section 3.5 qui constitue une part importante des problèmes réels de tournées. Nous traiterons également deux problèmes multi-période : le Period Vehicle Routing Problem l'Inventory Routing Problem en section 3.6 et le VRP avec flotte limitée en section 3.7. Enfin, la section 3.9 situera le problème étudié dans le cadre de ce stage par rapport à l'état de l'art présenté dans les sections précédentes.
L'ensemble des notations utilisées dans ce chapitre (tableau 3.1 page 17) : Les problèmes de tournées sont

	Notations
$[a_i, b_i]$	la fenêtre de temps au noeud i.
s_i	le temps de service au noeud i.
t_{ij}	le temps de transport associé à l'arc (i, j).
c_{ij}	le coût associé à l'arc (i, j).
M	un grand nombre.
	Variables
x_{ij}	(TSP) 1 si l'arc (i, j) est dans la solution.
x_{ij}^k	(VRP) 1 si l'arc (i, j) est dans la tournée de la ressource k.
y_i^k	1 si une tournée associée à la ressource k passe par le noeud i.
u_i^k	associée à l'heure d'arrivée du véhicule k au noeud i.

TAB. 3.1 – Notations et variables utilisées dans la modélisation des VRP

très variés, donc la définition de l'objectif aussi. Cependant, on peut proposer une classification simple :

1. Minimiser le coût total de parcours

2. Minimiser la somme des coûts fixes associés à l'utilisation des véhicules
3. Minimiser la somme des coûts fixes et des coûts de parcours

Le critère « coût total de parcours » est généralement équivalent au fait de minimiser la distance totale de parcours. On peut bien évidemment composer ces objectifs.

3.2 Traveling Salesman Problem (TSP)

On connaît mal l'origine exacte du TSP. Cependant, il s'agit d'un des plus vieux problèmes combinatoires. Des mathématiciens s'y sont intéressés depuis le début du vingtième siècle cherchant à apporter une réponse à ce problème. La définition en est simple :

TRAVELING SALESMAN PROBLEM

Données :	Un ensemble de noeuds et d'arêtes munies de coûts.
Trouver :	Un circuit hamiltonien, c'est-à-dire le circuit passant par tous les noeuds une et une seule fois.
Minimisant :	Le coût total de transport lié aux arcs empruntés par le voyageur.

Une formulation a été proposée par (Dantzig *et al.*, 1954). Pour les clients i et j on pose c_{ij} le coût du trajet entre ces deux clients, et x_{ij} une variable binaire indiquant si le trajet est compris dans la solution. Le TSP se modélise alors comme suit :

$$\min z = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} x_{ij}c_{ij} \tag{3.1}$$

sous les contraintes :

$$\sum_{i\in\mathcal{N}} x_{ij} = 1 \qquad \forall j \in \mathcal{N} \tag{3.2}$$

$$\sum_{j\in\mathcal{N}} x_{ij} = 1 \qquad \forall i \in \mathcal{N} \tag{3.3}$$

$$\sum_{i,j\in\mathcal{S}} x_{ij} \leq |\mathcal{S}| - 1 \qquad \forall \mathcal{S} \subset \mathcal{N} \ t.q. \quad 2 \leq |\mathcal{S}| \leq n-2 \tag{3.4}$$

Les contraintes (3.2) et (3.3) assurent que le voyageur entre et sort une seule fois de chaque sommet (conservation des flots). La contrainte (3.4) est une formulation classique pour éviter les sous-tours.

3.3 Vehicle Routing Problem (VRP)

Le VRP constitue une généralisation du TSP à plusieurs voyageurs. D'un point de vue terminologie, on parle de véhicules.

VEHICLE ROUTING PROBLEM

Données :	Un ensemble de noeuds client et d'arêtes (munies de coûts) et une flotte illimitée de véhicules partant d'un unique dépôt.
Trouver :	Un ensemble de routes (tournées de véhicules) recouvrant tous les noeuds client qui partent et reviennent au dépôt.
Minimisant :	Le coût total de transport lié aux arcs empruntés par les véhicules et/ou les coûts fixes associés à l'utilisation des véhicules.

Un modèle a été donné par (Fisher & Jaikumar, 1978) et (Fisher & Jaikumar, 1981) . Il s'agit d'une extension de la formulation du TSP vue précédemment. Pour les n clients et les M véhicules, on définit x_{ij}^k, variable binaire indiquant si le véhicule (la tournée) k effectue le trajet (i, j), et y_i^k, variable binaire indiquant si le véhicule k visite le client i. Voici le modèle étendu :

$$\text{Minimiser } z = \sum_{k\in K}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} x_{ij}^k c_{ij} \tag{3.5}$$

sous les contraintes :

$$\sum_{i>1} y_i^k \leq |J| \times y_0^k \qquad \forall k \in K \tag{3.6}$$

$$\sum_{k\in K} y_i^k = 1 \qquad \forall i \neq 1 \in \mathcal{N} \tag{3.7}$$

$$\sum_{i\in\mathcal{N}} x_{ij}^k = y_j^k \qquad \forall j \in \mathcal{N}, \forall k \in K \tag{3.8}$$

$$\sum_{i\in\mathcal{N}} x_{ij}^k = y_i^k \qquad \forall j \in \mathcal{N}, \forall k \in K \tag{3.9}$$

$$\sum_{i,j\in\mathcal{S}} x_{ij}^k \leq |\mathcal{S}| - 1 \qquad (k = 1,\ldots,M; \mathcal{S} \subset \mathcal{N}\ t.q.\quad 2 \leq |\mathcal{S}| \leq n-2) \tag{3.10}$$

Chaque client doit être visité une fois, ce qui est assuré par la contrainte (3.7). La contrainte (3.6) garantit que chaque tournée passe par le dépôt. Les contraintes (3.8) et (3.9) sont le pendant pour le VRP de (3.2) et (3.3) (on arrive et on part de chez chaque client). Enfin, on retrouve les contraintes d'élimination des sous-tours en (3.10).

3.4 Capacited Vehicle Routing Problem (CVRP)

Il s'agit du même problème que le VRP, à l'exception que chaque véhicule a maintenant une capacité Q et qu'on associe a chaque demande un poids q_i.

CAPACITED VEHICLE ROUTING PROBLEM

Données :	Un ensemble de noeuds client (ayant une demande) et d'arêtes (munies de coûts) et une flotte illimitée de véhicules de capacité uniforme Q partant d'un unique dépôt.
Trouver :	Des tournées de véhicules, satisfaisant chaque demande une et une seule fois et respectant les contraintes de capacité.
Minimisant :	Le coût total de transport lié aux arcs empruntés par les véhicules et/ou les coûts fixes associés à l'utilisation des véhicules.

Le modèle linéaire associé à ce problème est celui du VRP avec une contrainte supplémentaire (3.11) qui assure le respect des contraintes de capacité.

$$\sum_{i\in\mathcal{N}} q_i y_i^k \leq Q \forall k \in K \tag{3.11}$$

3.5 Vehicle Routing Problem with Time Windows (VRPTW)

3.5.1 L'énoncé

Le terme « fenêtre de temps » désigne un intervalle pendant lequel une visite chez un client est possible. Le VRP avec fenêtres de temps (VRPTW, VRP with Time Windows) désigne l'ensemble des problèmes de tournées de véhicules pour lesquels les interventions sont soumises à des fenêtres de temps. Ces problèmes sont fréquemment rencontrés dans la vie réelle, dans lesquels les contraintes de type « rendez-vous » (visite en fonction de la disponibilité du client) sont très présentes. Les fenêtres de temps peuvent être de deux types : dures et souples. Les fenêtres dures représentent des contraintes du problème et doivent absolument être respectées, alors que le non-respect d'une fenêtre souple entraîne une pénalité de coût. La définition du problème est la suivante :

VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Données :	Un ensemble de noeuds client ayant une demande et des fenêtres de temps (souples ou dures) ; un ensemble d'arêtes (munies de coûts) et une flotte illimitée de véhicules de capacité fixée Q partant d'un unique dépôt.
Trouver :	Un ensemble des tournées de véhicules, satisfaisant chaque demande une et une seule fois en respectant les contraintes de capacité des véhicules et les contraintes de fenêtres de temps dures.
Minimisant :	Le coût total de transport lié aux arcs empruntés par les véhicules et/ou le coût lié au non-respect des fenêtres de temps souples.

Il est possible d'étendre le modèle de Fisher et Jaikumar pour tenir compte des fenêtres de temps. Pour cela, nous avons besoin de données et de variables temporelles. Nous définissons :

- $[a_i, b_i]$ la fenêtre de temps au noeud i.
- s_i le temps de service au noeud i.
- t_{ij}le temps de transport entre les noeuds i et j.
- la variable u_i^k associée à l'heure d'arrivée du véhicule k au noeud i.
- M une grande valeur.

Nous allons présenter les contraintes traitant les fenêtres de temps :

$$a_i \leq u_i^k \leq b_i \quad \forall i \in \mathcal{N}, \forall k \in K \tag{3.12}$$

$$u_i^k + s_i + t_{ij} - M \times (1 - x_{ij}^k) \leq u_j^k \quad \forall i \in \mathcal{N}, \forall j \neq 1 \in \mathcal{N}, \forall k \in K \tag{3.13}$$

La contrainte (3.12) garantit que chaque demande est servie dans sa fenêtre de temps. La contrainte (3.13) assure la cohérence temporelle des tournées. En effet, elle vérifie que pour deux tâches exécutées consécutivement par une même ressource, le début de la seconde tâche commence après la fin d'exécution de la première tâche augmentée du temps de déplacement nécessaire pour se rendre sur le second site.

Les méthodes de résolution associées au Vehicle Routing Problem with Time Window se divisent en deux groupes : les méthodes approchées et les méthodes exactes. On peut consulter une synthèse sur le Vehicle Routing Problem with Time Window (Cordeau *et al.*, 2000) et sur les méthodes approchées (Braysy & Gendreau, 2005a) et (Braysy & Gendreau, 2005b). Nous présenterons plus en détails les méthodes approchées en section 6.1.

La première méthode exacte permettant de résoudre des instances de taille raisonnable du Vehicle Routing Problem with Time Window a été proposée par (Desrochers *et al.*, 1992). Cependant, l'objectif est la minimisation de la distance totale de parcours, alors que pour les métaheuristiques, l'objectif est hiérarchique, l'objectif primaire étant la minimisation du nombre de véhicules. La méthode utilisée est un algorithme de Branch and Price. Le Branch and Price est un algorithme de recherche arborescente comme le Branch and Bound, mais repose généralement sur une formulation comme un problème de partitionnement ou de recouvrement. Le nombre de variables est alors trop grand pour qu'on puisse toutes les énumerer et on se contente de générer un sous-ensemble de variables utiles à la résolution optimale du problème. La relaxation linéaire du modèle est résolue à chaque noeud de l'arbre de recherche par un algorithme de génération de colonnes. L'article (Barnhart *et al.*, 1998) donne une très bonne description des principes de fonctionnement du Branch and Price.

Le sous-problème de l'algorithme de génération de colonnes de (Desrochers *et al.*, 1992) est un problème de plus court chemin avec contraintes de ressources (SPPRC Shortest Path Problem with resource Constraints). La résolution de ce problème avec un algorithme inspiré de celui de Bellman pouvait poser des problèmes en terme de complexité temporelle et spatiale. On diminue cette difficulté en ne considérant que des chemins élémentaires (Feillet *et al.*, 2004).

Il existe très peu de méthodes exactes de résolution du Vehicle Routing Problem with Time Window qui ne sont pas basées sur la génération de colonnes et nous ne les présenterons pas ici.

3.6 Problèmes de tournées multi-périodes

Plus récents que les problèmes de tournées de véhicules classiques, les problèmes de tournées multi-périodes considèrent un horizon de planification où un véhicule peut effectuer plusieurs tournées. De ce

fait, les véhicules peuvent effectuer plusieurs voyages. Nous considérons dans cette section deux types de problèmes multi-périodes : le PVRP (Period Vehicle Routing Problem) et l'IRP (Inventory Routing Problem). Ces deux problèmes correspondent généralement à des problématiques industrielles similaires mais abordées de manières différentes. Parmi ces problématiques on compte généralement la livraison de gaz liquide et de pétrole, la distribution des boissons, et plus généralement les distributions périodiques de marchandises consommables et ayant besoin de plusieurs livraisons sur l'horizon de planification.

3.6.1 Period Vehicle Routing Problem (PVRP)

Dans le PVRP, on considère que chaque client doit être servi un certain nombre de fois dans l'horizon, et que ce nombre constitue une donnée du modèle. Le but est de servir chaque client autant de fois que nécessaire. A chaque client est attribué un ensemble de séquences de livraisons possibles, correspondant aux jours de livraison. Le PVRP peut donc se décomposer en deux problématiques : affectation de séquences de livraison aux clients et résolution d'un problème de tournées « classiques » par jour de l'horizon.

PERIOD VEHICLE ROUTING PROBLEM

Données :	Un ensemble de noeuds client ayant une demande dans un horizon de planification ; un ensemble d'arêtes (munies de coûts) et une flotte illimitée de véhicules de capacité fixée Q partant d'un unique dépôt.
Trouver :	Un ensemble des tournées de véhicules, satisfaisant chaque demande autant de fois que nécessaire dans l'horizon de planification, en respectant les contraintes de capacité des véhicules.
Minimisant :	Le coût total de transport lié aux arcs empruntés par les véhicules.

Ce problème a été formulé la première fois par (Beltrami & Bodin, 1974) concernant la collecte d'ordures ménagères. Deux approches sont proposées :

- Affecter à chaque client un séquence de livraison puis résoudre un problème de tournée pour chaque jour de l'horizon.
- Construire des tournées puis les affecter aux jours en respectant les séquences de livraison.

Cependant, la fréquence de visite des clients est une contrainte dure ce qui peut pénaliser la résolution. Des méthodes plus récentes relâchent cette contrainte et autorise un client à être visité plus souvent que nécessaire. Cette relaxation est particulièrement utile dans le Period Vehicle Routing Problem with Service Choice (Peter Francis, 2004) où la fréquence de visite est une variable de décision du problème.

3.6.2 Inventory Routing Problem (IRP)

Depuis vingt ans, l'IRP suscite beaucoup plus d'intérêt de la part des chercheurs. Il s'agit d'une approche plus globale que pour le PVRP, dans la mesure où il intègre également des problématiques de gestion des stocks. Dans l'Inventory Routing Problem, le concept de séquence de livraison devient implicite, et n'est plus une donnée du problème. Chaque client a un taux de consommation qui est généralement connu, et un stock de produit en début d'horizon. Le but est d'éviter les ruptures de stock chez le client. On peut alors découper l'IRP en trois problématiques : gestion des stocks de chaque client et des quantités à livrer afin d'éviter les ruptures de stocks, affectation des clients aux jours de livraison, et conception et optimisation des tournées.

INVENTORY ROUTING PROBLEM

Données :	Un ensemble de noeuds client ayant un stock et une consommation de ce stock dans l'horizon de planification ; un ensemble d'arêtes (munies de coûts) ; une flotte illimitée de véhicules de capacité fixée Q partant d'un unique dépôt.
Trouver :	Un ensemble des tournées de véhicules, évitant toute rupture de stock chez le client, en respectant les contraintes de capacité des véhicules.
Minimisant :	Le coût total de transport lié aux arcs empruntés par les véhicules.

3.7 Problèmes de tournées avec flotte limitée

Depuis quelques années, un nouveau type de problèmes de tournées suscite un intérêt grandissant : les problèmes de tournées avec flotte limitée, appelés m-Vehicle Routing Problem (m-VRP). S'il existe une affectation permettant de satisfaire toutes les demandes sur l'horizon de planification, le problème est dit satisfiable et on le résout en fonction de l'objectif prédéfini. Dans le cas contraire, on définit un objectif secondaire et on relance la résolution du problème. Typiquement, cet objectif visera à maximiser le nombre de demandes satisfaites. Cependant, il peut varier en fonction des contraintes (fenêtre de temps dure ou souple, etc). La satisfaction de chaque demande devient alors une variable de décision, et il faut donc adapter l'objectif en conséquence.

Nous allons définir deux énoncés du problème en fonction de ces deux alternatives

M-VEHICLE ROUTING PROBLEM WITH TIME WINDOWS (SATISFIABLE)

Données :	Un ensemble de noeuds client ayant une demande et des fenêtres de temps ; un ensemble d'arêtes (munies de coûts) ; flotte limitée de véhicules .
Trouver :	Un Ensemble des tournées des véhicules respectant les contraintes de capacité des véhicules, les fenêtres de temps dures et les contraintes sur les dépôts.
Minimisant :	Le coût total de transport lié aux arcs empruntés par les véhicules.

M-VEHICLE ROUTING PROBLEM WITH TIME WINDOWS (INSATISFIABLE)

Données :	Un ensemble de noeuds client ayant une demande et des fenêtres de temps ; un ensemble d'arêtes (munies de coûts) ; flotte limitée de véhicules .
Trouver :	Un Ensemble des tournées des véhicules respectant les contraintes de capacité des véhicules, les fenêtres de temps dures des jobs effectivement réalisés et les contraintes sur les dépôts.
Maximisant :	Le nombre de clients satisfaits.

On différencie les flottes sur un critère d'homogénéité. On dit qu'une flotte est homogène si chaque véhicule est identique en tous points. Certaines méthodes de résolution décomposent alors le problème en deux sous-problèmes, l'affectation des jobs aux ressources, puis la création de tournées pour chaque ressource. On peut se référer à (Libertad Tansini *et al.*, 2001) pour le cas du problème de tournées de véhicules multi-dépôts (les véhicules ont des points de départ et d'arrivée spécifiques).

3.8 Split Delivery Vehicle Routing Problem

A la différence des problèmes de tournées de véhicules classiques, le Split Delivery Vehicle Routing Problem permet qu'une demande soit satisfaite par plusieurs tournées. Précisons l'énoncé :

SPLIT DELIVERY VEHICLE ROUTING PROBLEM

Données :	Un ensemble de noeuds client ayant une demande ; un ensemble d'arêtes (munies de coûts) ; des véhicules ayant une capacité.
Trouver :	Un Ensemble des tournées des véhicules respectant les contraintes de capacité des véhicules, les contraintes sur les dépôts.
Minimisant :	Le coût total de transport lié aux arcs empruntés par les véhicules.

Une méthode qui nous a paru intéressante est une heuristique basée sur un programme d'optimisation mathématiques (Archetti *et al.*, 2005a). Il s'agit de générer un sous-ensemble de routes intéressantes à l'aide d'une recherche tabou puis de faire une seconde passe en optimisant les tournées réalisables grâce à ce sous-ensemble de routes à l'aide d'un programme d'optimisation mathématique. Cependant, on peut se demander quel gain est obtenu en autorisant le split, Nous nous réferons aux conclusions de (Archetti *et al.*, 2005b) :

– La réduction de coût obtenu en autorisant le split est principalement due à la réduction du nombre de tournées.
– Les bénéfices obtenus grâce à l'autorisation du split dépendent de la demande moyenne, la capacité de la flotte et de la variance de la demande. Il n'est pas apparu de dépendance au positionnement des clients.

Dans notre cas, on est obligé d'autoriser le split, que nous avons appelé préemption, quel enseignement peut-on tirer de l'étude de ce problème ? Les chercheurs qui se sont penchés sur ce problème ont pu mettre à jour une propriété des solutions optimales (Archetti *et al.*, 2006) que nous allons présenter :

Définition 3.1 (k-split cycle) *Étant donné k clients et k tournées. La route 1 visite les clients i_1 et $i-2$, la route 2 visite les clients i_2 et i_3, la route $k-1$ visite les clients i_{k-1} et i_k et la route k visite les clients i_k et i_1. Le sous-ensemble de clients $i_1, i_2, \ldots, i_k$ est appelé k-split cycle.*

Propriété 3.1 *Si la matrice de distance satisfait l'inégalité triangulaire, alors il existe une solution optimale du SDVRP pour laquelle il n'y a pas de k-split cycle (pour tout k).*

On peut alors en tirer le corollaire suivant :

Corollaire 3.1.1 *Si la matrice de distance satisfait l'inégalité triangulaire, alors il existe une solution optimale du SDVRP pour laquelle aucun couple de routes n'a plus d'un client splité en commun.*

3.9 Positionnement du problème traité dans le cadre de ce stage

Notre problème appartient donc à la classe des problèmes de tournées de véhicules. En effet, il est impossible de dissocier l'affectation des tâches au technicien ou l'orodonnancement des tâches de la construction de tournées réalisables.
Cependant, nous nous heurtons à certaines différences majeures avec les problèmes de tournées de véhicules classiques :

– Souvent, le temps passé chez un client est négligeable (livraisons de journaux, ramassage d'ordures, etc), alors qu'il s'agit d'un facteur important de notre problème. La difficulté combinatoire des problèmes classiques est due au très grand nombre de tournées réalisables, alors que dans notre cas le nombre de noeuds visités est limité par le temps minimum d'une intervention (environ 1h) et par la durée d'une journée de travail.
– Nous traitons le cas multi-périodes, c'est-à-dire avec un horizon de planification de plusieurs jours. Cependant, il s'agit de tournées de service, et chaque demande doit être satisfaite une seule fois, contrairement aux problèmes multi-périodes classiques comme l'Inventory Routing Problem (IRP). Une période de validité est ici associée à chaque demande, et représente un ensemble de jours de l'horizon pendant lesquels la demande peut être satisfaite.
– Il s'agit de tournées de service effectuées par des techniciens, donc les problématiques liées à la capacité sont absentes de ces problèmes ; cependant, la durée totale de chaque tournée est bornée par la durée d'une journée de travail, et le temps disponible peut être vu comme une capacité. C'est d'ailleurs le cas dans certains travaux concernant les problèmes avec capacité, qui considèrent également le temps disponible.
– La taille de la flotte est limitée et cette limite est une contrainte forte. Une tournée étant effectuée par un technicien, le nombre de techniciens disponibles le jour d est la borne supérieure du nombre de tournées associé au jour d. Les disponibilités des techniciens étant connues a priori, le nombre total de tournées possibles sur l'ensemble de l'horizon est connu.
– La notion de multi-dépôts est présente ici. De plus, chaque technicien dispose de points de départ et d'arrivée propres.
– Nous négligerons le problème de la pause déjeuner car les techniciens n'ont pas de points de restauration imposés. Nous nous contenterons donc de réduire le temps de travail d'une journée de la durée du déjeuner.
– Il existe des jobs exigeant la présence de deux techniciens.

La figure 3.1 illustre le positionnement de notre problème par rapport aux problèmes de tounées de véhicules présentés dans ce chapitre.

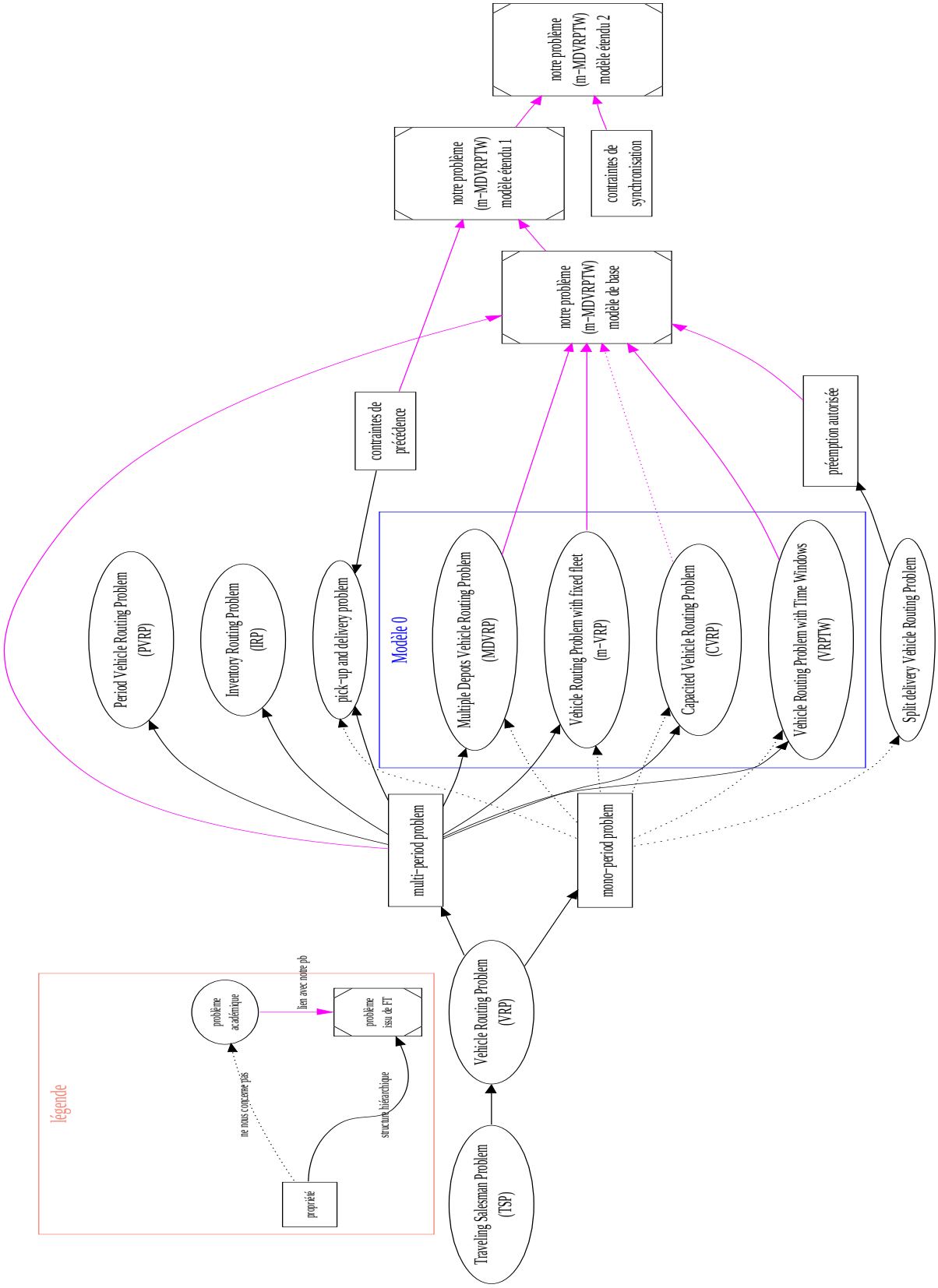

FIG. 3.1 – Situation de notre problème par rapport aux problèmes de tournées de véhicules.

CHAPITRE 4

MODÈLES MATHÉMATIQUES

Dans ce chapitre, nous présentons des modèles mathématiques linéaires associés à notre problème. Nous commencerons par détailler les informations sur l'intervention recueillies par l'UIC en section 4.1, puis nous présenterons les définitions et notations utilisées dans la modélisation en section 4.2 page 26, enfin nous modeliserons les contraintes et objectifs.
Pour modéliser notre problème, nous procédons en plusieurs étapes. Nous présenterons un modèle pour le problème de tournées de véhicules multi-périodes avec fenêtres de temps, flotte hétérogène limitée et l'intégration des jobs longs (section 4.3 page 29) inspiré de (Fabien Tricoire, 2006), puis nous montrerons les modifications et ajouts nécessaires à l'intégration de nouvelles contraintes : des contraintes de précédence (section 4.4 page 33) qui correspondent au besoin d'aller chercher du matériel à un dépôt ; des contraintes de synchronisation (section 4.5 page 34) qui correspondent à la nécessité d'envoyer plusieurs ressources pour satisfaire une demande. Enfin, nous présenterons la fonction économique associée aux coûts.

4.1 Les données

Grâce au travail du vendeur et du RO, nous disposons d'un certain nombre d'informations sur le client et sur l'intervention à réaliser. Dans cette section, nous présentons les informations recueillies ainsi que le pré-processing auquel nous les soumettons afin qu'elles soient utilisables par un modèle mathématique.

4.1.1 La géographie

Nous avons besoin d'un certain nombre de données géographiques :

1. Les dépôts France Telecom
2. Les bases de départ des techniciens
3. Les points d'arrivée des techniciens
4. Les sites correspondant aux clients

4.1.2 Les techniciens

On définit l'ensemble des techniciens $K = \{1, \ldots, m\}$. Pour chaque technicien k, on a les informations suivantes :

- Ses compétences.
- Son planning.
- Une base de départ et d'arrivée.
- Une zone géographique.

4.1.3 Les clients

Les informations sur le client sont inutiles lorsque les jobs sont modélisés. Elles peuvent cependant servir à gérer des clients privilégiés. Ces informations peuvent être utilisées lors du pre-processing pour gérer l'importance attribuée aux jobs.

4.1.4 Les interventions

Nous avons défini différents types d'intervention : pré-visites, jobs courts et jobs longs. Nous allons maintenant présenter les informations nécessaires à leur modélisation dans le tableau 4.1 page 26. On remarque que le produit concerné par l'intervention définit les compétences nécessaires à un technicien. De plus, la durée d'une tâche peut dépendre du niveau de compétence du technicien.

On peut noter qu'un des objectifs de notre modèle est d'augmenter la satisfaction client, ceci passera par la définition d'une fonction objectif qui dépendra éventuellement de la date d'échéance, la date de fin souhaité, la priorité de la tache, le prix du contrat, du client,...

Donnée	**pré-visite**	**intervention courte**	**intervention longue**
site client	oui	oui	oui
durée	fixe	$\leq 1j$	$\geq 1j$
date contractuelle	non	oui	oui
fin souhaitée	oui	oui	oui
date de début au plus tôt	non	oui	oui
compétences	non	oui	oui
une priorité	oui	oui	oui
dépôt de livraison matériel	non	si nécessaire	si nécessaire
zone géographique	oui	oui	oui

TAB. 4.1 – Tableau récapitulatif des données en fonction du type de job.

4.1.5 Pre-processing

Lors du pre-processing, on calcule les matrices de temps d'exécution d'un job j par une ressource k, noté p_j^k, de la manière suivante : Si un job n'est pas dans la zone géographique du technicien ou s'il n'a pas les compétences pour l'exécuter, la durée du traitement de la tâche par le technicien est considérée infini ; sinon on met la durée estimée en fonction du niveau de compétence.

4.2 Définitions et notations

4.2.1 Définitions

Nous allons maintenant définir formellement les notions que nous avons présentées dans les sections précédentes.

Définition 4.1 (Ressource ou véhicule) *Un technicien sera assimilé à une ressource (ou véhicule). L'ensemble des ressources sera noté* $K = \{1, \ldots, m\}$.

Définition 4.2 (Job) *Une intervention sera appelée un job. L'ensemble des jobs est* $J = \Theta \cup \Delta \cup \Omega$, *où* Θ *correspond aux prévisites,* Δ *à l'ensemble jobs courts et* Ω *à l'ensemble des jobs longs. On a aussi* $\Delta = \Delta_1 \cup \Delta_2$ *et* $\Omega = \Omega_1 \cup \Omega_2$ *où l'indice correspond au nombre de techniciens nécessaire à la réalisation du job.*

Définition 4.3 (Fenêtre de temps) *Une fenêtre de temps désigne un intervalle de temps pendant lequel une ressource est disponible ou une demande est satisfaisable. Une fenêtre de temps est dite dure si elle doit être absolument respectée et souple si le non respect de cette fenêtre entraîne une pénalité.*

Nous avons défini trois types de jobs, les pré-visites, les jobs courts et les jobs longs. Nous avons dit que les jobs longs avaient une durée supérieure à une journée de travail, nous allons donc définir la notion de préemption pour ces jobs longs.

Définition 4.4 (Préemption) *La préemption signifie qu'une ressource peut interrompre un job en cours d'exécution pour le reprendre plus tard.*

Définissons maintenant le vocabulaire utilisé pour désigner les contraintes.

Définition 4.5 (Contrainte de précédence) *Une contrainte de précédence entre deux jobs i et j, notée $i \rightarrow j$, impose que l'exécution du job j commence après la fin du job i. Nous imposons aussi que ce soit le même technicien qui exécute les deux jobs.*

Définition 4.6 (Contrainte de synchronisation) *Une contrainte de synchronisation sur un job i impose l'envoi de deux ressources sur ce job.*
Une contrainte de synchronisation totale impose que ces ressources réalisent la totalité du job simultanement. Une contrainte de synchronisation partielle impose que les ressources exécutent une partie de la tâche simultanément.

Définition 4.7 (Contrainte de connexité) *On appellera contrainte de connexité, la contrainte imposant qu'un job long soit réalisé sur un intervalle de jours consécutifs.*

Dans notre problème, nous devons donc vérifier que chaque tournée soit réalisable, mais nous devons également nous assurer de la cohérence entre les tournées. Nous pouvons donc définir deux catégories de contraintes :

Définition 4.8 (Contrainte interne, Contrainte externe) *Une contrainte est dite interne si elle assure la faisabilité d'une tournée. Une contrainte est dite externe si elle assure la cohérence entre les tournées*

On voit donc que les fenêtres de temps et les bases de départ et d'arrivée peuvent être vues comme des contraintes internes, alors que les contraintes de synchronisation et connexité sont des contraintes externes. Les contraintes de précédence peuvent être selon les cas internes ou externes.

4.2.2 Propriétés utiles

Rappel de la théorie des graphes

Rappelons quelques notions essentielles de la théorie des graphes.

Définition 4.9 (Graphe orienté) *Un graphe orienté $\mathcal{G} = (\mathcal{N}, \mathcal{V})$ est constitué d'un ensemble fini de nœuds et d'un ensemble fini d'arcs. Un arc est une paire de nœuds (X, Y) telle que $X, Y \in \mathcal{N}$ et $X \neq Y$. On dit que X est le parent de Y et Y est l'enfant de X.*

On dit aussi qu'un sous-graphe $\mathcal{G}'$ d'un graphe $\mathcal{G}$ est un graphe induit par un sous-ensemble de nœuds $N \subseteq \mathcal{N}$ et tous les arcs de $\mathcal{V}$ joignant deux sommets de N.

Définition 4.10 (Chemin) *Un chemin d'un nœud X à un nœud Y dans un graphe orienté $\mathcal{G}$ est un ensemble d'arcs $E = \{(X_1, X_2), (X_2, X_3), \ldots, (X_{n-1}, X_n)\}$ tel que $E \subseteq \mathcal{V}$ et $X = X_1$ et $X_n = Y$; la longueur du chemin E est $|E|$.*

Définition 4.11 (Chemin Elémentaire) *Un chemin $E = \{(X_1, X_2), (X_2, X_3), \ldots, (X_{n-1}, X_n)\}$ est élémentaire si chacun des sommets du parcours est visité une seule fois : $\forall i, j$ tels que $1 \leq i, j \leq n$ et $i \neq j$ on a : $X_i \neq X_j$.*
Un chemin élémentaire est donc un chemin simple (chaque arête est empruntée une seule fois) et sans cycle.

Propriété 4.1 *Tout chemin élémentaire d'un graphe $\mathcal{G} = (\mathcal{N}, \mathcal{V})$ passant par n noeuds est de longueur au plus $n - 1$.*

En effet, si le chemin possède $k > n - 1$ arcs, alors il passe par $k + 1 > n$ sommets, il visite donc un sommet deux fois et le chemin contient donc un cycle et n'est donc pas élémentaire.

Intervalles de $\mathbb{N}$

Nous allons maintenant rappeler la définition d'un intervalle dans l'ensemble des entiers naturels afin de montrer une propriété qui nous sera utile dans la rédaction des contraintes. Nous utiliserons l'opérateur . pour la concaténation.

Définition 4.12 (Intervalles de $\mathbb{N}$) *Soit I une partie de $\mathbb{N}$. On dit que I est un intervalle si, pour tous $x < y$ appartenant à I, pour tout z avec $x < z < y$, alors z est élément de I.*

Propriété 4.2 *Soit $\{x_i\}_{0<i<T+1}$ un ensemble de variable binaire. Les assertions suivantes sont équivalentes :*

1. *$I = \{i \in [1, \ldots, T] \mid x_i = 1\}$ est un intervalle fermé de $[1, \ldots, T]$;*
2. *$\forall i < j < k \in [1, \ldots, T] \quad x_i.x_j.x_k \neq 1.0.1.$*

Démonstration $1 \Rightarrow 2$: supposons 2 fausse pour $i_0 < j_0 < k_0$, alors $I \subset [1, x_{j_0}[\cup]x_{j_0}, T]$ et n'est donc pas un intervalle. $2 \Rightarrow 1$: par définition, un intervalle $[x, y]$ contient l'ensemble $\{z | x \geq z \leq y\}$. ♠

4.2.3 Représentation graphique du problème

A partir des données géographiques présentées en section 4.1, nous exposerons une représentation graphique de l'énoncé du problème à l'aide d'un graphe orienté $\mathcal{G} = (\mathcal{N}, \mathcal{V})$, puis nous détaillerons comment construire les noeuds et les informations associées à chaque noeud.

Chaque technicien a une base de départ (respectivement d'arrivée) unique $s^k \in S$ (respectivement $f^k \in F$) auquel est attaché l'information a^{kd} (respectivement b^{kd}). Le temps de service associé à ces noeuds est nul.

Chaque job est représenté par un noeud auquel est associé sa fenêtre de temps, sa durée totale en fonction des compétences des ressources. Lorsqu'un job j nécessite d'aller chercher du matériel à un dépôt, on crée un nouveau job i lié à j par une contrainte de précédence $i \rightarrow j$ et le noeud associé à ce job a la même localisation que le dépôt. De cette manière, les dépôts n'apparaissent pas dans le graphe. Lorsqu'on doit envoyer deux techniciens sur un même job, on ajoute un marqueur sur le noeud représentant le job. Il existe donc une bijection entre les jobs et le sous-ensemble de noeud du graphe représentant les sites clients. Par la suite, on utilisera la notation $J = \Theta \cup \Delta_1 \cup \Delta_2 \cup \Omega_1 \cup \Omega_2$ aussi bien pour désigner l'ensemble des jobs que l'ensemble des noeuds associés à ces jobs. On a donc $\mathcal{N} = S \cup F \cup J$. Nous énonçons quelques restrictions sur la structure du graphe :

- Les points de départ et d'arrivée ne sont utilisés qu'une seule fois par jour. Si plusieurs ressources partagent ces points, il suffit de les dupliquer.
- Si une ressource part d'un dépôt, il suffit de rajouter un noeud de départ et un noeud d'arrivée relié au dépôt par un arc de longueur 0.
- Il n'existe pas d'arc joignant deux points de départs ou deux points d'arrivée.
- Aucun arc n'a pour destination un point de départ ni pour origine un point d'arrivée.

On peut noter qu'un arc (i, j) peut être supprimé si :

$$r_i + \min_k p_i^k + t_{ij} + \min_k p_j^k > d_j^e$$

Nous noterons $\mathcal{N}^-$, l'ensemble des noeuds qui sont origine d'un arc, soit $\mathcal{N}^- = S \cup J$ et $\mathcal{N}^+$, l'ensemble des noeuds ou un arc arrive, soit $\mathcal{N}^+ = F \cup J$.

On peut s'intéresser à deux types de coûts associés aux déplacements, la distance ou le temps de parcours. Quel que soit le critère choisi, la matrice représentant ces coûts possédera les propriétés suivantes :

$$\begin{array}{rlcll} \forall i, j & d_{ii} & = & 0 & \\ \forall i, j, k & d_{ij} & \leq & d_{ik} + d_{ki} & \text{(sur-additivité)} \end{array}$$

4.2.4 Remarque sur la linéarité

Différents modèles linéaires décrivent le problème. Cependant, certaines contraintes ne sont à priori pas linéaires, par exemple les contraintes liées aux fenêtres de temps. On utilise alors un artifice mathématique pour les linéariser. Cette méthode est illustrée sur l'exemple suivant ou $x \in \mathbb{R}^+$ et $y \in \{0, 1\}$ sont deux variables :

$$x \times y \leq D \Leftrightarrow x - M \times (1 - y) \leq D$$

On peut donc remarquer qu'il faudra choisir judicieusement la valeur de M en fonction des données.

4.3 Modèle de base

Nous présentons un premier modèle inspiré de la formulation du Vehicle Routing problem with Time Windows.

4.3.1 Hypothèses et notations

Liste exhaustive des hypothèses retenues pour ce modèle :

1. Chaque ressource a une base de départ et d'arrivée unique.
2. Chaque ressource a une fenêtre de temps quotidienne.
3. Le nombre de ressources est limité.
4. Chaque ressource peut réaliser un sous-ensemble des jobs.
5. Chaque job doit être entièrement exécuté par la même ressource avec possibilité de préemption pour un sous-ensemble de jobs.
6. Chaque job doit être réalisé dans l'intervalle de temps correspondant à sa fenêtre de temps.
7. Chaque job pour lequel la préemption est autorisée doit être réalisé sur un intervalle de jours consécutifs.
8. La durée d'un job dépend de la ressource qui l'exécute.
9. Il n'y a pas de contraintes de précédence ni de synchronisation.

Ces hypothèses nous situe dans le cadre d'un problème de tournées de véhicule multi-périodes avec fenêtre de temps et flotte hétérogène limitée (voir 3.7 et 3.5).
Les notations utilisées pour le modèle de base sont présentées dans le tableau 4.2 page 30.

Horizon de planification	
$\|D\| = T$	horizon en jours
$[0, t_{max}]$	horizon continu
d_{max}	temps de travail maximal d'une journée
Géographie $\mathcal{G} = (\mathcal{N} = S \cup F \cup J, \mathcal{V})$	
S	bases de départ des ressources
F	bases d'arrivée des ressources
J	sites des jobs
$\mathcal{V}$	ensemble des arcs
t_{ij}	temps de transport du noeud i au noeud j.
Ressources $\|K\| = m$	
s^k	base de départ
f^k	base d'arrivée
q_j^k	niveau de compétence de la ressource k pour le job j
$\left[a^{kd}, b^{kd}\right]$	fenêtre de temps de la ressource k le jour d
Jobs $J = \Theta \cup \Delta \cup \cup \Omega \quad \|J\| = n$	
Θ	prévisites
Δ	jobs sans préemption (job court)
Ω	jobs pour lesquels la préemption est autorisée (job long)
p_j^k	durée du job en fonction du technicien (temps de service)
r_j	date de début au plus tôt (release date)
ds_j	date de fin souhaitée (due date)
dc_j	date de fin contractuelle (deadline)
Modéle de base	
M	un grand nombre
variables	
y_i^k	variable binaire valant 1 si une tournée associée à la ressource k passe par le noeud i.
y_i^{kd}	variable binaire valant 1 si la tournée associée à la ressource k passe par le noeud i le jour d.
x_{ij}^{kd}	variable binaire valant 1 si la tournée associée à la ressource k emprunte l'arc (i, j) le jour d.
$u_i^{kd} \in R$	heure d'intervention au noeud i de la ressource k le jour d.
$p_i^{kd} \in R$	temps de travail d'une ressource k pour un job $i \in \Omega$ le jour d.

TAB. 4.2 – Tableau des notations relatives aux données et au modèle de base.

4.3.2 Formulation mathématique

Nous allons maintenant présenter un premier modèle pour le problème de tournées de véhicule multi-période avec fenêtre de temps et flotte limitée n'incluant ni contraintes de précédence, ni contraintes de synchronisation.

Remarque 2 *Souvent, les algorithmes pour les problèmes multi-période fonctionnent itérativement : lorsque tous les jobs ne sont pas réalisables sur une seule période, on utilise un objectif secondaire (par exemple minimiser le nombre de jobs restant à exécuter), puis on réitère le raisonnement sur la période suivante. Ceci n'est plus possible dans notre cas. En effet, il existe des contraintes externes qui imposent la prise en compte des tournées construites au cours des périodes précédentes. Nous sommes donc obliger de considérer l'ensemble de l'horizon de planification pour la construction des tournées.*

Contraintes retenues pour ce modèle :

$$\sum_{i \in J} x_{s^k i}^{kd} - \sum_{i \in J} x_{i f^k}^{kd} = 0 \quad \forall k \in K, \forall d \in D \tag{4.1}$$

$$\sum_{j \in J} x_{s^k j}^{kd} \leq 1 \quad \forall k \in K, \forall d \in D \tag{4.2}$$

La conservation des flots sur l'utilisation des ressources est assurée par la contrainte (4.1) : si un arc sort du point de départ d'une ressource un jour donné, alors un arc doit rentrer dans le point d'arrivée

de cette ressource le même jour. La contrainte (4.2) assure qu'un technicien ne sort de sa base qu'au plus une fois par jour.

$$\sum_{k\in K} y_i^k = 1 \quad \forall i \in J \tag{4.3}$$

$$y_i^k - \sum_{d\in D} y_i^{kd} = 0 \quad \forall i \in \Theta \cup \Delta, \forall k \in K \tag{4.4}$$

$$y_i^k - \sum_{d\in D} y_i^{kd} \leq 0 \quad \forall i \in \Omega, \forall k \in K \tag{4.5}$$

$$|D| \times y_i^k - \sum_{d\in D} y_i^{kd} \geq 0 \quad \forall i \in \Omega, \forall k \in K \tag{4.6}$$

La contrainte (4.3) impose la satisfaction de chaque demande. La contrainte (4.4) impose que les prévisites et les jobs courts soient exécutés sans préemption. La contrainte (4.5) impose qu'un technicien affecté à un job le réalise pendant au moins une journée. La contrainte (4.6) empêche un technicien non affecté à un job de travailler dessus.

$$y_j^{kd} - \sum_{i\in\mathcal{N}^-} x_{ij}^{kd} = 0 \quad \forall j \in J, \forall k \in K, \forall d \in D \tag{4.7}$$

$$y_i^{kd} - \sum_{j\in\mathcal{N}^+} x_{ij}^{kd} = 0 \quad \forall i \in J, \forall k \in K, \forall d \in D \tag{4.8}$$

Les contraintes (4.7) et (4.8) imposent la cohérence entre l'affectation des demandes aux tournées et la trace des tournées. Elles assurent aussi qu'une ressource débute sa tournée par un arc (i, j) ou $i \in S$ et la finit par un arc (i_1, j_1) ou $j_1 \in F$. Ces contraintes associées aux contraintes (4.1) et (4.2) vérifie les contraintes associées aux bases de départ et d'arrivée.

$$\sum_{i,j\in\mathcal{S}} x_{ij}^k \leq |\mathcal{S}| - 1 \quad \forall k \in K, \forall \mathcal{S} \subset \mathcal{N} \text{ t.q. } 2 \leq |\mathcal{S}| \leq n-2 \tag{4.9}$$

La contrainte (4.9) interdit la présence de sous-tours dans une solution. Elle a une formulation classique que l'on retrouve dans le TSP ou le VRP.
Nous avons mis la formulation brute de cette contrainte mais nous y ajoutons une restriction. En effet, chaque job ayant un temps de service, tous les sous-tours ne sont pas possibles, nous montrerons les restrictions sur cette contrainte en section 5.1.1.

$$\sum_{d\in D} p_i^{kd} = p_i^k \times y_i^k \quad \forall i \in \Omega, \forall k \in K \tag{4.10}$$

La contrainte (4.10) vérifie qu'une tâche exécutée en plusieurs fois est réalisée intégralement. En effet, nous verrons que la contrainte (4.17) impose que $p_i^{kd} = 0$ si un technicien n'a pas effectué le job i le jour d. Donc, la contrainte (4.10) vérifie la quantité de la tâche qui a été réalisée. La variable p_i^{kd} n'est défini que pour les jobs longs, c'est-à-dire les jobs pour lesquels la préemption est autorisée. On réduit ainsi le nombre de variables mais aussi de contraintes nécessaires à la modélisation.

$$y_i^k \leq q_i^k \quad \forall i \in J\backslash\Theta, \forall k \in K \tag{4.11}$$

La contrainte (4.11) assure la compatibilité entre les ressources et les demandes. La contrainte (4.11) assure qu'une ressource ayant un niveau de compétence nul pour un job ne l'effectuera pas.

$$u_i^{kd} + p_i^k + t_{ij} - M \times (1 - x_{ij}^{kd}) \leq u_j^{kd} \quad \forall i \in \Theta \cup \Delta, \forall j \in \mathcal{N}^+, \forall k \in K, \forall d \in D \tag{4.12}$$

$$u_i^{kd} + p_i^{kd} + t_{ij} - M \times (1 - x_{ij}^{kd}) \leq u_j^{kd} \quad \forall i \in \Omega, \forall j \in \mathcal{N}^+, \forall k \in K, \forall d \in D \tag{4.13}$$

$$u_{s^k}^{kd} + t_{s^k j} - M \times (1 - x_{s^k j}^{kd}) \leq u_j^{kd} \quad \forall j \in J, \forall k \in K, \forall d \in D \tag{4.14}$$

Les contraintes (4.12), (4.13) et (4.14) assurent le non chevauchement de deux tâches consécutives i et j. En effet, on vérifie que la date de début du job j est postérieure à la date de début du job i augmentée du temps d'exécution de i et du temps de parcours entre i et j. La contrainte (4.14) traite le départ d'un technicien depuis sa base et on lui associe donc un temps d'exécution nul. L'arrivée du technicien à sa

base est déjà traitée par les contraintes (4.12) et (4.13) car $f^k \in \mathcal{N}^+$. On remarque que nous ne traitons pas tous les couples $(i,j) \in \mathcal{N} \times \mathcal{N}$, mais ceci est justifié par les restrictions présentées en 4.2.3.

$$\begin{aligned} u_i^{kd} &\geq r_i \times y_i^{kd} && \forall i \in J, \forall k \in K, \forall d \in D && (4.15)\\ u_i^{kd} &\leq (dc_i - p_i^k) \times y_i^{kd} && \forall i \in \Theta \cup \Delta, \forall k \in K, \forall d \in D && (4.16)\\ u_i^{kd} + p_i^{kd} &\leq dc_i \times y_i^{kd} && \forall i \in \Omega, \forall k \in K, \forall d \in D && (4.17) \end{aligned}$$

La contrainte (4.15) assure qu'un job est commencé après sa date de début au plus tôt. Les contraintes (4.16) et (4.17) vérifient qu'un job finit avant sa date contractuelle. De plus, la contrainte (4.17) assure que si y_i^{kd}=0, alors u_i^{kd}=0 et p_i^{kd}=0.

$$\begin{aligned} u_{s^k}^{kd} &\geq a^{kd} && \forall k \in K, \forall d \in D && (4.18)\\ u_{f^k}^{kd} &\leq b^{kd} && \forall k \in K, \forall d \in D && (4.19) \end{aligned}$$

La contrainte (4.18) vérifie qu'un technicien réalisant une tournée part de sa base de départ en respectant l'horaire de début de journée. La contrainte (4.19) vérifie qu'un technicien réalisant une tournée arrive à sa base d'arrivée en respectant l'horaire de fin de journée.

Nous imposons que les jours passés sur une intervention soient consécutifs, c'est-à-dire, l'ensemble $\{i | x_i = 1\}$ est un intervalle de $[1, \ldots, T]$. Nous allons utiliser la propriété présenté en 4.2 pour construire une contrainte linéaire vérifiant cette condition.

$$\left(y_i^{kd_0} - y_i^{kd_1}\right) - \left(y_i^{kd_1} - y_i^{kd_2}\right) \leq 1 \quad \forall i \in \Omega, \forall k \in K, \forall d_0 < d_1 < d_2 \in D \qquad (4.20)$$

La contrainte (4.20) garantit que la tâche est exécutée sur un intervalle de jours. En effet, nous illustrons la discrimination due à cette contrainte dans le tableau 4.3 où l'unique triplet interdit est grisé.

triplet	000	001	010	011	100	101	110	111
(4.20)	0	1	-2	-1	1	2	-1	0

TAB. 4.3 – Tableau indiquant pour chaque triplet possible la valeur de la contrainte (4.20).

Remarque 3 *Grâce aux contraintes* (4.16) *et* (4.17)*, on a les propriétés suivantes :*

$$\begin{aligned} &(4.16) \quad si\ y_i^{kd} = 0 \quad \Rightarrow \quad u_i^{kd} = 0\ et\ p_i^{kd} = 0 \quad \forall i \in \Omega\\ &(4.17) \quad si\ y_i^{kd} = 0 \quad \Rightarrow \quad u_i^{kd} = 0 \quad \forall i \in \Theta \cup \Delta \end{aligned}$$

Remarque 4 *En supprimant toutes les contraintes associées uniquement aux jobs* $\in \Omega$*, on retrouve un modèle pour le problème de tournées de véhicules multi dépôts avec fenêtre de temps, flotte hétérogène limitée.*

Toutes les contraintes du modèle de base sont présentées en page 33.

$$\sum_{i\in J} x_{s^k i}^{kd} - \sum_{i\in J} x_{if^k}^{kd} = 0 \qquad \forall k \in K, \forall d \in D \tag{4.1}$$

$$\sum_{j\in J} x_{s^k j}^{kd} \leq 1 \qquad \forall k \in K, \forall d \in D \tag{4.2}$$

$$\sum_{k\in K} y_i^k = 1 \qquad \forall i \in J \tag{4.3}$$

$$y_i^k - \sum_{d\in D} y_i^{kd} = 0 \qquad \forall i \in \Theta \cup \Delta, \forall k \in K \tag{4.4}$$

$$y_i^k - \sum_{d\in D} y_i^{kd} \leq 0 \qquad \forall i \in \Omega, \forall k \in K \tag{4.5}$$

$$|D| \times y_i^k - \sum_{d\in D} y_i^{kd} \geq 0 \qquad \forall i \in \Omega, \forall k \in K \tag{4.6}$$

$$y_j^{kd} - \sum_{i\in\mathcal{N}^-} x_{ij}^{kd} = 0 \qquad \forall j \in J, \forall k \in K, \forall d \in D \tag{4.7}$$

$$y_i^{kd} - \sum_{j\in\mathcal{N}^+} x_{ij}^{kd} = 0 \qquad \forall i \in J, \forall k \in K, \forall d \in D \tag{4.8}$$

$$\sum_{i,j\in\mathcal{S}} x_{ij}^k \leq |\mathcal{S}| - 1 \qquad \forall k \in K, \forall \mathcal{S} \subset \mathcal{N} \ t.q. \quad 2 \leq |\mathcal{S}| \leq n-2 \tag{4.9}$$

$$\sum_{d\in D} p_i^{kd} = p_i^k \times y_i^k \qquad \forall i \in \Omega, \forall k \in K \tag{4.10}$$

$$y_i^k \leq q_i^k \qquad \forall i \in J \backslash \Theta, \forall k \in K \tag{4.11}$$

$$u_i^{kd} + p_i^k + t_{ij} - M \times (1 - x_{ij}^{kd}) \leq u_j^{kd} \qquad \forall i \in \Theta \cup \Delta, \forall j \in \mathcal{N}^+, \forall k \in K, \forall d \in D \tag{4.12}$$

$$u_i^{kd} + p_i^{kd} + t_{ij} - M \times (1 - x_{ij}^{kd}) \leq u_j^{kd} \qquad \forall i \in \Omega, \forall j \in \mathcal{N}^+, \forall k \in K, \forall d \in D \tag{4.13}$$

$$u_{s^k}^{kd} + t_{s^k j} - M \times (1 - x_{s^k j}^{kd}) \leq u_j^{kd} \qquad \forall j \in J, \forall k \in K, \forall d \in D \tag{4.14}$$

$$u_i^{kd} \geq r_i \times y_i^{kd} \qquad \forall i \in J, \forall k \in K, \forall d \in D \tag{4.15}$$

$$u_i^{kd} \leq (dc_i - p_i^k) \times y_i^{kd} \qquad \forall i \in \Theta \cup \Delta, \forall k \in K, \forall d \in D \tag{4.16}$$

$$u_i^{kd} + p_i^{kd} \leq dc_i \times y_i^{kd} \qquad \forall i \in \Omega, \forall k \in K, \forall d \in D \tag{4.17}$$

$$u_{s^k}^{kd} \geq a^{kd} \qquad \forall k \in K, \forall d \in D \tag{4.18}$$

$$u_{f^k}^{kd} \leq b^{kd} \qquad \forall k \in K, \forall d \in D \tag{4.19}$$

$$\left(y_i^{kd_0} - y_i^{kd_1}\right) - \left(y_i^{kd_1} - y_i^{kd_2}\right) \leq 1 \qquad \forall i \in \Omega, \forall k \in K, \forall d_0 < d_1 < d_2 \in D \tag{4.20}$$

$$x_{ij}^{kd} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{V}, \forall k \in K, \forall d \in D$$

$$y_i^k \in \{0,1\} \qquad \forall i \in J, \forall k \in K$$

$$y_i^{kd} \in \{0,1\} \qquad \forall i \in J, \forall k \in K, \forall d \in D$$

$$u_i^{kd} \in \mathbb{R}^+ \qquad \forall i \in \mathcal{N}, \forall k \in K, \forall d \in D$$

$$p_i^{kd} \in \mathbb{R}^+ \qquad \forall i \in \Omega, \forall k \in K, \forall d \in D$$

contraintes	y_i^k	y_i^{kd}	x_{ij}^{kd}	u_i^{kd}	p_i^{kd}
(4.1) & (4.2) & (4.9)			■		
(4.3) & (4.11)	■				
(4.4) & (4.5) & (4.6)	■	■			
(4.7) & (4.8)		■	■		
(4.10)	■				■
(4.12) & (4.15) & (4.16)			■	■	
(4.13) & (4.17)			■	■	■
(4.18) & (4.19)				■	
(4.20)		■			

TAB. 4.4 – Modèle de base pour les tournées de véhicules multi dépôts avec fenêtre de temps, flotte hétérogène limitée et préemption. Représentation des variables et contraintes couplantes.

4.4 Modèle avec contraintes de précédence

Nous devons ici étendre le modèle de base afin de prendre en compte des contraintes de précédence.

4.4.1 Hypothèses et notations

1. Hypothèses retenues en 4.3.1.
2. Il existe des contraintes de précédence.

4.4.2 Formulation mathématique

On ajoute des contraintes pour les jobs liés par des contraintes de précédence $i \rightarrow j$. D'après nos hypothèses l'ensemble $\{(i,j)|i \neq j \text{ et } i \rightarrow j\}$ est un sous-ensemble de $\Delta \times \Delta \cup \Omega$.

$$y_i^k = y_j^k \quad \forall i \rightarrow j \subset \Delta \times (\Delta \cup \Omega), \forall k \in K \tag{4.21}$$

$$\sum_{k \in K} \left(y_j^{kd} + y_j^{kd+1}\right) \geq \sum_{k \in K} y_i^{kd} \quad \forall i \rightarrow j \subset \Delta \times (\Delta \cup \Omega), \forall d \in D \tag{4.22}$$

$$u_i^{kd} \leq u_j^{kd} \quad \forall i \rightarrow j \subset \Delta \times (\Delta \cup \Omega), \forall k \in K, \forall d \in D \tag{4.23}$$

Le même technicien doit exécuter les deux jobs concernés par la contrainte de précédence (4.21). La contrainte (4.22) impose que le technicien commence le job j au plus tard le lendemain de l'exécution du job i. Cette contrainte signifie qu'un technicien ne doit pas passer chercher le matériel nécessaire à une intervention trop longtemps avant de commencer cette intervention. La contrainte (4.23) nous assure que si les deux tâches sont effectuées le même jour, alors la tâche i est exécutée avant la tâche j.

4.5 Modèle avec contraintes de synchronisation

4.5.1 Hypothèses et notations

1. Hypothèses retenues en 4.4.1.
2. Il existe des jobs liés par des contraintes de synchronisation.

Nous présentons les notations supplémentaires associés au modèle avec contraintes de synchronisation dans le tableau 4.5 page 34.

Modèle avec contraintes de synchronisation	
Δ_1	jobs sans préemption, sans contrainte de synchronisation
Δ_2	jobs sans préemption, avec contrainte de synchronisation
Ω_1	jobs avec préemption, sans contrainte de synchronisation
Ω_2	jobs avec préemption, avec contrainte de synchronisation
$\alpha \in [0,1]$	pourcentage des tâches synchronisées devant être effectué en commun par les deux techniciens.
variables	
t_i^d	temps de travail commun aux deux techniciens pour un job $i \in \Omega_2$ un jour d.
y_i^d	variable binaire valant 1 si un technicien quelconque travaille sur le job $i \in \Omega_2$ le jour d.

TAB. 4.5 – Tableau des notations relatives au modèle avec contraintes de synchronisation.

4.5.2 Formulation mathématique

L'ensemble des jobs est maintenant $J = \Theta \cup \Delta_1 \cup \Delta_2 \cup \Omega_1 \cup \Omega_2$.

Une contrainte de synchronisation consiste à envoyer deux techniciens sur une même intervention, elle concerne donc les jobs $i \in \Delta_2 \cup \Omega_2$. Nous distinguons plusieurs types de synchronisation. Quel que soit le type de synchronisation choisi, il faut transformer certaines contraintes du modèle de base :

$$(4.3) \rightarrow \sum_{k \in K} y_i^k = 2 \quad \forall i \in \Delta_2 \cup \Omega_2 \tag{4.24}$$

$$\sum_{k \in K} q_i^k \times y_i^k \geq 2 \quad \forall i \in \Delta_2 \cup \Omega_2 \tag{4.25}$$

$$(4.10) \rightarrow \sum_{d \in D} \sum_{k \in K} p_i^{kd} \geq \sum_{k \in K} \left(p_i^k \times y_i^k\right) \quad \forall i \in \Omega_2 \tag{4.26}$$

La contrainte (4.24) impose que deux techniciens effectuent les jobs avec contraintes de synchronisation. La contrainte (4.25) impose qu'un des deux techniciens présents sur l'intervention a un niveau de compétence supérieur à 2. La contrainte (4.26) nous assure que les techniciens ont achevé le job. En effet, leur temps de travail cumulé est supérieur à $2p$ (le temps qu'aurait pris le technicien le plus qualifié seul).

Si un job j possède une contrainte de synchronisation et une contrainte de précédence $\rightarrow j$, on peut vouloir affaiblir la contrainte de synchronisation pour le job i en n'autorisant qu'un seul technicien a l'exécuter. Ceci peut être réalisé en transformant une contrainte :

$$(4.21) \rightarrow y_i^k \quad \leq \quad y_j^k \quad \forall i \rightarrow j \subset \Delta \times (\Delta \cup \Omega), \forall k \in K (4.16) \tag{4.27}$$

Nous allons maintenant présenter les contraintes supplémentaires associées à la synchronisation totale et à la synchronisation partielle.

synchronisation totale

Les contraintes à ajouter au modèle pour inclure la synchronisation totale sont les suivantes :

$$u_i^{kd} \leq u_i^{k'd} + M \times (1 - y_i^k) \quad \forall i \in \Delta_2 \cup \Omega_2, \forall k, k' \in K, \forall d \in D \tag{4.28}$$
$$p_i^{kd} \leq p_i^{k'd} + M \times (1 - y_i^k) \quad \forall i \in \Omega_2 \in \mathcal{N}, \forall k, k' \in K, \forall d \in D \tag{4.29}$$

Les deux techniciens commencent aux mêmes moments (4.28) et travaillent pendant la même durée (4.29). Ces deux contraintes représentent en fait des contraintes d'égalité pour les techniciens réalisant les opérations, grâce à la symétrie de k et k'. On peut choisir $M = t_{max}$.

On remarque que ce type de synchronisation peut limiter le nombre d'heure de travail du technicien le plus proche du site du job.

synchronisation partielle pour les jobs longs

On n'impose plus que la totalité de la tâche soit exécutée simultanément mais seulement un pourcentage $\alpha \in [0, 1]$ pour les jobs $i \in \Omega_2$. On garde la contrainte de synchronisation totale pour les jobs courts ($\in \Delta_2$). Nous allons exposer le raisonnement qui aboutit au système de contraintes que nous ajoutons. Pour un jour donné, soit une tâche J_1 que le technicien 1 exécute dans l'intervalle $[a_1, b_1]$ et le technicien 2 exécute dans l'intervalle $[a_2, b_2]$, le temps passé en commun t_c est donné par la formule :

$$t_c = \max(0, \min(b_1, b_2) - \max(a_1, a_2))$$

Cette contrainte peut se modéliser de manière linéaire avec le système de contraintes suivant :

$$\begin{aligned} t_c &\leq b_1 - a_1 \\ t_c &\leq b_2 - a_2 \\ t_c &\leq b_1 - a_2 \\ t_c &\leq b_2 - a_1 \\ t_c &\geq 0 \end{aligned}$$

Or si les deux intervalles d'exécution sont disjoints $a_1 > b_2$ ou $a_2 > b_1$ ce système est insatisfiable car $b_1 - a_2 < 0$ ou $b_2 - a_1 < 0$. Donc, si on choisit de modéliser la synchronisation partielle par ce système de contraintes, on impose que deux techniciens travaillant un même jour sur un job se croisent. Cette contrainte est donc implicite mais on peut la justifier par la nécessité d'échanger des informations sur l'avancement du travail. Sous réserve d'accepter cette hypothèse, on peut donc créer un système de contraintes similaire adapté à notre problème :

$$t_i^d \leq u_i^{kd} + p_i^{kd} - u_i^{k'd} + M \times (2 - y_i^{kd} - y_i^{k'd}) \quad \forall i \in \Omega_2, \forall k, k' \in K, \forall d \in D \tag{4.30}$$
$$t_i^d \in \mathbb{R}^+ \quad \forall i \in \Omega_2, \forall d \in D \tag{4.31}$$

La contrainte (4.30) représente les deux dernières équations de l'exemple précédent et impose que le temps passé en commun est inférieur à l'intersection des intervalles. Nous devons vérifier le pourcentage du travail réalisé par les deux techniciens.

$$\sum_{d\in D} t_i^d \quad \geq \quad \alpha/2 \times \sum_{k\in K} \left(p_i^k \times y_i^k\right) \quad \forall i \in \Omega_2, \forall k \in K \tag{4.32}$$

La contrainte (4.32) garantit que les techniciens ont travaillé plus de $\alpha\%$ du temps ensemble.

On remarque que l'on doit modifier la contrainte (4.20) car le travail est réalisé par les deux techniciens et cette contrainte peut être violé par l'un des deux techniciens. On s'aperçoit sur l'exemple du tableau 4.6 que l'emploi du temps du technicien 2 ne respecte pas la contrainte (4.20). Or si on considère les jours travaillés sur le job (dernière ligne du tableau), on s'aperçoit que les jours où les techniciens ont travaillé sont consécutifs. Il faut modifier la contrainte (4.20) pour intégrer le fait que les jours travaillés sur un job peuvent être dus à l'un ou à l'autre des techniciens.

Planning \ Jour	1	2	3	4
Tech 1	0	1	1	0
Tech 2	1	0	1	0
job	1	1	1	0

TAB. 4.6 – Exemple où l'affectation de deux techniciens à un job viole la contrainte (4.20) pour un des deux techniciens.

La contrainte (4.20) devient $\rightarrow$

$$|K| \times y_i^d \quad \geq \quad \sum_{k\in K} y_i^{kd} \quad \forall i \in \Omega_2, \forall d \in D \tag{4.33}$$

$$y_i^d \quad \leq \quad \sum_{k\in K} y_i^{kd} \quad \forall i \in \Omega_2, \forall d \in D \tag{4.34}$$

$$\left(y_i^{d_0} - y_i^{d_1}\right) - \left(y_i^{d_1} - y_i^{d_2}\right) \quad \leq \quad 1 \quad \forall i \in \Omega_2, \forall d_0 < d_1 < d_2 \in D \tag{4.35}$$

$$y_i^d \quad \in \quad \mathbb{R}^+ \quad \forall i \in \Omega_2, \forall d \in D$$

Les contraintes (4.33) et (4.34) assignent la valeur 1 à la variable y_i^d si un technicien travaille sur le job i le jour d. La contrainte (4.35) est une réécriture de la contrainte (4.20) avec cette nouvelle variable. On a étendu la contrainte de connexité à un job réalisé par plusieurs techniciens.

4.6 Discussion sur les objectifs

Le but de la problématique est donc de réaliser la planification, avec comme base l'ensemble de données présenté en section 4.1, en fonction de différents objectifs. On distingue deux types d'objectifs, la minimisation des coûts et la maximisation de la satisfaction client.

Les coûts sont liés aux trajets des techniciens, au respect des dates contractuelles et à l'homogénéité des plans de charge pour les techniciens. La satisfaction des clients est liée au respect des dates contractuelles et si possible au respect des dates souhaitées, ainsi qu'à la qualité des interventions (compétences du technicien, temps de latence court, ...).

4.7 Objectif : minimisation des coûts

4.7.1 Objectif du modéle avec fenêtres de temps dures

Nous pouvons dès maintenant définir une fonction économique associée à la minimisation des coûts qui intègre les déplacement des techniciens :

$$\min \sum_{d\in D} \sum_{k\in K} \sum_{(i,j)\in \mathcal{V}} \left(\tau \times (t_{ij} \times x_{ij}^{kd})\right)$$

4.7.2 Objectif avec relaxation des contraintes associées aux dates contractuelles

Si on relaxe les contraintes (4.16) et (4.17) associées aux dates contractuelles, on choisit de les intégrer comme une pénalité pour la fonction objectif. On doit alors créér la variable d_i représentant la date de fin d'exécution d'un job $i \in J$. Malgré l'assouplissement des fenêtres de temps, on veut garder la propriété $y_i^{kd} = 0 \Rightarrow u_i^{kd} = 0$ et si $i \in \Omega$ $p_i^{kd} = 0$. On va donc modifier les contraintes (4.16) et (4.17).

$$
\begin{array}{rrcll}
(4.16) \rightarrow & u_i^{kd} & \leq & M \times y_i^{kd} & \forall i \in \Theta \cup \Delta, \forall k \in K, \forall d \in D \\
(4.17) \rightarrow & u_i^{kd} + p_i^{kd} & \leq & M \times y_i^{kd} & \forall i \in \Omega, \forall k \in K, \forall d \in D \\
& d_i & \geq & \displaystyle\sum_{k \in K} \sum_{d \in D} (u_i^{kd} + p_i^k \times y_i^{kd}) & \forall i \in \Theta \cup \Delta_1 \qquad (4.36) \\
& d_i & \geq & \displaystyle\sum_{d \in D} (u_i^{kd} + p_i^k \times y_i^{kd}) & \forall i \in \Delta_2, \forall k \in K \qquad (4.37) \\
& d_i & \geq & \displaystyle\sum_{k \in K} (u_i^{kd} + p_i^{kd}) & \forall i \in \Omega_1, \forall d \in D \qquad (4.38) \\
& d_i & \geq & u_i^{kd} + p_i^{kd} & \forall i \in \Omega_2, \forall k \in K, \forall d \in D \qquad (4.39) \\
& d_i & \in & \mathbb{R}^+ & \forall i \in J \qquad (4.40)
\end{array}
$$

$$
\min \sum_{d \in D} \sum_{k \in K} \sum_{(i,j) \in \mathcal{V}} \left(\tau \times (t_{ij} \times x_{ij}^{kd})\right) + \sum_{i \in J} (c_i \times \max(0, d_i - dc_i))
$$

Cette seconde fonction économique peut être utilisée dans le cas où le problème est insatisfiable avec les contraintes (4.16) et (4.17).

4.8 Objectif : maximisation de la satisfaction client

La réalité incontournable du nouvel environnement industriel est une compétition extrême qui fait qu'aujourd'hui le client est placé au centre. Nous cherchons donc à améliorer son niveau de satisfaction et introduisons ce dernier comme objectif dans notre modèle. Ce niveau de satisfaction peut dépendre, dans notre cas, de plusieurs facteurs : le délai de livraison, la qualité du produit, le service après vente, etc. L'aspect qui nous intéresse ici est lié aux délais de livraison, qui est un élément très important pour l'image de marque de l'entreprise. Jusqu'ici notre objectif, en terme de satisfaction client, se limitait uniquement à l'aspect financier des pénalités de retard, ce qui nous parait insuffisant. En effet, s'il est pertinent de considérer que la satisfaction client est liée au respect des dates contractuelles par un opérateur, elle dépend également de sa date souhaitée (liée à son activité) d'installation du service.

Cependant, en raison de la durée limitée de ce stage, nous n'avons pas pu proposer un objectif intégrant la durée totale de trajet (nécéssaire à la résolution d'un Vehicle Routing Problem), le respect des dates contractuelles et le respect des dates souhaitées. Nous avons donc limité l'aspect satisfaction client au respect des engagements pris par France Telecom (les dates contractuelles).

CHAPITRE 5

IMPLÉMENTATION ET TESTS

Afin de valider nos modèles linéaires, nous les avons implémenter en JAVA en utilisant le solveur commercial CPLEX. Pour toute information sur cette implémentation ou sur l'utilisation du logiciel, vous pourrez vous referez au manuel de l'utilisateur fourni en annexe A page 64.

5.1 Implémentation et modèles

Dans cette section, nous allons présenter certaines différences entre les modèles et l'implémentation. Ces différences concernent des optimisations de la contrainte de sous-tours ainsi que des contraintes supplémentaires et des propriétés de la solution optimale.

5.1.1 La contrainte d'élimination des sous-tours

Optimisation implémentée

La combinatoire de la contrainte (4.9) explosent avec la taille de l'instance et se révèle donc un obstacle important pour la résolution grâce à CPLEX. En effet, la contrainte des sous-tours (4.9) concerne normalement tous les sous-ensembles de S de cardinal 2 à $n-2$, soit $O(2^n)$ sous-ensembles.
Cependant tous les sous-ensembles de tâches ne sont pas réalisables à cause des contraintes temporelles. Vu la croissance exponentielle du nombre de contraintes de sous-tours, nous avons tenté d'en éliminer le maximum. Nous avons exploré deux pistes :

- Limiter de manière empirique la taille maximale des sous-tours interdits
- Utiliser la durée des tâches pour éliminer des sous-ensembles de sommets.

Le premier critère est fixer par un pré-traitement ayant analysé les données.
Le second critère est facile à fixé pour les jobs $\in \Theta \cup \Delta$, mais plus difficile pour les jobs de Ω (préemption autorisée) car on ne peut pas prévoir à l'avance la durée réalisée du job. Nous avons donc renforcé la contrainte (4.9) en éliminant les sous-ensembles tels que la somme de la durée des tâches du sous-ensemble est supérieure à la durée maximale d'une journée de travail :

$$\sum_{i,j \in \mathcal{S}} x_{ij}^k \quad \leq \quad |\mathcal{S}| - 1 \quad \forall k \in K, \forall \mathcal{S} \subset \mathcal{N} \text{ t.q. } \quad 2 \leq |\mathcal{S}| \leq n-2$$

$$\text{et} \sum_{i \in \mathcal{S} \cap (\Theta \cup \Delta)} p_i^k \leq d_{max}$$

Optimisation réalisable dans le cas sans préemption

Tout d'abord, nous noterons $p_i = \min_{k \in K} p_i^k$. Soit un sous-ensemble $E \subset \mathcal{N}$ de jobs, nous noterons $C(E)$ la condition : $\sum_{i \in E} p_i \leq d_{max}$.

Borne supérieure sur la taille des sous-ensembles Soit $\mathcal{N} = \{j_1, \ldots, j_n\}$ tel que $\forall i p_{j_i} \leq p_{j_{i+1}}$, on peut trouver la borne supérieure sur le cardinal des sous-ensembles $E \subset \mathcal{N}$ tels que : $w(E) = \sum_{i \in E} p_i < d_{max}$. En effet, il est facile de démontrer que pour tout sous-ensemble $E \subset \mathcal{N}$ on a $w(E) \geq \sum_{l=1}^{l=|E|} p_{j_l}$. Ainsi, la borne supérieure sur la taille des sous-ensembles est donné par l'équation :

$$\arg_m \min(\sum_{l=1}^{l=m} p_{j_l} \geq d_{max})$$

Élimination dynamique de sous-ensembles Nous allons donner un exemple des sous-ensembles de cardinal 3 générés par notre algorithme pour 6 jobs dans le tableau 5.1 page 39. Si on garde la liste

012	013	014	015	023	024	025	034	035	045
123	124	125	134	135	145				
234	235	245							
345									

TAB. 5.1 – Énumération des sous-ensembles de cardinal 3 d'un ensemble de cardinal 6.

triée et vu notre algorithme de génération des sous-ensembles, on voit que dès qu'un sous-ensemble d'une taille donnée viole la contrainte, il est inutile de générer les suivants qui la violeront aussi. En effet, on remarque que si le sous-ensemble 023 viole la condition C(023), alors les conditions C(024) et C(025) ne seront pas respectées, ainsi que pour le reste de la ligne du tableau.

5.1.2 La contrainte k-split

Nous avons aussi rajouté une contrainte spécifique au SDVRP présentée en section 3.8 :

$$\sum_{k \in K, d \in D} x_{ij}^{kd} \leq 1 \quad \forall (i,j) \in \mathcal{V}$$

Cette contrainte est plus forte que celle impliquée par la propriété du k-split cycle, car elle traite les tâches sans préemption. Mais cette contrainte accélère souvent la résolution des programmes en nombre entiers (voir (Archetti *et al.*, 2005a)). L'ajout de cette contrainte ne modifie pas l'ensemble des solutions admissibles, mais nous espérons qu'elle puisse accélérer la résolution.

5.1.3 Solution admissible et Solution optimale

Lors des tests, il arrivait fréquemment que le programme ne trouve pas la solution optimale dans le temps imparti (1h). Nous analysions alors la meilleure solution entière trouvée.

Or, soit s_0, une solution admissible pour notre modèle de base (figure 4.4) telle que :

$$\exists i \in J, k \in K, d \in D \text{ tels que : } y_i^{kd} = 1 \text{ et } p_i^{kd} = 0$$

Soit s_0' une solution qui ne diffère de s_0 qu'en supprimant ce passage par le job i. Alors s_0' aura une valeur de l'objectif inférieure ou égal à s_0 à cause de l'inégalité triangulaire et la tâche i finira au plu tard à la même date que pour s_0.

On peut donc si nécessaire ajouter une méthode permettant de « lisser » la solution en enlevant tous les déplacements ou $p_i^{kd} = 0$.

5.2 Tests

Nous n'avons pas pu réaliser d'expériences sur des jeux de données issus de la réalité opérationnelle de France Telecom. En effet, nous n'avons pas pu obtenir les informations concernant le réseau géographique. Nous avons donc analysé la répartition des commandes et avons créé un générateur d'instances.

5.2.1 Génération des instances

Les fichiers de commandes ont été générés avec le script genData.perl. Pour chaque fichier de commandes, on a fait varier un certains nombres de paramètres :

1. Le nombre de jours de planification : entre 4 et 7.
2. Le nombre de techniciens : entre 4 et 6.
3. Éventuellement les fichiers de géographie (définissant le réseau géographique).

Tous les tests ont été réalisés avec une limite de temps fixée à 1 heure. Les instances sont de petites tailles car l'explosion combinatoire des contraintes de sous-tours complique énormément la résolution quand elles ne provoquent pas une erreur de la pile mémoire de CPLEX.

5.2.2 Génération des graphiques

Le protocole de test implémenté nous permettait de générer cinq types de graphiques :

1. Taux de solution optimale trouvée : une solution est considérée comme optimale si le problème est infaisable, non bornée ou la solution optimale a été trouvée.
2. taux aucune solution entière trouvée
3. nombre moyen de noeuds visités quel que soit le résultat des calculs
4. gap initial moyen : concerne toutes les instances pour lesquels on a trouvé au moins une solution entière
5. gap final moyen : concerne les instances stoppées par la limite de temps(1h).

5.2.3 Résultats

Instances sans préemption

On peut remarquer deux phénomènes sur la série de tests présentée sur les deux premières lignes de la figure 5.1 41 :

- L'importance du nombre de jours de planification choisi pour trouver rapidement une solution optimale ainsi que pour minimiser le nombre de noeuds visités.
- L'augmentation très importante de la complexité lorsqu'on ajoute un job. En effet, pour les instances de 7 jobs, on trouve la solution optimale même avec une mauvaise estimation du nombre de jours, alors que ce n'est plus le cas pour des instances à 8 jobs. La difficulté à trouver la solution optimale est aussi révélée par le nombre de noeuds visités lors de la résolution.
- On remarque que la variation du nombre de techniciens n'influe pas énormément sur la difficulté de la résolution.

Nous avons voulu tester l'influence de la géographie sur la difficulté du problème, nous avons donc réutiliser le fichier de commandes avec 7 jobs mais en utilisant une autre localisation des sites associés aux commandes. Nos présentons les résultats sur la dernière lignes des figures 5.1.

On remarque que ce seul changement de géographie complique la résolution. Peu de solutions optimales sont trouvées, mais surtout, il y a un nombre non négligeable d'instances pour lesquelles aucune solution entière n'a été trouvée dans le temps imparti. On voit également que l'augmentation du nombre de noeuds visités illustre la complexité.

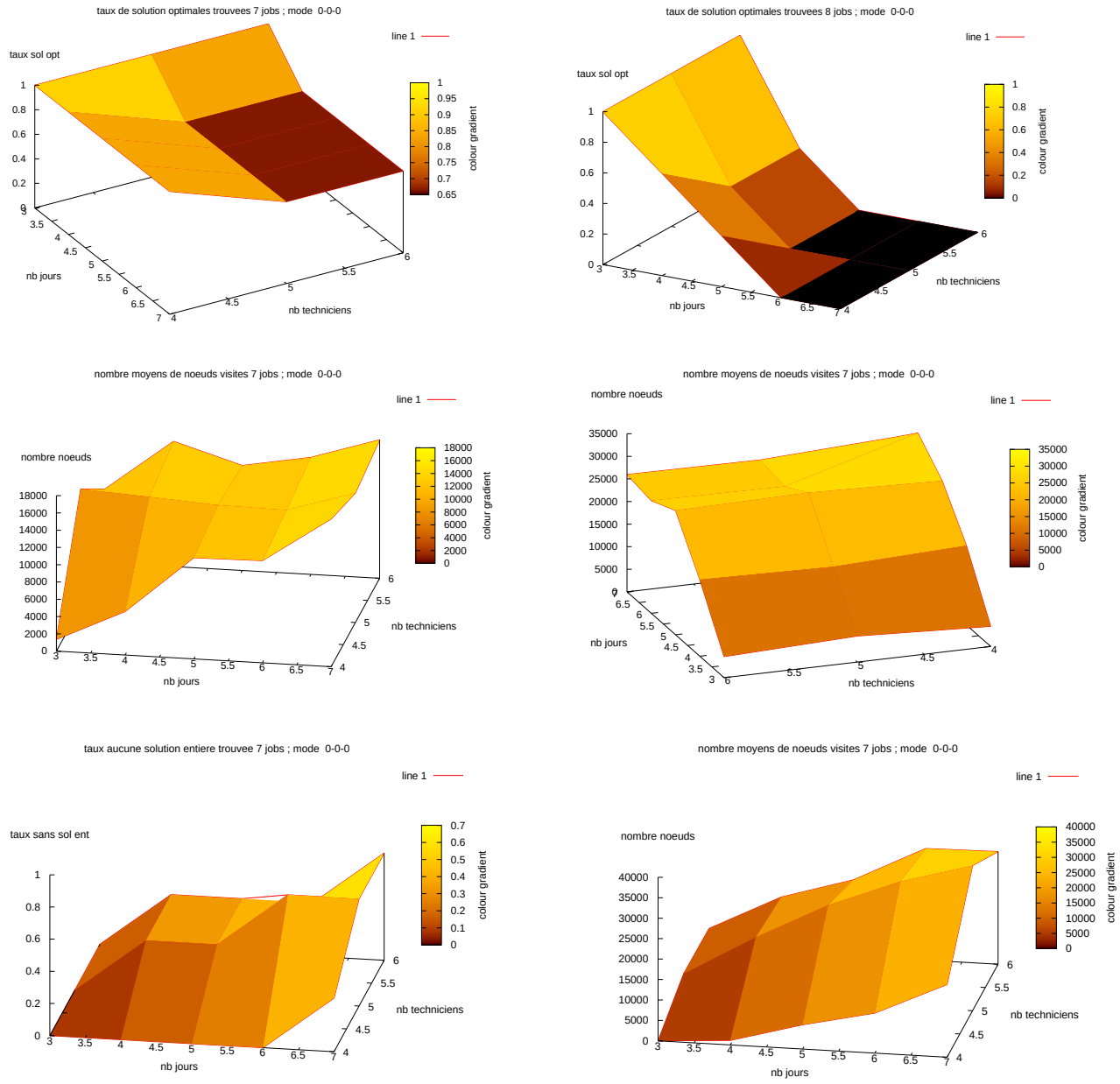

FIG. 5.1 – Résultats des tests pour 3 instances de 7 et 8 jobs sans préemption, sans précédence et sans synchronisation avec une limite de temps d'1h (2 premières lignes géographie random, dernière ligne random and cluster voir A). Errata : 2ème ligne à gauche 8 jobs au lieu de 7.

Instances avec préemption

Nous avons lancé une autre série de tests avec des instances contenant des jobs longs, c'est-à-dire des instances pour lesquelles la préemption est autorisée. Nous avons fait varier les paramètres dans les mêmes intervalles, les problèmes ont été détectés comme infaisables pour 3 et 4 jours de planification. Nous présentons donc les résultats pour lesquels le nombre de jours de planification varie entre 5 et 7 jours. On peut remarquer facilement en regardant le modèle la complexité supplémentaire lié à l'ajout de la préemption. Ce phénomène a été confirmé par l'expérimentation. Nous présentons les résultats en figure 5.2. On remarque pour la première fois que l'augmentation du nombre de techniciens peut faciliter le problème. Un choix judicieux du nombre de jours de planification est toujours nécessaire pour être capable de prouver l'optimalité d'une solution. En effet, CPLEX n'arrive plus à élaguer suffisamment de branches de l'arbre de recherche et ne peut donc pas prouver l'optimalité de la solution. On remarque que le nombre moyen de noeuds visités est plus petit que dans les expériences précédentes ; ceci est

principalement du au temps nécessaire pour résoudre les sous-problèmes, car l'ajout de la préemption ajoute des variables entières p_i^{kd}. De plus le nombre de noeuds confirme que l'augmentation du nombre de techniciens peut simplifier le problème.

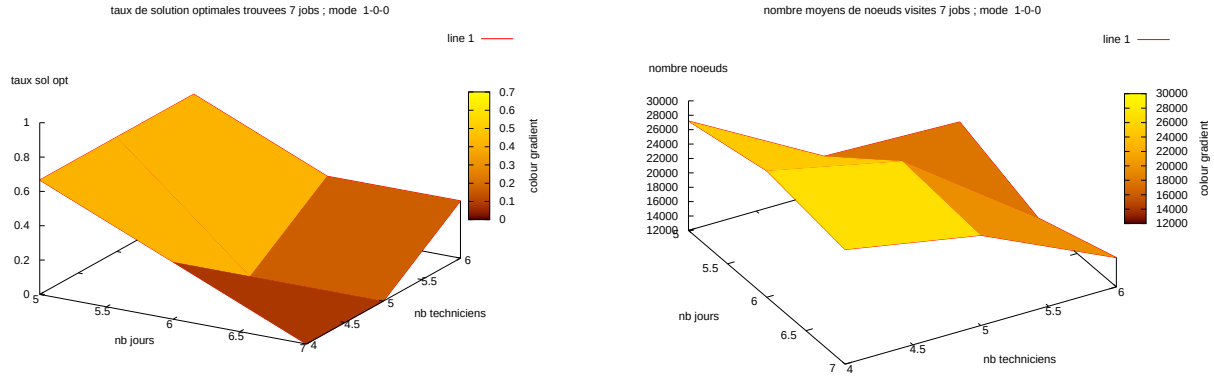

FIG. 5.2 – Résultats des tests pour 3 instances de 7 avec préemption, sans précédence et sans synchronisation avec une limite de temps d'1h (géographie random).

Un constat empirique que nous avons pu réaliser est que l'augmentation du nombre de jours de planification ne détériore pas toujours la qualité de la solution (la valeur de la fonction objectif) mais elle empêche le solveur de prouver son optimalité car celui-ci n'a pas le temps d'éliminer toutes les branches de l'arbre d'exploration. Nous n'avons pas réalisé de séries de tests sur les modèles avec précédence ou synchronisation, car on voit que la complexité du problème nous pousse à trouver d'autres moyens de résolutions qu'une formulation linéaire orienté arcs. Cependant, nous avons validé les quatre modèles sur de petits exemple, mais nous n'avons pas réalisé de tests de performance sur tous les modèles.

Conclusions

- **Une bonne estimation du nombre de jours de planification facilite l'obtention de solution de qualité ainsi que la preuve de l'optimalité**
- **L'augmentation du nombre de techniciens peut faciliter la résolution du problème**
- **La résolution d'instances de taille raisonnable grâce aux modèles combinés à l'utilisation de CPLEX n'est pas envisageable.**
- **Une optimisation de la contrainte des sous-tours améliorerait les capacités de résolution du programme.**

CHAPITRE 6

MÉTHODES APPROCHÉES

Les limitations constatées lors nos approches de résolution par méthodes exactes nous a contraint à explorer des approches de résolution par heuristique. Dans un démarche hiérarchique, nous nous sommes d'abord intéressé à une problématique plus simple en termes de modéle correspondant à la problématique traitée précedemment pour un sous-ensemble des services de France Telecom (services aux particuliers).

Nous allons d'abord présenté les méthodes approchées associées au Vehicle Routing Problem with Time Window et l'énoncé du problème qui peut être considéré comme un cas particulier du Vehicle Routing Problem with Time Window avec flotte hétérogène ou un sous-problème de notre modèle de base (voir chapitre 4). Nous présenterons ensuite une heuristique de construction très utilisée dans les problèmes de tournées de véhicules et enfin l'adaptation cette heuristique à notre problème, ainsi que les limites et améliorations éventuelles de notre méthode.

6.1 Méthodes approchées pour le Vehicle Routing Problem with Time Window

Étant donné la complexité du Vehicle Routing Problem with Time Window , de nombreuses heuristiques ont été développées. Nous établissons une différence entre les heuristiques simples permettant la construction d'une solution et les métaheuristiques qui essaient d'améliorer la solution initiale obtenue par une heuristique grâce à des méthodes de recherche locale tentant d'éviter les optimums locaux.

6.1.1 Heuristiques simples

Construction de tournées

Cette famille d'algorithme construit une solution réalisable en ajoutant à chaque itération un job qui n'appartient à aucune tournée. Ce processus peut s'exécuter de manière séquentiel (une tournée à la fois) ou parallèle (plusieurs tournées simultanément). Deux facteurs principaux déterminent la qualité de ces heuristiques : quel job insérer à chaque itération ? dans quelle tournée insérer ce job ?

Cependant, la présence de fenêtres de temps complique la construction de solutions réalisables. La méthode *Max Savings* de (Clarke & Wright, 1964) utilisée pour le Vehicle Routing Problem ne fonctionne plus de manière satisfaisante. Une heuristique d'insertion séquentielle a alors été proposée (Solomon, 1987). Cette heuristique est en deux phases ; premièrement chaque job est inséré de façon à minimiser la distance et le temps supplémentaire ; deuxièmement la méthode sélectionne le client à insérer en utilisant le concept de *Max Savings*. Une version parallèle a été proposé par (Potvin & Rousseau, 1993). Une dernière heuristique proposée par (Ioannou *et al.*, 2001) utilise le concept d'insertion séquentielle mais en utilisant un nouveau critère pour la sélection du job. L'idée de base étant que l'insertion du job sélectionné doit minimiser l'impact sur les jobs déjà planifiés, sur les jobs non planifiés et sur la fenêtre de temps du job lui-même.

Amélioration de tournée

Ces méthodes améliorent itérativement une solution en effectuant une recherche locale (dans un voisinage de la solution). Pour créer une méthode de recherche locale on a besoin de spécifier certains choix :

- Comment est générée la solution initiale ?
- Quel est le voisinage choisi ?
- Le critère d'admissibilité d'une nouvelle solution.
- Le critère d'arrêt de la recherche.

Dans notre cas, la solution initiale sera générée par une méthode de construction de tournée.
Les voisinages les plus utilisés pour le Vehicle Routing Problem with Time Window sont basés sur des échanges d'arcs ou de noeuds. Par exemple, un voisinage très utilisé pour les problèmes mono tournée est le voisinage λ-opt qui consiste en des échanges d'arcs : un mouvement retire λ arcs d'une tournée et à les réinsère en les croisant. Une solution est dite λ-optimale lorsqu'il n'existe plus de croisement permettant de l'améliorer. Cependant, même pour des valeurs de λ petite ces voisinages sont très grands. Pour les problèmes multi-tournées, nous reprenons la classification de (Fabien Tricoire, 2006) :

- *String cross* (croisement de chaîne) consiste à croiser deux tournées en échangeant des chaînes de noeuds entre ces tournées.
- *String exchange* (échange de chaîne) consiste à échanger entre deux tournées des chaînes de noeuds de longueur bornée.
- *String relocation* (déplacement de chaîne) consiste à déplacer une chaîne de noeuds d'une tournée à l'autre.
- *String mix* consiste à choisir le meilleur mouvement entre les échanges et déplacements possibles.

Pour des raisons combinatoires, la longueur des chaînes pour le déplacements et l'échange est souvent courte (1 ou 2). Par contre il n'est pas *à priori* possible de réduire la taille des croisements de cette manière.
Lorsqu'on utilise ces opérateurs, il faut faire attention que la tournée soit encore réalisable. En effet, dans le cas du croisement il est nécessaire que les deux tournées aient les mêmes points de départ et d'arrivée. Dans tous les autres cas, des contraintes de fenêtres de temps ou de compatibilité peuvent être violées.
Les critères d'admissibilité les plus répandus sont first-accept (FA) et best-accept (BA). Le critère first-accept accepte la première solution améliorante, alors que BA examine tous les voisins et sélectionne le meilleur.
Les critères d'arrêts peuvent dépendre du temps ou de l'évolution des solutions courantes.

6.1.2 Métaheuristiques

Toutes les familles de métaheuristiques ont été développées pour traiter le Vehicle Routing Problem with Time windows. Nous allons présenter brièvement les principes de ces métaheuristiques.

- *Recherche Tabou* : La recherche tabou explore l'espace sélectionnant à chaque itération la meilleure solution dans un sous-ensemble du voisinage de la solution courante. Des solutions moins bonnes peuvent être acceptées afin d'éviter de retourner dans des zones déjà explorées. Pour éviter les cycles, les solutions précédemment explorées sont déclarées tabou.
- *Recuit Simulé* : La stratégie des algorithmes de recuit consiste, en effectuant une exploration aléatoire de l'espace d'état, à favoriser les descentes, mais sans interdire tout à fait les remontées. Plus précisément, on se donne une chaîne de Markov sur l'espace d'état, et on accepte ou on refuse une transition avec une probabilité 1 si l'objectif F décroît, et une probabilité égale à $\exp(\frac{\Delta F}{T}$si F croît. Ici, T est un paramètre, appelé température par analogie avec la mécanique statistique. Tout au long de l'algorithme, T va décroître, ni trop vite pour ne pas rester bloqué autour d'un minimum local, ni trop lentement si on veut avoir un résultat en un temps raisonnable.
- *Algorithme génétique* : Une métaheuristique adaptative basée sur l'évolution des gènes. On génère une première génération de chromosomes, puis à l'aide d'opérateurs inspirés des mécanismes génétiques, on produit itérativement les générations suivantes avec un critère de convergence. Ce critère peut être le nombre maximum de génération ou la convergence vers une population de chromosomes identiques. On sélectionne ensuite le meilleur chromosome qui représente la meilleure solution trouvée.
- *Algorithme évolutionaire* : Cet algorithme est de la même famille que les algorithmes génétiques

mais présente des différences sur la façon dont sont représentées les solutions et sur l'importance des opérateurs (plus de mutations que de croisements).

– *Colonie de fourmis* : L'idée de base est qu'un grand nombre d'agents simples partageant une mémoire commune (similaire aux phéromones déposées par les fourmis) sont capables de construire une bonne solution pour des problèmes d'optimisations difficiles. Les fourmis artificielles sont implémentées comme des processus parallèles dont le rôle est de construire des solutions en utilisant leur mémoire commune, les données du problèmes et des heuristiques utilisées pour évaluer les paliers pour la construction de solutions (Gambardella *et al.*, 1999)

Les méthodes les plus récentes essaie de combiner plusieurs de ces métaheuristiques, comme par exemple (Homberger & Gehring, 2005) qui propose une méthode en deux phases : minimiser le nombre de véhicules utilisés puis minimiser la distance totale parcourue avec une recherche tabou. (Olli Bräysy & Gendreau, 2004) arrivent à la conclusion que les méthodes actuellement les plus efficaces sont les algorithmes évolutionnaires, hybridés avec des méthodes de recherche locale, comme par exemple la recherche taboue.

6.2 Énonce du problème

6.2.1 Hypothèses et notations

Nous allons présenter les hypothèses de ce nouveau problème. Ce problème peut être vu comme un sous-problème de notre modèle de base ou comme un cas particulier du Capacited Vehicle Routing Problem with Time Window avec flotte hétérogène.

1. Chaque ressource a une base de départ et d'arrivée unique.
2. La durée maximale d'une tournée associée à une ressource est d_{max}.
3. Le nombre de ressources est limité.
4. Chaque ressource peut réaliser un sous-ensemble des jobs.
5. Chaque ressource travaille tous les jours de l'horizon.
6. Les jobs sont indivisibles.
7. Chaque job possède une fenêtre de temps qui représente un intervalle de jours pour lesquels le job est réalisable.
8. La durée p_j d'un job j ne dépend pas de la ressource qui l'exécute.

Remarque 5 *Une particularité des hypothèses est que les fenêtres de temps des jobs sont en jours. Donc, si on choisit les jobs à réaliser pour un jour donné, on n'a plus de contraintes de fenêtre de temps pour la construction des tournées.*

Nous allons définir la notion de fenêtre de temps pour une tournée.

Définition 6.1 (Fenêtre de temps d'une tournée) *Soit $t = (i_0, \ldots, i_n)$ une tournée, soit $[r_l, dc_l]$ la fenêtre de temps du job i_l appartenant à la tournée, alors nous définissons la fenêtre de temps de t par : $[r_t, dc_t] = \cap_{i_l \in t}[r_l, dc_l]$. Cette fenêtre de temps correspond à l'intervalle de jour pendant lequel une tournée est réalisable en respectant les fenêtres de temps de chacun des jobs.*

Nous noterons o_i^k la variable binaire indiquant si le technicien k peut effectuer le job i.

6.2.2 Modélisation linéaire

Étant donné le travail accompli au début de ce stage, il a été facile d'adapter les modèles linéaires du chapitre 4 à ce nouvelle énoncé. En effet ce problème peut être traité comme un sous-problème de celui initialement traité. L'intérêt pratique de ce modèle est double :

– Résoudre à l'exact de petites instances et les comparer avec les résultats de l'heuristique.
– Résoudre la relaxation continue de ce modèle pour de plus grosses instances afin d'obtenir une borne inférieure sur la valeur de l'objectif.

Le modèle :

$$\sum_{i\in J} x^{kd}_{s^k i} - \sum_{i\in J} x^{kd}_{i f^k} = 0 \qquad \forall k \in K, \forall d \in D \tag{6.1}$$

$$\sum_{j\in J} x^{kd}_{s^k j} \leq 1 \qquad \forall k \in K, \forall d \in D \tag{6.2}$$

$$\sum_{k\in K} y^k_i = 1 \qquad \forall i \in J \tag{6.3}$$

$$y^k_i - \sum_{d\in D} y^{kd}_i = 0 \qquad \forall i \in \Theta \cup \Delta, \forall k \in K \tag{6.4}$$

$$y^{kd}_j - \sum_{i\in \mathcal{N}^-} x^{kd}_{ij} = 0 \qquad \forall j \in J, \forall k \in K, \forall d \in D \tag{6.5}$$

$$y^{kd}_i - \sum_{j\in \mathcal{N}^+} x^{kd}_{ij} = 0 \qquad \forall i \in J, \forall k \in K, \forall d \in D \tag{6.6}$$

$$\sum_{i,j\in \mathcal{S}} x^k_{ij} \leq |\mathcal{S}| - 1 \qquad \forall k \in K, \forall \mathcal{S} \subset \mathcal{N} \ t.q. \quad 2 \leq |\mathcal{S}| \leq n-2 \tag{6.7}$$

$$y^k_i \leq o^k_i \qquad \forall i \in J \backslash \Theta, \forall k \in K \tag{6.8}$$

$$u^{kd}_i + p_i + t_{ij} - M \times (1 - x^{kd}_{ij}) \leq u^{kd}_j \qquad \forall i \in \Theta \cup \Delta, \forall j \in \mathcal{N}^+, \forall k \in K, \forall d \in D \tag{6.9}$$

$$u^{kd}_{s^k} + t_{s^k j} - M \times (1 - x^{kd}_{s^k j}) \leq u^{kd}_j \qquad \forall j \in J, \forall k \in K, \forall d \in D \tag{6.10}$$

$$u^{kd}_i \geq r_i \times y^{kd}_i \qquad \forall i \in J, \forall k \in K, \forall d \in D \tag{6.11}$$

$$u^{kd}_i \leq (dc_i - p_i) \times y^{kd}_i \qquad \forall i \in \Theta \cup \Delta, \forall k \in K, \forall d \in D \tag{6.12}$$

$$u^{kd}_{s^k} \geq a^{kd} \qquad \forall k \in K, \forall d \in D \tag{6.13}$$

$$u^{kd}_{f^k} \leq b^{kd} \qquad \forall k \in K, \forall d \in D \tag{6.14}$$

$$x^{kd}_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{V}, \forall k \in K, \forall d \in D$$

$$y^k_i \in \{0,1\} \qquad \forall i \in J, \forall k \in K$$

$$y^{kd}_i \in \{0,1\} \qquad \forall i \in J, \forall k \in K, \forall d \in D$$

$$u^{kd}_i \in \mathbb{R}^+ \qquad \forall i \in \mathcal{N}, \forall k \in K, \forall d \in D$$

Pour une description précise de chaque équation, veuillez consulter le chapitre 4 page 25.

6.3 Heuristique d'insertion séquentielle de Solomon

6.3.1 Principe

L'heuristique séquentielle d'insertion de (Solomon, 1987) appelé I1 est une heuristique de construction de tournées pour le Vehicle Routing Problem with Time Window .

Une route est d'abord initialisée avec un client grâce à un critère d'initialisation et les clients restant sont alors insérés dans cette route jusqu'à ce qu'elle soit pleine à cause des contraintes de capacité ou des fenêtres de temps. Le critère d'initialisation est basé sur l'insertion du client le plus éloigné par rapport au dépôt ou le client avec la plus petite date de début au plus tôt. Après avoir initialisée la route, la méthode utilise deux critères $c_1(i,u,j)$ (estimation de la meilleure insertion réalisable pour un job donné) et $c_2(i,u,j)$ (balance entre l'économie dû à l'insertion du job et le coût de création d'une nouvelle tournée avec ce job) pour déterminer le prochain client non servi u à insérer entre deux clients consécutifs i_{p-1} et i_p d'une route partiellement construite $(i_0, i_1 \ldots, i_m)$ où i_0 est la base de départ du véhicule et i_m sa base d'arrivée.
Pour chaque client u non servi, on calcule d'abord le coût de la meilleure insertion réalisable :

$$c_1(i(u),u,j(u)) = \min_{p=1,\ldots,m} (c_1(i_{p-1},u,i_p) \tag{6.15}$$

Puis, le meilleur client u^* est celui tel que :

$$c_2(i(u),u,j(u)) = \max_u \{c_2(i(u),u,j(u)) | u \text{ n'est pas servi et la route est réalisable}\} \tag{6.16}$$

Le client u^* est ensuite insérée entre $i(u^*)$ et $j(u^*)$. Quand plus aucun client n'a d'insertion réalisable, la méthode commence une nouvelle route.

Détaillons à présent le calcul de $c_1(i,u,j)$:

$$\begin{aligned} c_1(i,u,j) &= \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) && (6.17)\\ \text{où } \alpha_1 + \alpha_2 &= 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0 \\ c_{11}(i,u,j) &= t_{iu} + t_{uj} - \mu t_{ij} && (6.18)\\ c_{12}(i,u,j) &= b_{ju} - b_j && (6.19) \end{aligned}$$

t_{ij} est le temps de parcours entre les clients i et j, b_{ju} l'heure de début d'exécution de j quand u est inséré, b_j l'heure de début d'exécution de j avant l'insertion. Le paramètre μ contrôle l'importance du « savings ». Le critère $c_2(i,u,j)$ est alors calculé de la manière suivante :

$$c_2(i,h,j) = \lambda t_{0u} - c_1(i,u,j) \tag{6.20}$$

Le paramètre λ est utilisé pour déterminer l'importance de la distance au dépôt par rapport au temps supplémentaire nécessaire à la visite du client par le véhicule courant. Dans le même article Solomon proposait deux autres heuristiques que nous ne détaillerons pas ici,car l'heuristique I1 a donné les meilleures résultats (Braysy & Gendreau, 2005a).

6.3.2 Exemple

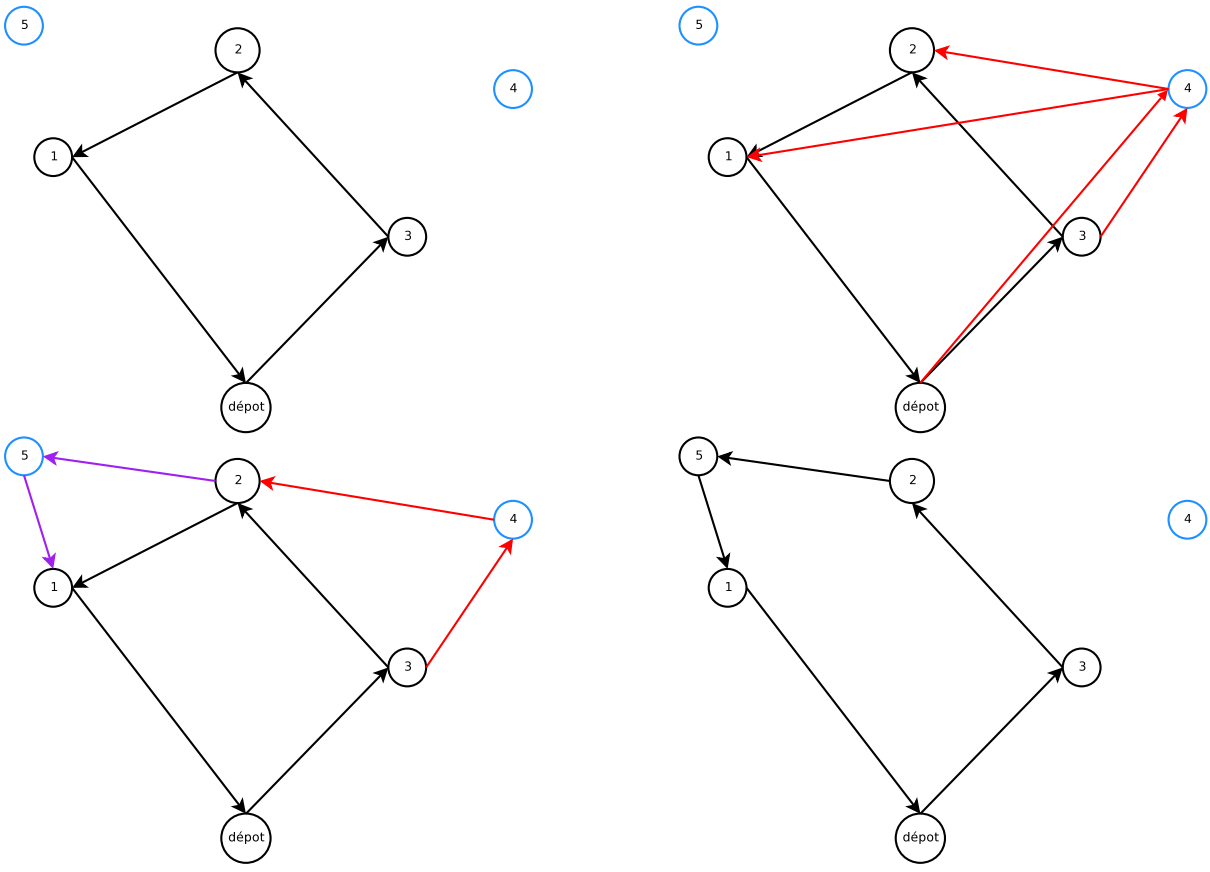

FIG. 6.1 – Exemple d'insertion utilisant l'heuristique de Solomon.

Prenons un exemple afin d'illustrer le principe de fonctionnement de l'heuristique de Solomon. Dans cet exemple, nous considérerons que les contraintes temporelles et de compétences sont respectées pour chaque insertion. Considérons la tournée de la figure 6.1 (en haut à gauche) et les jobs 4 et 5 sont à insérer dans la tournée courante.
On cherche d'abord la meilleure insertion selon le critère c_1 parmi les insertions réalisables du job 4 dans

la tournée(figure 6.1 en haut à droite). On réalise la même étape pour le job 5. Après avoir sélectionne la meilleure insertion réalisable pour chaque job non servi (6.1 en bas à gauche), on sélectionne la meilleure insertion sur l'ensemble des jobs à l'aide du critère c_2. on réalise l'insertion du job sélectionné dans la tournée (6.1 en bas à droite).
On réitère l'opération jusqu'à ce qu'il n'y ait plus de job ou jusqu'à ce qu'aucun job ne puisse être inséré dans la tournée.

6.4 Notre heuristique de construction

Notre heuristique de construction est basée sur l'heuristique de Solomon pour la construction des tournées et sur un algorithme d'affectation dynamique des tournées sur l'horizon.

6.4.1 Principe de l'heuristique de construction

Notre heuristique va exploiter la spécificité des fenêtres de temps pour « séparer » la construction des tournées de leur affectation sur l'horizon. La construction des tournées sera basée sur l'heuristique de Solomon adaptée à quelques particularités du problème. En effet, une tournée est réalisable par un technicien si :

- Il possède les compétences pour réaliser chacun des jobs.
- La durée totale de la tournée est inférieur ou égal à d_{max}.
- La fenêtre de temps de la tournée, qui est l'intersection des fenêtres de temps de tous les jobs appartenant à la tournée n'est pas nulle.
- Elle est compatible avec le planning courant du technicien, c'est-à-dire on peut lui affecter un jour appartenant à sa fenêtre de temps qui n'est affecté à aucune autre tournée.

La construction des tournées sera donc effectuée grâce à l'heuristique de construction de Solomon adaptée à notre problème (compétences, bases de départ et d'arrivée, capacité). Lors de la création ou l'insertion de jobs dans une tournée, nous vérifierons qu'il existe une affectation des tournées de chaque technicien sur un ensemble de jours distincts.

Après avoir détaillé l'algorithme d'affectation dynamique des tournées sur l'horizon basé sur une idée originale qui utilise une structure de graphe biparti, nous présenterons l'heuristique proprement dite.

6.4.2 Affectation dynamique des tournées sur l'horizon de planification

Lors de cette phase, nous utilisons une structure de graphe biparti dans lequel nous maintenons un couplage maximal. Commençons par rappeler les notions de graphe biparti et couplage, puis nous présentons notre méthode.

Rappels : graphe Biparti et couplage

Notions de couplage dans un graphe biparti (Berge, 1969) :

Définition 6.2 (Couplage) *Étant donné un graphe simple non orienté $\mathcal{G} = (\mathcal{N}, \mathcal{V})$, un couplage est un sous-ensemble d'arête $K \subset \mathcal{V}$ tel que deux arêtes quelconques de K ne sont pas adjacentes.*
Un sommet i est dit saturé par le couplage K s'il existe une arête de K incidente à i.
Un couplage K qui sature tous les sommets du graphe est appelé couplage parfait.
Un couplage maximal est un couplage de cardinalité maximale.

Théorème 6.1 (Graphe biparti) *Pour un graphe $\mathcal{G}$, les conditions suivantes sont équivalentes :*

- *$\mathcal{G}$ est biparti ;*
- *$\mathcal{G}$ n'a pas de cycles élémentaires de longueur impaire ;*
- *$\mathcal{G}$ n'a pas de cycles de longueur impaire.*

Nous allons maintenant présenté un théorème d'existence d'un couplage dans un graphe biparti.

Théorème 6.2 (P. Hall 1934 « théorème de König-Hall ») *Dans un graphe biparti $\mathcal{G} = (X, Y, \mathcal{V})$, on peut coupler X dans Y si et seulement si on a :*

$$|\Gamma_G(A)| \geq |A| \quad \forall A \subset X$$

où $\Gamma_G(A) = \cup_{x \in A} \Gamma_G(x)$ *et* $\Gamma_G(x)$ *est l'ensemble des voisins du sommet* x.

Hélas, la vérification de cette condition nécessite l'énumération de tous les sous-ensembles de X, elle a donc une complexité exponentielle. Nous ne vérifierons donc pas cette condition et nous contenterons de trouver un couplage maximal de poids minimal (nos arêtes sont pondérées). Nous allons maintenant présenter la notion de chaîne alternée utilisée dans la plupart des algorithmes de recherche de couplages :

Définition 6.3 (Chaîne alternée) *Si* $K \subset A$ *est un couplage de* $\mathcal{G} = (\mathcal{N}, \mathcal{V})$*, on appelle chaîne alternée (relativement a* K*) une chaîne élémentaire de* $\mathcal{G}$ *dont les arêtes appartiennent alternativement a* K *et* $K = A \backslash K$*. Une chaîne alternée augmentante ou chaîne améliorante est une chaîne alternée joignant deux sommets insaturés.*

Théorème 6.3 (Berge,1957) *Un couplage* K_0 *est maximum si et seulement si il n'existe pas de chaîne alternée reliant un sommet insaturé à un autre sommet insaturé.*

Algorithme d'affectation dynamique

Structure de donnée Notre algorithme utilise une structure de données de graphe biparti pour affecter les tournées au jours de travail des techniciens. Le graphe biparti $G_k = (X, Y, \mathcal{V})$ (voir section 6.4.2) est construit de la manière suivante :

- X représente les tournées affectées à un technicien k.
- $Y = \cup_{t \in X}[r_t, dc_t]$ représente les jours de l'horizon de planification disponibles pour ces tournées, l'union des fenêtres de temps des tournées.
- $(x, y) \in \mathcal{V}$ signifie que la tournée associée à x est réalisable le jour associé à y.

L'affectation des tournées aux jours de l'horizon est équivalente à l'existence d'un couplage de X dans Y. De plus, il faut que cette structure soit dynamique afin de pouvoir modifier le graphe tout en gardant un couplage de X dans Y. Chacune de ces opérations sera annulée si elle ne conserve pas un couplage de X dans Y. Illustrer différents cas de figures possibles lors de ces opérations.

Opération sur la structure de donnée Avant chacune des opérations, on connaît un couplage de X dans Y. Les opérations possibles sont :

- Création d'une tournée.
- Ajout d'un job à une tournée existante.

Lors de la création d'une tournée, on ajoute un noeud au graphe représentant la tournée, puis on crée les arêtes liant la tournée aux sommets correspondant à sa fenêtre de temps (si ces noeuds n'existent pas on les crée). S'il existe une chaîne alternée liant la tournée à un sommet non saturé ($\in Y$), on effectue le transfert. Dans le cas contraire, on annule l'opération.
Lors de l'ajout d'un job à une tournée, on effectue l'intersection entre la fenêtre de temps de la tournée et la fenêtre de temps du job à insérer. Trois cas sont alors à envisager :

1. L'intersection est vide, on ne peut pas insérer le job.
2. L'intersection contient l'arête appartenant au couplage courant, on supprime les arêtes n'appartenant pas à l'intersection.
3. L'intersection ne contient pas l'arête appartenant au couplage, on supprime les arêtes n'appartenant pas à l'intersection. Si une chaîne alternée issue de la tournée existe, on effectue le transfert, sinon on annule l'opération.

Exemple Supposons qu'un technicien k à un planning représenté par le graphe biparti en figure 6.2 (à gauche) . Les arêtes rouges représentent les arêtes appartenant au couplage et les arêtes en gras représentent les arêtes d'une chaîne alterné.

Dans le premier cas de figure, l'ajout d'un job à la tournée 2 et la suppression d'arêtes consécutive à cet ajout ne modifie aucune arête appartenant au couplage, le couplage courant est encore valide. Dans le second cas, l'arête appartenant au couplage est retirée du graphe. Il faut donc mettre à jour le couplage si cela est possible. On cherche une chaîne alterné qui se résume à l'arc liant la tournée 2 au jour 3.

En partant du même état courant, nous illustrons maintenant l'ajout d'une tournée au planning en figure 6.3. Nous ajoutons donc le noeud et les arêtes associées à la nouvelle tournée (à gauche), puis nous

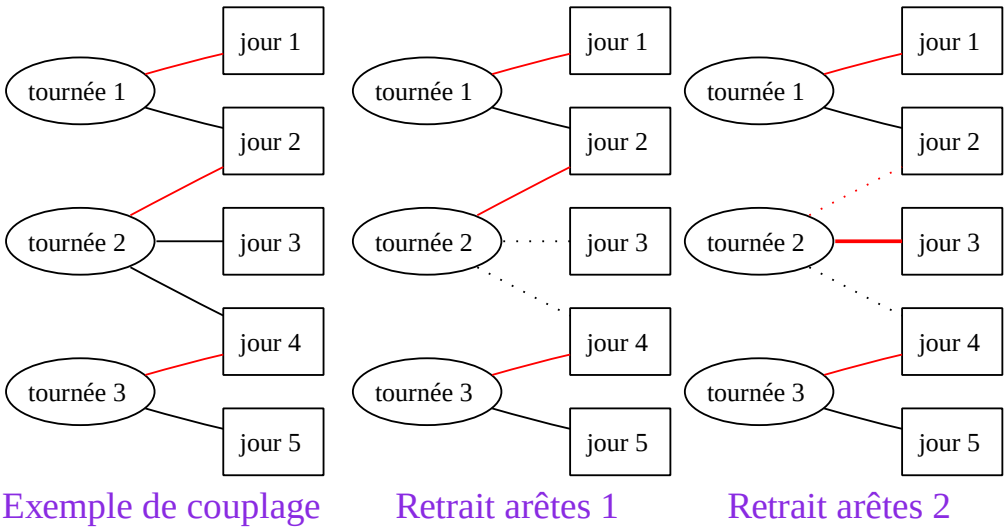

FIG. 6.2 – Couplage courant et retrait d'arêtes dû à l'ajout d'un job dans la tournée.

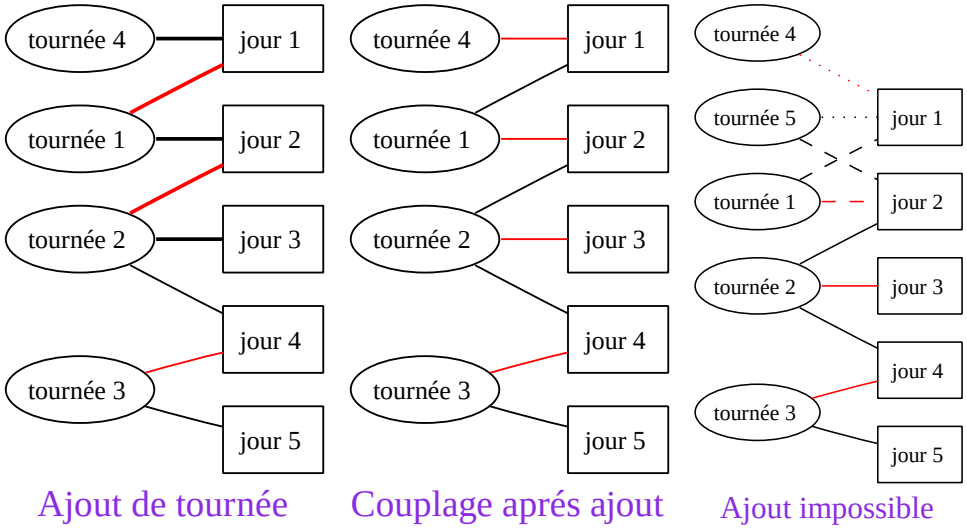

FIG. 6.3 – Ajout de tournée au planning d'un technicien.

effectuons le transfert le long d'une chaîne alternée afin de déterminer un nouveau couplage maximal (au milieu). Enfin, nous essayons de rajouter une dernière tournée (en effet, il est nécessaire que $|X| \leq |Y|$), mais nous nous apercevons qu'il n'existe pas de chaîne alternée valide (les chaîne en pointillé). Il est donc impossible d'ajouter cette nouvelle tournée au planning courant. Notons que cette situation peut aussi se produire lors d'un retrait d'arêtes et donc interdire l'insertion d'un job dans une tournée.

6.4.3 Heuristique de Solomon modifiée

L'algorithme utilisé qui prends en argument :

- Le graphe $\mathcal{G}$ représentant le problème.
- Un ensemble de technicien K.
- Un ensemble de jobs J_0

Cet algorithme renvoie l'ensemble de tournées construites et donc une solution au problème.

Initialisation d'une tournée

La fonction CREATETOURNEE initialise une nouvelle tournée. Lorsqu'il n'y a qu'un dépôt et pas de compétences, on initialise une tournée avec le job le plus éloigné du dépôt.
Or, l'aspect multi-dépôts de notre problème nous oblige à changer les critères d'initialisation. Nous choisissons :

1. Sélectionner un job prioritaire. La priorité d'un job est déterminée en fonction de sa fenêtre de temps et de sa durée.
2. L'affecter à un technicien pouvant réaliser le job et n'étant pas trop éloigné du site.
3. Tenir compte de l'équilibrage des charges entre les techniciens.

Notre solution est de choisir un technicien puis de lui affecter le job le plus urgent (la liste $J_{restant}$ est ordonnée) tout en gardant un planning réalisable (c'est-à-dire qu'il existe toujours un couplage pour le graphe biparti représentant le planning du technicien).

Fonction `createTournee`($k, J_{restant}$)
Données : un technicien, ensemble de jobs
Résultat : retourne la nouvelle tournée affectée au technicien ou null
tant que `isEmpty`($J_{restant}$)=**faux faire**
  $j \leftarrow$ `pop`($J_{restant}$);
  si `isFeasibleBy`(j,k) **and** `timeFeasible`(j,k) **alors**
    tournee $\leftarrow j$;
    `setTechnicien`(*tournee*,k);
    si `updateMatching`(*tournee*)=**vrai alors**
      retourner *tournee*;
    fin
  fin
fin
retourner null;

Insertion de jobs dans une tournée

La fonction SORTBESTINSERTION présentée page 52 renvoie la liste ordonnée des meilleures insertions réalisables dans une tournée.

A la différence de l'heuristique de Solomon, nous ne gardons pas uniquement la meilleure insertion réalisable, car celle-ci ne conserve peut-être pas l'existence d'un couplage maximal pour le technicien.
A la place, nous réalisons la première insertion pour laquelle un couplage maximal existe (voir INSERTJOBAT page 52). On peut également noter que nous avons remplacer un des termes des fonctions de Solomon pour prendre en compte l'aspect multi-dépôts. Ainsi :

$$(6.20) \rightarrow c_2(i,h,j) = \lambda \times (t_{s^k u} + t_{u f^k}) - c_1(i,u,j)$$

Algorithme principal : ModifiedSolomon

Nous allons maintenant présenter l'algorithme 4 qui est la méthode principale de notre heuristique de construction de tournées. Elle construit une tournée en deux étapes :
- Initialisation de la tournée.
- Insertion de jobs dans la tournée tant que cela est possible.

Elle se termine lorsque tous les jobs ont été insérés ou lorsqu'on ne peut plus créer de nouvelle tournée tout en respectant les plannings actuels des techniciens. On remarque que la fonction renvoie le nombre de jobs qui n'ont pas pu être insérés dans une tournée.

Fonction `sortBestInsertion`$(\mathcal{G}, J_{restant}, t)$

Données : ensembles $\mathcal{G}, K, J_0$ (ensemble de jobs),t (tournée courante)
Résultat : les meilleures insertions triées dans la tournée t
$k \leftarrow$ `getTechnicien`(t) `/* tech affecté à la tournée */`;
pour chaque $j \in J_0$ **faire**
  si `isFeasibleBy`$(j,k) \cap$(`getDuree`$(t) + p_j \leq d_{max}) \cap ([r_j, dc_j] \cap$ `getTimeWindow`$(t) \neq \emptyset)$
  alors
    $c_2(t, place_j, j,) \leftarrow$ `computeBestInsertion`(j,t) `/* meilleure insertion du job` j `*/`;
  fin
fin
$L \leftarrow$ `sortInsertion`$(\cup_{J_0} c_2(t, place_j, j))$ `/* tri décroissant des meilleures insertions */`;
retourner L;

Fonction `insertJobAt`$(j, place, t)$

Résultat : vrai si on a réussi à mettre à jour les affectations des tournées aux jours après insertion du job
$t \leftarrow$ `insertJob`$(j,place,t)$ `/* insertion du job dans la tournée */`;
si `updateMatching` (t)=**vrai alors**
  retourner vrai `/* Le couplage est mis à jour */`;
fin
sinon
  $t \leftarrow$ `removeJob`(t,j) `/* suppression du job de la tournée */`;
  retourner faux `/* pas de couplage maximal possible : annulation */`
fin

Algorithme 4 : ModifiedSolomon

Données : ensembles $\mathcal{G} = (\mathcal{N}, \mathcal{V}), J, K$
Résultat : ensemble de tournées de véhicules : T
$J_{restant} \leftarrow$ `jobSort`(J);
$T \leftarrow \emptyset$;
$k \leftarrow 0$;
tant que $J_{restant} \neq \emptyset$ **faire**
  `/* Création d'une nouvelle tournée */`
  $tournee \leftarrow$ **null**;
  $cpt \leftarrow 0$;
  tant que $tournee =$ **null and** $cpt \leq |K|$ **faire**
    $cpt \leftarrow cpt + 1$;
    $tournee \leftarrow$ `createTournee`$(J_{restant},k)$;
    $k \leftarrow k + 1\ mod(|K|)$;
  fin
  si $tournee \neq$ **null alors**
    `/* Insertion de jobs dans la tournée courante */`
    répéter
      $inser =$ **faux**;
      $L_{insert} \leftarrow$ `sortBestInsertion`$(J_{restant}$,`getTechnicien`$(tournee))$;
      tant que $inser =$ **faux and** `isEmpty`$(L_{insert}) =$ **faux faire**
        $(j, place) \leftarrow$ `pop`(L_{insert}) ;
        si `insertJobAt`$(j,place,tournee) =$**vrai alors**
          $J_{restant} \leftarrow J_{restant} \backslash \{j\}$ `/* job ajouté à la tournée */`;
          $inser \leftarrow$**vrai** ;
        fin
      fin
    jusqu'à $inser =$**faux** ;
  sinon
    `/* on ne peut plus créer de nouvelles tournées */`
    break;
  fin
  Retour $|J_{restant}|$;
fin

6.4.4 Complexité

Déterminons la complexité de chacune des fonctions de l'heuristique afin de calculer la complexité générale de la méthode. Nous noterons d_{limit} la plus grande date contractuelle des jobs de l'énoncé. Nous rappelons que n est le nombre de jobs et m le nombre de techniciens.

Complexité de la fonction updateMatching

La fonction UPDATEMATCHING réalise un parcours en largeur du graphe biparti associé au planning du technicien. La complexité d'un algorithme de parcours en largeur d'un graphe $\mathcal{G} = (\mathcal{N}, \mathcal{V})$ est : $\max(\mathcal{N}, \mathcal{V})$. Dans le pire des cas, chaque sommet de X du graphe biparti $\mathcal{G} = (X, Y, \mathcal{V})$ est lié à tous les sommets de Y. Donc $O(updateMatching) = \max(X + Y, X \times Y)$. Mais on sait aussi qu'il existe un couplage de X dans Y, on a donc : $|X| < |Y|$ et $O(updateMatching) \leq \max(2Y, Y^2) = \max(2d_{limit}, d_{limit}^2)$. Or on peut supposer que $d_{limit} \geq 2$ car sinon nous ne sommes plus dans un contexte multi-périodes et il existe d'autres méthodes de résolution. Donc nous avons $O(updateMatching) = d_{limit}^2$.

Complexité des autres fonctions

Connaissant la complexité de la fonction UPDATEMATCHING, on peut calculer la complexité des autres fonctions de l'algorithme. Nous présentons ces résultats dans le tableau 6.1.

UPDATEMATCHING	INSERTJOBAT	CREATETOURNEE
d_{limit}^2	d_{limit}^2	$n \times d_{limit}^2$
COMPUTEBESTINSERTION	SORTBESTINSERTION	algorithme de tri
n	n^2	n^2

TAB. 6.1 – Complexité des fonctions de l'heuristique de construction SolomonModified.

Complexité de l'algorithme SolomonModified

Calculons la complexité de l'algorithme MODIFIEDSOLOMON :

$$O(ModifiedSolomon) = n^2 \times (n + m) \times d_{limit}^2$$

6.5 Amélioration locale de la solution obtenue

Nous allons maintenant présenter deux méthodes d'amélioration locale de la solution utilisant la programmation linéaire. On utilise la programmation linéaire plutôt qu'une heuristique d'amélioration locale afin de tirer bénéfice du travail réalisé pendant la première partie du stage 4.
Ce choix a aussi été guidé par le désir des encadrants de pouvoir utiliser ce travail pour développer d'autres heuristiques et résoudre des sous-problèmes de manière exacte.

6.5.1 Amélioration par tournée

La première méthode d'amélioration développée tente d'optimiser chaque tournée individuellement en résolvant un Traveling Salesman Problem (voir 3.2).
En effet, vu la petite taille des tournées, nous pensions résoudre ce sous-problème exactement en utilisant CPLEX. Cependant, pour la même raison, il y avait peu de chance que cette méthode améliore beaucoup la solution. Nous avons donc décider de résoudre un sous-problème plus important que nous présentons dans la section suivante.

6.5.2 Amélioration par jour

La seconde méthode d'amélioration est donc une méthode d'amélioration par jour. Après la construction de la solution, nous résolvons jour par jour un m-Capacited Vehicle Routing Problem avec flotte hétérogène (voir 3.3). Pour chaque jour, nous ressortons donc l'ensemble des jobs réalisés par tous les

techniciens et nous résolvons un problème de tournées de véhicules avec contraintes de capacité (la durée maximale d'une tournée) et flotte hétérogène (compétence des techniciens ainsi que les bases).

Cette méthode devrait donner de meilleures résultats d'amélioration, mais dès que le nombre de techniciens ou de jobs devient trop important, nous ne pourrons plus résoudre ce sous-problème.

Nous reprenons les équations présentées en section 3.3 auxquelles nous rajoutons les contraintes de capacité et de compatibilité :

$$\sum_{i \in J} p_i y_i^k + \sum_{(i,j) \in \mathcal{V}} t_{ij} x_{ij}^k \leq d_{max} \quad \forall k \in K \tag{6.21}$$

$$y_i^k \leq o_i^k \quad \forall i \in J \backslash \Theta, \forall k \in K \tag{6.22}$$

6.6 Limites et améliorations

Étudions d'abord les limites actuelles de notre méthode, puis nous discuterons des améliorations éventuelles que nous pouvons lui apporter.

6.6.1 Limites de la méthode

Tout d'abord il faut remarquer que la présence de fenêtre de temps dure peut rendre le problème insatisfiable. Lorsqu'on utilise un programme linéaire et que le problème est insatisfiable, les méthodes les plus classiques sont de relâcher des contraintes et/ou de changer l'objectif.
On peut par exemple relâcher la date contractuelle et introduire une pénalité sur les retards, ou changer d'objectifs en essayant de minimiser le nombre de jobs non servis.
Or, notre méthode ne nous assure pas de minimiser le nombre de jobs non servis. un exemple illustre ce phénomène (nous ne prenons pas en compte l'aspect spatial , les distances entre les sites sont nulles).

Exemple Soit un seul technicien k_1 devant réaliser l'ensemble de jobs du tableau 6.2 avec $d_{max} = 8$. Si le technicien réalise la tournée $t_0 = (j_1, j_2, j_3)$ le jour 0 (il est obligé), il ne peut plus réaliser le job j_4.

id	j_1	j_2	j_3	j_4
durée	5	1	2	3
fenêtre de temps	[0,0]	[0,1]	[0,1]	[0,0]

TAB. 6.2 – Ensembles de jobs à réaliser par le technicien.

Or, il existe des solutions servant tous les jobs, par exemple : $t_0 = (j_1, j_4)$ et $t_1 = (j_2, j_3)$. Cette exemple simple cherche met en lumière un phénomène qui peut se produire à cause des compétences ou de la décomposition entre la création des tournées et leur affectation à un jour de l'horizon.

Solutions proposées Pour contrer ce phénomène, on peut agir à plusieurs niveaux :
- Modifier la liste de priorité en tenant aussi compte du nombre de techniciens capable de réaliser un job (par exemple une pondération) ;
- Modifier le critère d'insertion de façon similaire.
- Perturber le critère de sélection (par exemple avec un critère tabou), ce qui permettrait de casser le déterminisme de notre algorithme.

Cependant, aucune de ces méthodes ne nous assurent de minimiser le nombre de jobs non servis. Elles permettent simplement d'intégrer un critère concernant la flotte dans l'heuristique.

6.6.2 Extension de l'heuristique

Lors d'une présentation interne à France Telecom, une critique importante concernait le fait que les hypothèses ne prenaient pas en compte le planning journalier des techniciens. Ce planning est le nombre d'heures de travail qu'un technicien peut fournir un jour donné.

Il fallait donc supprimer le critère d_{max} et le remplacer par une donnée h_k^d indiquant le nombre d'heure qu'un technicien k peut travailler le jour d. Nous avons donc envisager une amélioration permettant de prendre en compte ce critère. En effet, dans la méthode actuelle, on vérifie que la tournée est réalisable par un technicien en fonction de ces compétences et de d_{max}, puis on cherche une affectation de la tournée à un jour de l'horizon.
Pour cela, nous avons imaginé de transformer l'affectation dynamique des tournées sur l'horizon et de ne vérifier que la compétence du technicien dans les fonctions INSERTJOBAT et CREATETOURNEE. Nous transformons le graphe biparti en attribuant un potentiel aux noeuds ; pour les noeuds de tournées leur durée ; pour les noeuds jours la disponibilité du technicien. Puis, pour chaque arc, on le pondère par la différence de potentiels entre le noeud jour et le noeud tournée, en supprimant les arcs négatifs. La transformation du graphe biparti est illustrée en figure 6.4 page 55.
Ensuite, on cherche un couplage maximal de poids minimum, ce qui peut être réalisé en un temps

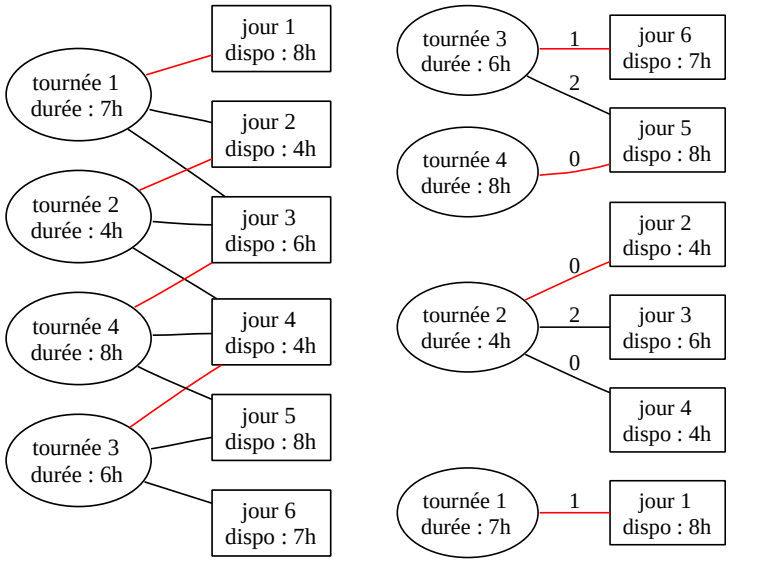

FIG. 6.4 – Modification du graphe biparti afin de prendre en compte les disponibilités journalières des techniciens.

polynomial. L'intérêt de chercher un couplage maximal de poids minimum est ainsi de libérer le maximum du temps pour la phase de création des tournées. Avec cette modification, le retrait d'arête qui a éventuellement lieu lors de l'ajout d'un job à une tournée est réalisé en fonction de la fenêtre de temps du job et de la durée de la tournée.

CHAPITRE 7

RÉSULTATS EXPÉRIMENTAUX

Nous présentons dans ce chapitre le protocole expérimental et les résultats obtenus lors des tests de l'heuristique de construction et des méthodes d'amélioration locale.

7.1 Implémentation

Pour implémenter l'heuristique et les méthodes d'améliorations, nous avons implémenter des modules supplémentaires au programme VrpSolver. Nous avons rajouter un package pour les classes de l'heuristique de construction et nous avons utilisé la classe abstraite du solver pour implémenter les classes associées à l'amélioration locale.

7.2 Protocole

7.2.1 Données générées et paramétrage

La géographie est un fichier de 100 villes généré par genVilles.perl. Les fichiers de commandes ont été générés par genData.perlavec le réglage présenté dans le tableau 7.1.
Lors de la phase d'amélioration locale, nous résolvons des sous-problème à l'aide de CPLEX. Nous fixons

nb jobs	nb techniciens	durée	release date	deadline
50, 100, 150	6	short [60,100]	0-6	rd + 1-30
200, 300, 400		long [60,300]	équiprobable	équiprobable

Tab. 7.1 – Type de données générées pour le test des méthodes heuristiques.

une limite de temps pour chaque résolution d'un sous-probléme :
- Traveling Salesman Problem : 5 minutes.
- m-Capacited Vehicle Routing Problem avec flotte hétérogène : 10 minutes.

Nous avons testé deux réglages des paramètres de l'heuristique que nous présentons dans le tableau 7.2

paramètre	α_1	λ	μ
réglage 1	0.5	0.5	1
réglage 2	0.9	0.5	1

Tab. 7.2 – Réglage des paramètres de l'heuristique.

7.2.2 Analyse des résultats

Pour chacune des exécutions, nous avons relevé les indicateurs suivantes :
- Le nombre de jobs de l'instance ;

- le nombre de jobs non servis à la fin de l'heuristique ;
- la valeur du problème relaxé associé à la solution lorsque nous pouvions le résoudre ;
- la valeur de l'objectif après l'heuristique de construction ;
- Le temps d'exécution de l'heuristique de construction ;
- la valeur de l'objectif après l'amélioration par tournée ;
- Le temps d'exécution de l'amélioration par tournée ;
- la valeur de l'objectif après l'amélioration par jour ;
- Le temps d'exécution de l'amélioration par jour ;

A l'aide de ces valeurs, nous avons généré plusieurs types de graphiques :

1. Un graphique affichant la moyenne du rapport entre la valeur de l'objectif lors des différentes phases de notre objectif et la valeur du problème relaxé associé (borne inf) à la solution courante en fonction du nombre de jobs de l'instance.
2. Un graphique affichant la moyenne du rapport entre la valeur de l'objectif avant et après les méthodes d'amélioration locale en fonction du nombre de jobs de l'instance. Ce second type de graphique est utile lorsqu'on ne peut plus résoudre la relaxation.
3. Un graphique indiquant la moyenne du nombre de jobs non servis en fonction du nombre de jobs de l'instance.
4. Un graphique indiquant la moyenne du temps d'exécution des différentes phases en fonction du nombre de jobs de l'instance.

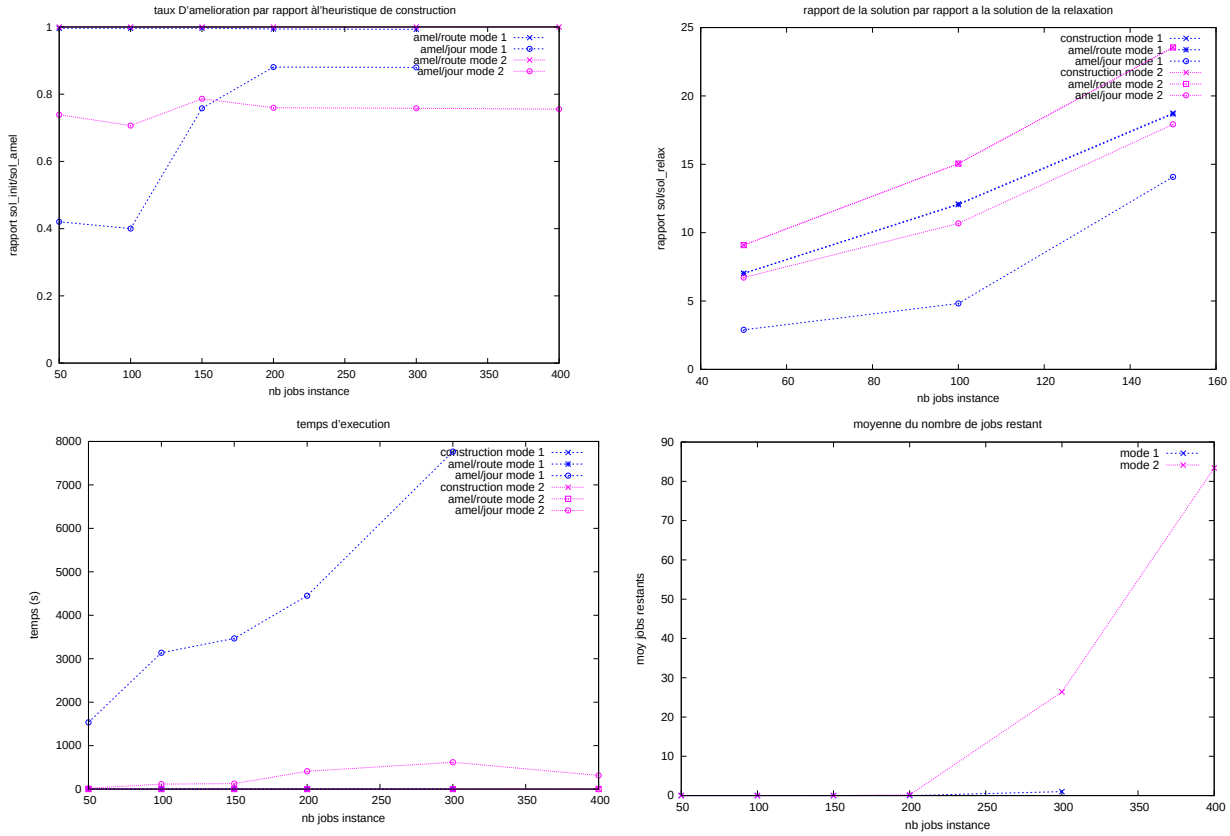

FIG. 7.1 – Comparaison des performances de l'heuristique de construction et des méthodes d'amélioration locales pour des instances contenant des jobs courts (mode 1) et d'autres instances contenant des jobs long (mode 2).

7.3 Comparaison selon le type d'instance

Lors de la première phase de tests, nous avons voulu comparer l'efficacité de notre heuristique en fonction du type d'instances. Nous avons donc comparer les résultats obtenus pour le réglage 1 avec les instances short (mode 1) et les instances long (mode 2) présentés en figure 7.1.
Notons que pour des instances de type short, la méthode d'amélioration par jour n'a pas pu être mené à bout pour des instances comprenant 400 jobs alors que notre heuristique de construction fonctionnait encore.

Nous pouvons tirer des résultats les conclusions suivantes :

1. Pour les deux types d'instances, l'amélioration par route a un effet négligeable (voir graphique taux d'amélioration par rapport à l'heuristique de construction en haut à gauche).
2. La qualité de l'amélioration par jour décroît rapidement avec la taille pour les instances courtes alors qu'elle reste constante pour les instances longues. Ceci est dû au fait que le programme n'arrive plus à résoudre les M-CVRP avec flotte hétérogène sous-problèmes. Ce phénomène est illustré par l'augmentation du temps nécessaire à la résolution des sous-problèmes (voir graphiques temps d'exécution en bas à gauche) pour les jobs courts.
3. Le rapport de la valeur de l'objectif sur la valeur de l'objectif de la relaxation est cependant meilleure pour les jobs courts (graphique en haut à gauche).
4. La moyenne du nombre de jobs restant est quasiment nulle pour les jobs courts alors qu'il semble y avoir une limite inférieure à 300 jobs sur l'horizon de planification (graphique en bas à droite). La taille de l'horizon étant borné, cela peut indiquer que certaines instances sont insatisfiables.

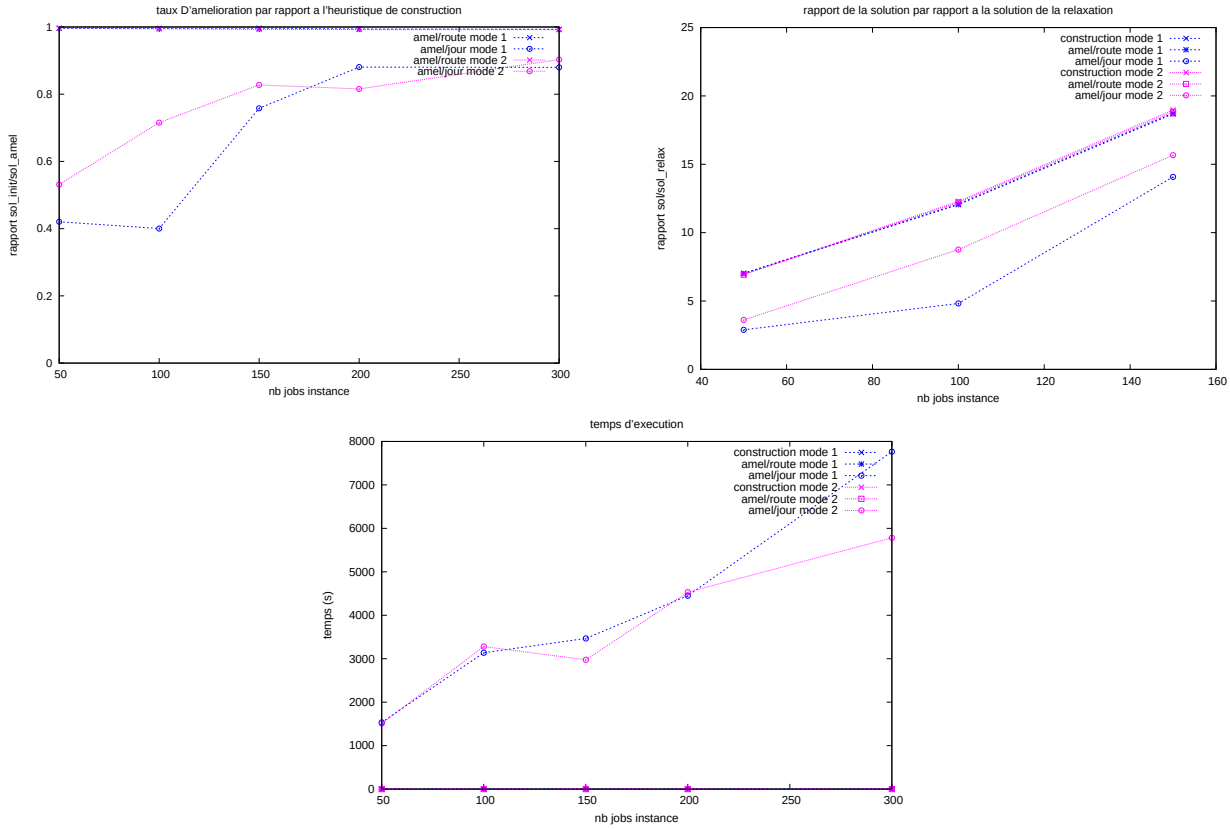

FIG. 7.2 – Comparaison des performances de l'heuristique de construction et des méthodes d'amélioration locales pour le réglage 1 (mode 1) et le réglage 2 (mode 2).

7.4 Comparaison selon le paramétrage de l'heuristique

Lors de cette phase de tests, nous avons voulu comparer deux paramétrages de l'heuristique (voir tableau 7.2) et voir de quel manière ils peuvent influencer la phase d'amélioration locale. Nous avons utiliser les instances short car elles rendent plus difficiles la phase d'amélioration locale (voir section 7.3). Le réglage 1 correspond au mode 1 et le réglage 2 au mode 2. Nous présentons les résultats des tests en figure 7.2.

Nous pouvons tirer des résultats les conclusions suivantes :

1. La méthode d'amélioration par tournée est toujours inefficace.
2. Le changement de paramétrages ne semble pas modifier la qualité de la solution initialement construite (voir graphique en haut à droite).
3. Par contre les différents paramétrage semble modifier la qualité de l'amélioration par jour (graphiques en haut) ainsi que le temps d'exécution de la méthode d'amélioration par jour (graphique en bas).
4. La moyenne du nombre de jobs non servis est négligeable quel que soit le réglage.

7.5 Conclusions

- **Le fait de ne pas minimiser le nombre de jobs non servis ne semble pas problématique.**
- **La méthode d'amélioration par tournée est inefficace.**
- **La méthode d'amélioration par jour est fortement limitée par la taille des sous-problèmes liée au nombre de techniciens et aux durées des jobs.**
- **Les performances de l'heuristique de construction semblent stables. On a pu construire des solutions pour des problèmes de taille importante (une vingtaine de techniciens et un millier de jobs) en un temps raisonnable (quelques secondes).**

CHAPITRE 8

PERSPECTIVES

Nous présentons dans ce chapitre les perspectives et pistes afin de continuer le travail entrepris. Ces perspectives concernent aussi bien l'optimisation de la résolution des modèles linéaires par CPLEX que l'amélioration et la création des méthodes approchées.

8.1 Optimisation de la résolution par CPLEX

Nous voudrions explorer deux pistes principales afin d'optimiser la résolution des problèmes à l'aide de CPLEX :

- Implémenter les optimisations concernant la contrainte d'élimination des sous-tours présentée en section 5.1.1.
- Mesurer l'efficacité de la contrainte k-split présentée en section 5.1.2.

8.2 Méthodes heuristiques

Les perspectives concernant les méthodes heuristiques sont :

- tester l'équilibrage des charges entre les techniciens.
- Implémenter l'heuristique de construction modifiée pour prendre en compte les contraintes de planning (voir section 6.6.2).
- Créer une structure de voisinage adaptée à notre problème afin d'utiliser une méthode d'amélioration locale heuristique (voir section 6.1.1).
- Utiliser une métaheuristique (voir section 6.1.2).

Pour cette problématique des jeux de données seront prochainement disponibles. Il pourrait être intéressant de comparer la solution construite à l'aide de notre méthode au planning qui a été effectivement suivi par les techniciens afin d'évaluer le gain potentiel pour l'entreprise.

CONCLUSION

Nous avons introduit en première partie de ce rapport la problématique concernant les unités d'intervention clients. Il s'agit d'un problème original, issu de la réalité industrielle et comportant un certain nombre de contraintes spécifiques comme par exemple les contraintes de synchronisation de passage des techniciens pour une intervention. Ce problème est central et se situe à la base de problématiques industrielles existantes comme le problème des tournées multi-période, , les problèmes avec fenêtre de temps ou encore les problèmes avec flotte limitée.

Dans un premier temps nous avons formalisé et modélisé le problème concret posé par l'entité opérationnelle. J'ai appris en premier lieu le travail en équipe et la difficulté de concilier les objectifs des différents acteurs.

Une fois la modélisation du problème arrêtée nous l'avons validée avec les premières solutions. Certaines de mes connaissances ont pu être approfondies dans les domaines de l'optimisation, la programmation JAVA, et les outils de développement collaboratif ; en particulier, cette première partie m'a permis de découvrir et maîtriser un solveur majeur pour la programmation mathématique : CPLEX.
La résolution de ces modèles à montré des limites attendues. Elle a été néanmoins utile pour valider notre modèle et nous donne un outil d'évaluation de petites instances pour les heuristiques développées par la suite.

En conséquence, pour le traitement d'un problème connexe, nous avons choisi de développer des méthodes heuristiques. Les premiers tests effectués ont montré l'efficacité de notre heuristique en terme de réalisabilité, mais nous ne disposons d'aucune borne de référence sur de grandes instances (à l'exception de la valeur de l'objectif du problème relaxé). Cette partie m'a permis de mieux appréhender le travail de recherche, en synthétisant plusieurs articles et en proposant une méthode originale.

BIBLIOGRAPHIE

Archetti, C., Savelsbergh, M.W.P., & Speranza, M.G. 2005a. *An Optimization-Based Heuristic for the Split Delivery Vehicle Routiong Problem.* optSDVRP.pdf, Submitted to Transportation Science.

Archetti, C., Savelsbergh, M.W.P., & Speranza, M.G. 2005b. *To Split or Not To Split : That is the Question.* ratio-sdvrp-revision.pdf, To appear in Transportation Research E.

Archetti, C., Savelsbergh, M.W.P., & Speranza, M.G. 2006. Worst-Case Analysis for Split Delivery Vehicle Routing Problems. *Transportation Science*, **40**(2), 226. SDVRPrevision.pdf.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. 1998. Branch-and-price : column generation for solving huge integer programs. *Operations Research*, **46**, 316–329. barnhart96branchprice.pdf.

Beltrami, E., & Bodin, L. 1974. Networks and vehicle routing for municipal waste collection. *Networks*, **4**, 65–94. article historique PVRP.

Berge, Claude. 1969. *Graphes et Hypergraphes.* Dunod.

Braysy, Olli, & Gendreau, Michel. 2005a (February). Vehicle Routing Problem with Time Windows, Part I : Route Construction and Local Search Algorithms. *Pages 104–118 of : Transportation science*, vol. 39. trsc.1030.0056.pdf.

Braysy, Olli, & Gendreau, Michel. 2005b (February). Vehicle Routing Problem with Time Windows, Part II : Metaheuristic. *Pages 104–118 of : Transportation science*, vol. 39. trsc.1030.0057.pdf.

Clarke, G., & Wright, J. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Pages 568–581 of : Operation Research*, vol. 12. a commander.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., & Soumis, F. 2000. VRP with Time Windows. *In : cahiers du GERAD.* juste en papier.

Dantzig, G.B., Fulkerson, D.R., & Johnson, S.M. 1954. Solution of a large-scale traveling-salesman problem. *Operations Research*, **2**, 393–410. article historique TSP.

Desrochers, Martin, Desrosiers, Jacques, & Solomon, Marius. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.*, **40**(2), 342–354. a commander.

Fabien Tricoire. 2006. *Optimisation des tournées de véhicules et de personnels de maintenance : application à la distribution et au traitement des eaux.* Ph.D. thesis, Université de Nantes. these_tricoire_full.pdf.

Feillet, Dominique, Dejax, Pierre, Gendreau, Michel, & Gueguen, Cyrille. 2004. An exact algorithm for the elementary shortest path problem with resource constraints : Application to some vehicle routing problems. *Networks*, **44**(3), 216–229. shortest_path.pdf.

Fisher, M.L., & Jaikumar, R. 1978. *A decomposition algorihtm for large-scale vehicle routing.* Tech. rept. 78-11-05. University of Pennsylvania. article historique VRP.

Fisher, M.L., & Jaikumar, R. 1981. A generalized assignment heuristic for vehicle routing. *Networks*, **11**, 109–124. article historique VRP.

Gambardella, L. M., Taillard, E., & Agazzi, G. 1999. *MACS-VRPTW : A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows.* Tech. rept. IDSIA-06-99. IDSIA, Corso Elvezia 36, 6900 Lugano, Switzerland. tr-idsia-06-99.pdf.

Homberger, Jörg, & Gehring, Hermann. 2005. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational research*, **162**(march), 220–238. sdarticle.pdf.

Ioannou, G, Kritikos, M, & Prastacos, G. 2001. A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, **52**(5), 523–537. a commander.

Libertad Tansini, Maria Urquhart, & Omar Viera. 2001. *Comparing assignment algorithms for the Multi-Depot VRP*. Tech. rept. University of Montevideo. TR0108.pdf.

Olli Bräysy, Wout Dullaert, & Gendreau, Michel. 2004. Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows. *Pages 587–611 of : Journal of Heuristics*, vol. 10. Springer Netherlands. a commander.

Peter Francis, Karen Smilowitz & Michal Tzur. 2004 (september). The period vehicle routing problem with service choice. WP_04_005.pdf.

Potvin, J.-Y., & Rousseau, J.-M. 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Pages 331–340 of : European Journal of Operational Research*, vol. 66. a commander.

Solomon, M.M. 1987. Algorithms for the vehicle routing and scheduling problem with time windows constrains. *Operations Research*, **35**, 254–265. article histo heuristique best insertion.

ANNEXE A

ASPECT LOGICIEL

A.1 Introduction

Ce document est le guide de l'utilisateur de l'application VrpSolver. VrpSolver est une application java développé sous Eclipse et utilisant le solver commercial CPLEX 10.0. VrpSolver et tous les outils connexes ont été développés dans le cadre d'un stage de fin d'étude à France Telecom R&D. Cette application sert à la modélisation de problèmes de tournées de véhicules issus de la réalité industrielle et à leur résolution à l'aide de CPLEX. Nous allons étudier le principe de fonctionnement général du programme, puis son utilisation proprement dite. Enfin, nous présenterons les outils en ligne de commandes développés pour générer ou exploiter les données ainsi qu'une partie plus technique à destination des programmeurs.

machine virtuelle	java version 1.5.0_07
solveur	ILOG CPLEX 10.000
développement	Eclipse SDK 3.2.0
6 packages, 39 classes, 8400 lignes de code	

Tab. A.1 – Carte d'identité VrpSolver

A.2 Présentation générale

Nous allons présenter le fonctionnement de VrpSolver, Nous allons présenter certaines classes du programme :

- La classe principale UIC.
- Les classes de tests de performances.
- La classe permettant de régler la plupart des variables statiques du programme.

A.2.1 La classe UIC

Fonctionnement

UIC est la classe mère du programme, elle synthétise toutes les actions du programmes associées à un problème. Elle parse les fichiers d'entrées du problème, gère l'interaction avec CPLEX et crée les fichiers générés au cours d'une exécution. Nous présentons un schéma illustrant le principe général de fonctionnement du programme en figure A.1. Nous donnerons plus de détails sur les fichiers d'entrée du programme dans le paragraphe suivant et en section A.3.

Les arguments et options du programmes

Nous allons présenter les arguments nécessaires au programme pour fonctionner :
- Fichier compétences des techniciens.
- Fichier des commandes.
- Fichier de géographie.
- Répertoire d'export (pour les fichiers générés).
- Nom des fichiers d'export sans extension.
- Nombre de jours de planification.

Nous rappelons que la machine virtuelle installée sur la machine doit être liée à la bibliothèque CPLEX pour un fonctionnement correct du programme. De plus, afin d'améliorer les performances du programme, on peut régler la mémoire dont dispose la machine virtuelle lors de son exécution grâce aux options :
- -Xmsn : Specifies the initial size of the memory allocation pool. This value must be greater than 1000.
- -Xmxn : Specifies the maximum size of the memory allocation pool. This value must be greater than 1000.

Exemple de ligne de commande lançant l'application :
cd /workspace2/vrp ;
java -classpath /usr/ilog/cplex100/lib/cplex.jar :/home/atom8230/workspace2/vrp
-Djava.library.path=/usr/ilog/cplex100/bin/x86_rhel4.0_3.4 -Xms1000m -Xmx1000m -Xss3M
elements.UIC heuristic /home/atom8230/data/middle_instance/uic_villes.txt
/home/atom8230/data/middle_instance/competences.txt
/home/atom8230/data/middle_instance/commandes.txt /home/atom8230/resultats instance 5

Les sorties du programmes

Fichiers de sortie du programme :
- Répertoire jour_* et fichiers .sgeo .bgeo : les fichiers décrivant l'ensemble des sites et les tournées de chaque technicien pour un jour donné.
- .aff : le fichier d'affectation des techniciens à un job.
- .log : le fichier de log du programme.
- cplex.log : le fichier ou a éventuellement été redirigé la sortie standard de CPLEX.
- .xml : le fichier décrivant les tournées au format xml (voir section A.6) et sa feuille de style XSL associée permettant un affichage dans n'importe quel navigateur.
- sol.xml : le fichier généré par CPLEX décrivant la solution au format xml..

A.2.2 La classe Launcher

La classe Launcher est une interface graphique permettant d'initialiser la classe UIC. Lors du démarrage, elle affiche une fenêtre (copie d'écran en figure A.2) permettant la saisie de chacun des champs nécessaire au lancement de la classe UIC.

Une fois la résolution effectuée, la classe Launcher affiche le fichier de log dans une nouvelle fenêtre (figure A.4 71). Cette affichage est aussi disponible dans la classe UIC, en mettant dans le fichier de configuration .vrpsolver la ligne SHOW_LOG=0. des champs nécessaires au lancement de la classe UIC.

A.2.3 Les classes de tests

Nous avons développé plusieurs classes de tests, chacune répondant à des besoins spécifiques. La classe Test nous permet de tester les performances sur un ensemble d'instances et la classe TestHeuristic teste les performances de l'heuristique et des méthodes d'amélioration locales.

Test

Avec cette classe, on teste un ensemble d'instances en faisant varier les paramètres suivants : le fichier de géographie, le nombre de techniciens, le nombre de jours de planification. Nous allons présenter les arguments nécessaires au lancement de cette classe :

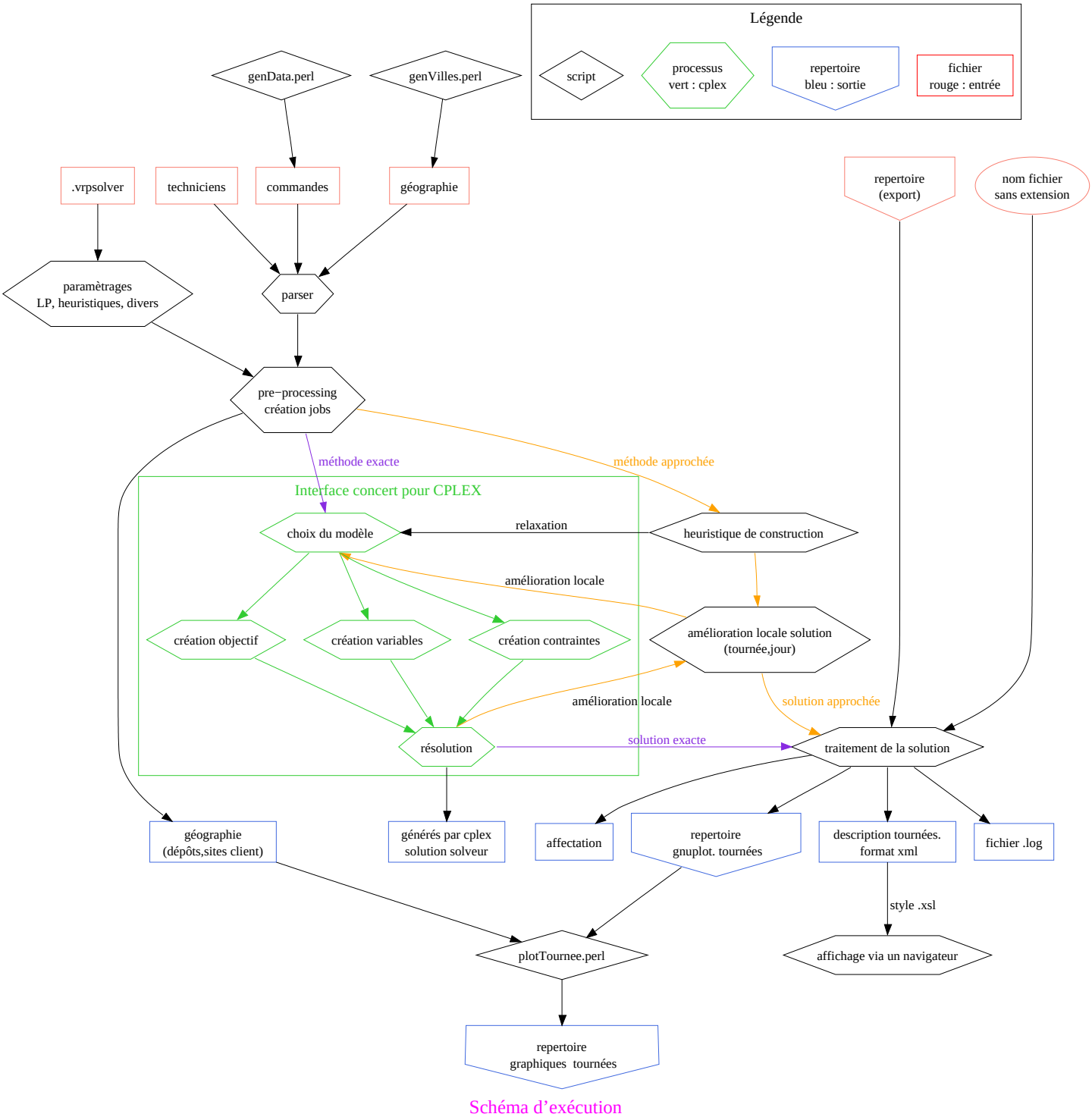

FIG. A.1 – Principe de fonctionnement de VRPSOLVER. Légende forme : diamant script ; boîte fichier ; maison inversé répertoire ; polygone action du programme. Légende couleur : bleu entrée ; rouge. sortie ; vert CPLEX

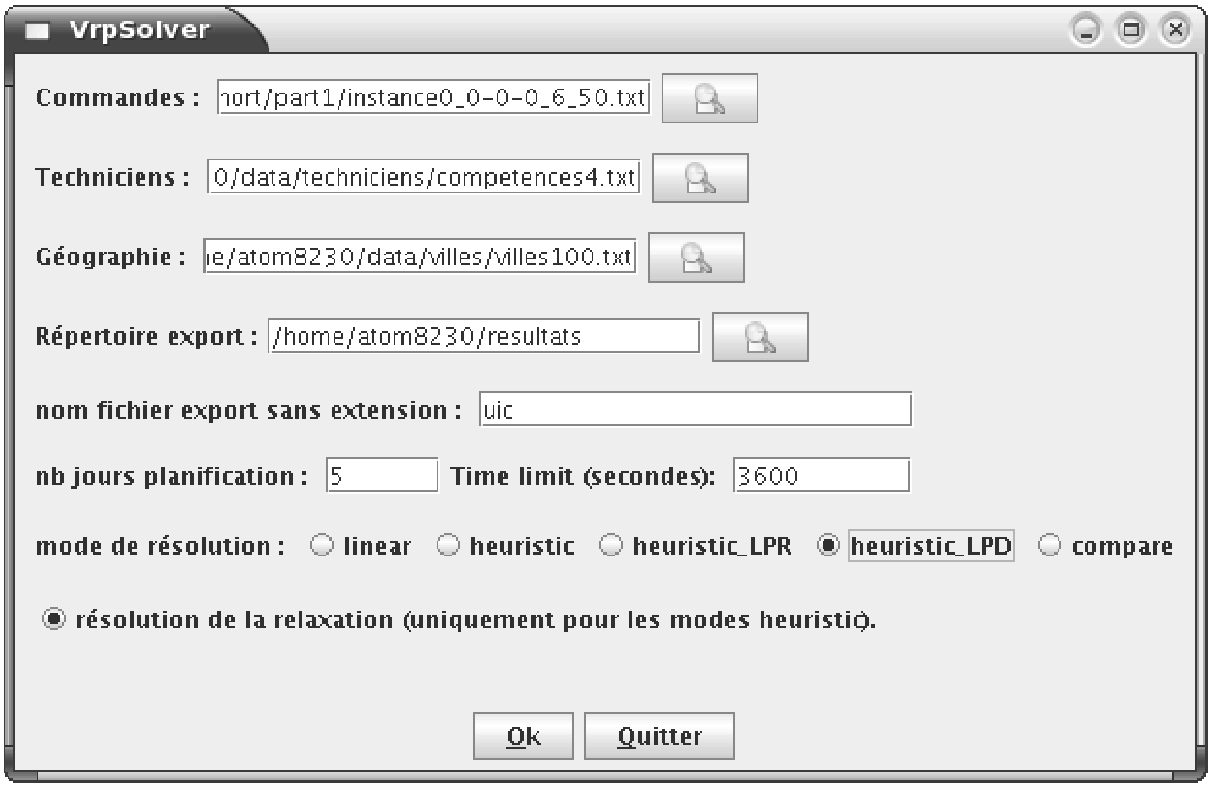

FIG. A.2 – Copie d'écran de la fenêtre de lancement de la classe Launcher.

- Le nombre de jobs des instances.
- Le répertoire contenant les instances.
- Le répertoire d'export.

Les sorties sont toutes écrites dans le répertoire d'export. On place dans un répertoire spécifique toutes les sorties associées à un paramétrage et à une instance. On écrit aussi un fichier destiné à être lu par un script octave (langage proche de Matlab) afin de traiter les résultats. Ce fichier contient :

- Le paramétrage.
- Le statut CPLEX à la fin de l'exécution.
- Le nombre de noeuds visités.
- Les gap initiaux et finaux.

TestHeuristic

Cette classe sert à tester les performances de l'heuristique de construction et des procédures d'amélioration locale pour un jeu d'instances et un paramétrage donné. Les arguments nécessaires au lancement de cette classe sont :

- Le répertoire contenant les instances.
- Le répertoire d'export.

Les sorties sont toutes écrites dans le répertoire d'export. On génère aussi une matrice destinée à octave contenant pour chaque instance :

- Nombre de jobs.
- Nombre de jobs restant.
- La valeur de l'objectif du problème relaxé.
- La valeur de l'objectif après la construction.
- La durée de la construction.
- La valeur de l'objectif après l'amélioration par tournée.
- La durée de l'amélioration par tournée.
- La valeur de l'objectif après l'amélioration par jour.

– La durée de l'amélioration par jour.

A.2.4 Config

La classe Config sert à la configuration d'un certain nombre de variables statiques du programme. La configuration est lu dans le fichier « .vrpsolver » de votre répertoire home. Ce fichier est lu lorsque vous lancez une exécution des classes décrites ci-dessus. Cependant, cette classe possède une méthode main parsant votre fichier de configuration s'il existe, puis vous affichant la configuration courante au format du fichier de configuration.

Pour plus de détails sur les variables à configurer, qui sont toutes des variables statiques du programme, veuillez vous reporter au manuel du programmeur (javadoc).

A.3 Fichiers d'entrée du programme

A.3.1 Géographie

La structure du fichier correspondant à la géographie peut avoir deux syntaxes :
– ligne 1 : nombre de villes $=n$
– ligne 2 : type de syntaxe choisi = 0
– ligne 3 : $ville_1; codepostal_1; zone_1; x_1; y_1$
– ...
– ligne $k+2$: $ville_k; codepostal_k; zone_k; x_k; y_k$
– ...
– ligne $n+2$: $ville_n; codepostal_n; zone_n; x_n; y_n$

x_k et y_k correspondent à des coordonnées euclidiennes associées à la ville. Si le type de syntaxe choisi est 1, on remplace ses coordonnées par la ligne d'une matrice de distance correspondant à la ville.

A.3.2 Techniciens

Le fichier fournissant les informations associées aux techniciens a la forme suivante :
– ligne 1 : *nb_competences*.
– lignes 2 : nom_tech_1 ; $code\_tech_1$; $zone\_geographique_1$;$base_1$; $competence_1^1$; ... ; $competence_1^{nb}$.
– lignes 3 : nom_tech_2 ; $code\_tech_2$; $zone\_geographique_2$; $base_2$;$competence_2^1$; ... ; $competence_2^{nb}$.
– ...
– lignes k+1 : nom_tech_k ; $code\_tech_k$; $zone\_geographique_k$; $base_k$; $competence_k^1$; ... ; $competence_k^{nb}$.

A.3.3 Commandes

Le fichier représentant les commandes de l'UIC a la structure suivante :
– ligne 1 : en-tête décrivant le format d'entrée des commandes : « id_comm ; nom ; type_intervention ; duree(minutes) ; id_ville ; passage_depot ; priorite ; competence ; date_arrivee_si ; release_date ; date_souhaitée ; date_contractuelle ».
– ligne 2 : id_comm_1 ; nom_1 ; $type\_intervention_1$; $duree_1$; id_ville_1 ; $passage\_depot_1$; $priorite_1$; $competence_1$; $date\_arrivee\_si_1$; $release\_date_1$; $date\_souhaitee_1$; $date\_contractuelle_1$.
– ligne 3 : id_comm_2 ; nom_2 ; $type\_intervention_2$; $duree_2$; id_ville_2 ; $passage\_depot_2$; $priorite_2$; $competence_2$; $date\_arrivee\_si_2$; $release\_date_2$; $date\_souhaitee_2$; $date\_contractuelle_2$.
– ...
– ligne l+1 : id_comm_l ; nom_l ; $type\_intervention_l$; $duree_l$; id_ville_l ; $passage\_depot_l$; $priorite_l$; $competence_l$; $date\_arrivee\_si_l$; $release\_date_l$; $date\_souhaitee_l$; $date\_contractuelle_l$.

A.4 Outils en ligne de commande

Nous allons présenter quelques outils en ligne de commande développés pour la génération de donnée mais aussi leur exploitation.

A.4.1 genVilles.perl

Le script perl genVilles.perl sert à générer le fichier associé à la géographie du problème.
Nous utilisons une liste de site fournie par L'UIC. Cependant, nous n'avons pas d'informations concernant les distances inter-sites, nous devons donc les générer. Nous n'avons pas implémenté de méthodes de générations pour la matrice de distance.
Pour la génération de coordonnées euclidiennes associées aux sites, nous nous servons des instances de Solomon qui sont les instances utilisées dans la plupart des articles sur le Vehicle Routing Problem. Cependant, nous ne pouvons les utiliser en l'état vu les entrées attendues du programme. Nous les utilisons pour attribuer les coordonnées aux sites. Le script perl genVilles.perl lit séquentiellement les deux fichiers et attribue au site n les coordonnées du n-ième site des instances de Solomon (http ://www2.imm.dtu.dk/-jla/solomon.html). Il existe trois types d'instances de Solomon :

1. Random : les coordonnées sont générées aléatoirement.
2. Cluster : les coordonnées appartiennent à des clusters.
3. Random and Cluster : les coordonnées sont générées alternativement comme aléatoire ou appartenant à un cluster.

A.4.2 Techniciens

Lors des tests, nous avons utilisé de 4 à 6 techniciens, les quatre premiers techniciens permettent de recouvrir l'ensemble des compétences. Nous rajoutons ensuite un ou deux techniciens.

A.4.3 genData.perl

La génération des commandes a été la partie la plus complexe, réalisée grâce au script genData.perl, notamment car nous n'avons pas pu rassembler assez d'information sur les échéanciers réels de l'UIC. La génération des commandes est effectuée grâce au script perl genData.perl.

Arguments

genData.perl prends les arguments suivants :

1. Le nombre de jobs.
2. Le temps maximum pour la date de début au plus tôt.
3. Le nombre de sites.
4. Le nombre de compétences.
5. Le mode choisi (par exemple 0-0-0).
6. défaut ou 0 mode incrémentale, un chiffre mode multi.

Expliquons les deux derniers paramètres : le mode est codé sur 3 bits (0 absence,1 présence), le premier correspond à la préemption, le second à la précédence, le troisième à la synchronisation.

Le mode incrémentale crée des instances allant de MIN_JOB (variables globales du programme) au nombre de jobs, l'instance de taille $n+1$ ne différant de l'instance de taille n que par le job $n+1$. Le mode multi permet de à générer plusieurs instances aléatoires d'une taille donné.

Génération des champs

Nous allons maintenant détailler la génération de chaque champs :

1. L'ID corresponds à l'ordre de génération des instances.
2. Le nom est tiré aléatoirement dans une liste de noms.
3. Le type du job dépend d'un vecteur de probabilité lui-même dépendant du mode choisi.
4. Pour chaque type de job, on tire aléatoirement une durée dans un intervalle donné.
5. On tire le passage au dépôt avec une probabilité donné par le mode.
6. On tire la priorité aléatoirement grâce à un vecteur de probabilité.

7. La compétence nécessaire est tirée aléatoirement.
8. On tire le site du job aléatoirement.
9. La date d'entrée dans le Système d'information est la date courante.
10. La date de début au plus tôt est tirée aléatoirement entre 0 et le second argument du programme.
11. La date de début au plus tôt est augmentée du nombre de jours minimum nécessaire à l'exécution du job plus une quantité aléatoire entre 0 et MAX_DC (variables globales du programme) qui nous donne la date de fin souhaitée et la date contractuelle (elles sont égales).

Toutes les probabilités servant à la génération des données peuvent être modifiées en réglant les variables globales au début du script et le bloc de code choisissant les vecteurs de probabilités selon le mode de l'instance choisi.

A.4.4 plotTournee.perl

Le script perl PLOTTOURNEE.PERL permet d'exploiter les sorties du programme représentant les tournées dans le plan. En effet, on crée un répertoire par jour de planification contenant un ensemble de fichiers représentant les tournées. Ce script crée des fichiers gnuplot associés à chaque jour de planification, puis trace les graphiques. On peut facilement modifier le format d'export des graphiques en transformant deux variables de celui-ci. Nous présentons un exemple de graphe généré à l'aide de ce script en figure A.4.

A.5 Utilisation avancée

Nous allons maintenant détailler quelques points qui seront utiles aux utilisateurs désireux de modifier des paramètres avancés du programme ou à un éventuel programmeur. Nous allons présenter un diagramme d'héritage de toutes les classes utilisant le solveur CPLEX en figure A.3 page 70.

Pour plus de précisions sur le code, veuillez vous reportez à la javadoc.

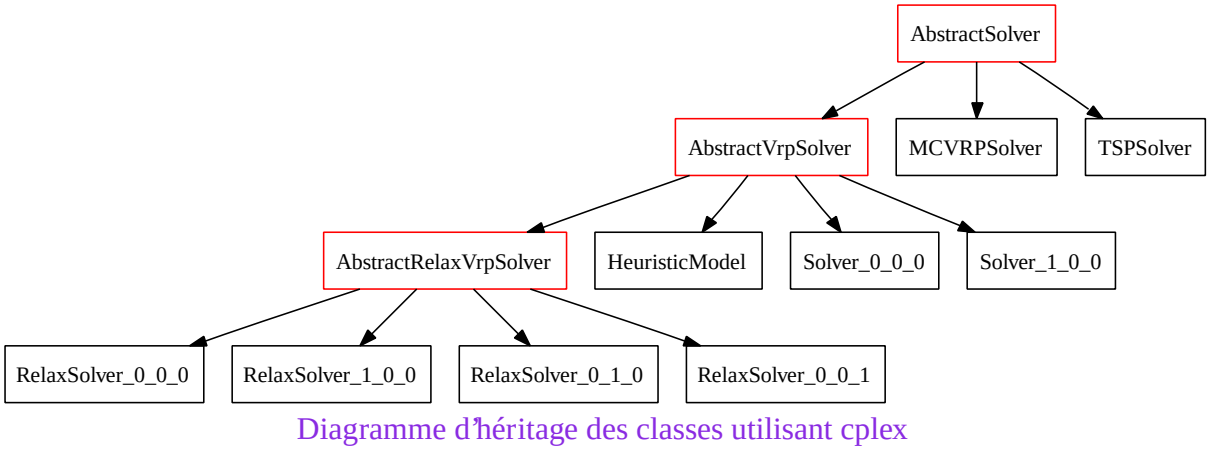

FIG. A.3 – Diagramme d'héritage des classes utilisant le solveur commercial CPLEX.

A.6 Exemple de fichiers générés

Les fichiers générés par PLOTTOURNEE.PERL représentant les tournées pour un jour donné.

journal : uic.log

```
fichier de log du 24/8/2006 à 11h1mn

%%%%%%%%%%%%%%%% INITIALISATION %%%%%%%%%%%%%%%%


fichier de villes parsé :/home/atom8230/data/villes/villes100.txt
fichier de techniciens parsé :/home/atom8230/data/techniciens/competences4.txt
fichier des commandes parsé
:/home/atom8230/data/heuristic_short/part1/instance0_0-0-0_6_50.txt
site clients ecrit dans le fichier : /home/atom8230/resultats/uic.sgeo
bases des techniciens ecrites dans le fichier : /home/atom8230/resultats/uic.bgeo

mode de l'instance : 0-0-0
Le nombre de techniciens  est : 4
Le nombre de jobs créé est : 50
repartition par type : 8 42 0 0 0
HORIZON ------>5
Ecriture de la liste des jobs dans le fichier : /home/atom8230/resultats/uic.job


%%%%%%%%%%%%%%%%%%%% HEURISTIQUE DE CONSTRUCTION %%%%%%%%%%%%%%%%%%

nombre de jobs non servis : 0
temps construction : 0,02 s
nombre de jours de planification: 8
nombre de tournées créées : 14
valeur de objectif : 1950,68

optimisation par jour : 3 jours améliorées
temps amélioration : 28.973 s
nouvelle valeur de objectif : 1426,61

ecriture des tournées dans :/home/atom8230/resultats/uic_heur.xml
ecriture des couplages : /home/atom8230/resultats/uic.match

%%%%%%%%%%%%%%%%%%%%%% CPLEX %%%%%%%%%%%%%%%%%%%%%%

export Model to /home/atom8230/resultats/uic.lp :ok
valeur objectif problème relaxé :274,2

%%%%%%%%%%%% fin %%%%%%%%%%%%
```

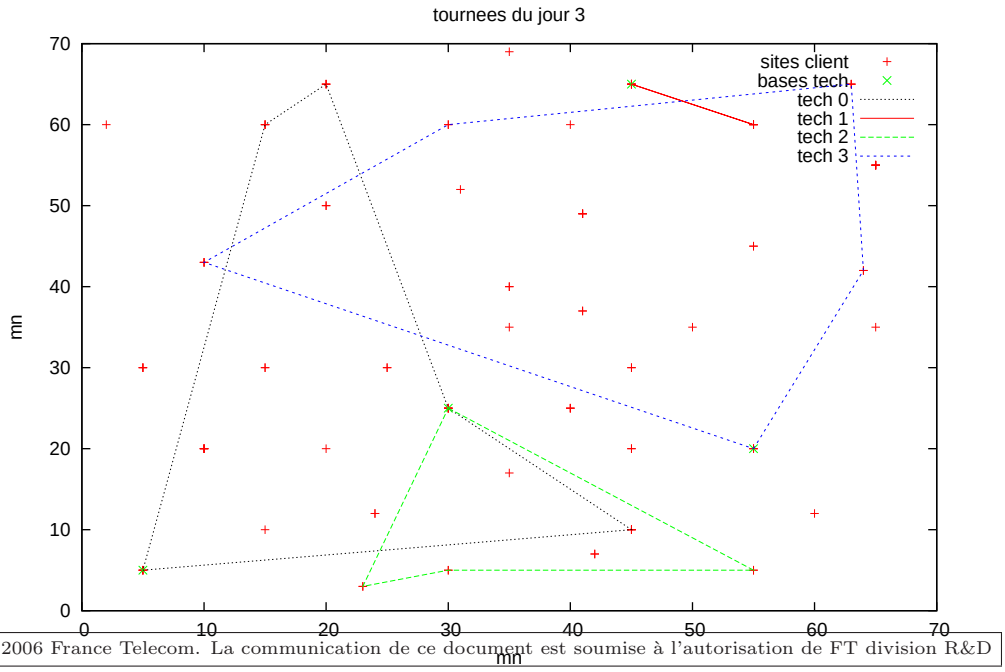

FIG. A.4 – Exemple de fichier de log et de fichier généré par PLOTTOURNEE.PERL représentant les tournées d'une journée.

TABLE DES MATIÈRES

INDEX