$N°$d'ordre : 3324

# THÈSE

**présentée à**

# L'UNIVERSITÉ BORDEAUX I

**ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE**

**par Sophie MICHEL**

**POUR OBTENIR LE GRADE DE**

## DOCTEUR

**SPÉCIALITÉ : Mathématiques Appliquées**

**********

## OPTIMISATION DES TOURNEES DE VEHICULES COMBINEES A LA GESTION DE STOCK

**********

Soutenue le : 11 Décembre 2006

Après avis de :

| | | | | |
|---|---|---|---|---|
| **MM.** | Pierre DEJAX | Professeur | Ecole des Mines de Nantes | **Rapporteurs** |
| | Frédéric SEMET | Professeur | Université de Valenciennes | |

Devant la commission d'examen formée de :

| | | | | |
|---|---|---|---|---|
| **MM.** | Paul MOREL | Professeur | Université Bordeaux 1 | **Président** |
| | Pierre DEJAX | Professeur | Ecole des Mines de Nantes | **Rapporteurs** |
| | Frédéric SEMET | Professeur | Université de Valenciennes | |
| | Fabien RODES | Industriel | Synoptis (France) | **Examinateurs** |
| | François VANDERBECK | Professeur | Université Bordeaux 1 | |

**- 2006 -**

# Remerciements

Je tiens avant tout à remercier François Vanderbeck pour la confiance qu'il m'a accordée durant ces trois années de thèse. Son regard critique et ses conseils me permirent de mener cette recherche.

Je remercie les professeurs Pierre Dejax et Frédéric Semet pour l'intérêt qu'ils ont porté à mon travail en acceptant d'être rapporteurs et pour leur participation au jury. Merci également au professeur Paul Morel d'avoir présidé le jury.

Cette thèse a été effectuée en collaboration avec l'entreprise Synoptis. Je remercie Fabien Rodes pour la confiance qu'il m'a témoignée, sans oublier le personnel pour son accueil lors de mes différents séjours.

Je tiens aussi à mentionner le plaisir que j'ai eu à travailler au sein de l'IMB, et je remercie ici mes collègues (et ex-), en particulier Pierre pour ses remarques constructives et sa disponibilité ainsi que Marie.

Merci à ma famille...

# Table des matières

# Liste des figures

# Liste des tableaux

# Liste des exemples

# Introduction

Dans le secteur des transports, les modèles mathématiques de la recherche opérationnelle sont largement utilisés. Le modèle de base consiste à déterminer une tournée de longueur minimale pour un véhicule passant une seule fois chez chaque client, c'est le problème du voyageur de commerce (TSP, "Travelling Salesman Problem"). Dans une version plus complexe, la planification logistique consiste à construire les routes effectuées par des véhicules qui vont livrer aux clients les quantités demandées. C'est le problème de tournées de véhicules (VRP, "Vehicle Routing Problem"). Les progrès récents de la recherche opérationnelle permettent de mieux résoudre ces problèmes. Comme les industries de transport et de logistique ont un besoin continuel d'améliorer leur gestion et de réduire leurs coûts, des problématiques de plus en plus complexes apparaissent. Pour répondre à leur demande, les fournisseurs de logiciel doivent s'adapter en proposant des outils d'aide à la décision compétitifs. Cette thèse a été motivée par une demande de notre partenaire industriel Synoptis qui développe un outil de ce type. L'application traitée est la collecte des points de recyclage d'apports volontaires sur un site français comportant 260 bornes de verre à recycler.

La variante du problème de tournées de véhicules qui nous intéresse allie la planification des tournées de livraison et la gestion des niveaux de stock chez les clients. Elle est connue dans la littérature sous le nom d'IRP, "Inventory Routing Problem". Ainsi, aux modèles opérationnels de construction de tournées s'ajoutent des modèles de planification tactique sur un horizon de temps plus long. C'est un problème sur de multiples périodes. Pour chacune de ces périodes, les décisions à prendre sont : (i) quel client servir ?, (ii) quelle quantité livrer ou enlever ?, (iii) quelles routes utiliser ? Suivant les hypothèses retenues, dans le choix de l'horizon de temps et les contraintes prises en compte pour les routes, ce problème se situe à la frontière des modèles de décisions à long terme (niveau tactique) et à court terme (niveau opérationnel). Dans le

cadre de notre étude, nous avons adopté le point de vue tactique. La solution tactique pourra aussi servir de base pour la construction des solutions opérationnelles en lui apportant une vue à long terme.

La problématique d'optimisation des tournées sur un horizon assez grand dans le cas concret de la collecte de bornes de déchets recyclables combine trois difficultés : celle des problèmes de tournées de véhicules classiques, celle de la planification tactique des collectes de bornes et enfin le fait d'être confronté à des problèmes de grandes dimensions. Néanmoins, notre problème a une structure particulière avec deux composantes bien identifiées : un modèle de tournées de véhicules dans lequel on détermine les routes de livraison pour chaque période, et un modèle de gestion des stocks dans lequel on planifie les visites des clients en fonction de leur taux de consommation et leur capacité. De plus dans notre application, la gestion de stock sous-jacente est simplifiée : à chaque collecte, la borne est entièrement vidée. Pour exploiter cette structure, les méthodes de décompositions du type Dantzig-Wolfe sont bien adaptées.

Les chapitres 1 à 5 de ce mémoire introduisent le problème et les méthodes de résolution. Ils présentent les différentes modélisations possibles pour le problème de tournées de véhicules et le problème de planification dictée par la gestion des stocks. Cette étude permet de choisir la modélisation la mieux adaptée. Les chapitres 6 et 7 sont consacrés à l'algorithme de résolution pour la modélisation choisie. Les tests sont effectués sur un ensemble de jeux de données générées aléatoirement et issues de l'application réelle. Les résultats sur les données industrielles sont présentés au chapitre 8.

Le premier chapitre définit en détail le problème traité. Différents choix pour modéliser les tournées et divers scénarios pour la planification tactique sont proposés. Le but premier est de définir un planning périodique, mais trois scénarios sont envisagés, selon que la situation initiale est connue ou non, et que l'horizon de temps est fini ou non. Lorsque l'horizon est infini, à chaque tournée est associée une périodicité choisie dans un ensemble fini. L'objectif principal est de minimiser le nombre de véhicules utilisés. L'objectif secondaire est de construire des routes de collecte qui minimisent la durée des trajets tout en dessinant un découpage régionalisé de l'espace. Des formulations naturelles du problème sont données. Vient ensuite une classification des modèles traités dans la littérature.

Le chapitre 2 donne un aperçu de l'approche de décomposition utilisée pour

résoudre des problèmes de grande taille. En effet, les formulations naturelles de notre problème font intervenir des variables de planification des visites chez les clients et des variables de construction de routes empruntées par les véhicules. La méthode de décomposition de Dantzig-Wolfe exploite cette structure pour reformuler le problème, et le résoudre ensuite par génération de colonnes. Différentes décompositions sont discutées. Suit une revue de la littérature sur les heuristiques basées sur cette approche de décomposition.

Le chapitre 3 est consacré au problème de tournées à un seul véhicule et sur une période. Il s'agit de déterminer l'ensemble de clients à visiter et pour chacun d'eux optimiser les quantités collectées. Une première modélisation définit une route opérationnelle qui démarre du garage, visite les clients et finit au vidage (terme employé pour désigner la déchetterie). Une deuxième modélisation consiste à définir une route comme un ensemble homogène de clients (le coût de transport est alors estimé par la somme des distances à un centre). Chacune de ces modélisations mène à un sous-problème classique. Une revue des méthodes de résolution proposées dans la littérature est faite.

Le chapitre 4 traite du modèle de gestion du stock d'un seul client. Ce problème consiste à élaborer un planning des visites. Le type de solution obtenue dépend du scénario choisi pour la planification tactique. Lorsque l'horizon de temps est fini, les différentes formulations possibles sont inspirées de la littérature sur le problème de production par lot. Lorsque l'horizon est théoriquement infini, la politique imposant aux routes d'être périodiques restreint l'espace des solutions, le problème de gestion de stock n'est alors plus indépendant du problème de tournées. Afin de mieux comparer ces approches, un exemple simple du problème global avec deux clients est utilisé pour illustrer l'inconvénient de chacun d'eux.

Dans le chapitre 5, le problème global est formulé en utilisant les différentes modélisations des problèmes de tournées de véhicules et de gestion des stocks. La relaxation linéaire de ce problème est résolue par génération de colonnes. Deux formulations sont explicitées : l'une sur un horizon fini et l'autre sur un horizon infini. Quelques résultats numériques aident à comparer ces deux formulations et l'impact du modèle de tournées sur le temps de calcul. La formulation la mieux adaptée à notre problème est basée sur les ensembles homogènes de clients et sur un horizon des temps infini, l'ensemble des périodicités des routes étant restreint. Cette formulation est appelée formulation discrète.

Le Chapitre 6 est consacré au calcul des bornes duales. Tout d'abord, on fait la constatation que la formulation discrète souffre d'une symétrie : les solutions obtenues en appliquant une permutation cyclique des dates de départ définissent une classe d'équivalence. Une formulation définie en agrégeant les périodes est alors proposée. Nous montrons que ces deux formulations donnent la même borne duale. Ensuite, une version tronquée de la méthode de Branch-and-Price-and-Cut appliquée à la formulation agrégée permet d'obtenir la borne duale qui servira à évaluer nos heuristiques primales.

Dans le chapitre 7, différentes heuristiques basées sur l'approche de décomposition sont testées. En particulier, diverses versions de l'heuristique qui arrondit itérativement des colonnes de la solution continue sont comparées. Ensuite pour tenter d'améliorer les solutions entières obtenues, une heuristique de recherche locale est développée. Ces deux heuristiques utilisent les deux formulations, discrètes et agrégées.

Enfin, dans le chapitre 8 sont présentés les résultats de notre algorithme sur les données réelles fournies par notre partenaire industriel. Les solutions obtenues sont visualisées et différentes restrictions de l'espace sont essayées. De plus, nous effectuons une comparaison du comportement moyen sur une semaine entre la meilleure solution tactique obtenue par notre algorithme et la solution opérationnelle de l'année 2005.

# 1

# Le Problème de tournées de véhicules combinées à la gestion de stock

Optimiser des tournées de véhicules ou gérer des stocks sont des problèmes qui, pris séparément, sont bien connus. Cependant il existe des domaines où il est nécessaire de planifier des tournées en fonction de la gestion des stocks. Ce chapitre est consacré à la présentation détaillée du problème que nous traitons et à un état de l'art sur les problèmes de ce type.

## 1.1 Problème traité

Le problème de tournées de véhicules ("Vehicle Routing Problem" : VRP) est un des problèmes d'optimisation combinatoire les plus étudiés (voir les travaux de revue de la littérature [7, 80]). Etant donné un ensemble de clients demandant une livraison, il s'agit de déterminer une route pour chaque véhicule d'une flotte homogène satisfaisant les contraintes :

- la route démarre et finit au dépôt ;
- les quantités livrées par un véhicule ne peuvent excéder sa capacité ;
- chaque client doit être affecté à un véhicule.

L'objectif est de minimiser les distances parcourues.

Plusieurs variantes du problème de tournées de véhicules existent. On peut notamment citer le cas où la flotte est hétérogène [78], c'est à dire que différents types de véhicules sont disponibles. Des fenêtres de temps durant lesquelles les clients doivent être livrés peuvent être définies [34]. Les véhicules peuvent aussi

partir de plusieurs dépôts [77], livrer certains points et charger de la marchandise en d'autres [32]. Plusieurs objectifs sont aussi possibles ; les temps de parcours peuvent être choisis à la place des distances. Pour déterminer la taille de la flotte, le nombre de véhicules est minimisé.

Le cas d'un seul véhicule sans capacité est le problème du voyageur de commerce ("Travelling Salesman Problem" : TSP [61]), qui consiste à trouver un tour optimal passant une fois chez chaque client. Dans le problème de tournées sur les arcs ("Arc Routing Problem" : ARP [36]), les demandes ne se trouvent plus sur les sommets mais sur les arcs. Il s'agit donc de trouver un ensemble de tournées minimisant la distance et couvrant l'ensemble des tronçons du réseau.

Ces problèmes varient aussi suivant l'horizon de temps considéré pour la planification. Quand on doit déterminer des tournées sur plusieurs périodes en s'assurant que chaque client est visité périodiquement, soit la fréquence de passage chez chaque client est prescrite, soit c'est l'écart maximum et minimum entre deux visites qui est prescrit, ou encore à chaque client est associé un ensemble de scénarios de visites possibles. Ces variantes sont connues sous le nom de PVRP qui est la version périodique du VRP (voir [67] pour une classification et une revue de la littérature).

Notre problématique est plus complexe que le PVRP. La planification des tournées de livraison doit tenir compte de la gestion des niveaux de stock sur les sites de livraison. Un même point de demande requiert plusieurs visites sur l'horizon de temps : celles-ci sont espacées selon une fréquence dictée par le modèle de gestion du stock de ce client et non prédéterminée comme dans le PVRP. De plus la rupture de stock doit être évitée.

Mathématiquement, le problème de tournées de véhicules combinées à la gestion de stock se traduit par un modèle de planification sur un horizon de temps où les décisions sont de trois types :
($i$) affecter des dates de passage à tous les points de demande,
($ii$) définir les volumes de livraison des points de demande à chaque passage,
($iii$) construire des tournées pour chaque date de façon à visiter les points qui y sont affectés.
L'objectif poursuivi consiste typiquement à minimiser les coûts : l'un fixe correspondant à l'utilisation des véhicules, les autres variables liés aux tournées et à la gestion des stocks.

Ce problème, connu sous le nom d'IRP pour "Inventory Routing Problem", se situe à la frontière des modèles de décisions à moyen terme (niveau tactique) et à court terme (niveau opérationnel) suivant le type d'horizon choisi et les hypothèses retenues. Beaucoup de variantes sont discutées dans la littérature (voir section 1.3). L'IRP est $NP-$difficile puisque l'on retrouve le problème de tournées de véhicules comme sous-problème. En effet, lorsque le planning de livraison est fixé ou lorsque c'est le client qui gère son stock et place lui même sa commande, on doit résoudre pour chaque période un VRP, i.e. répondre à la question "Quelles routes utiliser ?".

Le problème de gestion de stock pour un client consiste à trouver une politique optimale de commande sur un horizon infini ou fini. L'objectif est de minimiser les coûts de stockage, de commande et de rupture de stock (lié aux demandes différées ou perdues). Ces problèmes sont bien connus [55], et de nombreux modèles sont proposés selon les hypothèses choisies.

Dans le cas d'un horizon infini, les demandes sont déterminées à l'aide d'un taux de consommation fixe (déterministe) ou aléatoire (stochastique). Quelques modèles sont cités ci-dessous (voir [55] pour plus de détails). Dans le cas des modèles avec demandes déterministes, on a le modèle de Wilson (EOQ, "Economic Order Quantity") qui interdit la rupture de stock. Il s'agit de définir la commande optimale $Q$ toutes les $T$ unités de temps en équilibrant les coûts de stockage et de commande. Dans une solution optimale, la commande se fait quand le stock arrive à zéro. Si les demandes différées sont permises, alors des coûts de rupture de stock s'ajoutent. Cette fois ci la solution optimale est un compromis entre les coûts de rupture de stock, de stockage et de commande. La solution définit la commande optimale $Q$ et le nombre de demandes différées $s$. Dans le cas de demandes aléatoires, il faut prévoir un stock de sécurité. On trouve des modèles à point de commande, comme le modèle $< Q, r >$ où il s'agit de commander une quantité $Q$ lorsque le niveau de stock est au niveau $r$. Une surveillance continue des mouvements est nécessaire pour réagir quand ce niveau $r$ est atteint. Dans les modèles $< R, T >$ et $< R, r, T >$, la surveillance du stock se fait toutes les $T$ unités de temps. Le stock est ramené au niveau $R$ toutes les $T$ unités de temps pour le premier modèle, et seulement si le niveau de stock est inférieur à $r$ pour le second.

Le cas d'un horizon fini où les demandes sont non stationnaires et définies pour chaque période se modélise comme un problème de production par lot (lot sizing) [72]. Pour chaque période, il s'agit de définir :

- si on commande (ou produit),
- quelle quantité on commande (ou produit),
- quelle quantité on stocke jusqu'à la période suivante.

Des demandes différées peuvent aussi être prises en compte.

Remarquons que pour la gestion de stock, deux points de vue sont possibles :

1. Dans le cas de la livraison, le client $i$ possède un stock de capacité $C_i$ qu'il consomme à un taux $d_i$. A chaque livraison, le niveau de stock augmente. Le véhicule part de l'entrepôt, livre une quantité chez certains clients et rentre au garage.
2. Dans le cas du ramassage, le client $i$ produit un stock avec un taux $d_i$, sa capacité de stockage est de $C_i$. A chaque ramassage, le niveau de stock est diminué. Le véhicule part du garage, emporte une partie de la quantité accumulée chez certains clients et va au vidage.

Le premier point de vue est le plus courant dans la littérature. Il correspond au cas du réapprovisionnement de produit comme le gaz. Le second point de vue est le cas rencontré dans l'application qui motive cette thèse. Il s'agit de la collecte des bornes de dépôt de produit à recycler tel que le verre ou le papier.

Notre partenaire industriel, Synoptis, nous a soumis cette problématique de collecte non résolue à ce jour. Synoptis nous a fourni en particulier les données de la collecte sur le verre aux points d'apport volontaires dissiminés sur plusieurs communes des Charentes. Les principales caractéristiques de l'instance reçue sont le nombre de bornes, 260, et le nombre de bornes qu'un véhicule ramasse en moyenne : environ 10. Comme chaque borne est collectée au maximum une fois par semaine, la période de base est la semaine. Les bornes ont la même capacité de stockage, leur fréquence de collecte varie de toutes les semaines à toutes les 6 semaines, voir même toutes les 12 semaines et exceptionnellement 14. La figure 1.1 illustre la répartition des bornes en fonction du nombre de périodes qui s'écoulent entre deux visites (notée P.). Trois types de milieu sont possibles pour la répartition des bornes : un milieu urbain dans lequel les bornes sont proches, un milieu rural où elles sont plus par petits groupes avec des écarts pouvant être grands entre elles et un milieu mixte où les bornes sont plus uniformément réparties. Le garage et le vidage ne se trouvent pas forcément au même endroit, mais sont éloignés des centres villes.

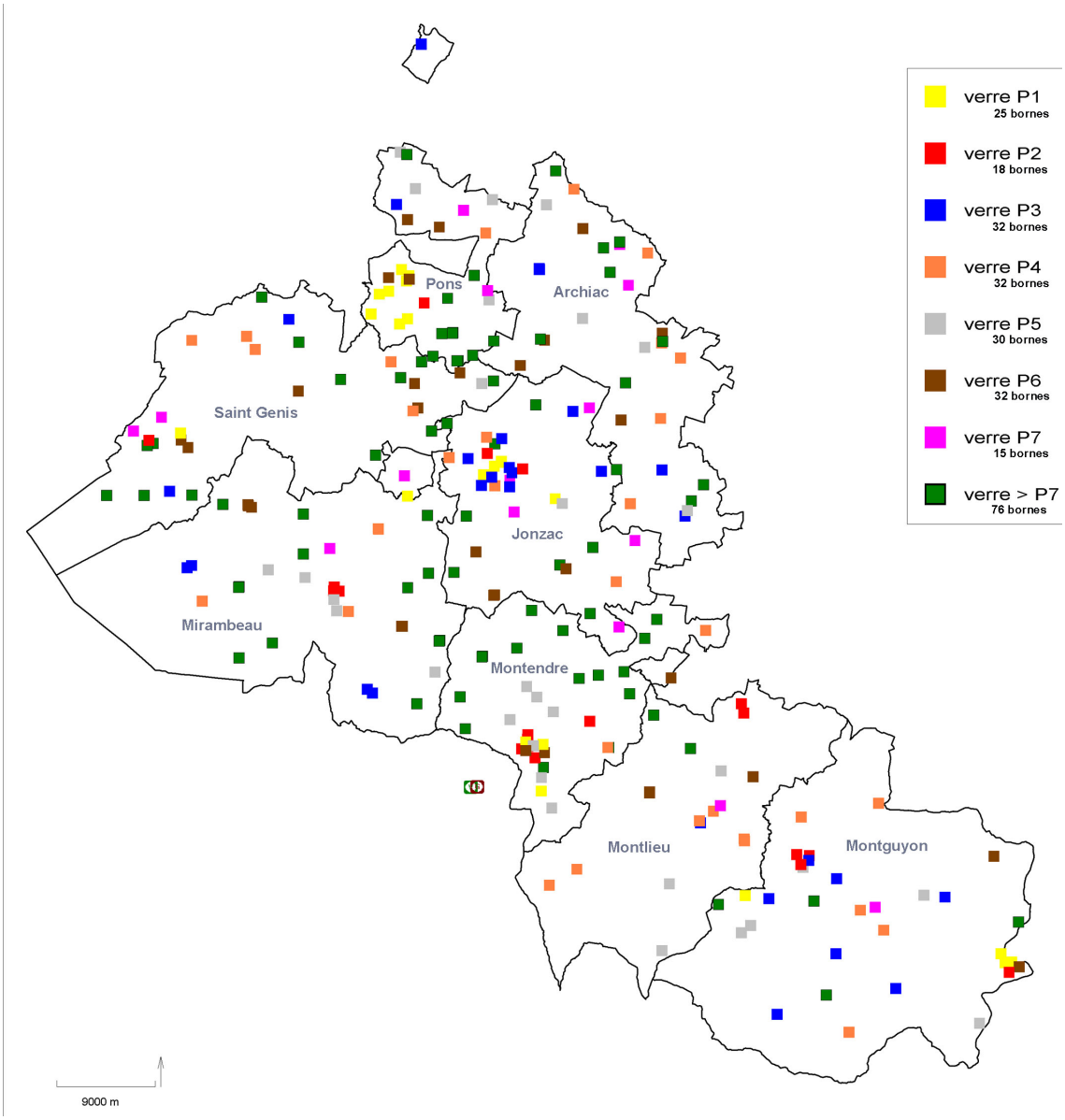



FIG. 1.1 – Répartition des bornes sur les communes des Charentes

Pour chaque période de l'horizon, les tournées de collecte doivent être optimisées. Il s'agit de définir les routes effectuées par chacun des véhicules. Deux points de vue, présentés dans la table 1.1, sont possibles pour modéliser ces routes.

**(i)** Le point de vue opérationnel consiste à générer des routes réalisables qui démarrent du garage et finissent au vidage ; les demandes affectées au véhicule ne peuvent pas excéder sa capacité.

**(ii)** Le point de vue de la planification tactique consiste à définir des ensembles de clients, dits "clusters", tels que leur demande n'excède pas la capacité du véhicule. Le coût d'un cluster est une approximation du coût d'une route réalisable qui visiterait les clients du cluster. Chaque cluster est affecté à un véhicule ; le mot tournée est alors abusivement utilisé pour désigner un cluster. Un cluster homogène tend à rassembler les clients voisins et/ou ayant le même planning de visite.

TAB. 1.1 – Modèles de tournées : route opérationnelle ou cluster

L'avantage du choix (i) comme modèle de tournées est la possibilité d'inclure en plus du respect de la capacité du véhicule d'autres contraintes opérationnelles comme une limitation en temps, en distance et éventuellement des fenêtres de temps et des rotations (lorsque l'on passe au vidage, la capacité se réinitialise). De plus, le coût réel de l'itinéraire est connu. Les routes obtenues sont opérationnelles.

Le choix (ii) permet de modéliser un autre objectif important d'un point de vue tactique qui est de régionaliser l'espace. Dans ce cas, la seule contrainte prise en compte est la capacité du véhicule, les autres seront traitées au niveau opérationnel. En choisissant une structure coût en forme d'étoile, la régionalisation de l'espace se fait en connectant des clients autour d'un centre.

La modélisation de ces deux variantes et les algorithmes de résolution associés sont étudiés dans le chapitre 3. Il en résulte que le deuxième point de vue est plus adapté à notre problème.

**Définition 1.1.1** *Une solution du problème de tournées, appelée abusivement route, représente soit une route opérationnelle réalisable, soit un cluster homogène de clients visités en précisant la quantité collectée chez chaque client visité.*

Dans notre cas pratique, le problème de planification au niveau d'un client (noté client $i$) est basé sur un modèle de gestion de stock assez simple résumé dans la table 1.2.

**(i)** Un seul produit est considéré.

**(ii)** Le taux de remplissage est déterministe.

**(iii)** A chaque collecte le stock est entièrement vidé (politique d' "order up to level").

**(iv)** Il n'y a pas de coût de stockage.

TAB. 1.2 – Modèle de gestion de stock

Les hypothèses (ii) et (iii) permettent de normaliser la quantité collectée, $q_i$, en nombre de périodes $\ell$, i.e. $q_i = d_i * \ell$ avec $d_i$ le taux de remplissage. L'hypothèse (iv) se justifie en partie du fait de l'hypothèse (ii) : il n'y a pas de coût lié au dépassement de capacité, les seuls coûts d'une politique de gestion de stock sont les coûts liés aux tournées qu'elle génère. A partir de la capacité de stockage et du taux de remplissage, l'intervalle maximal entre deux visites, noté $tmax_i$, est déterminé. Notons $tmax$ l'intervalle maximal le plus grand parmi les clients $i$, i.e. $tmax = \max\{tmax_i\}$.

**Définition 1.1.2** *Un planning est une solution réalisable du problème de gestion des stocks des clients. Il est défini par les dates de collecte des clients telles que l'espacement entre les dates du client $i$ est d'au plus $tmax_i$ périodes.*

Avec les hypothèses (ii) et (iii), la connaissance d'un planning entraîne la connaissance des quantités ramassées pour chaque client à chaque date de collecte. Dans l'exemple 1.1, un planning est représenté, les lignes $i$ sont associées aux clients et les colonnes $t$ aux périodes. Une croix $\times$ dans la case $(i., t.)$ indique que le client $i$ est collecté à la période $t$. Par exemple dans ce planning, pour le client $i2$ une quantité $2d_{i2}$ est collectée toutes les 2 périodes, tandis que le client $i5$ se voit retirer successivement une quantité $5d_{i5}$ (le planning est cyclique), puis $3d_{i5}$, puis $4d_{i5}$.

**Définition 1.1.3** *Le planning pour un seul client est appelé planning individuel.*

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i2 | × | | × | | × | | × | | × | | × | |
| i3 | | × | | | × | | | × | | | × | |
| i4 | | | | × | | | | × | | | | × |
| i5 | | | × | | | × | | | | × | | |
| i6 | × | | | | | | × | | | | | |

Exemple 1.1: Un planning

Notre objectif est de déterminer le mode de fonctionnement en équilibre du système. On parle de solution d'équilibre par opposition à une solution destinée à gérer le transitoire. Nous nous plaçons donc dans le cadre de décision à long ou moyen terme (niveau dit stratégique ou tactique) et ignorons les aspects opérationnels du problème. A ce niveau, l'aide à la décision adresse des questions telles que le dimensionnement de la flotte de véhicules nécessaires ou l'évaluation de la prise en charge de clients supplémentaires. De plus, une solution tactique nourrit le niveau opérationnel en apportant une vue à long terme. Le mode "normal" de fonctionnement défini par le niveau tactique prescrit non seulement les fréquences optimales de visites des clients mais aussi les regroupements de clients basés sur des raisons géographiques et/ou de compatibilité de fréquence. Nous voulons que cette solution d'équilibre se répète au cours du temps.

**Définition 1.1.4** ***Politique périodique et cycle de régénération***
*Une route est dite $p-$périodique (ou de périodicité $p$) si elle est effectuée toutes les $p$ périodes, $p$ étant choisie parmi un ensemble $P \subseteq \{1, \cdots, Pmax\}$, (la donnée $Pmax$ dénote la plus grande périodicité).*
*Un planning est dit périodique s'il se compose de routes périodiques.*
*Le cycle de régénération du planning est l'intervalle de temps de longueur minimal au bout duquel la solution se répète à l'identique. La longueur du cycle de régénération est notée $H$. C'est le plus petit commun multiple des périodicités $p$ des routes du planning.*
*Un planning individuel est $h$ périodique, où $h$ est le plus petit commun multiple des périodicités des routes visitant ce client, $h \leq H$.*

**Remarque:** La périodicité $p$ d'une route ne peut pas être inférieure à la date de sa première utilisation.

Dans l'exemple 1.1, prenons une route qui visite les clients $i2$ et $i6$ toutes les 6 périodes, la première occurrence de cette route étant à la date $t1$. Le client $i2$ est visité par deux autres routes, chacune de périodicité 6 et commençant

respectivement à la date $t3$ et à la date $t5$. Par conséquent, les plannings individuels des clients $i2$ et $i6$ sont de période 6. Si le client $i4$ est visité par une seule route de périodicité 4, son planning est alors $4-$périodique. Par contre, le planning du client 5 est $12-$périodique : il se compose de trois collectes de quantités différentes et donc appartenant à trois routes différentes, chacune ayant une périodicité de 12.

Pour garantir une solution périodique, trois scénarios de jeux d'hypothèses sont considérés : ils sont notés A, B et C et sont présentés dans la table 1.3.

A : Solution opérationnelle

- **(H1)** Le cycle de régénération $H$ est borné par $T$, ce qui équivaut à travailler sur un horizon de temps $T$ fini.
- **(H2)** Le stock initial est connu, le stock final doit être au plus égal au stock initial.

B : Solution sur un horizon restreint

- **(H1)** Le cycle de régénération $H$ est borné par $T$, ce qui équivaut à travailler sur un horizon de temps $T$ fini.
- **(H2)** Le stock initial n'est pas fixé mais doit être égal au stock final.

C : Solution sur un ensemble de périodicités restreint

- **(H1)** Le cycle de régénération $H$ est libre mais l'ensemble des périodicités permises pour les routes est fixé à priori, $P \subseteq \{1, \cdots, Pmax\}$ où $Pmax$ est la plus grande périodicité. La définition de $P$ induit une borne sur $H$ : soit $T = ppcm\{p : p \in P\}$ alors $H \leq T$. Bien que l'horizon de temps soit théoriquement infini, on peut donc se contenter d'examiner la solution sur l'horizon fini $T$.
- **(H2)** Le stock initial n'est pas fixé mais doit être égal au stock final.

TAB. 1.3 – Type de solutions recherchées au niveau tactique : 3 scénarios

Les solutions de ces 3 scénarios sont $H-$périodiques, mais le mode de contrôle de $H$ diffère selon l'hypothèse (H1). Pour les scénarios A et B, $H$ est borné par $T$. Pour le scénario C, le cycle de régénération est borné par le plus petit commun multiple des périodicités permises. Notons que la vraie valeur de $H$ n'est connue qu'une fois la solution établie ($H$ est égal au ppcm des périodicités des routes utilisées).

L'ensemble $P$ des périodicités pour les routes permises n'est pas explicitement donné dans les scénarios A et B, mais il comprend $T$ et ses diviseurs. $T$ doit être choisi suffisamment grand, peut être $T \geq 3tmax$, pour permettre plusieurs visites afin d'avoir une vue plus globale. Le planning du scénario A commence dans des conditions proches de celles observées, la situation initiale est connue (H2). La solution obtenue est "opérationnelle" et dépend de la situation initiale fournie par l'utilisateur. Dans le scénario B, la solution a l'avantage de ne plus dépendre de la donnée d'une situation initiale. Cependant elle dépend de $T$, en effet, si $T$ est impair un client $i$ ayant $tmax_i = 2$ ne peut pas être visité toutes les 2 périodes. Le scénario C évite ces effets de bords liés au choix de $T$. Les solutions obtenues dépendent du choix de $P$. Cependant il est plus facile de déterminer $P$ pour ne pas passer à coté d'une bonne solution que de fixer $T$. Par exemple, en prenant $P$ égal à $\{1, 2, \cdots, tmax\}$, chaque route est au plus $tmax$ périodique et la solution n'est pas éclatée en n'ayant que des routes de très grandes périodicités, i.e. de périodicité $p$ bien plus grande que $tmax$. Ensuite chaque client peut être ramassé à sa fréquence optimale.

Le chapitre 4 est consacré à l'étude des modélisations possibles pour le problème de planification individuel dans chacun de ces trois scénarios. Ensuite les inconvénients de chaque scénario est illustré par un exemple avec deux clients et il s'en dégage que le choix du scénario C est préférable.

Enfin, les variantes de notre problème peuvent se distinguer aussi par le choix de l'objectif présenté dans la table 1.4.

**(i)** L'objectif consiste à minimiser le nombre de véhicules. Cet objectif mène à un modèle dont la réalisabilité est assurée et est utile pour dimensionner la flotte de véhicules.

**(ii)** L'objectif consiste à minimiser tous les coûts liés aux tournées. Cet objectif mène à un modèle économique qui reflète les préoccupations des opérateurs souhaitant optimiser les distances parcourues ou les durées des tournées

TAB. 1.4 – Choix de l'objectif

Le premier choix semble bien adapté au niveau tactique. Cependant deux solutions peuvent utiliser le même nombre de véhicules mais avoir des routes de qualité totalement différentes. Par conséquent, choisir une combinaison de ces deux objectifs est préférable. Il consiste à minimiser le nombre maximum

de véhicules utilisés et $\alpha$ fois le coût des routes, $\alpha$ étant un paramètre que l'utilisateur peut faire varier selon ses préférences.

La table 1.5 résume les données du problème et définit les notations utilisées par la suite. Il s'agit de planifier les dates de collecte d'*un produit* auprès de $N$ *clients* sur un *horizon fini ou infini*. Le but est de trouver des tournées de collecte *périodiques* en cherchant à minimiser l'utilisation des véhicules et les coûts liés au temps de parcours entre chaque client et au temps de service. Au niveau tactique, la période de base peut être une agrégation de plusieurs jours si durant cet ensemble de jours aucun client ne nécessite de multiples visites. La période peut être la semaine, l'unité de temps peut être la minute et celle de masse le kilogramme.

- l'horizon $T$ (prescrit dans les scénarios A et B) ou l'ensemble des périodicités $P \subseteq \{1, \cdots, Pmax\}$ (prescrit dans le scénario C) : par exemple $P = \{1, 2, 3, 4, 5, 6\}$, qui détermine $T = ppcm(p \in P)$, dans cet exemple $T = 60$.
- le garage 0 et le vidage 1
- pour chaque client $i = 2, \cdots, N+1$ :
  - son *taux de remplissage* moyen hebdomadaire $d_i$ (en kg/sem)
  - l'*intervalle maximal* entre deux collectes $tmax_i$ (en nombre de semaines), peut être calculé : $tmax_i = \min(\lfloor \frac{C_i}{d_i} \rfloor, Pmax)$ où $C_i$ est la capacité de stockage (en kg)
  - son *temps de service* (i.e. de collecte), $f_i$ (en min)
  - les *temps de parcours* entre le client $i$ et les autres clients ainsi que le garage et le vidage, $c_{ij}$ (en min) avec $i \in \{0, 2, \cdots, N+1\}$ et $j \in \{1, 2, \cdots, N+1\}$,
  - la fenêtre de temps pour le service $[a_i, b_i]$ (dans le cas des routes opérationnelles),
  - pour le scénario A, le stock initial normalisé $tinit_i < tmax_i$ (quantité produite avant la première période).
- la *capacité* $W$ (en kg) des véhicules et dans le cas des routes opérationnelles la limite $C$ en temps (en min).

TAB. 1.5 – Les données de notre problème

La longueur du cycle de régénération $H$ n'est pas une donnée mais est calculée une fois la solution $S$ connue. En notant $P(S)$ l'ensemble des périodicités

utilisées par les routes de la solution $S$, $H = ppcm(p \in P(S))$. On note $V$ une borne supérieure sur le nombre de véhicules nécessaires, $V \geq \lceil \frac{\sum_i C_i}{W} \rceil$. Les solutions ayant donc plus de $V$ véhicules sont écartées.

## 1.2 Formulations mathématiques

Le problème de tournées de véhicules combinées à la gestion de stock peut se formuler de différentes manières, selon les choix du modèle de tournées de la table 1.1 et du scénario de la table 1.3. Pour asseoir la présentation du problème, nous donnons ici une première formulation correspondant au scénario A avec le modèle de route opérationnelle. D'abord définissons le graphe $G = (N^+, A)$ où $N^+ = N \cup \{0, 1\}$ est l'ensemble des sommets, i.e. les clients, le garage et le vidage et $A = \{(i, j), i \in N^+ \backslash \{1\}, j \in N^+ \backslash \{0\}\}$ est l'ensemble des arcs. Les variables de décisions sont celles qui déterminent les plannings individuels dictés par la gestion des stocks et celles qui définissent les routes opérationnelles constituant la solution au problème de tournées :

- pour chaque période $t = 1, \cdots, T$, définissons $y_{it} = 1$ si le client $i \in N$ est visité durant la période $t$, zéro sinon ; $x_{it}$ le nombre de périodes de production de stock qui sont ramassées chez le client $i \in N$ en période $t$ ; et $s_{it}$ la quantité en stock chez le client $i \in N$ en fin de période $t$ correspondant au nombre de périodes passées depuis la dernière collecte.
- pour chaque période $t = 1, \cdots, T$ et chaque véhicule $v = 1, \cdots, V$, définissons $q_{ivt}$ la quantité normalisée en nombre de périodes ramassée chez le client $i \in N$ par le véhicule $v$ en période $t$ ; $r_{ivt} = 1$ si le client $i \in N^+$ est visité par le véhicule $v$ en période $t$, zéro sinon ; $z_{ijvt} = 1$ si l'arc $(i, j) \in A$ est constituant de la route suivie par le véhicule $v$ en période $t$ (i.e. si le client $i$ est visité après $j$), zéro sinon et $L_{ivt}$ est la charge du véhicule $v$ lorsqu'il quitte le client $i \in N^+$ en période $t$.
- définissons par $Vmax$ le nombre de véhicules utilisés dans la solution.

Le problème de tournées de véhicules combinées à la gestion de stock s'écrit comme suit :

$$\min \quad Vmax + \alpha \sum_{vt,(i,j)\in A} (c_{ij} + f_j) z_{ijvt} \tag{1.1}$$

$$x_{it} \leq tmax_i \, y_{it} \qquad \forall i, t = 2, \cdots, T \tag{1.2}$$

$$x_{i1} = (tinit_i + 1) \, y_{i1} \qquad \forall i \tag{1.3}$$

$$s_{it} \leq (tmax_i - 1)(1 - y_{it}) \qquad \forall i, t = 1, \cdots, T-1 \tag{1.4}$$

$$s_{iT} \leq tinit_i \, (1 - y_{iT}) \qquad \forall i \tag{1.5}$$

$$s_{it-1} + 1 = x_{it} + s_{it} \qquad \forall i, t, \tag{1.6}$$

$$\sum_j z_{ijvt} = r_{ivt} \qquad \forall i, v, t \qquad (1.7)$$

$$\sum_j z_{jivt} = r_{ivt} \qquad \forall i, v, t \qquad (1.8)$$

$$r_{0vt} = r_{1vt} \qquad \forall v, t \qquad (1.9)$$

$$L_{ivt} + d_j q_{jvt} \leq L_{jvt} + W(1 - z_{ijvt}) \qquad \forall i, j, v, t \qquad (1.10)$$

$$L_{ivt} \leq W r_{0vt} \qquad \forall i, v, t \qquad (1.11)$$

$$q_{ivt} \leq tmax\ r_{ivt} \qquad \forall i, v, t \qquad (1.12)$$

$$\sum_v r_{ivt} = y_{it} \qquad \forall i, t \qquad (1.13)$$

$$\sum_v q_{ivt} = x_{it} \qquad \forall i, t \qquad (1.14)$$

$$\sum_v r_{0vt} \leq Vmax \qquad \forall t \qquad (1.15)$$

$$y_{it} \in \{0, 1\} \qquad \forall i, t \qquad (1.16)$$

$$x_{it},\ s_{it} \in I\!N \qquad \forall i, t \qquad (1.17)$$

$$q_{ivt} \in I\!N \qquad \forall i, v, t \qquad (1.18)$$

$$r_{ivt},\ z_{ijvt} \in \{0, 1\} \qquad \forall i, v, t \qquad (1.19)$$

$$L_{ivt} \geq 0 \qquad \forall i, v, t \qquad (1.20)$$

$$Vmax \in I\!N. \qquad (1.21)$$

L'objectif (1.1) est de minimiser le nombre de véhicules utilisés et un coût variable lié aux tournées. Trois groupes de contraintes apparaissent :

1. les contraintes (1.2-1.6) de gestion des stocks définissant un planning ;
2. les contraintes (1.7-1.12) définissant les routes ;
3. les contraintes liantes : (1.13-1.14) lient le planning aux routes utilisées par l'intermédiaire des variables des visites et des quantités ramassées.

Les contraintes (1.15) définissent le nombre de véhicules utilisés. Cette structure particulière définie par ces trois groupes de contraintes est exploitée pour reformuler ce problème à l'aide de la méthode de décomposition de Dantzig-Wolfe. Les différentes décompositions possibles sont abordées au chapitre 2.

La formulation définie par les contraintes (1.1-1.21) est facilement adaptable au cas du scénario B de la table 1.3. Les modifications se font au niveau de la gestion des stocks et sont étudiées dans le chapitre 4. En revanche, dans le cas du scénario C de la table 1.3, une route n'est plus définie pour une date $t$ mais avec une périodicité $p \in P$ et une date $s$ de première occurrence sur l'horizon $T$ (i.e la date de départ $s \leq p$). Ainsi l'indice $t$ des variables $q$, $r$ et $z$ est remplacé par les indices $p$ et $s$. Comme chaque route assure un planning partiel pour quelques clients, le problème de gestion de stock consiste à

ordonner ces plannings partiels pour obtenir un planning global (voir le chapitre 4), i.e le stock produit à chaque période $t \in T$ doit être collecté par une route. Les variables de décision concernant le planning de gestion de stock sont $x_{i\ell ps} = 1$ si le client $i \in N$ est collecté pour $\ell \leq \min(tmax_i, p)$ périodes toutes les $p \in P$ périodes, sa première collecte étant à la date $s \leq p$, 0 sinon. La matrice des coefficients associée aux variables $x$ est une matrice $\delta$ indiquant pour chaque planning partiel les demandes couvertes. Lorsque $x_{i\ell ps}$ vaut 1 alors les demandes couvertes sont celles des périodes $t$ telles que $t \in \{s, s+p, \cdots\}$ mais aussi les périodes $t \in \{s-\ell+1, \cdots, s-1, s-\ell+1+p, \cdots, s-1+p, \cdots\}$. La demande de la période $t$ est couverte s'il existe $\tau \in \{0, \cdots, \ell-1\}$ tel que soit $t = kp + s - \tau$ avec $k \in I\!N$ si $t - \tau \geq 0$, soit $t + T = kp + s - \tau$ avec $k \in I\!N$ si $t - \tau < 0$. Comme $T$ est multiple de $p$, cela revient à vérifier si $\frac{t+\tau-s}{p} \in Z\!\!\!Z$, ainsi :

$$\delta_{it}^{\ell ps} = \begin{cases} 1 & \text{si le stock du client } i \text{ produit à la période } t \text{ est collecté} \\ & \text{par le planning partiel associé à la variable } x_{i\ell ps} \\ & \text{i.e. si } \exists \tau \in \{0, \cdots, \ell-1\} \text{ tel que } \frac{t+\tau-s}{p} \in Z\!\!\!Z \\ 0 & \text{sinon} \end{cases}$$

Les contraintes de gestion des stocks (1.2-1.6) deviennent :

$$\sum_{ps\ \ell} \delta_{it}^{\ell ps}\ x_{i\ell ps} = 1 \quad \forall i, t = 1, \cdots, T$$

et les contraintes liantes (1.13-1.14) :

$$\begin{aligned} \sum_{v} r_{ivps} &= \sum_{\ell} x_{i\ell ps} \qquad \forall i, p, s \\ \sum_{v} q_{ivps} &= \sum_{\ell} \ell x_{i\ell ps} \qquad \forall i, p, s. \end{aligned}$$

De même que pour les variables $x$, les variables $r_{0vps}$ indiquent un planning partiel d'utilisation des véhicules et la matrice des coefficients associée est une matrice indiquant les périodes d'utilisation du véhicule de ce planning partiel. Lorsque $r_{0vps}$ vaut 1 alors un véhicule est utilisé aux périodes $t$ tel que $t \in \{s, s+p, \cdots\}$, i.e. si $t$ est de la forme $t = kp + s$ avec $k \in I\!N$, ainsi :

$$\delta_{0t}^{vps} = \begin{cases} 1 & \text{si un véhicule } v \text{ est utilisé à la période } t \text{ par le planning partiel} \\ & \text{associé à la variable } r_{0vps}, \text{ i.e. si } \frac{t+\tau-s}{p} \in I\!N \\ 0 & \text{sinon} \end{cases}$$

Les contraintes (1.15) deviennent :

$$\sum_{vps} \delta_{0t}^{vps} r_{0vps} \leq Vmax \ \ \forall t = 1, \cdots, T.$$

Les contraintes de tournées restent quant à elles inchangées. Dans la fonction objectif (1.1), les coûts liés aux tournées sont multipliés par $\frac{1}{p}$ pour avoir le coût moyen par période de la solution.

La formulation peut aussi changer en fonction du modèle de tournées utilisé. Ainsi, pour avoir un aperçu sur l'ensemble des formulations possibles, nous donnons une deuxième formulation correspondant au scénario C avec le modèle de cluster. Un cluster est défini comme une étoile autour d'un centre-client $i \in N$, le coût d'ouverture du centre $i$ est $c'_i$ et le coût de connexion du client $j$ au centre $i$ est $c'_{ij}$ (ces coûts sont définis en (3.29)). Les variables $z$ ne définissent plus la route utilisée mais le cluster. Pour chaque périodicité $p$, date de départ $s$ et véhicule $v$, définissons $z_{iivps} = 1$ si le client $i \in N$ est un centre, zéro sinon ; et $z_{ijvps} = 1$ si le client $j \in N$ est connecté au centre $i$, zéro sinon. La formulation du problème devient :

$$\min \quad Vmax + \alpha \sum_{vps} \frac{1}{p} \sum_{i,j} (c'_i z_{iivps} + c_{ij} z'_{ijvps}) \tag{1.22}$$

$$\sum_{ps\,\ell} \delta_{it}^{\ell\, ps}\, x_{i\ell ps} = 1 \qquad \forall i, t = 1, \cdots, T \tag{1.23}$$

$$\sum_{i} z_{iivps} = r_{0vps} \qquad \forall v, p, s \tag{1.24}$$

$$z_{ijvps} \leq z_{iivps} \qquad \forall i, j, v, p, s \tag{1.25}$$

$$\sum_{j} z_{jivps} = r_{ivps} \qquad \forall i, v, p, s \tag{1.26}$$

$$\sum_{i} d_i q_{ivps} \leq W r_{0vt} \qquad \forall v, p, s \tag{1.27}$$

$$q_{ivps} \leq tmax\ r_{ivps} \qquad \forall i, v, p, s \tag{1.28}$$

$$\sum_{v} r_{ivps} = \sum_{\ell} x_{i\ell ps} \qquad \forall i, p, s \tag{1.29}$$

$$\sum_{v} q_{ivps} = \sum_{\ell} \ell x_{i\ell ps} \qquad \forall i, p, s \tag{1.30}$$

$$\sum_{vps} \delta_{0t}^{vps} r_{0vps} \leq Vmax \qquad \forall t = 1, \cdots, T \tag{1.31}$$

$$x_{i\ell ps} \in \{0, 1\} \qquad \forall i, \ell, p, s \tag{1.32}$$

$$q_{ivps} \in I\!N \qquad \forall i, v, p, s \tag{1.33}$$

$$r_{ivps},\ z_{ijvps} \in \{0, 1\} \qquad \forall i, v, p, s \tag{1.34}$$

$$Vmax \in I\!N. \tag{1.35}$$

Dans ces deux formulations du problème de tournées de véhicules combinées à la gestion de stock, des jeux distincts de variables pour la gestion de stock et les tournées sont maintenus afin de faire apparaître la décomposition mais

ces deux formulations pourraient être formulées d'un manière plus compacte. La forme compacte de la formulation définie par les contraintes (1.1-1.21) est obtenue en utilisant les contraintes de lien (1.13) et (1.14) pour éliminer les variables $y$ et $x$. Pour la formulation définie par les contraintes (1.22-1.35), il faut utiliser la variable $x_{i\ell vps} = 1$ si le client $i \in N$ est collecté par le véhicule $v$ pour $\ell \leq \min(tmax_i, p)$ périodes toutes les $p \in P$ périodes, sa première collecte étant à la date $s \leq p$, 0 sinon, alors $r_{ivps} = \sum_\ell x_{i\ell vps}$, $q_{ivps} = \sum_\ell \ell x_{i\ell vps}$ et $x_{i\ell ps} = \sum_v x_{i\ell vps}$.

Les différentes modélisations sont étudiées aux chapitres 3 à 5, la section suivante propose une classification des problèmes IRP de la littérature avec un aperçu des méthodes utilisées pour les résoudre.

## 1.3 Revue de la littérature sur l' "Inventory Routing Problem"

Ce problème apparaît dans la littérature dans de nombreuses variantes et applications : le tableau 1.6 classifie les différentes variantes rencontrées. Plusieurs classifications sont possibles, tout dépend du premier critère de sélection (voir Federgruen et Simchi-Levi [39] pour une première étude sur l'IRP).

### 1.3.1 Classification

Notre premier critère de classification concerne le *nombre de types de produits* transportés. La plupart des problèmes de type VRP considèrent le transport d'un seul type de produit. Cependant certains articles différencient les produits. Ces derniers se classent en deux sous-parties, selon que les produits peuvent être mis ensemble dans les véhicules, ou qu'un véhicule comporte plusieurs compartiments, chacun étant destiné à un seul type de produit.

Notre deuxième critère est le *type d'horizon* sur lequel le problème est traité. Deux types d'horizons existent : les horizons finis et les horizons infinis. Dans le premier cas, la solution est construite sur $T$ périodes, $T$ étant fini et connu. Dans le deuxième, la solution est définie en considérant le long terme.

Les horizons finis sont eux mêmes séparés en deux catégories : les mono et les multi périodiques. Les mono-périodiques simplifient le problème en n'opti-

misant que sur une période (tournées quotidiennes). Pour chaque nouvelle période, le processus d'optimisation est réitéré. Cette optimisation hiérarchique permet d'obtenir des solutions opérationnelles mais sous optimales, elle manque de vue à long terme. Par contre, des informations plus précises sur les données pour les jours suivants peuvent être apportées, soit d'après les valeurs réelles récupérées lors des tournées quotidiennes [53], soit en ayant les informations directement chaque matin [20] ; ce qui permet d'ajuster les tournées. Pour avoir une vue sur le plus long terme, il faut utiliser un horizon multipériodiques, durant lequel un client peut ne nécessiter qu'une seule visite [86]. Deux approches sont proposées. La première consiste à travailler sur un horizon glissant : un planning est élaboré sur $T$ périodes, mais seulement les routes des $t < T$ premières périodes sont implémentées. Après les livraisons des $t$ périodes, un ajustement est réalisé au vu des dernières informations reçues (stock réel constaté lors des tournées, ...) et un nouveau planning est construit pour les $T$ périodes suivantes (de $t$ à $t + T$). La deuxième consiste à travailler sur $T$ périodes directement. Dans ce cas, soit les routes sont planifiées sur tout l'horizon connaissant la situation initiale [12, 22], soit elles sont périodiques [50, 3].

L'horizon infini sert quant à lui surtout au niveau tactique où on cherche à minimiser les coûts moyens sur le long terme. Dans ce cas, la recherche d'une solution peut s'avérer très difficile et la solution obtenue peut devenir vite impraticable (il peut n'y avoir pratiquement aucune répétition dans les routes à effectuer, comme le montre l'exemple 1.2). Pour remédier à ces difficultés,

Exemple 1.2 – La solution sur un horizon infini peut être impraticable

Soit un véhicule de capacité non restrictive et 12 clients $i$ tel que $tmax_i = i$. Une solution est de visiter chaque client $i$ toutes les $tmax_i$ périodes. Le chauffeur visite les 12 clients à la période 1. Il refera cette route à la période $27720 = ppcm\{1, \cdots, 12\}$.

l'espace des solutions est habituellement restreint par différentes stratégies de réapprovisionnement, par exemple :
$(i)$ le réapprovisionnement direct ("direct shipping"). Une tournée ne livre qu'un seul client.
$(ii)$ le réapprovisionnement quand le stock du client devient nul ("zero inventory policy"). Il peut être très difficile de trouver la structure optimale des solutions, ainsi une autre politique lui est préférée : la politique de partition fixe [14]. Elle consiste à partitionner l'ensemble des clients en sous ensembles disjoints (clusters) ; chaque sous ensemble est servi séparément et indépendam-

ment des autres et lorsqu'un client d'un sous ensemble est livré, tous les autres le sont.
(*iii*) le réapprovisionnement suivant la politique de puissance de 2 [28]. Les intervalles de réapprovisionnement sont des multiples d'une puissance de deux.

Enfin, le dernier critère de classification est le *taux de consommation* : déterministe versus stochastique. Le cas stochastique est plus proche de la réalité : le risque que le client soit en rupture de stock existe. Si le niveau de stock réel est connu, alors une livraison urgente avec un coût plus élevé est faite pour pallier au manque de stock ; cela peut être traité seulement au niveau opérationnel. Pour prendre en compte les aléas au niveau de la planification tactique, une probabilité de rupture de stock est définie. Cette probabilité permet soit d'ajouter un terme d'erreur au taux moyen [53], soit de calculer les quantités moyennes des demandes en rupture de stock et les pénalités associées [76] (le niveau tactique est possible car ces quantités peuvent être calculées pour toute longueur d'intervalle).

Pour simplifier le problème ou lorsque les données sont peu fluctuantes, on préfère un taux déterministe. Le taux stochastique peut être ramené à un taux déterministe moyennant des seuils de sécurité : pour chaque client $i$, un intervalle maximum entre deux visites est défini tel que la probabilité d'être en rupture de stock n'excède pas la probabilité donnée $p_i$ [85] ; un stock de sûreté est prévu [28] ou un espace tampon dans le véhicule [50]. La gestion des aléas peut se faire à l'aide d'une interface avec l'utilisateur pour des modifications manuelles possibles au vu des vraies données (dans [9, 86] des heuristiques sont développées pour la réoptimisation quotidienne afin de faire face à l'apparition de demandes supplémentaires).

Un dernier cas particulier consiste à supposer qu'à tout moment on peut avoir accès aux niveaux de stock exact (à l'aide de capteur ou par appel). Ce scénario a son intérêt au niveau opérationnel, avant chaque planification les données sont mises à jour.

Dans le tableau de classification 1.6, certaines variations dans la modélisation ne sont pas prises en compte. Par exemple dans [37, 12, 85], les auteurs ont fait l'hypothèse qu'à chaque livraison le niveau de stock revient à son maximum, c'est la politique "order up to level". Les quantités livrées sont donc dictées par le planning et ne sont pas des variables de décision.

D'autre part la fonction objectif diffère selon le statut du transporteur, le plus fréquent étant la minimisation des coûts liés aux tournées. S'il est aussi propriétaire des magasins (ou du dépôt), il minimise en plus les coûts de stockage. Si son revenu dépend des livraisons effectuées, alors il maximise son profit [9, 20]. Le nombre de véhicules disponibles peut être connu et fixé ou illimité [85, 5]. Dans ce dernier cas, alors l'utilisation moyenne des véhicules peut être minimisée [85].

Enfin, les ordres de grandeur des données varient en fonction du type d'application. Citons notamment le nombre de clients qui peut passer de 15 dans la distribution d'ammoniac par transport maritime [22] à 3000 dans la distribution de propane liquide par route [53]. Ensuite des différences existent au niveau du véhicule, par exemple dans le nombre de clients visités : dans [50] 2 supermarchés sont livrés, dans [9] 4 stations essences et dans notre application 10 bornes sont collectées. Un même véhicule peut être utilisé pour une seule tournée ou plusieurs de suite en retournant à chaque fois au dépôt [53, 3]. Sa tournée dure quelques heures [86] ou plusieurs jours dans le cas du transport maritime [22]. Les différences au niveau des caractéristiques du client se font sur son taux de consommation, habituellement exprimé en jour mais peut l'être en heure [50, 3] ; et sur sa capacité de stockage, limitée ou non [14, 3].

### 1.3.2 Problèmes à produit unique

Le cas d'un seul produit est le plus fréquent. Même si les applications comportent plusieurs produits, souvent on peut n'en considérer qu'un seul car soit les différents produits peuvent être agrégés en un seul avec un taux moyen de consommation commun [50], soit ils sont pratiquement indépendants [15, 86] (transportés dans des véhicules différents et un client est égal à un produit) ou indifférentiables [9] (transportés dans un même véhicule et un client est égal à un produit). Des méthodes sont développées pour les quatres types d'horizon. On commence par l'horizon le plus petit (mono-périodique) pour aller vers le plus grand (infini).

Golden et al. (1984, [53]) développent une heuristique pour optimiser un système de planification intégrant les livraisons pour une grande compagnie d'énergie qui distribue du propane liquide à 3000 clients. La compagnie propose deux systèmes de planification/livraison pour chaque district : le premier (système route) réapprovisionne les clients tous les 60 jours, le deuxième (système de demande de livraison) prévoit la prochaine livraison en fonction de

- **1 produit** :
  - **horizon mono-périodique** :
    - **demandes déterministes** : [20] ;
    - **demandes stochastiques** : [53] (propane liquide) ;
  - **horizon glissant** :
    - **demandes en ligne** : [9] (T= 2 à 5 jours, t=1 jour, gaz : oxygène, nitrogène...) ;
    - **demandes déterministes** : [27] (T=2 périodes, t=1 période), [86] (T=1 semaine, t=1 jour, distribution de gaz), [15, 16] (T= 1 mois, t=2 jours, industrie de gaz) ;
    - **demandes stochastiques** : [37] (T=1 an, t= 1 semaine), [58] (T=2 semaines, t= 1 semaine) ;
  - **horizon multi périodiques** :
    - **demandes déterministes** : [50] (T= 1 semaine, chaîne de supermarché, solution périodique), [3] (T=1 an, réapprovisionnement cyclique), [12], [22, 21] (transport maritime de l'ammoniac) ;
  - **horizon infini** :
    - **demandes déterministes** : [5], [14], [19], [85] ;
- **plusieurs produits,** transport dans le **même compartiment** :
  - **horizon infini** :
    - **demandes déterministes** : [28] (produits congelés) ;
    - **demandes stochastiques** : [76] ;
- **plusieurs produits,** transport dans **différents compartiments** :
  - **horizon mono-périodique** :
    - **demandes déterministes** : [79], [6], [64] (stations essences) ;

TAB. 1.6 – Classification des différents modèles pour l' "inventory routing problem"

l'historique (dernière livraison et estimation du taux de consommation). Pratiquement, beaucoup de districts préfèrent le deuxième système. Les auteurs ont amélioré le système de demande de livraison : les coûts des tournées ont été minimisés avec une meilleure performance (moins de rupture de stock nécessitant un service urgent et plus de productivité). Dans la prévision des clients à visiter ce jour et pour estimer le coût des tournées, ils utilisent la solution d'un TSP avec collecte de prix (le profit associé à un client dépend de la capacité de ce client, de son niveau de stock et de son taux de consommation). Ensuite les routes sont déterminées à l'aide d'un VRP résolu (par l'algorithme heuristique de Clarke Wright [26]) sur cet ensemble de clients. Puis comme un véhicule

peut faire plusieurs voyages, les routes obtenues sont assignées aux camions en résolvant un problème d'empaquetage dans des boîtes ("bin packing") par l'heuristique "first fit decreasing".

Dans Chien et al. (1989, [20]) la quantité de produit en stock au dépôt est limitée. Au début de chaque jour, le dépôt reçoit les informations sur les stocks clients, son but est de déterminer les livraisons du jour pour maximiser son profit total (le revenu obtenu par unité livrée moins les coûts des tournées et de rupture de stock). Ils formulent le problème comme un problème entier mixte et développent une approche de résolution basée sur la relaxation Lagrangienne. Le problème résultant se décompose en deux sous-problèmes : un problème d'allocation du stock du dépôt aux clients et un problème d'affectation des clients à l'utilisation d'un véhicule. Les deux sous-problèmes sont des problèmes de sacs-à-dos continus et peuvent être résolus relativement efficacement. Une solution réalisable est obtenue par une heuristique basée sur la solution Lagrangienne.

Bell et al. (1983, [9]) décrivent le développement d'un système d'aide à la décision à Air Products and Chemical INC. Après la collecte des données (prévision de l'usage des clients) et la détermination d'un ensemble de routes ayant au maximum 4 clients, un module d'optimisation sélectionne les routes de livraison sur un horizon de 2 à 5 jours. Dans ce module, ils formulent ce problème à l'aide d'un programme entier mixte qui va fixer le volume des livraisons, l'affectation des véhicules aux routes et des dates aux routes et chercher à maximiser le coût lié à la quantité livrée moins le coût des tournées. Un algorithme de relaxation Lagrangienne résout ce problème par décomposition en programmes plus simples (un problème de sac-à-dos pour chaque véhicule). Le système réalise des bénéfices par rapport aux coûts opérés des solutions mises en place par l'industriel.

Witucki et al. (1997, [86]) proposent un système de deux modules pour aider un distributeur de gaz industriel à livrer et gérer le stock client, ils peuvent se limiter à 60 clients. Le premier module optimise la planification des routes sur un horizon d'une semaine en cherchant à minimiser les coûts des tournées. Ils résolvent le problème par génération de colonnes, chaque client nécessite une seule visite durant la semaine, celle ci devant se faire entre 2 dates données. Le deuxième module est un module de post-optimisation journalière : connaissant les quantités réelles livrées le jour précédent, le planning est réoptimisé par génération de colonnes sur l'horizon décalé d'un jour. Cette réoptimisation

est plus rapide que la première optimisation car la génération de colonnes est initialisée avec la première solution.

Cousineau-Ouimet (2002, [27]) a développé une recherche tabou pour le problème de livraison de gaz : son voisinage est l'échange de clients entre les routes et l'échange de jours ou combinaisons de jours de visite ; les dépassements de la capacité du véhicule et du temps du trajet sont permis mais pénalisés. Au préalable l'auteur génère les combinaisons possibles de visites sur deux périodes puis résout seulement la première période et recommence.

Le travail de Campbell et al. (1999, [15] ; 2004, [16]) est lui aussi motivé par une application d'une industrie qui livre du gaz sous forme liquide. Ils construisent un planning sur un mois mais les routes sont seulement définies pour les quelques premiers jours. Ils utilisent une approche en deux phases. Dans la première, un programme entier détermine pour chaque période des ensembles de clients qui sont visités ("route") avec les volumes de livraison pour chacun d'eux tout en minimisant les coûts des tournées. Pour accélérer la résolution, ils agrègent les jours en semaine sur la fin de l'horizon et considèrent un sous-ensemble de routes pré-générées. La réduction du nombre de routes est basée sur le fait que des clients ne peuvent être dans une même route que s'ils appartiennent à un même cluster. Ainsi un bon ensemble de clusters disjoints devra être identifié pour couvrir tous les clients. Le coût d'un cluster est une estimation des coûts de distribution pour servir ses clients pendant un mois, il dépend de la position géographique des clients et de la compatibilité de leur capacité de stockage et de leur taux de consommation. Un grand ensemble de clusters est généré et un problème de partition est résolu pour les sélectionner. Dans la deuxième phase, une heuristique d'insertion construit les routes (ordre des clients avec fenêtre de temps, quantités livrées) pour les deux premiers jours, la solution de la phase 1 leur servant de suggestion. Pour se comparer aux solutions industrielles, ils implémentent une approche basée sur les règles les plus utilisées en pratique : créer des voyages autour des clients qui reçoivent une livraison le jour considéré ; tous les clients d'un voyage sont remplis à pleine capacité, sauf le dernier ; le véhicule est totalement vidé. Ils améliorent l'existant.

Dror et Ball (1987, [37]) présentent une procédure pour convertir le problème long terme (un an) en un problème court terme ($m$ jours) basée sur la définition de coûts reflétant l'impact des décisions du court terme sur le long terme. Les taux de consommation sont stochastiques. Pour chaque client, le

jour optimal de livraison $t^*$ est calculé en utilisant la probabilité qu'un client soit en rupture de stock un jour spécifié du planning et les coûts de livraison (ces coûts comprennent une estimation du coût moyen de livraison et le coût d'une livraison spéciale en cas de rupture de stock). Si $t^*$ tombe à l'intérieur du planning court terme de $m$ jours, le client est visité et une valeur $c_t$ est calculée pour chaque jour $t$ du planning court terme qui reflète l'augmentation attendue dans le coût si la livraison est faite le jour $t$ plutôt que le jour $t^*$ (une pénalisation). Un programme entier est alors résolu pour assigner ces clients à un véhicule et à un jour (ou seulement à un jour, et ensuite un VRP sur chaque jour est résolu). Pour quelques autres clients, un coût incitant à les réapprovisionner finalement durant ces $m$ jours est calculé. Le programme entier minimise la somme de ces coûts plus les coûts des tournées (basé sur le problème d'affectation généralisée de Fisher et Jaikumar pour le VRP [44]). Ainsi, les éléments stochastiques sont capturés dans les coûts et $t^*$, le programme entier est quant à lui déterministe. Comme dans [37], la méthode de Bard, Huang, Jaillet et Dror (1998) identifie en premier les clients devant être visités durant un horizon de deux semaines, ensuite un ajustement est fait pour équilibrer la demande quotidienne et accommoder les jours de la semaine. Les routes sont élaborées pour les clients planifiés pour la première semaine seulement. L'horizon des deux semaines est ensuite décalé d'une semaine et le processus est répété. Les principales justifications derrière l'identification du processus (la sélection des clients qui doivent être routés un jour spécifique de la semaine) sont contenues dans le papier [58]. Cette sélection est basée sur la distribution de la demande et sur la fréquence optimale.

Gaur et Fisher (2004, [50]) améliorent le système de planification d'un dépôt central d'une chaîne de supermarché qui tous les 3 mois détermine les dates de livraison de ses magasins affectés et les routes utilisées. L'optimisation est faite sur un horizon $T = 1$ semaine pour obtenir une solution qui est répétée, définissant ainsi l'IRP périodique. Ils considèrent la politique de partition fixe, dans leur cas un cluster peut être servi par deux types de routes : la route livrant tous les magasins du cluster ou les routes livrant un seul magasin (livraison directe). Le coût d'un cluster est obtenu en résolvant un plus court chemin dans un graphe où les nœuds correspondent aux périodes. Un arc entre deux périodes représente une livraison à la première période couvrant les demandes jusqu'à la deuxième, son coût est obtenu à l'aide d'un programme entier. Un problème de partitionnement est résolu pour choisir les clusters tels que le coût des tournées sur les T périodes soit minimum. Dans le cas particulier où un cluster comporte au maximum deux magasins (assez fréquent), le

problème de partitionnement devient un problème de matching généralisé de poids minimum où chaque sommet correspond à un seul magasin et chaque arc correspond à un cluster de deux magasins. Lorsqu'il y a plus de deux magasins par cluster, la résolution est heuristique. Il est facile d'ajouter des contraintes horaires (fenêtres de temps), par contre les contraintes sur la taille de la flotte hétérogène sont ignorées, mais elles sont prises en compte dans l'étape d'affectation des véhicules aux routes pour chaque jour. Comme la taille de la flotte est contraignante pour certains types de véhicules, le module IRP et le module d'affectation des véhicules sont associés dans une heuristique basée sur la relaxation Lagrangienne. Enfin une heuristique tente d'équilibrer la charge de travail (chargement, déchargement) du dépôt central.

Bertrazzi et al. (2002, [12]) analysent et comparent des variantes obtenues avec différentes fonctions objectives qui correspondent à divers points de vue : minimisation des coûts des tournées et/ou des coûts de stock chez les détaillants et/ou des coûts de stock chez le fournisseur. Ils comparent ces coûts sur le problème simplifié avec un seul produit et un seul véhicule. Leur solution est non périodique : le niveau de stock en fin d'horizon peut être différent du niveau de début. Ils utilisent une heuristique en deux étapes. La première construit une solution réalisable en insérant les détaillants un à un dans la solution partielle : le meilleur planning du prochain détaillant est calculé avec un algorithme de plus court chemin dans un graphe acyclique (le coût des arcs est basé sur la solution partielle) puis le détaillant est inséré dans la route pour chaque période de livraison. La seconde étape améliore la solution : les jours de livraison sont changés pour quelques détaillants.

Aghezzaf et al. (2006, [3]) se sont intéressés au cas où une tournée est composée de plusieurs tours. Sur tout l'horizon, chaque véhicule $v$ fait la même tournée qui est répétée toute les $T_v$ heures. Un client est affecté à un seul véhicule. Ils cherchent la longueur du cycle optimal $T_v$ pour chaque véhicule qui minimise les coûts des tournées et de stockage par heure. $T_v$ est borné inférieurement par $Tmin_v$, la longueur de la tournée ; et supérieurement par $Tmax_v$, la plus petite des durées maximales entre deux visites pour chaque tour qui dépendent de la capacité du véhicule. Ce problème est formulé à l'aide d'un programme non linéaire (dû à la variable $T_v$) entier mixte. Ils proposent un algorithme approximatif basé sur le processus de génération de colonnes : leur colonne représente une tournée, le problème maître affecte chaque client à une tournée. En fixant $T_v$ (sa valeur devient un paramètre), leur sous-problème est linéaire et résolu à l'aide d'une heuristique inspirée par celui de Clarke Wright.

Lorsqu'aucune colonne de coût réduit négatif n'est trouvée, ils résolvent le problème maître en nombre entier sur l'ensemble de toutes les colonnes générées. Ensuite pour chaque véhicule, connaissant $Tmin_v$ et $Tmax_v$, ils ajustent la longueur du cycle trouvé en fonction de la longueur optimale trouvée par la méthode EOQ. Leurs expériences montrent que les solutions obtenues en permettant aux véhicules de faire plusieurs tours sont meilleures qu' avec un seul tour.

Christiansen et Nygreen (1998, [22]) et Christiansen (1999, [21]) présentent un problème réel de planification pour les navires (circulant entre 15 ports, sur 1 mois), qui est une combinaison d'une variante du problème multi-véhicules avec chargement-déchargement et fenêtres de temps ("multi-vehicle pick-up and delivery problem with time windows") avec un modèle de gestion des stocks. Un seul produit est pris en compte : l'ammoniac. Il existe différents types de port : les ports externes (les quantités chargées et déchargées se font en fonction des négociations) et les ports internes (les ports producteurs et consommateurs) devant garder un niveau de stock entre deux limites. Seuls les coûts des tournées sont à minimiser. Dans le modèle mathématique proposé, ils dupliquent certaines variables et utilisent une décomposition de Dantzig-Wolfe : un sous-problème de routage avec horaire pour chaque navire (un plus court chemin élémentaire avec contraintes de ressources et fenêtre de temps) et un sous-problème de gestion de stock pour chaque port (un plus court chemin sur un graphe acyclique prenant en compte les fenêtres de temps et les quantités). Le problème est résolu par Branch-and-Price. Notons que certains éléments sont caractéristiques des problèmes maritimes : il n'y a pas de dépôt central, initialement et à la fin de l'horizon un navire peut se trouver n'importe où. De plus, les données déterminent le nombre d'arrivées possibles à chaque port avec les fenêtres de temps et un intervalle pour la quantité chargée (ou déchargée). Cette quantité est optimisée et en général représente une grande partie de la capacité du navire.

Anily et Federgruen (1990, [5]) considèrent un système avec un dépôt et $N$ magasins. Chaque magasin $j$ a une demande déterministe égale à $k_j$ fois un taux de base $\mu$, ainsi sans perte de généralité ils considèrent $k_j$ copies du magasin ayant un taux $\mu$. Ils font l'hypothèse de la politique de partition fixe à la différence qu'un magasin peut appartenir jusqu'à $k_j$ clusters : une fraction spécifique de sa demande est allouée à chacun des clusters. Le nombre de véhicules n'est pas limité, leur capacité est $W$. Connaissant une borne supérieure $f^*$ pour la fréquence des livraisons, ils déterminent le nombre de magasins $m^* = [\frac{f^*W}{\mu}]$

par cluster. Les magasins sont partitionnés géographiquement en $k_1$ clusters avec $m^*$ magasins et un cluster avec $k_2$ magasins (les plus proches du dépôt), $k_1$ et $k_2$ vérifiant $N = k_1 m^* + k_2$. Pour chaque cluster $S$, ils calculent $Q_S$ la quantité optimale de livraison qui est le minimum entre la quantité déterminée par le modèle EOQ et la capacité du véhicule, puis la fréquence optimale de livraison $f_S = \frac{m_S \mu}{Q_S}$ avec $m_S$ le nombre de clients dans le cluster. Ils minimisent les coûts moyens de stock et des tournées : le coût d'une tournée sur le cluster est le coût d'un TSP. Pour évaluer leur stratégie, ils calculent une borne inférieure sur le coût moyen. Cette borne est basée sur les distances des clients au dépôt et est asymptotiquement tendue (lorsque le nombre de magasins croît) par rapport à leur borne supérieure.

Bramel et Simchi-Levi (1995, [14]) présentent une heuristique basée sur le problème de localisation pour résoudre des problèmes de tournées qu'ils appliquent à l'IRP. Ils considèrent aussi la politique de partition fixe. Pour chaque cluster $S$, la durée du cycle optimal $t(S)$ est déterminée en se basant sur le modèle EOQ et sur la capacité des véhicules (la capacité de stockage est illimitée), et le coût par unité de temps $\phi(S)$ comprend les coûts d'un TSP sur $S$, de commande et de stockage. Une première phase définit $m$ centres $k_j$, un centre pouvant être formé de plusieurs clients, et calcule le coût de choisir un centre $k_j$, $v_j = \phi(k_j)$, et les coûts de connexion des clients $i$ à ce centre $k_j$, $c_{ij} = \phi(k_j \cup \{i\}) - \phi(k_j)$. Une seconde phase détermine la partition en résolvant un problème de localisation de dépôts. Une borne inférieure est calculée pour s'assurer de la qualité de leurs solutions. Cette borne est égale à la somme des coûts pour servir chaque client $i$. Le coût minimum pour servir un client est calculé par programme dynamique et le coût lié aux tournées consiste à une fraction du meilleur tour visitant ce client.

Chan et al. (1998, [19]) caractérisent l'efficacité asymptotique des classes des politiques de partition fixe et de commande à stock zéro. Elles sont liées à la constante de packing du problème de bin packing associé où les clients doivent être placés dans des boîtes de taille unitaire, la taille de chaque client est proportionnelle à leur taux de consommation relatif. Ils fournissent une borne inférieure sur le coût d'une politique réalisable et une borne supérieure sur le coût de la politique de partition fixe. Sous diverses hypothèses probabilistes (par exemple, supposons que les taux de consommation des clients sont comme dans [5], les coordonnées géographiques de ces clients et leur multiplicité sont distribués identiquement et indépendamment), ils analysent la déviation à l'optimum entre ces deux bornes.

Webb et Larson (1995, [85]) résolvent l'IRP stratégique sous l'hypothèse de la politique d' "order up to level" et cherchent à minimiser l'utilisation moyenne des véhicules pour estimer la taille de la flotte utile. Une route est initiée seulement si nécessaire (quand le stock d'un client devient nul ou quand la capacité du véhicule ne suffira plus pour livrer les demandes cumulées). Motivés par l'inefficacité dans certains cas du système de Larson (1988) qui considérait la politique de partition fixe ; ils vont chercher pour chaque cluster un ensemble de routes répété périodiquement pour réapprovisionner le cluster. Cet ensemble est défini par $|\mathcal{R}|$ routes ordonnées chronologiquement ($|\mathcal{R}|$ est tel que chaque client est réapprovisionné au moins une fois dans chaque répétition de l'ensemble et peut l'être toutes les $tmax$ périodes), et pour chaque client $i$ par la période $\tau_i$ (nombre de routes dans l'ensemble séparant deux livraisons) et la phase $\phi_i$ (occurrence de la première livraison). L'utilisation du véhicule est déterminée par le rapport entre la somme des durées des routes et la somme des intervalles de temps entre les dates d'utilisation des routes. Pour chaque cluster, ils calculent le nombre de routes $|\mathcal{R}|$ et proposent une procédure énumérative pour trouver les combinaisons $(i, \tau_i, \phi_i)$ minimisant l'utilisation moyenne des véhicules. Ils présentent un algorithme basé sur le concept de gain de Clarke Wright pour définir les clusters.

### 1.3.3 Problèmes à plusieurs produits

Les papiers traitant du problème de tournées dans différents compartiments se rapportent tous au problème de réapprovisionnement d'essence. Une des particularités est qu'un compartiment réapprovisionne une seule station, ainsi le nombre de stations maximum qu'un véhicule visite est assez petit (en général 4). Ils considèrent un horizon mono-périodique et leur objectif est de minimiser les coûts des tournées.

Taqa allah et al. (2000, [79]) proposent une formulation mathématique. Au vu du nombre de variables et contraintes pour les données de taille réelle, ils choisissent de résoudre le problème à l'aide d'heuristiques de type glouton. Ils comparent deux politiques d'approvisionnement : dans la première un véhicule ne réapprovisionne qu'une seule station par voyage, les tournées sont simplifiées, tandis que dans la deuxième il peut réapprovisionner plusieurs stations par voyage. Sur des instances aléatoires, la seconde politique conduit à une réduction des coûts de transport. Malépart et al. (2002, [64]) développent aussi des heuristiques constructifs. Ils considèrent deux types de stations : celles qui

passent une commande (les indépendantes) et celles dont le stock est géré par le transporteur (approvisionnement automatique). Dans Avella et al. (2004, [6]), chaque client place une commande avec une fréquence de un ou plusieurs jours et chaque commande doit être satisfaite le jour suivant. Pour un jour donné, ils résolvent le problème à l'aide d'un algorithme de Branch-and-Price. Une heuristique de packing/routing est développée pour résoudre les grandes instances rapidement et fournir un ensemble de colonnes initiales.

Les auteurs considérant plusieurs produits pouvant être transportés dans un seul compartiment travaillent sur un horizon infini et leur stratégie de réapprovisionnement détermine un cycle de régénération. Ils minimisent le coût moyen lié aux tournées et les coûts de stockage moyen sur le long terme qui correspond au coût sur le cycle de régénération divisé par la longueur du cycle.

Custódio et Oliveira (2001, [28]) considèrent le problème de la distribution de produits congelés et se restreignent à la politique de puissance de deux (la fréquence de réapprovisionnement est un multiple d'une puissance de 2). Ils développent une heuristique basée sur celle de Viswanathan et Mathur (1997) : la fréquence est déterminée en même temps que les routes pour les produits fréquents, les autres sont rajoutés par la suite. Dans Qu et al. (1999, [76]) c'est l'entrepôt qui va gérer son stock en se réapprovisionnant auprès de plusieurs fournisseurs. Leur stratégie est la politique de gestion de stock périodique $(R_i, T_i)$, c'est à dire que l'article $i$ est réapprovisionné jusqu'au niveau $R_i$ toutes les $T_i$ périodes, $T_i$ étant un multiple d'une périodicité de base à optimiser. Ils proposent une méthode de décomposition heuristique. Le problème maître de gestion de stock détermine $(R_i, T_i)$ indépendamment pour chaque article en prenant en compte les coûts liés aux tournées et les coûts de gestion de stock (comprenant une pénalité liée aux nombres probables de demandes différées), le problème à résoudre pour chaque article étant le modèle de gestion de stock classique à demande aléatoire $(R, T)$. Le sous-problème de tournées permet de générer des routes optimisées à l'aide d'un TSP pour chaque période du cycle et les nouveaux coûts des tournées par article sont donnés au problème maître. Le processus se répète. Les résultats sont meilleurs lorsque le nombre de fournisseurs est grand (une borne inférieure est calculée pour l'évaluation).

### 1.3.4 Conclusion

L'IRP admet de nombreuses variantes. Selon l'application sous-jacente, les hypothèses peuvent être très différentes. Suivant que les décisions sont prises à plus ou moins long terme (niveau tactique ou opérationnel) les hypothèses changent : au niveau opérationnel on choisit la mise à jour régulière des données et on construit des routes proches de la réalité en prenant en compte par exemple comme dans Bell et al (1983, [9]) une flotte hétérogène, les jours non ouvrables, les fenêtre de temps... Par contre au niveau tactique le problème est simplifié en considérant un nombre illimité de véhicules, en restreignant les stratégies de réapprovisionnement comme la stratégie de partition fixe pour Gaur et Fisher (2004 [50]), Anily et Federgruen (1990, [5]), Bramel et Simchi-Levi (1995, [14]) et/ou en ne prenant en compte que la contrainte de capacité... Ce grand choix parmi les hypothèses rend difficile les comparaisons des méthodes, d'autant plus qu'il n'y a pas de jeux de données de type "benchmark" existants.

Dans la littérature, la résolution de ces problèmes complexes se fait à l'aide de méthodes heuristiques avec en général deux phases comme dans Campbell et al (2004, [16]). La première phase de planification sur un horizon $T$ sélectionne les clients à livrer en chaque période et/ou détermine les volumes de livraison. La deuxième phase élabore les routes pour chaque période (typiquement par insertion itérative suivie d'une heuristique d'échange). Dans la première phase, on va essayer de prendre en compte la deuxième phase en estimant l'impact des coûts de transport (en résolvant un TSP pour Golden et al (1984 [53]), en estimant un coût fixe lié aux tournées pour Dror et Ball (1987, [37])). Ces deux phases peuvent aussi interagir en se passant des informations et en les résolvant itérativement, comme Qu et al (1999, [76]). Enfin dans le cas particulier de la politique de partition fixe, ces deux phases sont imbriquées car le choix d'un cluster définit la route utilisée (résolution d'un TSP) et le planning (résolution d'un plus court chemin pour Gaur et Fisher (2004, [50]), donné par le modèle EOQ pour Anily et Federgruen (1990, [5]), Bramel et Simchi-Levi (1995, [14])).

Les méthodes proposées utilisent souvent un principe de décomposition avec des sous-problèmes classiques comme le TSP, le VRP, le bin packing, le problème de sac-à-dos, le modèle EOQ... Les deux phases de l'heuristique consistent principalement à décomposer le problème avec la gestion de stock d'une part et l'élaboration des tournées d'autre part. Witucki et al (1997, [86]) utilise la décomposition exacte de Dantzig-Wolfe où le problème maître va gérer le stock et le sous-problème les routes. Bell et al (1983, [9]) et Chien et al

( 1989, [20]) utilisent la relaxation Lagrangienne pour décomposer le problème en sous-problèmes de sac-à-dos et servir de base à leur heuristique.

Signalons que du point de vue utilisateur, il est pratique de pouvoir modifier les solutions opérationnelles en fonction des aléas, ces modifications se font à l'aide d'interface manuelle mais peuvent aussi être aidées par des heuristiques très basiques qui doivent être très rapides, comme le module d'aide à la décision pour l'exploitation journalière de Witucki et al (1997, [86]).

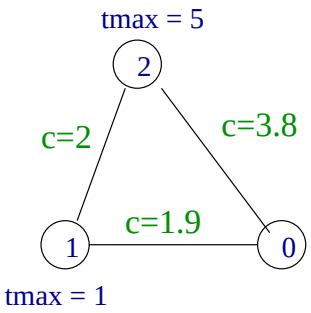
Notre problème se situe dans la catégorie "un produit, horizon infini et demandes déterministes avec l'hypothèse de la politique order up to level". Le seul article se rapprochant de nos hypothèses est celui de Webb et Larson (1995, [85]). Dans leur cas, les clients sont partitionnés en clusters et pour chaque cluster un ensemble de routes répété périodiquement est défini de sorte à minimiser l'utilisation moyenne du véhicule. Les clusters sont déterminés à l'aide d'une procédure basée sur le concept de gain de Clarke Wright. Dans notre cas, un cluster est une représentation homogène d'une route. Ainsi les clusters ne sont pas construits pour partitionner l'ensemble des clients mais le sont pour définir les routes. Nos "routes" sont générées en considérant à chaque fois tous les clients. Ensuite notre heuristique est basée sur une approche exacte de type décomposition. Enfin notre objectif qui est une combinaison linéaire des objectifs définis dans la table 1.4 est différent.

Webb et Larson (1995, [85]) ont montré que la politique de partition fixe est très restrictive dans le cas de l'hypothèse de la politique d' "order up to level". Notons d'abord que dans la littérature, on ne se soucie pas du nombre de véhicules (illimité pour [5] et majoré par $m$ pour [14]). Avec la politique de partition fixe, la livraison d'un cluster $R$ se fait quand le niveau de stock de chaque client devient nul, toutes les $p^{R*}$ périodes, mais cette livraison ne ramène pas forcément chaque niveau de stock à son maximum. Dans le cas de l'hypothèse de la politique d' "order up to level", pour ramener tous les niveaux de stock à leur maximum, il faut que les clients du cluster aient le même $tmax$, alors $p^{R*} = \min\{tmax, \lfloor \frac{W}{\sum_i d_i} \rfloor\}$. Cette politique est notée PF0. Lorsque les coûts de stockage ne sont pas pris en compte, une politique de partition fixe adaptée à l'hypothèse de la politique d' "order up to level" consiste à relâcher le fait que la livraison se fasse quand le stock de chaque client devient nul. Cette politique moins restrictive, notée PF, permet de livrer plus tôt, lorsque le premier niveau de stock tombe à zéro, alors $p^{R*} = \min\{\min_{i \in R}\{tmax_i\}, \lfloor \frac{W}{\sum_i d_i} \rfloor\}$. Le stock de certains clients du cluster peut n'être jamais nul. PF n'est plus

une politique de stock zéro mais reste une politique de partition fixe dans le sens où les clients sont partitionnés en clusters. De plus, lorsqu'un client du cluster est livré, tous les autres le sont. PF0 est un cas particulier de PF. La politique périodique du scénario C de la table 1.3 est comparée au chapitre 7.4 avec PF. Dans l'exemple 1.3, nous reprenons l'exemple donné dans [85] pour montrer que PF reste restrictive.

Exemple 1.3 – La politique de partition fixe est restrictive

Soit l'exemple de la figure suivante :




Supposons la capacité des véhicules suffisante.
La solution optimale obtenue avec la politique de partition fixe est

1. soit de coût $2 + (\frac{2*1.9}{1} + \frac{2*3.8}{5})\alpha = 2 + 5.32\alpha$ par période et visite :
   - 1 toutes les périodes,
   - 2 toutes les 5 périodes.
2. soit de coût $1 + \frac{1.9+2+3.8}{1}\alpha = 1 + 7.7\alpha$ par période et visite :
   - 1 et 2 toutes les périodes.

Avec la politique étendue du scénario C, la solution optimale, de coût $1 + (4\frac{2*1.9}{5} + \frac{1.9+2+3.8}{5})\alpha = 1 + 4.58\alpha$ par période, devient :

- 1 est visité 4 fois tout seul toutes les 5 périodes.
- 1 et 2 sont visités ensemble 1 fois toutes les 5 périodes.

# 2

# Approche de décomposition

Ce chapitre est consacré à l'approche utilisée pour résoudre notre problème. Les formulations naturelles, par exemple celle définie par les contraintes (1.1-1.21), font typiquement intervenir des variables de planification des visites chez les clients et des variables de construction de routes empruntées par les véhicules. La méthode de décomposition de Dantzig-Wolfe exploite cette structure pour reformuler le problème, et le résoudre ensuite par génération de colonnes. Dans une première section, la méthode de génération de colonnes est présentée brièvement. Notre problème présentant des symétries dans la représentation des solutions, la section 2.2 introduit des techniques classiques pour contrer les symétries. Les sections suivantes sont axées sur l'approche de décomposition. Les différentes décompositions possibles sont discutées dans la section 2.3. Suit une revue des méthodes heuristiques basées sur la décomposition. Les heuristiques utilisées dans la littérature sont les suivantes :

- la résolution du maître restreint en nombre entier (voir la section 2.4.1),
- les heuristiques de type glouton (voir la section 2.4.2),
- les heuristiques d'arrondi (voir la section 2.4.3),
- les heuristiques de recherche locale et méta-heuristiques (voir la section 2.4.4).

Enfin dans la section 2.4.5 sont présentées des heuristiques basées sur la programmation entière mais mal adaptées à l'approche de Branch-and-Price.

## 2.1 L'algorithme de Branch-and-Price

La technique de reformulation de Dantzig-Wolfe se résume comme suit. Considérons le problème suivant :

$$\begin{aligned} \min \quad & cx & (2.1)\\ Ax \quad & \geq \quad a & (2.2)\\ Bx \quad & \geq \quad b & (2.3)\\ x \quad & \in \quad \mathbb{N}. & (2.4) \end{aligned}$$

où les contraintes (2.2) sont vues comme des contraintes difficiles ou liantes alors que les contraintes (2.3) définissent un sous-problème bien structuré (par exemple le système de contraintes a une structure diagonale par block). Soit $X = \{x, Bx \geq b, x \in \mathbb{N}\}$ le sous-problème (on le suppose non vide et borné) et $Q$ l'ensemble énuméré des solutions du sous-problème, $X = \{x^q\}_{q \in Q}$, alors la reformulation de Dantzig-Wolfe s'écrit :

$$\begin{aligned} [M] \quad Z^M = \min \sum_{q \in Q} \quad & (cx^q) \quad \lambda_q & & (2.5)\\ \sum_{q \in Q} (Ax^q)\lambda_q \quad & \geq \quad a & (u) & (2.6)\\ \sum_{q \in Q} \lambda_q \quad & = \quad 1 & (\sigma) & (2.7)\\ \lambda_q \quad & \in \quad \{0,1\} & \forall \quad q \in Q. & (2.8) \end{aligned}$$

Le problème $[M]$ défini par les contraintes (2.5-2.8) est appelé maître. Notons $Z^M_{PL}$ la valeur de l'objectif de la relaxation linéaire du maître (où $\lambda_q \in [0,1]$).

La formulation $[M]$ met en jeu un grand nombre de variables, sa relaxation linéaire est résolue par génération de colonnes. Les colonnes sont ajoutées dynamiquement à la formulation, les étapes de la procédure d'optimisation sont les suivantes :

1. Résoudre à l'optimalité la relaxation linéaire du maître $[M]$ restreint au sous ensemble de colonnes générées $\overline{Q}$ jusqu'à présent, la solution duale $(u, \sigma)$ est enregistrée.
2. Vérifier s'il existe des colonnes de coût réduit négatif en résolvant un problème de pricing

$$\zeta(u) = \min\{(c - uA)x : \ x \in X\} \tag{2.9}$$

3. Si $\zeta(u) + \sigma < 0$, ajouter la colonne $x^q = argmin\ \zeta(u)$ à $\overline{Q}$ et retourner en 1. Sinon la solution courante est $PL$ optimale.

A chaque itération de la procédure de génération de colonnes, la valeur de l'objectif du maître $PL$ restreint ne constitue pas une borne duale valide pour le maître entier. Toutefois une borne duale valide $L(u)$ peut être obtenue en se basant sur la relaxation lagrangienne des contraintes (2.6). Le programme avec les contraintes (2.6) relaxées est le suivant :

$$L(u) = \min \sum_{q \in Q} (cx^q)\lambda_q \quad + \quad u(a - \sum_{q \in Q} (Ax^q)\lambda_q) \tag{2.10}$$

$$\sum_{q \in Q} \lambda_q \quad = \quad 1 \tag{2.11}$$

$$\lambda_q \quad \geq \quad 0 \quad \forall \quad q \in Q. \tag{2.12}$$

En regroupant les termes en $\lambda$ puis en prenant une fois la colonne $x^q = argmin\zeta(u)$, la borne lagrangienne devient :

$$L(u) = ua + \zeta(u). \tag{2.13}$$

La meilleure borne lagrangienne obtenue au cours du processus de génération de colonnes, notée $LD$, est enregistrée :

$$LD = \max_u L(u) \tag{2.14}$$

La connaissance de cette borne fournit un autre critère d'arrêt : le processus de génération de colonnes s'arrête lorsque la borne duale Lagrangienne courante $LD$ est supérieure ou égale à la valeur du maître $PL$ restreint.

Notons $conv(X)$ l'enveloppe convexe de l'ensemble des solutions de $X$, la théorie de la dualité Lagrangienne [51] nous dit que le maître $PL$ mène à une borne $Z^M_{PL}$ qui est équivalente à $LD$ :

$$Z^M_{PL} = \min\{cx : Ax \geq a, x \in conv(X)\} = LD \tag{2.15}$$

La reformulation de Dantzig-Wolfe définit un maître $PL$ qui est équivalent à une convexifixation implicite du sous-problème. La qualité de cette borne dépend de la définition de $X$. Si $X = conv(X)$, alors la borne $Z^M_{PL}$ est égale la borne obtenue par la relaxation linéaire du problème original (2.1-2.4).

Le programme maître est initialisé avec des variables artificielles. La résolution du maître $PL$ se fait en combinant les phases 1 et 2 du simplex grâce à

la gestion dynamique des coûts des variables artificielles. Lorsqu'il n'y a plus de colonne de coût réduit négatif mais qu'il reste des variables artificielles, leur coût est augmenté et la génération de colonnes recommence. Le nombre d'augmentations du coût des variables artificielles est paramétré (ce paramètre sera noté nbMaxMajPenalite), lorsque le nombre de mises à jour des coûts des variables artificielles est atteint, une phase 1 pure est résolue (le coût des colonnes est mis à zéro) pour prouver que le maître est soit irréalisable (l'objectif optimal est non nul), soit réalisable (on passe alors à la phase 2). Lorsque le problème est irréalisable, les phases 1 et 2 combinées sont peu efficaces. En effet, il faut de nombreux paliers d'augmentation des coûts des variables artificielles avant de pouvoir prouver l'irréalisabilité par comparaison de la borne duale (2.14) à une solution connue INC. Par contre lorsque le problème est réalisable, une phase 1 pure est moins efficace que les phases 1 et 2 combinées (dans la phase 1, les colonnes générées doivent juste satisfaire les contraintes quelque soit leur coût). Ainsi, il ne faut choisir nbMaxMajPenalite ni trop petit pour que la phase 1 ne soit pas appelée lorsque le problème est réalisable, ni trop grand pour que la phase 1 soit appelée lorsque le problème est irréalisable.

### 2.1.1 Méthode exacte

Lorsque les variables sont entières, on a recours à un arbre d'énumération à chaque nœud duquel on fait appel à la génération de colonnes, méthode connue sous le nom d'approche de "Branch-and-Price" (BaP). Cette technique permet de résoudre des problèmes entiers de grande taille [8]. Elle a été appliquée avec succès dans divers contextes : VRP avec fenêtres de temps [34], avec multi-dépôts [77], les problèmes de tournée avec gains sur les arcs [40], les problèmes de découpes ("cutting stock") [82] ...

Pour imposer l'intégralité des variables, il est nécessaire de brancher. Brancher directement sur une variable $\lambda_q = f \notin I\!N$ (i.e. $\lambda_q \leq \lfloor f \rfloor$ ou $\lambda_q \geq \lceil f \rceil$) n'est pas efficace : l'arbre est déséquilibré ($\lambda_q \leq \lfloor f \rfloor$ est moins restrictif que $\lambda_q \geq \lceil f \rceil$) et le problème de pricing doit être remanié pour ne pas régénérer la colonne $q$. Retourner dans l'espace des variables originales pour vérifier l'intégralité et dériver les contraintes de branchement est une première alternative. Une autre consiste à partitionner l'ensemble $Q = \hat{Q} \cup (Q \backslash \hat{Q})$ en choisissant $\hat{Q}$ tel que $\sum_{q \in \hat{Q}} \lambda_q = f \notin I\!N$ (alors $\sum_{q \in \hat{Q}} \lambda_q \leq \lfloor f \rfloor$ ou $\sum_{q \in \hat{Q}} \lambda_q \geq \lceil f \rceil$). Ces branchements peuvent modifier le problème de pricing en ajoutant des variables et/ou des contraintes.

Pour notre problème, nous branchons sur les variables du maître (voir section 6.4.2). Cependant, des règles de branchement auraient pu être utilisées en se basant sur des ensembles de colonnes. Un rapide aperçu des règles utilisées dans la littérature pour des problèmes similaires est donné ci-après.

Pour le VRPTW, Desrochers et al (1992, [34]) branchent d'abord sur le nombre de routes utilisées $\sum_{q\in Q}\lambda_q$ ($\hat{Q}=Q$), seul l'objectif du problème de pricing est modifié. Ensuite ils branchent sur les variables originales $x_{ij}=1$ si l'arc $(i,j)$ est dans la solution, 0 sinon ($\hat{Q}=\{q : x^q_{ij}=1 \text{ ou } 0\}$). Dans la branche où $x_{ij}=1$ ($\sum_q x^q_{ij}\lambda_q=1$), les arcs $(i,k\neq j)$ et $(k\neq i,j)$ sont alors supprimés dans le problème de pricing et un prix est associé à l'arc $(i,j)$ ; et dans la branche où $x_{ij}=0$ ($\sum_q x^q_{ij}\lambda_q=0$), l'arc $(i,j)$ est supprimé. La réoptimisation du nœud se fait avec le même oracle. Le branchement sur la variable $x_{ij}=\sum_q x^q_{ij}\lambda_q$ suffit à obtenir une solution entière en $x$ et en $\lambda$. Cependant, comme l'ont constaté Feillet et al (2005, [40]) pour le problème de tournée avec gains sur les arcs, la solution en $\lambda$ peut être fractionnaire mais entière une fois exprimée dans les variables de flot originales $x$. De plus, il n'est pas trivial de transformer la solution fractionnaire $\lambda$ en une solution entière réalisable de même coût en utilisant seulement les valeurs entières obtenues pour $x$. Dans ces cas, d'autres contraintes de branchements doivent finalement être utilisées.

### 2.1.2 Ajout de coupes

Lorsque la borne duale donnée par (2.14) est faible, une méthode pour l'améliorer consiste à ajouter des plans coupant les solutions fractionnaires du programme maître ("cutting plane"). La combinaison de la génération de colonnes avec la génération de coupes à chaque nœud de l'arbre d'énumération est ce qu'on appelle la méthode de Branch-and-Price-and-Cut (BPC).

Dans le problème de cutting stock, Vanderbeck (2000, [82]) remarque que lorsque le nombre de patrons différents est minimisé, au nœud racine l'écart relatif entre la borne duale et la meilleure solution réalisable est de 33.5% en moyenne. Après avoir ajouté des coupes, cet écart est égal à 13.8%. Les coupes sont basées sur des fonctions superadditives et sont exprimées dans les variables associées aux colonnes : l'objectif du problème de pricing est modifié et devient non linéaire dans certains cas, il faut alors modifier aussi l'oracle pour le résoudre.

Des instances du problème de tournées de véhicules sont résolues pour la première fois par Fukasawa et al. (2004, [48]) en combinant l'approche de Branch-and-Cut et la génération de colonnes (les colonnes représentent des q-routes sans petit cycle). Leur BPC est dit robuste car les coupes ajoutées ne modifient pas la structure du problème de pricing : les coupes sont générées à partir de la solution dans les variables originales, puis retraduites dans les variables du maître. Les coupes "rounded capacity" interagissent bien avec les q-routes pour améliorer la borne duale, alors que les autres familles de coupes connues ont un effet modeste, les coupes "generalized large multistar inequalities" sont même inutiles comme le montrent Letchford et Salazar-Gonzalez dans [62]. Poggi de Aragão et Uchoa [74] présentent une technique de reformulation qui étend l'application de l'algorithme BPC robuste à d'autres problèmes, en particulier il a déjà été testé sur les problèmes d'arbre de recouvrement de coût minimum avec capacité [47].

Chabrier (2003, [17]) propose un Branch-and-Price-and-Cut heuristique pour résoudre le problème de conception de réseau : il s'agit de dimensionner les arcs d'un réseau de télécommunication pour que chaque commodité puisse être routée sur un chemin unique. Cet article s'inspire du projet ROCOCO [75] : il s'agissait de trouver une bonne solution et/ou une borne inférieure pour de grandes instances en moins de 10 minutes. Il présente un modèle de décomposition basé sur les chemins et résolu par une approche de Branch-and-Price. La borne inférieure étant de mauvaise qualité (l'écart relatif entre la borne duale et la solution optimale peut être proche de 50%), il l'améliore en ajoutant des coupes : 4 sortes de coupes sont définies dont une seule va modifier l'objectif du problème de pricing, les autres ne faisant pas intervenir les variables associées au problème de pricing. Ne pouvant explorer tout l'arbre car étant limité par le temps, il fait appel à des heuristiques pour avoir une bonne borne supérieure (voir la section 2.4).

Pour améliorer notre borne duale, des inégalités valides dérivées des contraintes de notre formulation sont générées (voir section 6.4.1). Dans [68], on montre comment générer des inégalités valides : partant des inégalités initiales, on applique des fonctions superadditives aux coefficients qui ont des variables dans les contraintes initiales.

**Définition 2.1.1** *Une fonction $F : D \subseteq \mathbb{R}^m \to \mathbb{R}$ est appelée superadditive sur $D$ si*

$$F(d_1) + F(d_2) \leq F(d_1 + d_2) \quad \forall d_1, d_2, d_1 + d_2 \in D \tag{2.16}$$

Notons que $d_1 = 0$ mène à $F(0) + F(d_2) \leq F(d_2)$ c'est à dire $F(0) \leq 0$. Par la suite, lorsque $F$ est superadditive, l'hypothèse est faite que $F(0) = 0$ et $0 \in D$.

**Proposition 2.1.2** *[68] Soit $X^{\leq} = \{x \in \mathbb{Z}^n : Ax \leq b\}$ où $A$ et $b$ sont des matrices de taille respective $m \times n$ et $m \times 1$ avec des coefficients rationnels. Soit une fonction $F : \mathbb{R}^m \to \mathbb{R}$ superadditive et croissante, alors :*

$$\sum_{i \in n} F(A_i)x_i \leq F(b) \tag{2.17}$$

*est une inégalité valide pour $X^{\leq}$.*

Plusieurs exemples de fonctions superadditives sont proposées dans [68]. En prenant $F : \mathbb{R}^m \to \mathbb{R}$ définie par $F(d) = \lfloor ud \rfloor$ avec $u \in \mathbb{R}^m$, on obtient les inégalités d'arrondi à l'entier de Gomory. Ces inégalités peuvent être améliorées en prenant $F_\gamma : \mathbb{R} \to \mathbb{R}$ avec $0 \leq \gamma < 1$ définie par $F_\gamma(d) = \lfloor d \rfloor + \frac{(f_d - \gamma)^+}{1-\gamma}$ où $f_d = d - \lfloor d \rfloor$ et $(g)^+ = max(0, g)$ pour tout $g \in \mathbb{R}$.

### 2.1.3 Solutions approchées

La méthode de Branch-and-Price est une méthode exacte. Aux différentes étapes de la procédure (résolution du maître, du problème de pricing, parcours de l'arbre), des approximations peuvent être utilisées. La méthode de Branch-and-Price devient donc une méthode qui permet d'obtenir des solutions approchées et ainsi peut résoudre des problèmes réels de plus grande taille.

Desaulniers et al. (1999, [33]) résument les stratégies à tous les niveaux (préprocessing, résolution du problème de pricing, du maître, branchement, postprocessing) qui ont été développées pour accélérer la méthode de génération de colonnes dans la cas des problèmes de tournées de véhicules et d'horaires d'équipes de travail. Certaines stratégies, comme résoudre une relaxation du problème de pricing, diminuer la taille du graphe, tronquer la résolution du maître, brancher plus tôt, peuvent induire l'élimination de la solution optimale. Vance et al (1997, [81]) résolvent heuristiquement le problème d'appariement d'équipage dans les compagnies d'aviation en résolvant le problème de pricing approximativement et en n'explorant pas tous les nœuds de l'arbre (l'algorithme s'arrête lorsque la solution est acceptable).

## 2.2 Symétrie

Les procédures de résolution sont basées sur une énumération de solutions réalisables. La présence de symétries dans la formulation rend difficile la résolution du problème, typiquement dû au fait que la procédure retombe sur des solutions équivalentes. L'arbre d'énumération dans la méthode de Branch-and-Bound peut devenir vite conséquent et peu efficace en tombant sur des solutions équivalentes à différents nœuds. La génération de colonnes est plus instable, les variables duales n'ayant plus une réelle signification économique.

Les symétries les plus simples sont définies par des permutations d'indices : une solution équivalente peut être obtenue en appliquant une permutation à la solution courante. Dans les problèmes où les éléments sont dispatchés dans différents ensembles identiques ; la permutation des ensembles fournit des représentations différentes d'une même solution. Ces problèmes se formulent typiquement à l'aide de variables $x_{ik} = 1$ si l'élément $i$ est dans l'ensemble $k$ et $y_k = 1$ si l'ensemble $k$ est utilisé. Prenons comme exemple le problème de coloration de graphe dans lequel chaque sommet reçoit une couleur mais deux sommets reliés par une arête doivent être coloriés de couleurs différentes, l'objectif est de minimiser le nombre de couleurs utilisées. Définissons un graphe $G = (N, E)$ où $N$ est l'ensemble des sommets et $E$ l'ensemble des arêtes. Supposons que $K$ couleurs sont disponibles. Posons comme variables $y_k = 1$ si la couleur $k$ est choisie, zéro sinon et $x_{ik} = 1$ si le sommet $i \in N$ est colorié avec $k$, 0 sinon. Le problème de coloration de graphe se formule comme suit :

$$\min \quad \sum_{k=1}^{K} y_k \tag{2.18}$$

$$\sum_{k} x_{ik} = 1 \qquad \forall i \tag{2.19}$$

$$x_{ik} + x_{jk} \leq 1 \qquad \forall e = (i,j) \in E, k \tag{2.20}$$

$$x_{ik} \leq y_k \qquad \forall i, k \tag{2.21}$$

$$y_k, x_{ik} \in \{0,1\} \qquad \forall i, k. \tag{2.22}$$

Les contraintes (2.19) assurent que chaque sommet $i$ est colorié, (2.20) renforcent que deux sommets voisins $i$ et $j$ reçoivent des couleurs différentes et (2.21) mettent $y_k$ à 1 si la couleur $k$ est utilisée. L'objectif (2.18) minimise le nombre de couleurs utilisées. Cette formulation souffre d'une symétrie en $k$ : toute permutation de l'indice $k$ des couleurs fournit une représentation différente d'une même solution.

Pour éliminer les symétries, différentes techniques sont utilisées. Une première technique consiste à faire un changement de variables pour formuler différemment le problème. Dans les problèmes où les éléments sont dispatchés dans différents ensembles identiques ; un meilleur choix de variables consiste à définir la variable $z_{ij}$ égale à 1 si les éléments $i$ et $j$ sont dans le même ensemble. Pour le problème de coloration de graphe, ces variables sont $z_{ij} = 1$ si le sommet $i$ initialise une nouvelle couleur et le sommet $j$ reçoit la même couleur que le sommet $i$. Les variables $z$ sont définies pour tout sommet $i < j$ telles que l'arête $e = (i, j) \notin E$. $z_{ii} = 1$ signifie que le sommet $i$ est le sommet de plus petit indice qui initialise la couleur. La formulation avec les variables $z$ est :

$$\begin{array}{rcll} \min & & \sum_{i=1}^{N} z_{ii} & \\ \sum_{i \leq j} z_{ij} & = & 1 & \forall j \\ z_{ki} + z_{kj} & \leq & z_{kk} & \forall e = (i,j) \in E, k \\ z_{ij} & \leq & z_{ii} & \forall i < j \\ z_{ij} & \in & \{0,1\} & \forall i < j. \end{array}$$

Dans le problème de bin packing, il faut mettre des objets de différente taille dans des boîtes identiques de taille unitaire. La formulation est similaire à la formulation (2.18-2.22) excepté que les contraintes (2.20) sont remplacées par des contraintes de sac-à-dos $\sum_i w_i x_{ik} \leq W y_k$. Ce problème peut être formulé en indexant chaque boîte par le plus petit objet qu'elle contient. Poggi de Aragão et Uchoa [74] proposent une reformulation basée sur des plus courts chemins orientés avec capacité. Chaque solution correspond à un empaquetage différent.

Une deuxième technique pour éliminer la symétrie est la décomposition de Dantzig-Wolfe vue en section 2.3. Dans la formulation de la coloration de graphe, le sous-système (2.20-2.22) se décompose pour chaque $k$ en un problème qui consiste à définir un sous-graphe de $G$ sans arête, appelé un stable de $G$ ("Stable Set Problem"). Soit l'ensemble des stables de $G$ :

$$X^{SSP} = \{x \in \{0,1\}^N : x_i + x_j \leq 1 \ \ \forall e = (i,j) \in E\} = \{x^q\}_{q \in Q}$$

Posons $\lambda_q = 1$ si le stable d'indice $q$ est utilisé, zéro sinon. Chaque stable

est associé à une couleur et peut être colorié par n'importe quelle couleur. La reformulation de Dantzig-Wolfe est :

$$\begin{aligned} \min \quad & \sum_q \lambda_q \\ \sum_q x_i^q \lambda_q &= 1 \qquad \forall i \\ \lambda_q &\in \{0,1\}. \end{aligned}$$

Pour le problème de bin packing, le sous-système pour chaque boîte est un problème de sac-à-dos.

Une troisième technique est l'ajout de contraintes qui permettent d'éliminer les copies d'une même solution (ou d'une partie des solutions). La solution gardée est la solution de référence et typiquement est la première solution de la classe symétrique ordonnée lexicographiquement. Pour le problème de coloration de graphe, les variables $x_{ik}$ peuvent être définies pour $k \leq i$, i.e. le sommet 1 reçoit la couleur 1, le sommet 2 reçoit la couleur 1 ou 2, etc ... et les contraintes ajoutées sont de la forme :

$$\sum_{i<j} x_{ic} \geq \sum_{k>c} x_{jk} \quad \forall j, c$$

Elles signifient que le sommet $j$ peut recevoir une couleur $k$ d'indice plus grand que $c$ seulement si la couleur $c$ colore déjà un sommet $i$ d'indice plus petit que $j$. Généralisant cette idée, Kaibel et Pfetsch [59] dérivent une famille d'inégalités de taille exponentielle.

Une quatrième technique consiste à organiser le processus d'énumération dans l'arbre de Branch-and-Bound pour ne pas explorer un nœud menant à une solution équivalente à une solution déjà connue. Margot [65] présente un algorithme de Branch-and-Bound dans lequel seul le premier représentant de chaque groupe symétrique ordonné lexicographiquement est énuméré. Cependant il est nécessaire d'identifier chaque groupe symétrique.

Une cinquième technique est la fixation de variables. Cette fixation peut être exacte ou heuristique, c'est à dire que cette fixation peut être faite sous risque d'éliminer la solution optimale. Dans le problème de coloration de graphe, une fixation exacte consiste à colorier initialement deux sommets voisins avec deux couleurs différentes. Pour le problème de construction de grille d'emploi du temps dans le transport d'équipe sur des plate-formes pétrolières, Menezes et

al [66] évitent leur symétrie en temps en faisant des fixations itératives. Ces fixations se font à l'aide de contraintes de branchements locaux : les valeurs de certaines variables sont fixées par rapport à une solution connue. Initialement ils résolvent en nombre entier un problème restreint à un sous ensemble de contraintes et obtiennent une première solution entière. Itérativement, des contraintes du problème ainsi que des contraintes de branchements locaux sont ajoutées et une nouvelle solution entière est recherchée.

Notre problème souffre de plusieurs symétries. Une première symétrie est liée à l'utilisation des véhicules, chaque véhicule est indexé par $v$ dans les formulations proposées dans la section 1.2. Comme les véhicules sont identiques, le fait d'utiliser un véhicule plutôt qu'un autre pour assurer une même route ne change en rien la solution (la permutation des véhicules n'a pas d'incidence sur la solution). Pour éviter cette symétrie dans la formulation (1.22-1.35), l'indexation des véhicules peut se faire par le centre $i$. Les variables utilisées sont $x_{ij\ell ps} = 1$ si le client $j$ est connecté au centre $i$ et est collecté pour $\ell$ périodes toutes les $p$ périodes, sa première collecte étant à la date $s$ ($x_{ii\ell ps} = 1$ si le client $i$ est le centre). La formulation du problème de tournées de véhicules combinées à la gestion de stock devient :

$$\min \quad Vmax + \alpha \sum_{p,s} \frac{1}{p} \sum_{i,j,\ell} (c_i x_{ii\ell ps} + c_{ij} x_{ij\ell ps}) \tag{2.23}$$

$$\sum_{\ell} x_{ii\ell ps} \leq 1 \qquad \forall i, p, s \tag{2.24}$$

$$\sum_{\ell} x_{ij\ell ps} \leq \sum_{\ell} x_{ii\ell ps} \qquad \forall i, j, p, s \tag{2.25}$$

$$\sum_{j} d_j \sum_{\ell} \ell x_{ij\ell ps} \leq W \qquad \forall i, p, s \tag{2.26}$$

$$\sum_{i\ell ps} \delta_{jt}^{i\ell ps} x_{ij\ell ps} = 1 \qquad \forall j, t = 1, \cdots, T \tag{2.27}$$

$$\sum_{i\ell ps} \delta_{0t}^{i\ell ps} x_{ii\ell ps} \leq Vmax \qquad \forall t = 1, \cdots, T \tag{2.28}$$

$$x_{ij\ell ps} \in \{0, 1\} \qquad \forall i, j, \ell, p, s \tag{2.29}$$

$$Vmax \in I\!N. \tag{2.30}$$

Le nombre de variables est de l'ordre de $O(N^2 Pmax^3)$ et le nombre de contraintes de $O(N^2 Pmax^2 + NT)$ où $N$ est le nombre de clients, $Pmax$ est la plus grande périodicité et $T$ est la longueur maximale du cycle de régénération.

Dans le cas des scénarios B et C, une autre symétrie apparente est la symétrie en $t$. Comme la solution est périodique et que la situation initiale n'est

pas connue, la solution obtenue en ayant tout décalé d'une (ou plusieurs) période est identique à la solution sans décalage. De plus, des décalages sur des sous ensembles de clients ne changent pas la solution si Vmax reste inchangé. Cette symétrie diffère de la précédente car une permutation quelconque entre les dates peut ne pas mener à une solution réalisable (la permutation est du type cyclique). Nous verrons au chapitre 6 comment éliminer cette symétrie dans la résolution du maître $PL$. Mais pour le maître $PLE$, nous n'avons pas développer de technique autre que la fixation itérative heuristique.

## 2.3 Choix de décompositions

L'étude de notre problème requiert de coupler deux types de modèles :

**(i)** un modèle de planification des visites client dicté par la gestion des stocks,

**(ii)** un modèle de tournées avec capacité limitée.

Pour exploiter la structure de ce problème avec ses deux composantes bien identifiées, nous proposons d'utiliser une approche de décomposition de Dantzig-Wolfe. Avec cette approche, un grand nombre de sous-problèmes de petite taille et bien structurés est résolu plutôt que le problème original de grande taille et plus complexe. Les formulations naturelles de notre problème, définies par les contraintes (1.1-1.21) ou (1.22-1.35), font typiquement intervenir des variables de planification des visites chez les clients (les variables $y, x, s$), et des variables de construction de routes empruntées par les véhicules ou de clusters affectés aux véhicules (les variables $r, q, z$). Les contraintes se classent en trois groupes :

1. les contraintes définissant les plannings individuels des clients (imposant des collectes avant le dépassement de la capacité), (1.2-1.6) ou (1.23) ;
2. les contraintes liées à la problématique de transport qui garantissent la construction des routes ou clusters réalisables, (1.7-1.12) ou (1.24-1.28) ;
3. les contraintes liant ces deux sous-problèmes qui imposent un choix de planning de visites des clients compatible avec les routes, (1.13-1.14) ou (1.29-1.30).

Les contraintes (1.15) ou (1.31) sont aussi des contraintes liantes définissant le nombre de véhicules nécessaires. Schématiquement le problème pourrait s'écrire :

$$\begin{array}{lrcll} \text{(coût des tournées)} & \min Vmax & + & c(z) & (2.31) \\ \text{(élaboration du planning)} & A(x,y,s) & \geq & a & (2.32) \\ \text{(construction des tournées)} & B(z,r,q) & \geq & b & (2.33) \\ \text{(lien)} & D(x,y,q,r,Vmax) & \geq & d & (2.34) \end{array}$$

Trois types de décompositions sont possibles selon les contraintes relaxées :

1. on relaxe les contraintes liantes (2.34) et les contraintes construisant les tournées (2.33) ; le sous-problème résultant est alors un problème d'élaboration du planning dicté par la gestion des stocks.
2. on relaxe les contraintes liantes (2.34) et les contraintes d'élaboration du planning (2.32) ; le sous-problème résultant est alors un problème de construction des tournées.
3. on relaxe les contraintes liantes (2.34) ; alors deux sous-problèmes en résultent : un problème d'élaboration du planning et un de construction des tournées (on fait ce qu'on appelle la décomposition Lagrangienne).

Les contraintes de construction des tournées (1.7-1.12) ou (1.24-1.28) sont définies pour tout véhicule $v$, quelque soit la modélisation de la table 1.1. Les véhicules étant identiques, une symétrie apparaît. La définition du problème de tournées comme sous-problème dans la reformulation de Dantzig-Wolfe permet d'éviter cette symétrie en $v$ : pour une période donnée, un seul sous-problème est résolu et retourne une route (colonne) qui est valide pour n'importe quel véhicule. Le sous-problème de tournées pour un véhicule, une période donnée peut être résolu relativement efficacement à l'aide d'un programme dynamique (voir le chapitre 3). Afin d'éviter la symétrie résultant de l'indexation des véhicules identiques, la décomposition 1 n'est pas choisie.

Les contraintes d'élaboration du planning dicté par la gestion des stocks (1.2-1.6) ou (1.23) sont définies pour chaque client $i$. Dans le cas des scénarios A et B de la table 1.3, le problème d'élaboration d'un planning individuel (1.2-1.6) pour un client peut être formulé comme un problème de flot de taille raisonnable (voir le chapitre 4). Dans le cas du scénario A, ce problème de flot est un problème de plus court chemin et a la propriété d'intégralité. D'après l'égalité (2.15), formuler le problème d'élaboration du planning comme sous-problème n'améliorera pas la borne duale (2.14). Une décomposition où l'élaboration du planning est un sous-problème est donc peu attractive. Dans le cas du scénario B, le problème de flot est un problème de flot cyclique de coût minimum et la matrice des contraintes est presque totalement uni-modulaire (voir le chapitre 4). Nous pensons que formuler le problème d'élaboration du planning comme sous-problème n'améliorera que peu la borne duale (2.14). Pour ces deux scénarios, nous choisissons donc la décomposition 2, les contraintes de flot peuvent être formulées directement dans le maître (elles prennent le rôle des contraintes liantes).

Dans le cas du scénario C de la table 1.3, les contraintes d'élaboration du planning (1.23) (ou (2.27) pour la formulation compacte) sont des contraintes qui séquencent pour chaque client $i$ les plannings partiels individuels définis par les variables $x$ (voir le chapitre 4). Le sous-problème de tournées génère des routes (des colonnes) définissant les clients visités et les quantités collectées. Dans le maître des routes sont choisies avec une périodicité $p$ et une date $s$ de départ. En incluant la périodicité $p$ et la date de départ $s$ comme variables dans la colonne (il ne s'agit plus d'indices), la variable associée à la colonne $q$ dans le maître devient $\lambda_q$ et vaut 1 si la route $q$ de périodicité $p$ et commençant à la date $s$ est choisie dans la solution. Alors la colonne définit pour les clients visités dans la route un planning partiel. Cela consiste à inclure les variables $x$ dans la colonne : le sous-problème de tournées doit finalement optimiser la route avec les quantités collectées mais aussi la périodicité et la date de départ. Dans ce cas, la matrice $\delta$ qui permet d'exprimer les contraintes (1.23) ou (2.27) est aussi définie à partir de la colonne. Les contraintes d'élaboration du planning vont donc ordonner les colonnes et prennent le rôle des contraintes liantes. Pour le scénario C, nous choisissons donc aussi la décomposition 2 (de préférence à la décomposition 3 qui n'a plus lieu d'être vue que le problème d'élaboration du planning n'est plus indépendant du problème de construction de tournées) car elle est susceptible de produire de meilleure borne duale.

## 2.4 Solutions approchées basées sur la décomposition

La méthode de BPC est une méthode exacte qui peut aussi servir de base pour une résolution heuristique des problèmes de taille réelle en un temps raisonnable. Une bonne solution primale peut être obtenue en tronquant l'algorithme de BPC, mais aussi à l'aide d'une heuristique exploitant le principe de décomposition. Les heuristiques classiques de type glouton, d'arrondi de la solution continue ou de recherche locale, peuvent être adaptées à l'approche de décomposition. Intégrer à une méthode exacte de Branch-and-Price, elles permettent d'améliorer la borne primale et ainsi réduire le nombre de nœuds à explorer. Ces heuristiques permettent de trouver de "bonnes" solutions pour des problèmes réels de grande taille dont la structure se prête à une décomposition de Dantzig-Wolfe. Dans les sections qui suivent, les principales heuristiques sont décrites en pointant leur utilisation dans la littérature.

### 2.4.1 Résolution du problème maître restreint

Une façon naturelle d'obtenir des solutions entières au cours du processus de génération de colonnes, est de résoudre le problème maître en nombre entier restreint à ensemble de colonnes $\overline{Q}$. Le solveur utilisé peut avoir un temps et un nombre de nœuds limités, c'est alors la meilleure solution entière obtenue dans les limitations imparties qui est retournée.

Cet ensemble $\overline{Q}$ peut contenir les colonnes générées lors de la résolution de maître $PL$, c'est ce que fait Chabrier et al. (2002, [18]) pour le VRP avec fenêtre de temps. Pour le problème de conception de réseau, Chabrier (2003, [17]) teste deux types d'ensemble $\overline{Q}$ : le premier est local, le deuxième global. La résolution du maître en nombre entier est limitée. Dans le premier cas, le problème maître est restreint à l'état courant du nœud : toutes les contraintes de branchement et les coupes correspondant à ce nœud sont utilisées et $\overline{Q}$ est l'ensemble des colonnes courantes. Le but est de prendre avantage des éléments implémentés dans CPLEX MIP (les heuristiques et plans coupants). Dans le deuxième cas, $\overline{Q}$ est l'ensemble de toutes les colonnes générées à travers tous les nœuds du BPC, les coupes et les contraintes de branchements sont momentanément écartées. L'objectif de ce second choix de $\overline{Q}$ est de diversifier la recherche pour trouver des solutions différentes correspondant à des nœuds non explorés (les décisions locales ne sont pas prises en compte). Chabrier a constaté que la recherche sur l'ensemble local fournit la plupart des solutions. La résolution du problème en nombre entier n'est appelée qu'aux nœuds ayant un nombre de variables fractionnaires plus petit qu'un seuil.

Pour le VRP, Agarwal et al (1989, [2]) forment $\overline{Q}$ avec les colonnes de la solution $PL$ optimale et résolvent le problème entier avec les contraintes de recouvrement. Cependant dans la solution obtenue, des clients peuvent être surcouverts ; pour chacune des solutions réduites obtenues en enlevant les surcouvrements, les sous colonnes nécessaires sont générées. Ensuite $\overline{Q}$ est reformé avec les colonnes de la solution précédente et celles régénérées, puis le problème entier avec les contraintes de partitionnement est résolu.

Enfin, $\overline{Q}$ peut contenir des colonnes provenant d'un autre algorithme. Pour résoudre le problème de tournées de véhicules avec flotte hétérogène, Taillard (1999, [78]) génère des colonnes pour chaque type de véhicule à l'aide d'une méta-heuristique à mémoire adaptative puis résoud le programme entier avec ces colonnes.

### 2.4.2 Heuristiques de type glouton

Pour construire rapidement une solution, une heuristique possible est l'heuristique gloutonne ("greedy heuristic") dont le schéma est présenté à la table 2.1.

| | |
|---|---|
| **GH :** | ***Etape 0 :*** Génération d'une colonne réalisable dont le coût réduit est optimisé pour des variables duales fixées heuristiquement. |
| | ***Etape 1 :*** Sélection de cette colonne autant de fois que possible. |
| | ***Etape 2 :*** Mise à jour du problème maître. |
| | ***Etape 3 :*** Si la solution partielle n'est pas réalisable et que des colonnes peuvent encore être générées, alors retour à l'étape 0 ; sinon STOP. |

TAB. 2.1 – Algorithme de l'heuristique gloutonne

A l'étape 0, une nouvelle colonne peut être générée à partir soit d'une estimation des variables duales faite a priori, soit des variables duales du problème maître courant. Pour différents problèmes de cutting stock, Perrot (2005, [69]) teste l'heuristique soit en choisissant des valeurs duales estimant l'optimum $PL$, soit en utilisant les valeurs duales optimales. Ce dernier choix donne en moyenne de meilleures solutions.

A l'étape 2, lorsque le maître est mis à jour, les variables duales peuvent l'être aussi. Pour le problème de cutting stock avec plusieurs types de rouleaux, Belov et Scheithauer (2002, [10]) choisissent initialement les variables optimales puis après chaque fixation ils les mettent à jour : les nouvelles valeurs sont une combinaison linéaire entre l'ancienne valeur et un terme correctif représentant la consommation de matériel normalisée.

Cette heuristique peut être appelée en dehors de l'algorithme de génération de colonnes et l'étape 0 peut simplement être le choix d'une colonne parmi un ensemble connu. Pour le problème d'optimisation du traitement de l'ordre de fabrication dans l'industrie textile, Cimelière (2004, [25]) a généré son ensemble de colonnes dans une phase de préprocessing par énumération (exhaustive ou non) des placements possibles. Puis à l'étape 0, il choisit la colonne qui permet de découper le plus de vêtements.

Un cas particulier de l'heuristique gloutonne est lorsque le problème de pricing ne retourne pas de colonne réalisable, alors une étape 0bis est ajoutée qui consiste à rendre réalisable la colonne obtenue. Un autre cas plus particulier de l'heuristique gloutonne est lorsque la solution est formée d'une unique colonne, alors les étapes 2 et 3 ne sont plus nécessaires. Ces deux cas particuliers sont rencontrés dans la décomposition en un 1-arbre du problème de TSSP avec contrainte de sac-à-dos proposée par Göthe-Lundgren et al (2005, [54]).

### 2.4.3 Heuristiques d'arrondi

L'heuristique d'arrondi consiste à obtenir une solution entière réalisable en arrondissant la solution $PL$. C'est une heuristique de plongeon ("diving heuristics"). Cela peut être vu comme un plongeon dans l'arbre d'énumération, cependant la règle de branchement sous-jacente n'est pas celle que l'on utiliserait pour résoudre le problème exactement. Dans sa forme basique, l'heuristique d'arrondi ("Rounding Heuristic") peut se présenter en 4 étapes définies dans le schéma de la table 2.2.

| | |
|---|---|
| **RH :** | ***Etape 0 :*** Résolution du problème maître continue. |
| | ***Etape 1 :*** Fixation d'une colonne à une valeur entière. |
| | ***Etape 2 :*** Mise à jour du maître et du problème de pricing. |
| | ***Etape 3 :*** Si les critères d'arrêt sont faux alors retour à l'étape 0, sinon STOP. |

TAB. 2.2 – Algorithme de l'heuristique d'arrondi

Les critères d'arrêt sont : une solution est trouvée ou le problème maître est irréalisable ou la fonction objectif est moins bonne qu'une solution connue. De nombreuses variantes à partir de ce schéma permettent d'obtenir différentes versions.

L'étape 0 peut dépendre de l'itération à laquelle on se trouve. Initialement le problème maître continu est résolu à l'optimum (ou presque comme dans [49]). Ensuite lors de la réoptimisation du maître résiduel, la résolution peut être tronquée mais en générant toujours des colonnes [82], ou faire le choix de le résoudre sans générer de nouvelles colonnes si cela ne semble pas nécessaire [13]. Les paramètres de réoptimisation du maître (fréquence et nombre d'itérations) ont un impact sur le temps de calcul et sur la qualité de la solution.

En résolvant peu fréquemment le maître et en tronquant la génération de colonnes, l'heuristique est rapide cependant, avec un algorithme plus coûteux qui résout fréquemment le maître sans le tronquer, il y a plus de chances de trouver des solutions et d'avoir de meilleures solutions. Différents solveurs peuvent être utilisés pour résoudre le maître continu, par exemple par la méthode du simplex (le plus courant) ou par d'autres algorithmes comme la méthode de bundle inexacte [60] ou la méthode dual ascent coordonnée [13]. Dans [52], les auteurs utilisent l'algorithme de volume pour avoir la solution $PL$ optimale, mais avant la fixation des variables le problème est résolu à nouveau en utilisant le simplex car la solution obtenue est moins fractionnaire.

A l'étape 1, la colonne fixée à une valeur entière peut être choisie comme celle ayant la plus grande valeur [82], ou selon un critère glouton [13], ou aléatoirement [63]. La fixation déterministe est la plus fréquente. Comme pour les branchements, fixer une colonne à 0 est peu efficace et invoque un remaniement du problème de pricing. En général, les colonnes sont arrondies à l'entier inférieur et l'une d'elle est fixée à l'entier supérieur [69], ou plusieurs d'entre elles [60, 10, 49, 52]. Dans ce dernier cas, les colonnes sont fixées soit itérativement, tant que le maître reste réalisable [10, 60], soit indépendamment, celles dont la valeur est supérieure à un seuil [49, 52], soit toutes les colonnes d'un ensemble $\hat{Q}$ [52]. Dans le cas où $\lambda_q \in \{0, 1\}$, $\hat{Q}$ doit par exemple satisfaire $\sum_{q \in \hat{Q}}(1 - \lambda_q) < 1$, cela permet de fixer beaucoup de colonnes qui ont une valeur proches de 1 et quelques unes plus éloignées. La règle de branchement utilisées sous-jacente à ces fixations est très restrictive. Au lieu de fixer des colonnes, on peut aussi choisir de fixer des variables pures du maître [52] (i.e. des variables du maître non liées aux colonnes) ou des variables du problème de pricing [30, 52]. Dans [52], cette dernière fixation utilise la règle de branchement de Ryan-Foster. Ensuite la colonne fixée peut être choisie parmi toutes les colonnes de la solution $PL$, le plus souvent, ou toutes les colonnes compatibles déjà générées. Parfois certaines d'entre elles sont éliminées, par exemple celles avec une très petite valeur primale ou selon un autre critère. Dans [13], l'auteur attribue un score à chacune des colonnes. Il base le score sur le coût réduit moyen et la non satisfaction de contraintes. Ensuite il améliore l'information du score par une technique de probing permettant de calculer des pénalités : il tente de fixer des variables et regarde la conséquence sur la réoptimisation du dual $PL$. Ensuite les colonnes ayant un grand score sont éliminées.

A l'étape 3, la solution partielle peut être complétée à n'importe quelle itération par une heuristique gloutonne [10, 60] ou une méthode exacte [30].

Si les critères d'arrêt sont satisfaits, une remise en doute d'une fixation peut être faite, de la même façon qu'on peut faire du "backtracking" dans l'arbre de Branch-and-Bound, en changeant la valeur d'une ou plusieurs variables puis en retournant à l'étape 2 [49, 10, 60, 30]. Le fait de compléter la solution partielle par une heuristique gloutonne et de remettre en cause une fixation, est une façon de mettre en œuvre une procédure de diversification consistant à démarrer le glouton avec des points de départ différents. Ainsi l'étape 3 avec ces deux modifications devient l'étape combinée de la table 2.3.

| | |
|---|---|
| ***Etape 3 :*** | ***"GH+backtracking"*** |
| | (a) Appel de l'heuristique gloutonne pour compléter la solution partielle. |
| | (b) Remise en cause des valeurs des variables fixées à l'étape 1 et retour à l'étape 2 du RH. |

TAB. 2.3 – Etape 3 combinée de l'heuristique d'arrondi

Enfin, la fréquence d'appel de l'heuristique va aussi avoir un impact : elle peut être appelée au cours du processus de génération de colonnes [30], avant chaque ajout de coupes [10] ou seulement sur la solution $PL$ optimale [52], à différents nœuds de l'arbre ou seulement à la racine. L'heuristique d'arrondi peut aussi être appelée successivement sur une même solution $PL$ [82] : la première variable fixée va donner une direction de plongeon. Ainsi pour diversifier les différentes passes (i.e. les différents appels sur la même solution $PL$), chaque plongeon est initialisé en fixant une variable différente.

Remarquons qu'en limitant le nombre d'itérations de la génération de colonnes (noté nbIterCG) de l'étape 0, l'heuristique d'arrondi commence à ressembler à l'heuristique gloutonne de la section 2.4.2. Si nbIterCG=0 et que la colonne fixée est choisie parmi l'ensemble des colonnes déjà générées, l'heuristique d'arrondi devient l'heuristique gloutonne utilisée dans [25]. Si nbIterCG=1 et que la colonne fixée est la colonne qui vient d'être générée sans mise à jour des variables duales, l'heuristique d'arrondi devient l'heuristique gloutonne utilisée dans [69].

Notons que les différents choix effectués dépendent souvent de l'application, et une variante performante sur un problème peut ne pas l'être sur un autre. Par exemple, dans [49] plusieurs colonnes sont fixées simultanément, ce

n'est pas toujours possible si la fixation d'une colonne rend d'autres colonnes incompatibles (i.e. deux colonnes ne pourront jamais avoir une valeur entière strictement positive dans une solution), comme c'est le cas pour notre problème. Nous proposons ci-dessous une revue des différentes versions utilisées par application.

Pour le problème de découpe, différentes versions de l'heuristique d'arrondi ont été utilisées. Sur différentes variantes du problème, Perrot (2005, [69]) compare, à la racine, l'heuristique d'arrondi sur la solution $PL$ optimale avec plusieurs passes et l'heuristique gloutonne vue à la section précédente. Les meilleures bornes primales sont obtenues par l'heuristique d'arrondi. Vanderbeck (2000, [82]) appelle cette heuristique à la racine et avant d'examiner un nouveau nœud pour aider la procédure de Branch-and-Bound en ayant une bonne borne primale. Les auteurs de [10, 60] s'arrêtent au nœud racine de l'arbre de Branch-and-Price et utilisent l'heuristique d'arrondi pour trouver des solutions entières. Après le plus grand nombre de fixations possibles à l'étape 1, ils utilisent l'étape 3 "GH+backtracking" de la table 2.3. Dans la phase (b) de remise en cause, certaines valeurs des variables fixées à l'étape 1 sont décrémentées. La taille du problème résiduel est de plus en plus grande. Belov et Scheithauer (2002, [10]) résolvent le problème de cutting stock avec plusieurs types de rouleaux en combinant la méthode des plans coupants et l'approche de génération de colonnes. L'heuristique d'arrondi les aide à trouver une solution optimale plus rapidement ; elle est appelée avant chaque ajout de coupes. Kiwiel (2005, [60]) teste différentes heuristiques constructives pour la phase (a) de l'étape 3 combinée : le "First Fit Decreasing" et l'heuristique gloutonne vue à la section précédente. Les variantes utilisant l'heuristique FFD à la phase (a) résolvent la plupart des instances industrielles de la littérature.

Pour calculer une borne supérieure au problème de production par lot avec capacité et temps de mise en route, Degraeve et Jans (2003, [30]) appellent une heuristique d'arrondi après chaque étape de pricing. Dans leur version, ils fixent à l'étape 1 toutes les variables de set-up qui appartiennent au sous-problème. Pour arrondir les variables de set-up, ils choisissent un point de coupe : si la valeur de la solution agrégée est supérieure à ce point alors on arrondit à 1, sinon à 0. Une fois que les variables de set-up sont fixées, une solution peut être trouvée en résolvant un problème de flot qui détermine les quantités produites optimales pour ce choix de set-up. Ils utilisent l'étape 3 "GH+backtracking" de la table 2.3. Pour remettre en cause certaines fixations, ils changent de point de coupe.

Günlük et al. (2005, [52]) présentent les composantes d'optimisation d'un système d'aide à la décision pour le problème du fournisseur de voitures avec chauffeur. Le système aide à établir le programme de conduite du chauffeur sur la journée pour satisfaire la demande des clients. Ils décrivent 3 modes : le mode offline qui planifie sur un jour, le mode continue qui planifie tous les quart d'heure en prenant en compte les dernières données (comme l'annulation, ou la création de nouvelles commandes) et le mode instantané qui met la solution à jour en fonction de la réalité. Pour les deux premiers modes ils proposent une formulation dont la relaxation linéaire est résolue par génération de colonnes. Une solution primale est obtenue à l'aide d'une heuristique d'arrondi ("Fix-and-Price heuristic"). Dans leur heuristique, l'appel au générateur de colonnes se fait après plusieurs fixations (lorsque la valeur du $PL$ s'est détériorée). De plus, plusieurs types de fixation sont possibles pour l'étape 1. Initialement, ils fixent à 1 toutes les colonnes ayant une valeur supérieure à 0.99, et les variables pures du maître supérieure à 0.95. Ensuite, pendant un certain nombres d'itérations ils fixent des variables du sous-problème (2 tours doivent obligatoirement se suivre dans un planning). Les fixations suivantes sont toutes les colonnes d'un ensemble $\overline{S}$ tel que $\sum_{i,j\in\overline{S}}(1-x_{ij}) < 1$, où $x_{ij} = 1$ si le chauffeur $i$ effectue le planning de la colonne $j$. La résolution du deuxième mode est limitée à 15min, comme la résolution du $PL$ est plus lente, l'heuristique est adaptée en prenant en compte le temps restant. Le temps de réponse du troisième mode doit être de 15s, une heuristique de recherche locale modifie la solution pour prendre en compte les retards, les demandes dans le planning réel.

Les deux articles suivants présentent une méthode heuristique basée sur la génération de colonnes pour établir le planning mensuel personnalisé des membres de l'équipage dans les transports aériens [49] ou pour planifier le service dans le transport public [13]. Gamache et al (1999, [49]) obtiennent une solution entière en explorant partiellement l'arbre d'énumération avec la stratégie en profondeur d'abord : la règle de branchement utilisée correspond à des fixations de colonnes. Ils s'arrêtent dès qu'une solution entière correcte est trouvée. Lorsque la valeur du $PL$ ne s'améliore plus pendant un certain nombre d'itérations, un nouveau nœud est créé en fixant à 1 toutes les colonnes ayant une valeur $\geq 0.6$, et s'il n'y en a pas la colonne de plus grande valeur est fixée à 1 (cela est fait à l'étape 1). Le fait de "brancher" plus tôt (i.e. de fixer des colonnes alors que la solution $PL$ n'est pas prouvée être optimale) permet d'accélérer l'algorithme. Ensuite, des fixations peuvent être remises en cause par backtracking, mais pratiquement ils observent qu'aucun backtracking n'est fait car la première solution trouvée est acceptable. L'efficacité de l'algorithme

est due à plusieurs particularités de leur problème, comme le fait que le membre de droite de la contrainte de recouvrement des services est plus grand que 1, que chaque employé peut être affecté à presque tous les services et que les services non couverts pourront l'être par des équipes supplémentaires. Borndorfer et al (2001, [13]) présentent une heuristique de plongeon dans laquelle la fixation d'une colonne (à l'étape 1) se fait en 2 phases. La première phase sélectionne un sous ensemble de 20 colonnes ayant les plus petits scores basés sur le coût réduit moyen et la non satisfaction de contraintes. La deuxième phase calcule par la technique de probing la pénalité de chacune de ces colonnes, celle ayant la plus petite pénalité est fixée à 1. Le générateur de colonnes est appelé si l'objectif du dual $PL$ restreint n'est pas une approximation acceptable du problème linéaire maître global.

### 2.4.4 Recherche locale et méta-heuristique

La recherche locale peut aussi être implémentée dans le cas de la génération de colonnes. La recherche locale pure consiste à explorer le voisinage d'une solution courante pour y rechercher une meilleure solution et réitérer la procédure autour de cette nouvelle solution. Les méta-heuristiques mettent en œuvre les techniques de diversification pour ne pas rester bloqué dans un extremum local et d'intensification pour mieux explorer le voisinage d'une solution prometteuse.

L'heuristique d'arrondi appelée itérativement, soit en faisant du backtraking [60, 30], soit en faisant plusieurs passes [82], peut être vue comme une recherche locale qui explore le voisinage de la solution $PL$ en faisant différents plongeons (parcours du voisinage). Dans ce cas, la solution initiale $S_{PL}$ est la solution $PL$. Cette première solution $S_{PL}$ permet d'obtenir une solution entière $S$ caractérisée par l'ensemble $F$ contenant la ou les premières fixations effectuées. Une solution voisine de $S$ est une solution entière obtenue en ayant modifié l'ensemble $F$, i.e. en ayant enlevé la fixation de certaines colonnes et/ou en ayant refixé d'autres. Le voisinage est exploré heuristiquement, à l'aide soit d'une heuristique de type glouton [60], soit de l'heuristique d'arrondi [82], ou il est exploré exactement [30]. La diversification se fait en appelant l'heuristique d'arrondi sur différentes solutions $PL$ (obtenues au cours du processus de génération de colonnes ou à différents nœuds). Cependant, le voisinage d'une solution $PL$ n'est pas celui d'une solution entière.

Nous présentons maintenant un algorithme de base de recherche locale à l'image de celui développé par Cimelière (2004, [25]). Ce dernier propose un recuit simulé basé sur la génération de colonnes pour le problème d'optimisation du traitement de l'ordre de fabrication dans l'industrie textile. Les différents éléments de l'algorithme de recherche locale sont :

**(i)** la définition d'une solution $S$ et son ensemble $\hat{Q}$ de colonnes associées ;

**(ii)** la définition d'une solution voisine $(S', \hat{Q}')$, typiquement obtenue par échange de colonnes, i.e. $\hat{Q}' = \hat{Q} \backslash Q1 \cup Q2$ ;

**(iii)** l'exploration du voisinage de $(S, \hat{Q})$.

Le schéma de l'algorithme de recherche locale est celui de la table 2.4.

| | |
|---|---|
| **LS :** | ***Etape 0 :*** Obtention d'une solution initiale $(S, \hat{Q})$. |
| | ***Etape 1 :*** Faire : |
| | (a) Exploration du voisinage de $(S, \hat{Q})$ et obtention de la solution $(S', \hat{Q}')$. |
| | (b) Si $S'$ est meilleure que $S$ alors $(S, \hat{Q}) = (S', \hat{Q}')$ et retour en (a), sinon STOP. |

TAB. 2.4 – Algorithme de recherche locale pour une approche de génération de colonnes

A l'étape 0, la solution peut être obtenue à l'aide d'une des précédentes heuristiques. L'étape 1 est l'étape d'amélioration itérative et s'arrête lorsqu'aucune meilleure solution n'a été trouvée durant l'exploration (heuristique) du voisinage. Pour diversifier la recherche locale, l'algorithme LS peut être appelée sur différentes solutions de départ, par exemple les solutions obtenues par les différentes passes de l'heuristique d'arrondi. Enfin cette heuristique LS peut servir de base à un méta-heuristique. Dans [25], Cimelière a développé un recuit simulé sur cette base.

La définition du couple $(S, \hat{Q})$ dépend de la méthode utilisée pour obtenir la solution $S$. $\hat{Q}$ peut être l'ensemble des colonnes à partir duquel $S$ est obtenue ($\hat{Q}$ est défini avant $S$). Dans [25], Cimelière obtient $S$ en résolvant le problème maître en nombre entier sur $\hat{Q}$ à l'aide d'une heuristique qui arrondit inférieurement les variables de la solution $PL$ et complète la solution partielle d'une manière gloutonne. $\hat{Q}$ peut aussi être l'ensemble des colonnes utilisées

dans la solution $S$. Si pour obtenir la solution $S$ des colonnes sont générées (par exemple en utilisant l'heuristique d'arrondi, ce qui est notre cas), alors $S$ est définie en même temps que $\hat{Q}$ (on ne sait pas à l'avance quelles colonnes vont être générées). Dans les deux cas, l'ensemble voisin $\hat{Q}'$ est initialisé en supprimant d'abord des colonnes de $\hat{Q}$ (on obtient $Q1$). Ensuite dans le premier cas, des colonnes sont ajoutées à $\hat{Q}'$, i.e. $\hat{Q}' = \hat{Q} \backslash Q1 \cup Q2$ et $S'$ est calculée à partir de $\hat{Q}'$. Dans [25], $\hat{Q}'$ est obtenu en remplaçant aléatoirement une colonne de $\hat{Q}$ par une autre. Le voisinage est exploré heuristiquement : la meilleure solution $S'$ est obtenue en résolvant heuristiquement le problème maître en nombre entier sur $\hat{Q}'$. Par contre, dans le deuxième cas, l'ensemble les colonnes initiales de $\hat{Q}' = \hat{Q} \backslash Q1$ définissent une solution partielle $S'$. Cette solution est complétée en ajoutant des colonnes à $Q2$, puis $\hat{Q}' = \hat{Q}' \cup Q2$. L'exploration du voisinage de $S$ se fait heuristiquement et consiste à trouver la meilleure solution voisine $S'$ en essayant quelques échanges de colonnes.

La recherche améliorante et de contradiction limitée ("Improved Limited Discrepancy Search") utilisée par Chabrier [17], pour le problème de conception de réseau, peut être vue comme une recherche locale. Chabrier utilise la stratégie ILDS pour parcourir l'arbre de Branch-and-Bound en un temps imparti. La stratégie ILDS permet de diversifier la recherche et explorer des parties de l'arbre différentes et prometteuses. En ayant un temps imparti, la stratégie habituelle de recherche en profondeur plonge dans une branche, la solution trouvée peut ne pas être bonne ; la recherche en largeur peut améliorer la borne duale et fermer la déviation à l'optimum mais sans trouver de solution. La stratégie originale de branchement se base sur la solution de la relaxation linéaire, une contradiction se réfère à un mouvement correspondant à une décision non préférée. La stratégie ILDS consiste à faire un parcours partiel de l'arbre en ne visitant que les nœuds correspondant à des solutions dans lesquelles le nombre de contradictions par rapport à la stratégie de branchement original est bornées.

La recherche locale peut être extérieure à la génération de colonnes mais collaborer avec celle ci en utilisant les informations venant du problème maître continu. Chabrier et al. [18] proposent une coopération entre génération de colonnes et recherche locale pour le VRP avec fenêtre de temps. Ils essaient d'améliorer la meilleure solution courante, obtenue par Branch-and-Price ou non, en enlevant certaines visites de référence. Une visite est enlevée avec une probabilité proportionnelle aux nombres de routes qui la couvrent et qui sont prises dans la dernière solution $PL$ si celle ci n'est pas entière, ou proportion-

nelle au nombre d'arcs supprimés par les règles de branchement dans lesquelles elle intervient. D'autres visites proches de ces visites de référence sont enlevées. Le voisinage consiste à réintégrer ces visites. Les deux algorithmes tirent profit de la coopération : les solutions issues de la génération de colonnes permettent de relancer la recherche locale à partir de nouvelles solutions qui peuvent être meilleures que des optima locaux et la recherche locale améliore les solutions entières issues de la génération de colonnes. De plus, les informations de la génération de colonnes guident et accélèrent la recherche locale et les solutions issues de la recherche locale permettent une diversification des colonnes prises en compte dans le problème maître.

### 2.4.5 Heuristiques basées sur la programmation entière mal adaptées à l'approche de Branch-and-Price

Dans le cas des problèmes linéaires quelconques en nombres entiers, d'autres techniques existent pour trouver une solution primale et améliorer celle ci. Citons l'approche de Fishetti et al (2005, [45]) qui, pour trouver une solution entière réalisable, utilise deux vecteurs solutions et une modification du problème original. A chaque itération, la solution $PL$ optimale est arrondie : $\widetilde{x}$ est entière mais non réalisable. Puis une solution $x_{PL}$ aussi proche que possible de $\widetilde{x}$ est cherchée en résolvant la relaxation linéaire du problème qui a pour objectif de minimiser la distance entre la solution $PL$ et $\widetilde{x}$. Du point de vue géométrique, l'heuristique génère deux trajectoires : les points $\widetilde{x}$ et les points $x_{PL}$ ayant un but complémentaire. $\widetilde{x}$ satisfait les contraintes d'intégralité et $x_{PL}$ les contraintes linéaires. Achterberg et Berthold (2005, [1]) améliorent cette heuristique en modifiant l'objectif : ils prennent une combinaison convexe de la distance et de l'objectif initial.

Pour améliorer les solutions, les concepts de recherche locale (de voisinage, d'intensification et de diversification) ont été appliqués par l'intermédiaire de l'algorithme de Branch-and-Cut. Dans le cas de problèmes binaires, Fishetti et al. (2003, [46]) définissent le voisinage $k - OPT$ d'une solution de référence $\overline{x}$ en définissant des contraintes de fixation douce basées sur la distance de Hammimg par rapport à cette solution, $\Delta(x, \overline{x}) \leq k$. Le voisinage est exploré par un solveur MIP. Si une meilleure solution est trouvée, alors elle devient solution de référence, sinon ils diversifient en élargissant le voisinage, $\Delta(x, \overline{x}) \geq k + 1$.

Danna (2004, [29]) propose un algorithme hybride basé sur la recherche locale et le Branch-and-Cut ("Relaxation Induced Neighborhood Search"). A

chaque étape, deux solutions sont connues : la meilleure solution entière réalisable et la solution fractionnaire du nœud courant. L'algorithme de RINS va explorer le voisinage de la solution entière et de la solution fractionnaire. A un nœud de l'arbre, les variables qui prennent les mêmes valeurs dans la solution entière et la solution fractionnaire sont fixées, le sous-problème est résolu par Branch-and-Cut. La diversification est obtenue par les changements de la relaxation continue d'un nœud à l'autre, le RINS est appelé seulement tous les $f$ nœuds pour éviter les voisinages semblables. Une autre stratégie simple à mettre en œuvre est le plongeon dirigé ("guided dives") qui va explorer implicitement un voisinage d'une solution entière en choisissant le nœud à explorer qui se rapproche le plus de cette solution.

Ces heuristiques sont mal adaptées à l'approche de Branch-and-Price. Elles ne sont pas compatibles avec la gestion dynamique des colonnes ; le fait que la fixation d'une colonne fixe une plus grande partie de la solution que la fixation d'une variable originale les rendent moins performantes. Dans les heuristiques de [45, 46] lorsque le problème est binaire, la distance définie par rapport à une solution entière $\hat{\lambda}$, dans le cas de la génération de colonnes, est : $\Delta(\lambda, \hat{\lambda}) = \sum_{q \in \hat{Q}}(1-\lambda_q) + \sum_{q \in Q \setminus \hat{Q}} \lambda_q$ où $\hat{Q} = \{q : \hat{\lambda}_q = 1\}$. Toutes les colonnes non encore générées auront un coefficient de 1 dans cette distance. Dans l'heuristique de [29], mettre à zéro les variables qui sont à zéro dans la solution entière connue et dans la solution $PL$ courante induit que toutes les colonnes non générées sont mises à zéro.

## 2.5 Conclusion

Dans ce chapitre sont proposées une décomposition permettant de formuler notre problème et une revue des méthodes heuristiques basées sur une approche de décomposition. Ainsi cette formulation nous permet naturellement d'avoir une borne duale, voir le chapitre 6, mais aussi une borne primale, voir le chapitre 7. Plus précisément, la section 7.3 décrit les adaptations de l'heuristique d'arrondi avec des tests de différentes variantes et la section 7.4 une heuristique d'échange local.

# 3

# Modèles de tournées pour un véhicule

Dans ce chapitre, on étudie les deux modélisations proposées dans la table 1.1 pour le problème de tournées à un véhicule et sur une période. Il s'agit de déterminer l'ensemble des clients à visiter et pour chacun d'eux optimiser les quantités ramassées tout en respectant la capacité du véhicule. Les données du problème sont définies dans la table 1.5. L'objectif consiste à minimiser les coûts des tournées qui incluent les coûts duaux venant du schéma de décomposition (lié à la relaxation des contraintes (2.2)). Ces derniers s'expriment comme une récompense associée à chaque client $i$ pour une quantité $\ell$ collectée.

Le premier modèle dans lequel des routes opérationnelles sont construites (voir section 3.1) est le modèle le plus classique qu'on retrouve comme sous-problème résultant de la décomposition de Dantzig-Wolfe des problèmes type VRP. C'est un problème de voyageur de commerce avec collecte de profit aux nœuds et contrainte de capacité. Souvent ce problème est résolu comme un problème de plus court chemin avec contraintes de ressources "Elementary Shortest Path Problem with Resource Constraint" (ESPPRC) ou une relaxation de ce dernier.

Le deuxième modèle consiste à déterminer un ensemble homogène de clients à collecter (cluster). Le coût de collecte est une estimation du coût d'une route visitant les clients de ce cluster (voir section 3.2). Cette version du problème permet de ramener la problématique de génération de tournées à un problème de sac-à-dos ou "Knapsack Problem" (KP).

## 3.1 Génération de routes opérationnelles

Dans le premier modèle, on détermine une route opérationnelle démarrant du garage, collectant les clients et finissant au dépôt (ou vidage) (voir la figure 3.1). Les quantités collectées respectent la capacité du véhicule. De plus, d'autres contraintes limitatives peuvent être prises en compte, comme la durée du trajet, la distance totale... ou encore des fenêtres de temps.

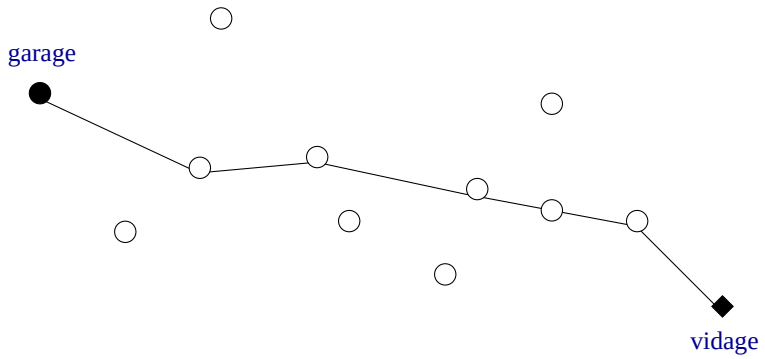



FIG. 3.1 – Route réalisable

Ce problème se modélise comme un voyageur de commerce avec collecte de profit aux nœuds et contrainte de capacité. Il peut se formuler à l'aide des contraintes (1.7-1.12) sans les indices $v$, $t$ et en posant $r_0 = 1$. Son objectif est $\max \sum_i (\pi_i r_i + \eta_i q_i) - \sum_{ij} c_{ij} z_{ij}$ où $\pi_i \in I\!R$, $\eta_i \geq 0$ sont les profits associés à la visite du client $i$ et à sa quantité collectée.

**Remarque:** Si on formule le problème en utilisant les variables $x_{i\ell}$ égale à 1 si la collecte du client $i$ couvre $\ell$ périodes, on calcule le profit associé à chacune de ces variables, $\pi_{i\ell} \in I\!R$. On retrouve les variables $r$ et $q$ en utilisant les changement de variables $r_i = \sum_\ell x_{i\ell}$ et $q_i = \sum_\ell \ell x_{i\ell}$.

Si la capacité du véhicule est non restrictive, le problème se réduit en un problème de voyageur de commerce avec profit, "Travelling Salesman Subtour Problem" (TSSP), présenté en section 3.1.1. Dans le cas où l'objectif consiste à minimiser les coûts des tournées moins les récompenses, le problème peut être résolu comme un problème de plus court chemin élémentaire.

Si la capacité du véhicule est restrictive, le chemin doit respecter cette capacité et un tel chemin est appelé une $q-$route élémentaire. Ce concept de

$q-$route a été utilisé par Christofides, Mingozzi et Toth (1981) pour calculer une borne inférieure au problème de tournées de véhicules [43]. Cependant ils ont relaxé le fait qu'un client doit être visité une seule fois, la contrainte de capacité empêchant de cycler indéfiniment. Un chemin non élémentaire pouvant compter des boucles, est appelé une $q-$route.

Lorsque plusieurs ressources sont considérées et limitées, par exemple le poids, le temps, la distance, ... ce problème de $q-$route élémentaire devient un problème de plus court chemin élémentaire avec contraintes de ressources. L'ESPPRC avec la considération de deux ressources : le temps et la capacité du véhicule, est le sous-problème résultant de la décomposition de Dantzig-Wolfe du problème de tournées de véhicules avec fenêtre de temps (VRPTW), [34, 18, 57, 42]. La complexité du programme dynamique pour résoudre l'ESPPRC étant exponentielle, plusieurs relaxations sont considérées pour obtenir une complexité pseudo-polynômiale.

Une première relaxation consiste à permettre la présence de boucles (chemin non élémentaire). Le problème obtenu est un plus court chemin avec contraintes de ressources (SPPRC). Cependant, l'avantage d'utiliser le sous-problème ESPPRC dans la décomposition de Dantzig Wolfe est qu'il fournit des bornes duales de meilleures qualités que le sous-problème SPPRC. Ceci permet d'avoir des plus petits arbres d'énumération dans la méthode de Branch-and-Price. En revanche, les temps de calculs peuvent devenir longs et limitent la taille des données.

Une relaxation moins forte consiste à ne permettre que des boucles de taille supérieure à $k$ : i.e. les $k-$cycles sont interdits, $2 \leq k \leq N$ (un $k-$cycle est un cycle formé par $k$ sommets distincts). Le SPPRC sans $k-$cycle est un compromis entre le SPPRC et l'ESPPRC. L'espace des solutions de l'ESPPRC est le plus petit, celui du SPPRC le plus grand.

Les algorithmes de résolution sont présentés dans la section 3.1.2 ainsi que l'adaptation du problème lorsque les quantités doivent être optimisées. Puis les résultats comparatifs de la littérature sur les instances de Solomon du VRPTW sont décrits.

### 3.1.1 Problème du voyageur de commerce avec profit

Définissons un graphe non orienté $G = (N \cup \{0\}, E)$ où $N \cup \{0\}$ est l'ensemble des sommets et $E$ est l'ensemble des arêtes. Notons $\delta(i)$ l'ensemble des arêtes incidentes au nœud $i$ et $E(S)$ l'ensemble des arêtes ayant les deux extrémités dans $S$. Les variables de décisions sont $y_i = 1$ si le client $i \in N$ est visité, zéro sinon ; $z_e = 1$ si l'arête $e \in E$ appartient à la route. Notons $z(J) = \sum_{e \in J} z_e$. Une formulation possible pour le problème du voyageur de commerce avec profit est la suivante :

$$\begin{aligned}
\min \sum_e c_e z_e \quad &- \quad \sum_i \pi_i y_i && && (3.1)\\
z(\delta(i)) \quad &= \quad 2\, y_i && \forall i \in N && (3.2)\\
z(\delta(0)) \quad &= \quad 2 && && (3.3)\\
z(E(S)) \quad &\leq \quad \sum_{j \in S \setminus \{i\}} y_j && \forall S \subseteq N,\ i \in S && (3.4)\\
y_i \quad &\in \quad \{0,1\} && \forall i && (3.5)\\
z_e \quad &\in \quad \{0,1\} && \forall e. && (3.6)
\end{aligned}$$

L'objectif (3.1) minimise les coûts des arêtes moins les profits $\pi_{i\ell} \in I\!R$ associés à chaque sommet. Les contraintes (3.2) garantissent que si le sommet $i$ est visité alors 2 arêtes doivent être incidentes à $i$. La contrainte (3.3) assure qu'un véhicule démarre et finit au dépôt. Les contraintes (3.4) empêchent les sous-tours.

Feillet et al. (2004, [41]) proposent une classification et une étude pour cette classe de problèmes. Trois problèmes génériques composent cette classe : le problème de tour profitable (3.1-3.6), "Profitable Tour Problem" (PTP), minimise le coût du trajet moins les récompenses ; le problème d'orientation, "Orienteering Problem" (OP), maximise le profit, le coût du trajet étant borné supérieurement dans une contrainte et enfin le problème de collecte de prix, "Prize collecting TSP", minimise le coût du trajet, la collecte des prix étant bornée inférieurement dans une contrainte. Ces problèmes sont $NP-$difficiles puisqu'une instance de TSP peut être transformée en une instance de TSSP en définissant de très grands profits sur chaque sommet.

Pour résoudre ces problèmes on retrouve les méthodes exactes de Branch-and-Cut et Branch-and-Bound basées sur différentes relaxations (la relaxation des contraintes de sous-tours (3.4) mène à un problème d'affectation, la relaxation des contraintes de degré des nœuds (3.2) mène à un 1-arbre de recouvrement) ; et les heuristiques (d'insertion, d'extension de chemin...) et méta-

heuristiques.

Notre objectif est le même que celui du PTP, ce dernier étant souvent rencontré comme sous-problème dans le schéma de génération de colonnes pour les problèmes de tournées. Dans ce cas les profits correspondent aux valeurs duales associées aux contraintes de recouvrement des visites des clients. Souvent, le PTP est résolu comme un problème de plus court chemin élémentaire entre deux copies du dépôt. Avec la contrainte de sac-à-dos, le problème de plus court chemin élémentaire devient le problème de plus court chemin élémentaire avec contraintes de ressources (ESPPRC).

### 3.1.2 Résolution du problème de plus court chemin élémentaire avec contraintes de ressources

Définissons un ensemble de ressources $R$ et un graphe pondéré $G = (N^+, A)$ avec $N^+ = N \cup \{0, 1\}$ l'ensemble des sommets et $A$ l'ensemble des arcs. Chaque sommet $i \in N$ a un profit $\pi_i$ (ce profit représente une récompense de visite moins un coût de service et peut prendre des valeurs négatives) et consomme $d_i^r \geq 0$ unités de ressource $r$ et chaque arc $(i, j)$ a un coût $c_{ij}$ qui peut aussi être la consommation d'une ressource (par exemple le temps). Un plus court chemin élémentaire respectant les contraintes de capacité pour chaque ressource est recherché entre le sommet 0 et le sommet 1. Un intervalle de valeurs $[a_i^r, b_i^r]$ borne l'accumulation de ressource $r$ au sommet $i$ (cet intervalle s'interprète comme une fenêtre de temps si la ressource est le temps).

Dans notre cas, deux ressources sont considérées : la capacité du véhicule et le temps, mais la ressource cumulée n'est bornée qu'au sommet terminal, noté 1, (il n'y a pas de fenêtre de temps sur les sommets intermédiaires). Comme les quantités collectées pour chaque client doivent être optimisées, un graphe étendu dans lequel les sommets sont dupliqués est construit : pour chaque client $i \in N$, $tmax_i$ sommets sont définis. Le sommet $(i, \ell)$, pour $1 \leq \ell \leq tmax_i$, a un profit $\pi_{i\ell}$, consomme $\ell d_i$ unités de capacité et $f_i$ unités de temps (voir les notations de la table 1.5). Pour chaque client $i$, un seul de ces sommets au plus doit être visité. Les sommets 0 et 1 ont un coût et une consommation nuls. La taille du graphe résultant est pseudo-polynômiale (à cause de la duplication de chaque sommet $i$ en $tmax_i$ copies).

Ce problème peut se formuler linéairement à l'aide des variables : $\phi_{i\ell} = 1$ si le sommet $(i, \ell)$ est visité, 0 sinon ($\phi_{01} = \phi_{1\,1} = 1$) ; $z_{ij} = 1$ si le sommet $j$ est

visité juste après le sommet $i$, 0 sinon. Les variables $t_i$ et $q_i$ sont respectivement le temps de parcours accumulé par le véhicule et la quantité chargée après avoir visité le client $i$ ($t_0 = q_0 = 0$). La formulation de l'ESPPRC est la suivante :

$$
\begin{aligned}
[ESPPRC] \quad & \min \quad \sum_{i,j} c_{ij}\, z_{ij} - \sum_{i,\ell} \pi_{i\ell}\, \phi_{i\ell} && && (3.7)\\
\sum_{\ell} \phi_{i\ell} \quad & \leq \quad 1 && \forall i \in N && (3.8)\\
\sum_{j} z_{ij} \quad & = \quad \sum_{\ell} \phi_{i\ell} && \forall i \in N && (3.9)\\
\sum_{j} z_{0j} \quad & = \quad \sum_{j} z_{j\,1} = 1 && && (3.10)\\
\sum_{j} z_{ij} \quad & = \quad \sum_{j} z_{ji} && \forall i \in N && (3.11)\\
q_i - q_j + W z_{ij} \quad & \leq \quad W - \ell d_j \phi_{j\ell} && \forall i \neq j \in N^+, \ell && (3.12)\\
\ell d_i\, \phi_{i\ell} \quad & \leq q_i \leq \quad W \sum_{k} \phi_{ik} && \forall i \in N^+, \ell && (3.13)\\
t_i - t_j + C z_{ij} \quad & \leq \quad C - c_{ij} - f_j && \forall i \neq j \in N^+ && (3.14)\\
t_i \quad & \leq \quad C \sum_{\ell} \phi_{i\ell} && \forall i \in N^+ && (3.15)\\
q_i,\ t_i \quad & \in \quad I\!R^+ && \forall i && (3.16)\\
\phi_{i\ell} \quad & \in \quad \{0,1\} && \forall i, \ell && (3.17)\\
z_{ij} \quad & \in \quad \{0,1\} && \forall i, j. && (3.18)
\end{aligned}
$$

Les contraintes (3.8) sont spécifiques à notre problème : parmi les sommets $(i, \ell)$, un seul au maximum doit être visité. Les contraintes (3.9-3.11) sont celles de plus court chemin. Les contraintes (3.12-3.15) sont liées aux ressources et empêchant les sous-tours. Notons $X^{ESPPRC}$ l'ensemble des solutions satisfaisant les contraintes (3.8-3.18). La considération des fenêtres de temps se fait en remplaçant les contraintes (3.15) par :

$$
\begin{aligned}
a_i \sum_{\ell} \phi_{i\ell} \quad & \leq t_i \leq \quad b_i \sum_{\ell} \phi_{i\ell} \qquad \forall i \in N\\
0 \quad & \leq t_1 \leq \quad C
\end{aligned}
$$

La relaxation linéaire de ce problème est faible, due aux constantes $W$ et $C$ dans les contraintes (3.12) et (3.14) qui sont la linéarisation des contraintes du type :$(r_i + d_i^r - r_j) z_{ij} \leq 0$, avec $r_i$ la variable représentant la consommation de ressource au départ de $i$. Une approche par programme dynamique permet de traiter directement ces relations quadratiques. Elle est donc préférée à une approche basée sur la formulation linéaire ci-dessus.

• **Programme dynamique classique ("backward recursion")**

Soit $F(S,(j,\ell),t,q)$ le coût minimum d'un chemin commençant au sommet 0, finissant au sommet $(j,\ell)$, ayant visité une seule fois tous les clients de $S$ et ayant pris $t$ unité de temps et chargé une quantité $q$.

$$F(S,(j,\ell),t,q) = \min_{(i,\tau),(i,j)\in A} \left\{ \begin{array}{l} F(S\backslash\{j\},(i,\tau),t',q') + c_{ij} - \pi_{j\ell}, \ \text{ tel que} \\ t' \leq t - c_{ij} - f_j, \ \ q' \leq q - \ell d_j, \quad \text{et} \ \ i \in S\backslash\{j\} \end{array} \right\}$$

$$\forall\, t \leq C, \ \ q \leq W \ \ \text{et} \ \ S \subseteq N, \ j \in S, 1 \leq \ell \leq tmax_j$$

Initialement :

$$F(\emptyset,0,0,0) = 0$$

Le coût optimal est :

$$\mathcal{O}pt = \min_{t\leq C-c_{j1}, \ q\leq W \ \text{ et } \ S\subseteq N, j\in S,(j,1)\in A, 1\leq\ell\leq tmax_j} \{F(S,(j,\ell),t,q) + c_{j1}\}$$

L'ajout des fenêtres de temps se fait à l'aide d'un test supplémentaire dans la récursion : il faut que $a_j \leq t \leq b_j$.

Dans l'état $(S,(j,\ell),t,q)$, l'ensemble $S$ impose que le chemin partiel du sommet 0 au sommet $(j,\ell)$ soit élémentaire et rend la complexité exponentielle, elle est égale à $O((N\ tmax)^2 CW2^N)$. Pour cette raison on peut relaxer l'espace des états ("state space relaxation") en permettant les visites multiples, i.e. on projette l'état $(S,(j,\ell),t,q)$ sur l'état $((j,\ell),t,q)$. Le programme dynamique est modifié en ne prenant pas en compte l'ensemble $S$, sa complexité devient pseudo polynômiale, elle est égale à $O((N\ tmax)^2 CW)$. On obtient le problème de plus court chemin avec contraintes de ressources non élémentaire, SPPRC. Le SPPRC fournit une borne inférieure sur l'ESPPRC.

Dans [35], Desrosiers et al. rappellent que le SPPRC revient à résoudre un problème de plus court chemin sur un graphe étendu acyclique comportant $\sum_{i,\ell}\prod_{r\in R}(b_i^r - a_i^r + 1)$ nœuds. Le temps de calcul et la quantité de mémoire peuvent ainsi devenir vite très importants si le nombre de nœuds est grand et les fenêtres de temps larges. En revanche il existe des algorithmes d'étiquetage efficaces car ils permettent de ne pas générer tous les nœuds du graphe étendu.

Dans ce qui suit, pour alléger les notations, les sommets $(j,\ell)$ sont indexés par $i = 2,\cdots,N\ tmax$ et $\pi_{j\ell}$ devient $\pi_i$.

• **Programme dynamique de type "forward recursion" avec gestion de liste d'étiquetage**

Pour résoudre le SPPRC et l'ESPPRC, on procède comme suit. A chaque chemin partiel $P_i$ du sommet 0 au sommet $i$ est associé une étiquette : $(c_{P_i}, q^r_{P_i})$ où $c_{P_i}$ est le coût et $q^r_{P_i}$ est la quantité de ressource $r$ consommée. On cherche à prolonger les chemins partiels en générant les étiquettes correspondant aux extensions possibles de ce chemin partiel $P_i$. Seuls les chemins partiels non dominés sont gardés : un nouveau chemin partiel $P_i^2$ est éliminé s'il existe un chemin partiel $P_i^1$ tel que $c_{P_i^1} \leq c_{P_i^2}$ et pour chaque ressource $r$, $q^r_{P_i^1} \leq q^r_{P_i^2}$.

Pour la version des chemins élémentaires (ESPPRC), Beasley et Christofides (1992) adaptent l'algorithme : dans l'étiquette du chemin $P_i$ est ajoutée une ressource supplémentaire $V^j_{P_i}$ pour chaque client $j$ (appelée extra-ressource). Cette ressource est consommée lors de la visite du client, le critère de dominance est étendu avec ces nouvelles ressources.

Chabrier et al ([18], 2002) et Feillet et al ([42], 2004) utilisent cet algorithme avec les ressources supplémentaires destinées à modéliser des chemins sans cycle pour la résolution du sous-problème ESPPRC dans le schéma de décomposition du VRPTW. Feillet et al introduisent la définition de sommets inaccessibles : un sommet est inaccessible s'il est dans le chemin ou si la valeur courante d'une des ressources empêche de l'atteindre. Cette nouvelle définition ne change pas la complexité de l'algorithme mais permet de réduire fortement le temps de résolution lorsque le problème est très contraint en éliminant plus rapidement des étiquettes dominées. Prenons le cas de deux chemins $P_i^1$ et $P_i^2$ tels que $c_{P_i^1} \leq c_{P_i^2}$ et pour chaque ressource $r$, $q^r_{P_i^1} \leq q^r_{P_i^2}$, et pour chaque sommet $j$, $V^j_{P_i^1} \leq V^j_{P_i^2}$, excepté pour le sommet $l$ où $V^l_{P_i^1} = 1 \geq V^l_{P_i^2} = 0$, i.e. le chemin $P_i^1$ visite le sommet $l$, mais pas le chemin $P_i^2$. Avec la définition classique, le chemin $P_i^2$ est gardé. Cependant, il se peut qu'une ressource de $P_i^2$ empêche de visiter le sommet $l$. Dans ce cas, on sait que toute extension de $P_i^2$ jusqu'au dépôt peut être répliquée pour $P_i^1$, et que le chemin résultant de l'extension de $P_i^1$ sera meilleur que le chemin résultant de l'extension de $P_i^2$. En posant $V^l_{P_i^2} = 1$ car le sommet $l$ est inaccessible, le chemin $P_i^2$ est éliminé par dominance.

Chabrier et al proposent une amélioration dans la règle de dominance qui compare un chemin $P_i^2$ à un chemin $P_i^1$ tel que certaines visites de $P_i^1$ ne sont pas dans $P_i^2$, i.e. $V(P_i^1) \backslash V(P_i^2) \neq \emptyset$ où $V(P_i)$ est l'ensemble des visites du

chemin $P_i$. Dans la règle classique, le chemin $P_i^2$ est conservé même si chacune des valeurs de son étiquette associée est respectivement plus mauvaise que celles de l'étiquette associée au chemin $P_i^1$. Il est possible d'améliorer cette règle en estimant le gain de coût qui pourrait être obtenu en passant par certaines des visites de $V(P_i^1)\backslash V(P_i^2)$. Prenons le cas d'une seule visite de différence, $V(P_i^1)\backslash V(P_i^2) = \{j\}$, accessible (sinon $P_i^2$ peut être éliminé). Alors pour toute prolongation $P_i^{2+}$ de $P_i^2$, il est possible de prolonger $P_i^1$ en $P_i^{1+}$ de la même manière, sauf si la prolongation considérée est $j$. Supposons que $P_i^{2+}$ est la prolongation de $P_i^2$ au sommet $j$. Notons $j_p$ et $j_s$ les prédécesseur et successeur de $j$ dans $P_i^{2+}$. $P_i^2$ est moins bon que $P_i^1$ et peut être éliminé si et seulement si le coût de $P_i^{2+}$, égal à $c_{P_i^2} + c_{j_p j} + c_{j j_s} - c_{j_p j_s} - \pi_j$, est moins bon que le coût de $P_i^{1+}$, qui reste égal au coût de $P_i^1$, i.e. si et seulement si :

$$c_{P_i^{2+}} - c_{P_i^{1+}} = c_{P_i^2} + c_{j_p j} + c_{j j_s} - c_{j_p j_s} - \pi_j - c_{P_i^1} \geq 0.$$

On peut estimer un minorant de $c_{j_p j} + c_{j j_s} - c_{j_p j_s}$ (par exemple 0), noté $minCout_j$, alors $P_i^2$ est dominé par $P_i^1$ si et seulement si :

$$c_{P_i^2} - c_{P_i^1} \geq \pi_j - minCout_j$$

Chabrier se limite à deux visites de différence, car minorer le coût de la prolongation de $P_i^2$ devenant dans la pratique vite difficile.

De plus, un préprocessing est effectué pour diminuer la taille du graphe. Il consiste à éliminer l'arc $(i1, i2)$ si pour aller du sommet $i1$ à n'importe quel sommet $j$, le détour en passant par $i2$ est plus coûteux que le chemin direct entre $i1$ et $j$ : i.e. si

$$\forall\ j, \quad c_{i1\,i2} - \pi_{i2} + c_{i2\,j} > c_{i1\,j}$$

et le sommet $i$ est supprimé s'il n'est jamais intéressant de le visiter : i.e. si

$$\forall\ j1,\ j2, \quad c_{j1\,i} - \pi_i + c_{i\,j2} \geq c_{j1\,j2}$$

Enfin, ces deux auteurs proposent un préprocessing plus drastique dans lequel des arcs $(i, j)$ sont supprimés si leur coût est jugé trop grand, i.e. tel que

$$c_{ij} \geq \{\% \max_k\{c_{ik}\}, (1 + \%) \min_k\{c_{ik}\}\}$$

où % est un paramètre. Ce preprocessing peut induire l'élimination de la solution optimale. En ce sens, ce preprocessing est dit heuristique. De plus, l'algorithme peut être tronqué en posant une borne supérieure sur le nombre

d'étiquettes à chaque nœud.

- **Le SPPRC sans $k-$cycle**

Pour construire des chemins élémentaires, une extra-ressource associée à chaque sommet est ajoutée dans l'étiquette, mais l'ordre des visites n'est pas connu. Lorsqu'on élimine les $k-$cycles, on doit connaître l'ordre des $k$ derniers sommets visités. Prenons $k = 3$, supposons un chemin $P_i$ défini par $(0, j1, j2, j3 = i)$. Les sommets $i$, $j2$, $j1$ et les chemins $(j4, i)$, $(j4, j2)$ sont des extensions interdites pour $P_i$ car elles créent une boucle de taille inférieure ou égale à 3. Par contre l'extension $(j4, j1)$ peut prolonger le chemin $P_i$. Irnich et al ([57], 2003) proposent un algorithme d'étiquetage qui élimine les $k-$cycles avec $k \geq 3$ pseudo polynômial à $k$ fixé (la complexité du pire des cas augmente exponentiellement avec $k$). Ils introduisent une nouvelle règle de dominance basée sur le concept d'extension possible : un chemin $P_i$ est dit dominé par un ensemble de chemins $\mathcal{P}_i$ si chaque chemin de $\mathcal{P}_i$ domine $P_i$ selon la règle classique et si en plus chaque extension de longueur $k$ de $P_i$ peut aussi étendre au moins un chemin de $\mathcal{P}_i$.

Lorsque $k = 2$, un chemin $P_i$ ne peut pas être prolongé à $j$ s'il est le prédécesseur de $i$. Dans ce cas, pour un même état $(i, t, q)$ deux étiquettes seulement sont gardées : la meilleure et la prochaine meilleure telles que les prédécesseurs de $i$ sont différents dans les deux chemins partiels [34]. En effet, prenons 3 chemins $P_i^1$, $P_i^2$ et $P_i^3$ tels que le prédécesseur de $i$ est différent dans $P_i^1$ et $P_i^2$ et $c_{P_i^1}, c_{P_i^2} \leq c_{P_i^3}$ et pour chaque ressource $r$, $q^r_{P_i^1}, q^r_{P_i^2} \leq q^r_{P_i^3}$. L'étiquette associée à $P_i^3$ peut être éliminée car toute prolongation de $P_i^3$ au sommet $j$ tel que $j$ n'est pas le prédécesseur de $i$ dans $P_i^3$ est dominée par la prolongation de $P_i^1$ au sommet $j$ si $j$ n'est pas le prédécesseur de $i$ dans $P_i^1$, ou par prolongation de $P_i^2$ au sommet $j$ sinon. La complexité reste pseudo-polynômiale. Desrochers et al ([34], 1990) développent un algorithme primal-dual pour résoudre le SPPRC : à chaque sommet, deux ensembles d'étiquettes sont gardés, l'un est associé aux chemins réalisables, l'autre est une borne inférieure. Cet algorithme est adaptable au cas sans $2-$cycle.

- **Comparaisons pour la résolution du VRPTW**

Ces différents auteurs ont appliqué leur algorithme au sous-problème obtenu dans le VRPTW résolu par Branch-and-Price. Les résultats obtenus sont résumés dans le tableau 3.1. Les instances R1, C1, RC1 ont des fenêtres de temps plus étroites que R2, C2, RC2 et sont par conséquent plus faciles à résoudre.

Dans la classe R, les coordonnées sont générées aléatoirement, dans la classe C elles sont générées géographiquement (pour avoir des groupes compacts), et dans la classe RC elles sont hybridées entre ces deux types.

| Référence | Méthode | Série |
|---|---|---|
| Desrochers et al. [34] 1992 | SPPRC sans $2-cycle$ | 1 |
| 27/87 instances sont prouvées à l'optimum<br>29/29 instances avec 25 clients résolues, 14/29 avec 50 et 7/29 avec 100 | | |
| Chabrier [18] 2002 | ESPPRC | 1 et 2 |
| tests des algorithmes différents où la génération de colonnes est couplée ou non avec la recherche locale<br>55/56 instances avec 25 clients sont résolues à l'optimum (dont 5 instances S2 fermées)<br>41/56 avec 50 (dont 10 S2 fermées) et 17/56 avec 100 (dont 2 S2 fermées) | | |
| Irnich et Villeneuve [57] 2003 | SPPRC sans $k-cycle$, $(k=1,\cdots,5)$ | 1 et 2 |
| tests de différentes valeurs de $k$<br>preuve de l'optimum pour plus de 15 nouvelles instances si $k=3,4$ | | |
| Feillet et al. [42] 2004 | ESPPRC | 1 et 2 |
| comparaison entre SPPRC-ESPPRC à la racine avec une limite de temps de 1h<br>ESPPRC : 115/168 instances sont résolues contre 146 pour le SPPRC<br>ESPPRC : 68/168 instances sont résolues à l'optimalité contre 22 pour le SPPRC,<br>la borne ESPPRC est 99 fois plus tendue que la borne SPPRC<br>ESPPRC est 52 fois plus rapide que SPPRC (lorsque les fenêtres sont larges) | | |

TAB. 3.1 – Résultats dans la littérature pour le VRPTW résolus par Branch-and-Price, sur les instances de Solomon (R, C, RC) avec $n = 25, 50, 100$

Les oracles ESPPRC et SPPRC sans $k-$cycle avec $k \geq 3$ ont permis d'améliorer les résultats de [34] en prouvant l'optimalité pour plus d'instances en un temps raisonnable. Pour plusieurs instances, Feillet et al. [42] ont trouvé dès la racine de meilleures bornes duales que Desrochers et al. [34] au bout de plusieurs nœuds. Pour la série 2, sur les instances résolues à l'optimalité par [18] et [57], le nombre de nœuds dans l'arbre de Branch-and-Bound est en général plus petit avec l'oracle ESPPRC. Le temps de calcul peut décroître avec ce gain en nombre de nœuds, Chabrier [18] met environ 1h17 pour prouver l'optimalité de r206.50 avec 67 nœuds et Irnich et Villeneuve [57] mettent environ 6h14 avec 1615 nœuds ; mais ce n'est pas systématique, Chabrier [18] met environ 55m pour prouver l'optimalité de r203.50 avec 15 nœuds et Irnich et Villeneuve [57] mettent environ 3m avec 19 nœuds.

Plus récemment, pour résoudre le VRP par BPC, Fukasawa et al ([48], 2004) utilisent des $q-$routes sans $k-$cycle avec $k = 2, 3, 4$. Ils constatent une diminution importante de la déviation à l'optimum à la racine avec leur algorithme par rapport au pur Branch-and-Cut lorsque le nombre moyen de clients par véhicule est petit car les $q-$routes sans petits cycles sont proches des routes réalisables.

## 3.2 Génération des ensembles homogènes de clients

Le deuxième modèle consiste à déterminer un ensemble de clients (ou cluster) homogène (les clients sont dans une même région et/ou nécessitent une même fréquence de visites). Les quantités collectées doivent respecter la capacité du véhicule. Le coût de collecte est une estimation du coût d'une route visitant les clients de ce cluster. Cette version du problème permet de ramener la problématique de génération de tournées à un problème de sac-à-dos (KP). En se basant sur la littérature des problèmes de transport avec plusieurs véhicules (VRP, PVRP), deux structures de coût possibles sont dégagées pour estimer le coût de la route ; chacune mène à un problème de sac-à-dos. Ce problème de sas-à-dos est formulé pour la structure choisie. Ensuite les algorithmes de résolutions sont présentés.

### 3.2.1 Structure de coût

Le VRP peut se formuler comme un problème d'affectation généralisée non linéaire :

$$\min \sum_{v} TSP(y_v) \tag{3.19}$$

$$\sum_{i} d_i \, y_{iv} \leq W \qquad \forall v \tag{3.20}$$

$$\sum_{v} y_{iv} = 1 \qquad \forall i \tag{3.21}$$

$$\sum_{v} y_{0v} = V \tag{3.22}$$

$$y_{iv} \in \{0,1\} \qquad \forall i, v. \tag{3.23}$$

où la variable $y_{iv}$ est égale à 1 si le véhicule $v$ visite le client $i$ et $TSP(y_v)$ est le coût d'un tour optimal sur les sommets de $N(y_v) = \{i, y_{iv} = 1\}$. Si la fonction $TSP(y_v)$ est remplacée par une fonction linéaire alors on obtient un problème d'affectation généralisée linéaire avec une contrainte de sac-à-dos (3.20) pour chaque véhicule.

Fisher et Jaikumar ([44], 1981) proposent une heuristique pour le VRP basée sur cette formulation en approchant le coût $TSP(y_v)$ par $\sum_i c_{iv} y_{iv}$ où $c_{iv}$ représente le coût de prise en charge du client $i$ par le véhicule $v$. A chaque véhicule $v$ est affecté un centre $i_v$ préalablement fixé, $c_{iv}$ est donc le coût de connexion du client $i$ au centre $i_v$ et vaut dans le cas symétrique $c_{iv} = c_{0i} + c_{ii_v} - c_{0i_v} \geq 0$. Les centres $i_v$ sont soit définis par l'utilisateur, soit

des points représentatifs de $V$ cônes partitionnant le plan. Pour déterminer les points $i_v$ dans ce dernier cas, l'espace est initialement partitionné en $N$ cônes ("cône-client"), chaque cône $i$ ayant un poids $d_i$. Les $V$ cônes sont ensuite formés par un groupe contigu de cônes-client ou de fraction de cône-client tel que son poids soit égal à $\frac{\sum_i d_i}{W}$. Le point $i_v$ se situe à l'intersection de la bissectrice du cône $v$ et du cercle de rayon $0.75\frac{\sum_i d_i}{W}$. Le graphe solution a une structure étoilée autour des centres $i_v$.

Bramel et Simchi-Levi ([14], 1995) ont proposé une heuristique similaire exceptée que les centres sont choisis en même temps que l'affectation des clients aux centres. Leur heuristique est basée sur la résolution d'un problème de localisation de concentrateurs avec capacité ou "Capacitated Concentrator Location Problem" :

$$
\begin{aligned}
[CCLP] \quad \min \sum_{ij} c_{ij} x_{ij} \quad &+ \quad \sum_{j} v_j y_j && && (3.24)\\
\sum_{i} d_i \, x_{ij} \quad &\leq \quad W_j && \forall j && (3.25)\\
\sum_{j} x_{ij} \quad &= \quad 1 && \forall i && (3.26)\\
x_{ij} \quad &\leq \quad y_j && \forall i, j && (3.27)\\
x_{ij}, y_j \quad &\in \quad \{0,1\} && \forall i, j. && (3.28)
\end{aligned}
$$

Un ensemble initial $J$ de centres est déterminé, ($|J|$ est une borne supérieure sur le nombre de véhicules utilisés). Il correspond à un sous ensemble de clients. Il peut y avoir typiquement un centre par client, des regroupements peuvent être fait en se basant sur l'expérience de l'utilisateur (par exemple deux clients sont mis ensemble s'ils sont obligatoirement visités par un même véhicule). Le coût d'ouverture $v_j$ du centre-client $j$ est égale à $2c_{0j}$ et la capacité résiduelle du véhicule associé au centre $j$ est $W_j$. Si le centre est défini par un ensemble de clients $S_j$ ; dans ce cas $v_j$ est le coût d'un TSP sur $S_j$. Soit $N_j$ le tour formé par $S_j$ et le dépôt, $N_j = \{0, 1, \cdots, l-1, l, l+1, \cdots, 0\}$. Le coût de connexion du client $i$ au centre $j$ est soit un coût direct, $c_{ij} = \min_{l \in N_j}\{2c_{il}\}$ (alors une borne supérieure sur le coût de la route finale est obtenue), soit le coût de la meilleure insertion dans la route initiale, $c_{ij} = \min_{l \in N_j}\{c_{li} + c_{il+1} - c_{ll+1}\}$. Le graphe solution a une structure étoilée autour des centres.

Christofides et Beasley ([23], 1983) proposent une heuristique pour le PVRP basée sur le sous-problème $1-$median avec capacité pour chaque jour. Pour chaque jour, un centre $j$ est fixé. Les $T$ centres sont choisis pour qu'ils soient

les plus éloignés des uns et des autres. Ils sont ensuite affectés aux jours les plus attractifs. Puis pour chaque client $i$, un planning est choisi en fonction des coûts d'affectations aux centres-jours $j$ (basés sur les distances $c_{ij}$) et du respect de la capacité du véhicule. Le graphe solution a une structure étoilée pour chaque jour.

Ensuite, les auteurs obtiennent la solution réalisable, i.e. les routes utilisées, en résolvant un TSP (ou VRP) sur chaque étoile, c'est une approche "cluster first, route second". Ces différents auteurs ont donc choisi d'utiliser une structure étoilée pour obtenir leur solution et approcher le coût d'un TSP, voir la figure 3.2 pour une étoile simple.

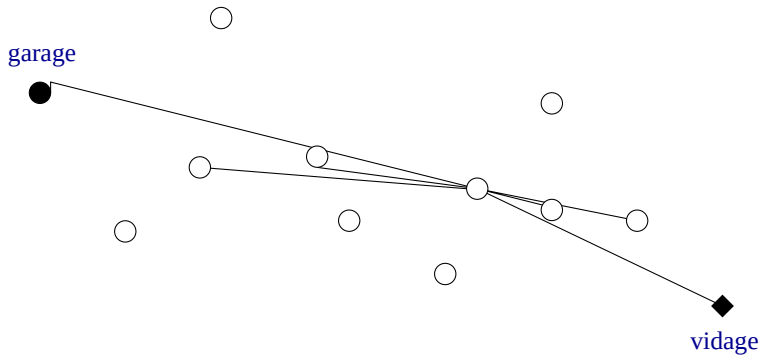



FIG. 3.2 – Cluster étoilé

Pour mesurer l'homogénéité d'un cluster de points $R$, Hansen et al. ([56], 1997) proposent différents critères. Ceux-ci se basent sur une matrice de dissimilarités $(c_{ij})$ entre ces points pour quantifier leurs différences. La matrice des dissimilarités peut être la matrice des temps entre les clients. Un de ces critères est le coût minimum d'une étoile sur $R$ : $st(R) = min_{i \in R} \sum_{j \in R} c_{ij}$. Le critère $st(R)$ mesure la plus petite somme de dissimilarités entre tout client de $R$ et les autres clients de $R$. Ainsi, la structure étoile permet d'évaluer l'homogénéité d'un cluster. De plus, Christofides et Eilon ([24], 1969) ont montré que la longueur attendue d'une route $D_0$ est monotiquement liée à la somme des distances radiales $D_r$ des clients à un centre donné $r$ : $D_0 \approx B\sqrt{a}\sqrt{D_r}$, où $B$ est une constante et $a$ est le côté du carré dans lequel se trouvent les clients.

Hansen et al. ([56], 1997) proposent d'autres critères d'homogénéité, comme la structure de clique qui mesure la somme des dissimilarités entre tous les

clients de $R$, $c_R = \sum_{i<j \in R} c_{ij}$. Graphiquement toutes les arêtes inter-clients sont comptées. Mourgaya ([67], 2004) résout le PVRP en utilisant la structure de clique pour former ses clusters car cette structure exprime l'homogénéisation en même temps que la séparation : en effet $\sum_{i<j} c_{ij} = \sum_{i<j\ intra-vehicule} c_{ij} + \sum_{i<j\ inter-vehicule} c_{ij} = cte$. Cependant le problème de transport (pour un véhicule) résultant est un problème de sac-à-dos quadratique, l'objectif étant $\min \sum_{i<j} c_{ij} y_i y_j$.

Dans notre cas, les clusters recherchés doivent être homogènes mais pas forcément séparés. De plus, dans une clique symétrique $\frac{n(n-1)}{2}$ arêtes sont comptées ce qui pratiquement a tendance à favoriser les petits clusters si le nombre de véhicules n'est pas limité.

Au vu de cette analyse, c'est la structure coût étoile qui est choisie pour notre modèle tactique. Le choix des centres est optimisé (cela fait partie des décisions à prendre dans notre modélisation). Le coût d'ouverture d'un centre $k$ est égal au coût d'un chemin du garage 0 au vidage 1 en passant par $k$, soit $c'_k = c_{0k} + c_{k1} + f_k$ ; le coût de connexion du client $i$ au centre $k$ peut être le coût direct $c'_{ik} = c_{ik} + f_i$. Cependant, pour prendre en compte les garage et vidage et permettre de connecter au cluster un client $i$ se trouvant sur le chemin du garage ou du vidage (comme illustré dans l'exemple 3.1), le coût de connexion est modifié comme dans [44] en le prenant égal au coût d'insertion du client $i$ dans le chemin du garage 0 au centre $k$ ou dans le chemin du centre $k$ au vidage 1, soit

$$c'_{ik} = c_{ik} + f_i + \min(c_{0i} - c_{0k}, c_{i1} - c_{k1}). \tag{3.29}$$

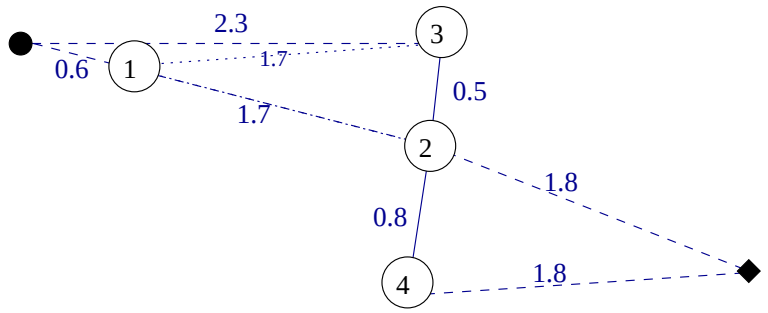
Notre structure coût étant définie, notre problème de transport peut être formulé. Lorsque le centre est fixé, le problème obtenu est un sac-à-dos à choix multiple, "Multiple Choice Knapsack Problem"(MCKP).

### 3.2.2 Problème de sac-à-dos

Le problème de transport consiste à former une étoile autour d'un centre-client tout en respectant la capacité du véhicule. Le choix d'un centre fait partie des décisions à prendre.
Posons les variables :
$z_{kk} = 1$ si le client $k$ est choisi comme centre, 0 sinon ;
$z_{ki} = 1$ si le client $i \neq k$ est connecté au centre $k$, 0 sinon ;

Exemple 3.1 – Coût de connexion

Soit la figure ci-dessous indiquant les temps entre les quatres clients et les deux dépôts.




Le coût de l'étoile $(2, 3, 4)$ est de 5.4 quelque soit le coût de connexion choisi, c'est aussi le coût d'un chemin passant par ces clients. Si le coût de connexion correspond à celui des distances, alors l'ajout du client 1 provoque une augmentation de 1.7 au coût de l'étoile. Si le coût de connexion correspond à celui de [44], alors l'ajout du client 1 ne modifie pas le coût de l'étoile et correspond au coût du chemin passant par $(1, 3, 2, 4)$.

$\phi_{i\ell} = 1$ si le client $i$ est collecté pour $1 \leq \ell \leq tmax_i$ périodes, 0 sinon.
Le problème peut se formuler comme suit :

$$[KP] \quad \max \sum_{i\ell} \pi_{i\ell}\phi_{i\ell} - \sum_{k} c'_k z_{kk} + \sum_{ki} c'_{ki} z_{ki} \tag{3.30}$$

$$\sum_{k} z_{kk} = 1 \tag{3.31}$$

$$z_{ki} \leq z_{kk} \quad \forall i \neq k \tag{3.32}$$

$$\sum_{\ell} \phi_{i\ell} = \sum_{k} z_{ki} \quad \forall i \tag{3.33}$$

$$\sum_{i\ell} \ell d_i \phi_{i\ell} \leq W \tag{3.34}$$

$$z_{ki} \in \{0, 1\} \quad \forall\, k, i \tag{3.35}$$

$$\phi_{i\ell} \in \{0, 1\} \quad \forall\, i, \ell. \tag{3.36}$$

où les coûts $c'_{ki}$ sont définis en (3.29). L'objectif (3.30) consiste à maximiser les profits $\pi_{i\ell}$ liés à la quantité collectée moins le coût d'ouverture du centre et les coûts de connexion. La contrainte (3.31) force à n'ouvrir qu'un centre et les contraintes (3.32) à ne pas connecter un client $i$ si le centre $k$ n'est pas ouvert. Les contraintes (3.33) lient les variables $\phi$ aux variables $z$. La contrainte (3.34) impose qu'on n'excède pas la capacité du sac-à-dos. Notons $X^{KP}$ l'ensemble des solutions satisfaisant les contraintes (3.31-3.36).

**Remarque:** Si dans l'objectif $\pi_{i\ell} = \ell\pi_i$ alors les variables $x_i = \sum_\ell \ell\phi_{i\ell} \in I\!N$ peuvent être utilisées à la place des variables $\phi_{i\ell}$. Dans ce cas, les contraintes (3.33) deviennent $x_i \leq tmax_i \sum_k z_{ki}$.

**Observation 3.2.1** *Lorsque le centre $k$ est fixé, i.e. $z_{kk} = 1$, le problème se réduit en un problème de sac-à-dos à choix multiple, MCKP.*

En effet, si le centre $k$ est fixé, les variables $z$ ne sont plus utiles et le problème devient un problème de sac-à-dos avec borne généralisée : pour chaque classe $i$, au plus un seul article $\ell$ doit être choisi, excepté pour le centre $k$ où exactement un article $\ell$ est choisi. Les contraintes (3.33) deviennent pour le centre $k$ :

$$\sum_\ell \phi_{k\ell} = 1$$

et pour tout $i \neq k$ :

$$\sum_\ell \phi_{i\ell} \leq 1 \quad \forall i \neq k$$

En rajoutant l'article nul (c'est à dire de profit et poids nuls) pour toutes les classes $i \neq k$, cela signifie définir les variables $\phi_{i\neq k,\ell}$ pour $0 \leq \ell \leq tmax_i$, le problème se modélise comme un sac-à-dos à choix multiple. Posons $p_{i\ell} = \pi_{i\ell} - c'_{ki}$ le profit de l'article $(i,\ell)$ et $w_{i\ell} = \ell d_i$ son poids $(w_{i0} < w_{i1} < \cdots < w_{itmax_i})$, alors la formulation du MCKP est la suivante :

$$[MCKP] \qquad \max \sum_{i\ell} p_{i\ell}\phi_{i\ell} \tag{3.37}$$

$$\sum_\ell \phi_{i\ell} = 1 \qquad \forall\, i \tag{3.38}$$

$$\sum_{i\ell} w_{i\ell}\phi_{i\ell} \leq W \tag{3.39}$$

$$\phi_{i\ell} \in \{0,1\} \qquad \forall\, i,\ell \tag{3.40}$$

La résolution de notre problème $[KP]$ se fait donc en itérant sur les centres potentiels, pour chaque centre $k$ un oracle renvoie la solution $(\phi)$ du problème $[MCKP]$ associée. Notons par $X^{MCKP}$ l'ensemble des solutions satisfaisant les contraintes (3.38-3.40).

Comme pour le problème de sac-à-dos, le MCKP peut être résolu à l'aide d'un programme dynamique. Sa complexité est $O(WN\, tmax)$.
Soit $V^i(q)$ = le profit maximum qui peut être atteint avec les articles de la classe $1, \cdots, i$ et une consommation de capacité inférieure ou égale à $q$.

Supposons, sans perte de généralité, que la classe $k$ corresponde à la classe 1. Initialement, un article de la classe 1 est choisi :

$$V^1(q) = \max_{1 \leq \ell \leq \min(\lfloor \frac{q}{w_{11}} \rfloor, tmax_1)} p_{1\ell} \quad \forall\, w_{11} \leq q \leq W$$

$V^i(q)$ est calculé par récurrence :

$$V^i(q) = \max\{V^{i-1}(q), \max_{1 \leq \ell \leq tmax_i} \{V^{i-1}(q - w_{i\ell}) + p_{i\ell},\ q - w_{i\ell} \geq w_{11}\}\}$$

$$\forall w_{11} \leq q \leq W,\ \ i \in N \backslash \{1\}$$

Le coût optimal du MCKP est :

$$\mathcal{O}pt = \max_{w_{11} \leq q \leq W} V^N(q)$$

Pisinger ([70], 1995) propose un programme dynamique basé sur la solution de la relaxation linéaire du MCKP pour énumérer un minimum de classes. On observe en effet qu'entre la solution entière et la solution $PL$, en général seules les solutions internes à quelques classes diffèrent, i.e. ce n'est pas le même article qui est choisi dans ces classes. Avant de le décrire, les particularités de la solution $PL$ optimale sont abordées ainsi que les algorithmes de résolution du $PL$ en se basant sur [38] et [70]. Dans ce qui suit, notons $N_i = \{\ell_1, \cdots, \ell_{n_i}\}$ l'ensemble des articles $\ell$ de la classe $i$ ($\ell_1$ est l'article nul pour les classes $i \neq k$ et $\ell_{n_i} = tmax_i$), et $n_i = |N_i|$.

- **Solution $PL$ du MCKP**

La solution $PL$ du problème de sac-à-dos peut être obtenue à l'aide d'un algorithme glouton. Cet algorithme est généralisable au MCKP. Posons deux propriétés.

**Propriété 3.2.2** *($PLE-$dominance)*
*Si $\ell_1, \ell_2 \in N_i$ sont tels que $w_{i\ell_1} \leq w_{i\ell_2}$ et $p_{i\ell_1} \geq p_{i\ell_2}$, alors il existe des solutions $PL$ et $PLE$ optimales avec $\phi_{i\ell_2} = 0$.*

**Propriété 3.2.3** *($PL-$dominance)*
*Soient $\ell_1, \ell_2, \ell_3 \in N_i$ tel que $w_{i\ell_1} < w_{i\ell_2} < w_{i\ell_3}$, $p_{i\ell_1} \leq p_{i\ell_2} \leq p_{i\ell_3}$ et $\frac{p_{i\ell_2} - p_{i\ell_1}}{w_{i\ell_2} - w_{i\ell_1}} \leq \frac{p_{i\ell_3} - p_{i\ell_1}}{w_{i\ell_3} - w_{i\ell_1}}$, alors il existe une solution $PL$ optimale avec $\phi_{i\ell_2} = 0$.*

La figure 3.3 illustre ces deux propriétés. Un point représente un article $\ell \in N_i$, l'abscisse est son poids $w_{i\ell}$, l'ordonnée son profit $p_{i\ell}$. La pente entre les

points consécutifs $(w_{i\ell_j}, p_{i\ell_j})$ et $(w_{i\ell_{j-1}}, p_{i\ell_{j-1}})$ (représentant deux articles adjacents), notée $\lambda_{ij} = \frac{p_{i\ell_j} - p_{i\ell_{j-1}}}{w_{i\ell_j} - w_{i\ell_{j-1}}}$, est la mesure du rapport profit-poids obtenue en choisissant l'article $\ell_j$ au lieu de l'article $\ell_{j-1}$ dans la classe $N_i$. La courbe est la frontière convexe supérieure de l'ensemble des points $(w_{i\ell}, p_{i\ell})_{\ell \in N_i}$. Les variables $\phi_{i\ell}$ dont les coefficients $(w_{i\ell}, p_{i\ell})$ ne sont pas sur la ligne sont $PL-$dominées, et celles dont les coefficients $(w_{i\ell}, p_{i\ell})$ sont en dessous de la région grisée sont $PLE-$dominées.

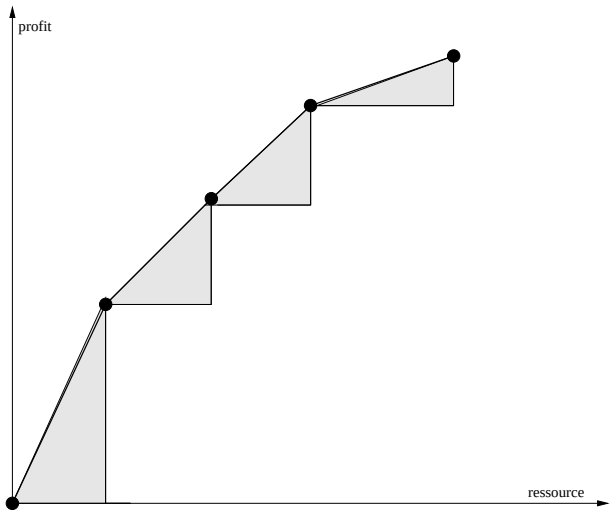



FIG. 3.3 – $PLE$ et $PL$ dominance au sein d'une classe

Au vu de la propriété 3.2.2, les algorithmes pour trouver la solution $PLE$ ne considèrent que les articles non $PLE-$dominés de la classe $N_i$. Au vu de la propriété 3.2.3, les algorithmes pour trouver la solution $PL$ ne considèrent que les articles non $PL-$dominés de la classe $N_i$.

Supposons la liste des articles non $PL-$dominés connue pour chaque classe $i$ et ordonnée telle que $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_{n_i}$, alors l'algorithme pour résoudre la relaxation linéaire du MCKP décrit dans [70] est celui de la table 3.2. Il consiste à remplir initialement le sac-à-dos avec le premier article de chaque classe. Puis tant que la capacité le permet, un échange d'article adjacent est effectué au sein d'une classe. Un dernier échange est effectué pour remplir le sac-à-dos à pleine capacité, mais seulement une fraction de l'article est échangée. Les échanges se font dans l'ordre décroissant des pentes $\lambda_{ij}$. La complexité de cet algorithme glouton pour trouver la solution $PL$ est due au tri des coefficient $\lambda_{ij}$ mesurant ces pentes et vaut $O(n \log n)$ où $n = \sum_i n_i$.

**glouton :**

- Pour chaque classe $i$, fixons $\phi_{i\ell_1} = 1$.

Soit le profit $P = \sum_i p_{i\ell_1}$ et le poids $Q = \sum_i w_{i\ell_1}$.

- Pour chaque classe $i$ et articles $\ell_j \neq \ell_1$, on définit la pente $\lambda_{ij} = \frac{p_{i\ell_j} - p_{i\ell_{j-1}}}{w_{i\ell_j} - w_{i\ell_{j-1}}}$. Les pentes $\{\lambda_{ij}\}$ sont rangées par ordre décroissant.
- Prenons la prochaine pente $\lambda_{ij}$ de $\{\lambda_{ij}\}$
- Tant que $Q + w_{i\ell_j} - w_{i\ell_{j-1}} \leq W$
  - Changeons d'article : $\phi_{i\ell_j} = 1$, $\phi_{i\ell_{j-1}} = 0$
  - $P = P + p_{i\ell_j} - p_{i\ell_{j-1}}$, et $Q = Q + w_{i\ell_j} - w_{i\ell_{j-1}}$.
  - Prenons la prochaine pente $\lambda_{ij}$ de $\{\lambda_{ij}\}$
- Si $Q = W$ alors la solution est entière ; sinon il y a deux variables fractionnaires : $\phi_{i\ell_j} = \frac{W-Q}{w_{i\ell_j} - w_{i\ell_{j-1}}}$ et $\phi_{i\ell_{j-1}} = 1 - \phi_{i\ell_j}$, appartenant à la même classe. Le profit devient $P = P + (W - Q)\lambda_{ij}$

TAB. 3.2 – Algorithme glouton pour résoudre le $[MCKP]$ continu

La solution $PL$ a la structure suivante, [70] :

**Propriété 3.2.4** *Une solution optimale $\phi^*$ à la relaxation linéaire du MCKP satisfait :*

1. *$\phi^*$ a au plus deux variables fractionnaires, $\phi^*_{f\ell}$ et $\phi^*_{f'\ell'}$.*
2. *Si $\phi^*$ a exactement deux variables fractionnaires, alors les variables appartiennent à la même classe $N_f$ et sont adjacentes (alors $f' = f$ au point 1).*
3. *Sinon la solution est entière et optimale pour le MCKP.*

Dans la solution $PL$ optimale, notons par $w_{f\ell^*}$, $p_{f\ell^*}$ le poids et le profit du premier article fractionnaire de $N_f$ et $w_{f\ell^{*\prime}}$, $p_{f\ell^{*\prime}}$ du deuxième ; et par $w_{i\ell^*}$, $p_{i\ell^*}$ le poids et le profit de l'article choisi pour toutes les autres classes $N_i$.

Dyer ([38], 1984) et Zemel (1984) ont développé indépendamment un algorithme ayant une complexité de $O(n)$ basé sur une procédure duale simplex. Graphiquement (voir la figure 3.4), une solution duale est réalisable lorsque la droite de pente $\lambda$ supporte la frontière convexe supérieure de l'ensemble des points $(w_{i\ell}, p_{i\ell})_{\ell \in N_i}$ au point $(w_{i\ell^*}, p_{i\ell^*})$ pour tout $i \neq f$, et aux points

$(w_{f\ell^*}, p_{f\ell^*})$, $(w_{f\ell^{*\prime}}, p_{f\ell^{*\prime}})$ sinon. Cette solution est primale réalisable si

$$\sum_{N_i, i\neq f} w_{i\ell^*} + w_{f\ell^*} \leq W < \sum_{N_i, i\neq f} w_{i\ell^*} + w_{f\ell^{*\prime}} \tag{3.41}$$

Leur algorithme consiste donc à trouver initialement une solution duale, i.e. une pente $\lambda$. Cette pente initiale $\lambda$ peut être la médiane (ou moyenne) des pentes initiales $\lambda_i^{init} = \frac{p_{i\ell_{jo+1}} - p_{i\ell_{jo}}}{w_{i\ell_{jo+1}} - w_{i\ell_{jo}}}$ pour chaque classe $i$, le point $(w_{i\ell_{jo}}, p_{i\ell_{jo}})$ satisfaisait $w_{i\ell_{jo}} \leq \frac{W}{N} \leq w_{i\ell_{jo+1}}$. Puis cette pente $\lambda$ est augmentée ou diminuée (par rotation) pour avoir une solution duale qui soit primale réalisable.

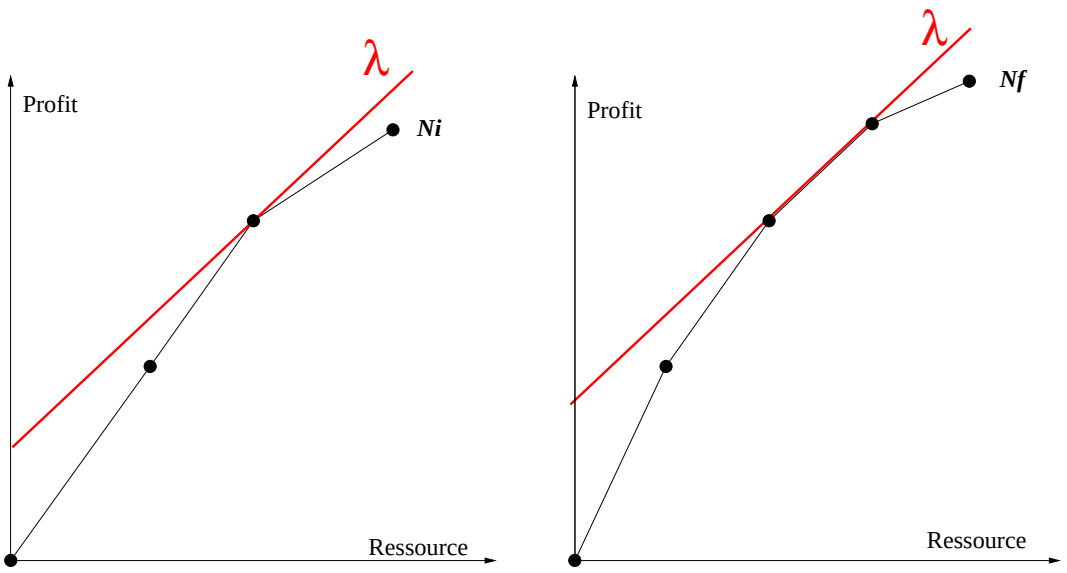



FIG. 3.4 – Solution duale

Pisinger [70] a modifié l'algorithme de Dyer pour avoir une approche primale plus facile à implémenter et de même complexité. Graphiquement (voir la figure 3.5), chaque article $\ell^* \in N_i$ maximise la projection orthogonale sur la droite passant par l'origine $O$ de pente $\lambda$. Soit $\overrightarrow{OM}$ le vecteur directeur de la droite de pente $\lambda$ (i.e. le point $M$ a pour coordonnées $(\delta_w, \delta_p)$), le repère étant orthonormé, la distance d'un point $(w_{i\ell}, p_{i\ell})$ à cette droite est égale à : $\frac{w_{i\ell}\delta_p - p_{i\ell}\delta_w}{||\overrightarrow{OM}||}$. Ainsi il s'agit de trouver $\lambda = \frac{\delta_p}{\delta_w}$ vérifiant les contraintes (3.41) et

$$\frac{p_{i\ell^*} - p_{i\ell}}{w_{i\ell^*} - w_{i\ell}} \leq \lambda \quad \forall i, \ell \tag{3.42}$$

Dans son algorithme, une pente $\lambda$ initiale est déterminée et cette pente définit une solution respectant les contraintes (3.42). Tant que la contrainte

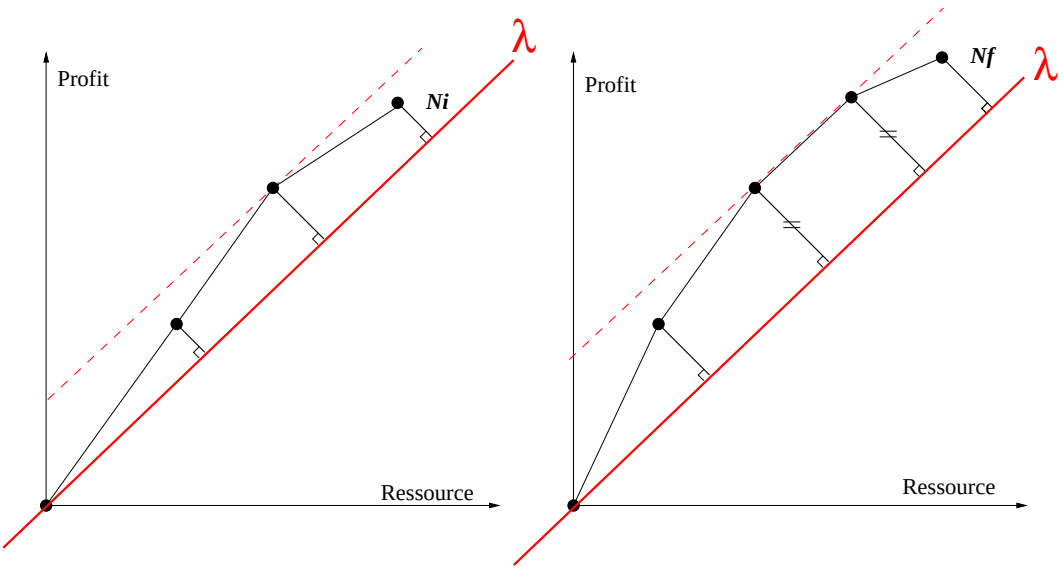



FIG. 3.5 – Vue primale

(3.41) n'est pas respectée, la pente est modifiée. A chaque modification de la pente, des articles ne pouvant se trouver dans le sac-à-dos sont supprimés. Pisinger démontre que si $\lambda$ est la médiane de $M$ pentes différentes alors à chaque étape au moins $\lceil \frac{M}{2} \rceil$ articles sont supprimés. De plus, si $M = N$ et que la taille de chaque classe est bornée par une constante alors l'algorithme s'exécute en $O(n)$. L'algorithme pour déterminer la solution $PL$ est celui de la table 3.3. Quelques notations utilisées dans la table 3.3 ont besoin d'être définies. Parmi les articles de la classe $N_i$, notons $\alpha_i$ l'article ayant le poids minimum et $\beta_i$ celui ayant le profit maximum. Pour chaque $i$, notons $L_i$ l'ensemble des articles $\ell^*$ vérifiant les contraintes (3.42). Parmi les articles de $L_i$ (il y en a un ou deux dans chaque classe), notons $\ell_g$ l'article ayant le poids minimum et $\ell_h$ celui ayant le poids maximum.

- **Programme dynamique : algorithme minimal de Pisinger [70]**

La connaissance de cette solution $PL$ est utilisée pour améliorer le programme dynamique classique. Pisinger définit le cœur $C = \{1, 2, \cdots, i_m\}$ comme l'ensemble courant des indices des classes énumérées. Soit $V_C(\overline{q})$ la solution optimale consommant au maximum $\overline{q}$ unités de capacité, $0 \leq \overline{q} \leq 2W$, pour le problème ayant pour cœur $C$ et où les variables en dehors du cœur sont fixées à leur valeur $PL$. Les capacités $\overline{q} \geq W$ doivent être acceptées dans les étapes intermédiaires car en diminuant la valeur des variables n'étant pas dans

**partition :**

*Etape 0-* $\lambda$ est la médiane de $M$ pentes $\lambda_i = \frac{p_{i\beta_i} - p_{i\alpha_i}}{w_{f\beta_i} - w_{i\alpha_i}}$.

*Etape 1-* Détermination pour chaque classe $N_i$ des ensembles $L_i$. Soit les poids $W' = \sum_i w_{i\ell_g}$ et $W'' = \sum_i w_{i\ell_h}$.

*Etape 2-* Si $W' \leq W \leq W''$ alors la partition est optimale. La solution est obtenue en prenant pour chaque classe $N_i$ l'article $\ell_g$ et tant que c'est possible on l'échange avec $\ell_h$. La première classe pour laquelle seulement une partie de l'article $\ell_h$ peut être prise est la classe fractionnaire $N_f$. STOP

Sinon, soit $W' > W$ alors la pente $\lambda$ est trop petite ; pour chaque classe $N_i$, $\beta_i = \ell_g$, et les articles $\ell \neq \beta_i$ tels que $w_{i\ell} \geq w_{i\beta_i}$ sont supprimés. Soit $W'' < W$ alors $\lambda$ est trop grand ; pour chaque classe $N_i$, $\alpha_i = \ell_h$, et les articles $\ell \neq \alpha_i$ tels que $p_{i\ell} \leq p_{i\alpha_i}$ sont supprimés. Retour à l'étape 0.

TAB. 3.3 – Algorithme pour résoudre le $[MCKP]$ continu

le cœur, la capacité du sac-à-dos peut à nouveau être respectée. Cependant soit $\overline{W} = \sum_{i \notin C} (w_{i\ell^*} - \min_{\ell \in N_i} w_{i\ell})$ le poids maximum qui peut être libéré dans les classes en dehors du cœur, alors seuls les états avec $\overline{q} \leq W + \overline{W} \leq 2W$ ont besoin d'être considérés. $V_C(\overline{q})$ correspond au problème suivant :

$$V_C(\overline{q}) = \max \left\{ \begin{array}{rcl} \sum_{i \in C} \sum_{\ell \in N_i} p_{i\ell}\phi_{i\ell} + \sum_{i \notin C} p_{i\ell^*} : & & \\ \sum_{i \in C} \sum_{\ell \in N_i} w_{i\ell}\phi_{i\ell} + \sum_{i \notin C} w_{i\ell^*} & \leq & \overline{q} \\ \sum_{\ell \in N_i} \phi_{i\ell} & = & 1 \quad \forall i \in C \\ \phi_{i\ell} & \in & \{0,1\} \quad \forall i, \ell \end{array} \right\}$$

A chaque fois que le cœur est étendu avec une nouvelle classe $N_i$, alors $V_{C+i}(\overline{q})$ peut être calculé en énumérant les différents échanges possibles :

$$V_{C+i}(\overline{q}) = \max \left\{ \begin{array}{l} V_C(\overline{q} - w_{i1} + w_{i\ell^*}) + p_{i1} - p_{i\ell^*} \\ \qquad \text{si } 0 \leq \overline{q} - w_{i1} + w_{i\ell^*} \leq 2W \\ V_C(\overline{q} - w_{i2} + w_{i\ell^*}) + p_{i2} - p_{i\ell^*} \\ \qquad \text{si } 0 \leq \overline{q} - w_{i2} + w_{i\ell^*} \leq 2W \\ \qquad \vdots \\ V_C(\overline{q} - w_{in_i} + w_{i\ell^*}) + p_{in_i} - p_{i\ell^*} \\ \qquad \text{si } 0 \leq \overline{q} - w_{in_i} + w_{i\ell^*} \leq 2W \end{array} \right\}$$

Initialement le cœur est vide et $V_\emptyset(\overline{q})$ correspond à la solution $PL$ si la capacité $\overline{q}$ est supérieure au poids de cette solution :

$$V_\emptyset(\overline{q}) = \left\{ \begin{array}{rcl} \sum_{i \in N} p_{i\ell^*} & \text{pour tout} & \overline{q} \geq \sum_{i \in N} w_{i\ell^*} \\ -\infty & \text{sinon} & \end{array} \right.$$

Le coût optimal est pour un cœur complet $C = \{1, 2, \cdots, N\}$ et une consommation de capacité inférieure $W$ :

$$\mathcal{O}pt = \max_{\overline{q} \leq W} V_C(\overline{q})$$

Pisinger a implémenté une version avec gestion de liste d'étiquetage de cet algorithme dynamique. Chaque étiquette représente un état $e = (\omega, \rho, v)$ avec $\omega$ le poids, $\rho$ le profit et $v$ le vecteur indiquant l'article pris pour chaque classe $N_i$ du cœur. Il utilise les propriétés de la solution $PL$ pour avoir un algorithme minimal et performant ; son initialisation et les tests effectués à chaque ajout d'une nouvelle classe $N_i$ au cœur $C$ sont détaillés ci-dessous.

Définissons le gradient de la classe fractionnelle $N_f$ :

$$\lambda = \frac{p_{f\ell^*} - p_{f\ell^{*\prime}}}{w_{f\ell^*} - w_{f\ell^{*\prime}}}$$

Pour toutes les autres classes $i \neq f$, définissons les gradients positifs

$$\lambda_i^+ = \max_{\ell \in N_i, w_{i\ell} > w_{i\ell^*}} \frac{p_{i\ell} - p_{i\ell^*}}{w_{i\ell} - w_{i\ell^*}}$$

et négatifs

$$\lambda_i^- = \min_{\ell \in N_i, w_{i\ell} < w_{i\ell^*}} \frac{p_{i\ell^*} - p_{i\ell}}{w_{i\ell^*} - w_{i\ell}}$$

Ces gradients mesurent respectivement le gain (la perte) par unité de poids si un article plus lourd (léger) est choisi à la place de l'article $PL$. Pisinger initialise son cœur $C$ avec la classe fractionnaire $N_f$. Puis il insère les classes $N_i$ dans $C$ dans l'ordre des gradients $\lambda_i^-$ croissants et $\lambda_i^+$ décroissants d'une manière alternative (il vérifie que la classe $N_i$ n'est pas déjà insérée vu qu'elle apparaît dans les deux listes des gradients).

Lors de l'extension du cœur $C$ avec une nouvelle classe $N_i$, le nombre d'articles est réduit en utilisant la technique de fixation par borne. Les articles $\ell$ de $N_i$ sont supprimés ($\phi_{i\ell}$ est mise à zéro) si une borne supérieure obtenue en prenant l'article $\ell$ au lieu de $\ell^*$ est moins bonne que la meilleure solution connue. La borne utilisée est celle de Dembo et Hammer ([31], 1980) : $u(\phi_{i\ell} = 1) = P - p_{i\ell^*} + p_{i\ell} + \lambda(W - Q + w_{i\ell^*} - w_{i\ell})$ (borne obtenue par relaxation Lagrangienne de la contrainte de capacité avec pour multiplicateur optimal $\lambda$), avec $P$ le profit courant et $Q$ le poids. S'il ne reste qu'un seul

article, la classe est fixée à sa valeur $PL$ et n'est pas ajoutée au cœur, et si cela est vrai pour toutes les classes non encore énumérées alors l'algorithme s'arrête.

Cette technique de fixation par borne et le principe de dominance permettent de réduire le nombre d'états. Après l'ajout de la classe $N_i$, de nouveaux états sont créés mais seulement une partie est gardée : les états non dominés et les prometteurs. Un état $e^1$ domine un état $e^2$ si $\rho_{e^1} \geq \rho_{e^2}$ et $\omega_{e^1} \leq \omega_{e^2}$. Un état $e$ est non prometteur si la borne supérieure déduite de cet état est moins bonne que la meilleure solution connue. Le cœur associé à l'état $e$ est formé des $m$ premiers gradients des listes $\lambda^+$ et $\lambda^-$, soit $\lambda_t^+$ et $\lambda_s^-$ les prochains gradients des listes, ils satisfont : $\lambda_t^+ \leq \min_{N_i \notin C} \lambda_i^+$ et $\lambda_s^- \geq \max_{N_i \notin C} \lambda_i^-$. Alors la borne supérieure déduite de cet état $e$ est égale à :
$u(e) = \left\{ \begin{array}{lll} \rho_e + (W - \omega_e)\lambda_t^+ & \text{si} & \omega_e \leq W \\ \rho_e + (W - \omega_e)\lambda_s^- & \text{si} & \omega_e > W \end{array} \right\}$

- **Méthodes de Branch-and-Bound**

Dyer et al ([38], 1984) ont développé un Branch-and-Bound basé sur la procédure duale simplex pour le calcul de la borne supérieure $Z_{PL}$. Ils parcourent l'arbre en profondeur d'abord. A chaque nœud une borne inférieure est calculée en choisissant pour la classe fractionnaire $N_f$ un article $\ell_j$ $PL-$dominé avec $w_{f\ell^*} < w_{f\ell_j} < w_{f\ell^{*\prime}}$ tel que $\sum_{i \in N, i \neq f} w_{i\ell^*} + w_{f\ell_j} \leq W < \sum_{i \in N, i \neq f} w_{i\ell^*} + w_{f\ell_{j+1}}$ pour remplir au mieux le sac-à-dos. Plusieurs stratégies de branchements sont testées, en commençant d'abord par :

1. mettre à 1 la variable fractionnaire $\phi_{f\ell^*}$
2. mettre à 0 la variable fractionnaire $\phi_{f\ell^*}$
3. ajouter la contrainte $\sum_{\ell \in N_f} \ell$ "à droite de $_{\ell_j}$" $\phi_{i\ell} = 1$

La réoptimisation pour le calcul de la borne supérieure après branchement (ajout de contraintes) ou dans la phase de retour en arrière (suppression de contraintes) se fait facilement en revoyant le liste des articles non $PL-$dominés si des articles sont mis à zéro, et en réajustant les pentes par rotations successives des droites support. De plus, avant le branchement ou à chaque sous-problème ils éliminent les articles non prometteurs en utilisant la borne de Dembo et Hammer : d'un point de vue graphique, tous les articles se trouvant à une distance verticale de plus de $Z_{PL} - INC$ de la droite de pente optimale $\lambda$ sont supprimés.

- **Tests numériques de ces deux algorithmes**

Les expérimentations de Pisinger [70] ont montré que l'algorithme minimal est très efficace : de grandes instances (avec $N = 10000, n_i = 10$ ou $N = 1000, n_i = 100$ ou $N = 100, n_i = 1000$ et les coefficients variant de 1 à 10000) sont résolues en une fraction de secondes. Seules les instances fortement corrélées demandent un effort plus important (mais toujours en moins de 30min). Pour les grandes instances non corrélées, la taille de l'ensemble des états le plus grand est de 4000. Dyer et al [38] ont testé leur algorithme sur des instances non corrélées. Les instances avec 8000 variables, des coefficients allant de 100 à 32000, n'ont pas pu être résolues car la profondeur de l'arbre était plus grande que 1500. Cependant aucune comparaison n'a été faite entre ces deux algorithmes sur un même jeu d'instances, en particulier sur des instances corrélées (tests qui nous auraient intéressé vu que nos données sont corrélées).

## 3.3 Conclusion

Finalement, les deux modélisations [ESPPRC] (3.7-3.18) et [KP] (3.30-3.36) pour le problème de tournées sont attrayantes. Pour élaborer les routes opérationnelles, nous avons choisi l'ESPPRC plutôt que le SPPRC ou le SPPRC sans $k-$cycle (avec $k = 2, 3, 4$) car dans les problèmes réels, le nombre de clients par véhicules est de l'ordre de 10, des cycles de longueur plus grand que 4 peuvent donc apparaître dans les routes. Comme nous cherchons des solutions heuristiques qui sont basées sur ces routes, les solutions obtenues risquent d'être de mauvaise qualité, car en enlevant les visites multiples, les routes deviendront sous chargées et finalement plus de véhicules seront nécessaires. Pour résoudre l'ESPPRC nous avons utilisé le programme de Feillet et al. [42] qui a été mis à notre disposition. L'adaptation à notre problème de tournées se fait en appliquant l'algorithme au graphe étendu. Les extra-ressources introduites artificiellement pour empêcher les cycles dans le parcours construit ne sont ajoutées que pour les clients $i \in N$ et non pour chaque sommet $(i, \ell)$. Un sommet $(i, \ell)$ devient inaccessible si le client $i$ est déjà dans le chemin ou si une ressource l'empêche de l'atteindre. A la différence de l'ESPPRC, le KP ne permet pas d'ajouter des contraintes opérationnelles liées au temps ne connaissant pas l'ordre réel de la tournée. Par contre, il répond aux exigences du niveau tactique en régionalisant l'espace par la formation de clusters homogènes. Lorsque le centre est fixé, le KP se réduit en un problème de sac-à-dos à choix multiple. Ce dernier est résolu en utilisant le programme dynamique de Pisinger disponible sur internet [71].

# 4

# Modèles de gestion de stock pour un client

Ce chapitre étudie la modélisation du problème d'élaboration d'un planning individuel dicté par la gestion du stock du client. Ce problème consiste à planifier les dates de collecte d'un produit chez le client. Les données ont été présentées dans la table 1.5. Selon les hypothèses tactiques retenues de la table 1.3, trois scénarios sont possibles. Dans le scénario A, on cherche une solution opérationnelle (le stock initial étant connu). Dans le scénario B, on cherche une solution sur un horizon restreint. Dans le scénario C, on cherche une solution sur un ensemble de périodicités restreint. Le planning obtenu avec ces trois scénarios est périodique. Sous les hypothèses de gestion de stock de la table 1.2, les quantités sont normalisées en nombre de périodes. Par conséquent la demande de chaque période est unitaire ; et la limitation sur le niveau de stock se traduit par un intervalle maximal entre deux visites $tmax$ et induit aussi une limitation sur la quantité ramassée.

Dans les deux premiers scénarios, l'horizon de travail est fini et le problème peut se formuler, pour le scénario A, à l'aide des contraintes (1.2-1.6). Le modèle du scénario A peut être adapté au cas du scénario B où le stock initial n'est pas connu. Les contraintes (1.2-1.6) définissent un problème de production par lot avec capacité, "Constant Capacitated Lot Sizing" (CCLS). Dans le cas des collectes, la quantité ramassée couvre les demandes des périodes passées. Les demandes sont donc rétrogrades, mais cela ne complique pas le problème (il est aussi simple que dans le cas des livraisons) ; il suffit simplement d'adapter les inégalités du problème CCLS classique en se basant sur le problème de lot sizing avec demandes rétrogrades, "Constant Capacitated Lot Sizing with

backlogged demand" (CCLSb) mais en posant les demandes non rétrogrades à 0. Les coûts de stock sont ceux issus du schéma de décomposition (en ayant relaxé les contraintes liantes (2.34)) : pour chaque période $t$, $\pi_t$ est le coût fixe de collecte, $\beta_t$ le coût unitaire de ramassage et $\gamma_t$ le coût unitaire de stockage. Enfin l'hypothèse de la politique d' "order up to level" restreint l'ensemble des solutions à une structure particulière : à chaque visite le niveau de stock devient nul et entre les visites il est strictement positif. Cette hypothèse rend le problème facile : il peut être résolu à l'aide d'un programme dynamique de complexité polynômiale. D'après [72], le problème de lot sizing du scénario A peut être reformulé comme un problème de localisation des dépôts ou un problème de flot particulier : un plus court chemin. De la même manière, nous reformulons le problème du scénario B. Dans le cas plus simple du scénario A, ces formulations ont la propriété d'intégralité, qui est perdue dans le cas du scénario B. Dans les deux cas, la taille du problème de flot est raisonnable.

Dans le cas du scénario C, l'horizon est théoriquement infini mais la connaissance de l'ensemble $P$ des périodicités des routes dans la politique restrictive (politique de la définition 1.1.4) définit un cycle de régénération de longueur maximal $T = ppcm\{p \in P\}$ : nous pouvons examiner la solution sur seulement $T$ périodes. Inclure la périodicité et la date de départ comme variables du problème de tournées fait que les variables de gestion de stock sont passées au problème de tournées et les contraintes définissant le planning individuel sont liantes (voir la section 2.3). Dans ce chapitre, nous ne donnons pas la formulation dans le cas d'un seul client mais illustrons la représentation d'un planning individuel obtenue par cette politique. Cette représentation met en évidence une symétrie due au temps, qui d'ailleurs affecte aussi le modèle du scénario B. Enfin, un exemple avec deux clients montre que le scénario C s'avère le plus adapté à notre problématique.

## 4.1 Solutions opérationnelles (scénario A)

Le premier scénario est le cas d'un horizon fini $T$ qui prend en compte un stock initial connu *tinit* et qui borne le stock final par *tinit*. Dans cette section, trois formulations sont étudiées. La première est celle du problème de lot sizing avec capacité constante (CCLS). Les deux autres sont des reformulations classiques du CCLS [72], comme un problème de localisation des dépôts et comme un problème de plus court chemin. Ces reformulations montrent que le problème est facile. Dans la section 4.1.1 est proposé un aperçu des diffé-

rentes formulations et des résultats obtenus. Ces résultats sont détaillés dans les sections suivantes.

### 4.1.1 Formulations

- **Formulation de lot sizing**

Le problème de gestion de stock du scénario A pour un client se formule comme un problème de lot sizing où les demandes sont unitaires et rétrogrades. Le stock et la quantité collectée sont bornés. De plus, à chaque collecte le niveau de stock revient à zéro. Chaque solution réalisable correspond à un flot dans le réseau de la figure 4.1. Les nœuds de 1 à $T$ correspondent aux périodes. Pour chaque nœud $t$, les flots entrants correspondent à la quantité produite à cette période (elle est égale à 1) et au niveau de stock de la fin de la période $t-1$ ; les flots sortants correspondent à la quantité ramassée à cette période $x_t$ et le niveau de stock restant à la fin de cette période $s_t$. Le nœud fictif 0 est associé aux quantités ramassées durant l'horizon. Les variables $y$ n'apparaissent pas sur la figure, mais une fois fixées, elles limitent le flot circulant sur les arcs (i.e. définissent des bornes sur les variables $x$ et $s$).

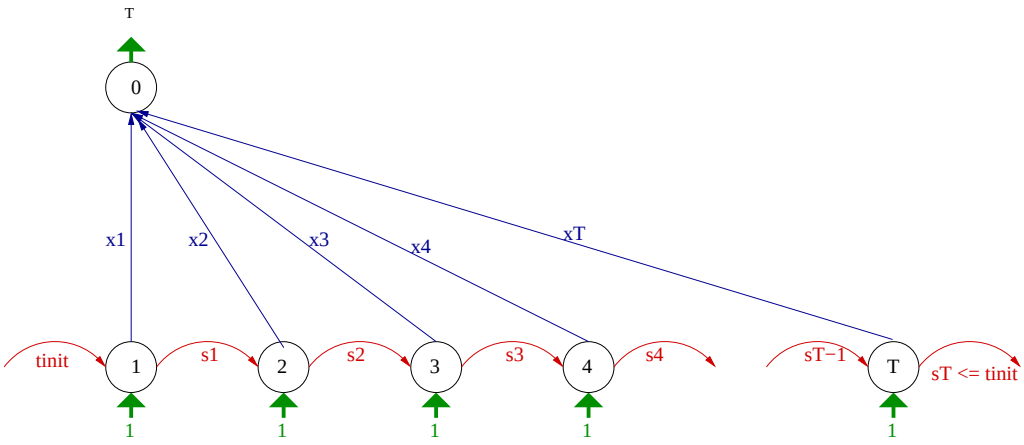



FIG. 4.1 – Modèle de planification : le stock initial est connu

Les variables de décisions sont :
$y_t = 1$ s'il y a un ramassage en période $t \in T$, 0 sinon ;
$x_t =$ la quantité ramassée en fin de période $t \in T$ ;
$s_t =$ le niveau de stock en fin de la période $t \in T$ ($s_0 = tinit$).
La formulation est la suivante :

$$[StockLSsxy] \qquad \min \sum_{t \in T} (\pi_t y_t + \beta_t x_t + \gamma_t s_t) \tag{4.1}$$

$$
\begin{array}{rcll}
s_{t-1} + 1 & = & x_t + s_t & \forall t = 1, \cdots, T \quad (4.2) \\
x_t & \leq & tmax\, y_t & \forall t = 2, \cdots, T \quad (4.3) \\
x_1 & = & (tinit + 1)\, y_1 & (4.4) \\
s_t & \leq & (tmax - 1)\,(1 - y_t) & \forall t = 2, \cdots, T \quad (4.5) \\
s_T & \leq & tinit\,(1 - y_T) & (4.6) \\
y_t & \in & \{0, 1\} & \forall t \quad (4.7) \\
x_t & \geq & 0 & \forall t \quad (4.8) \\
s_t & \geq & 0 & \forall t. \quad (4.9)
\end{array}
$$

(4.2-4.4) et (4.7-4.9) correspondent aux contraintes du problème de lot sizing avec les contraintes de conservation de flot (4.2) et les contraintes de capacité sur la quantité collectée (4.3)-(4.4). Les contraintes supplémentaires (4.5) et (4.6) sont celles issues de la politique "order up to level". Elles imposent un niveau de stock nul après chaque ramassage et borne le niveau de stock. Notons $X^{StockLSsxy}$ l'ensemble des solutions satisfaisant les contraintes (4.2-4.9).

**Observation 4.1.1** *La formulation* $[StockLSsxy]$ *utilise* $3T$ *variables et* $3T$ *contraintes.*

L'étude des problèmes classiques de lot sizing permet de :

- définir la structure des solutions extrêmes : si $y$ est entier alors les solutions $(x, s)$ extrêmes associées sont entières ;
- définir des inégalités $(k, l, I)$ adaptées aux demandes rétrogrades ;
- projeter la formulation sur un espace plus petit en supprimant soit les variables $s$, soit les variables $x$.

L'hypothèse de la politique "order up to level" permet de :

- renforcer la structure des solutions extrêmes : il y a unicité de la solution $(x, s)$ extrême associée à $y$ entier.
- définir des inégalités valides basées sur la variable $y$.

Ces résultats sont détaillés dans la section 4.1.2.

- **Formulation de localisation des dépôts**

Une première reformulation du problème $[StockLSsxy]$ présentée dans la littérature [72] est la reformulation en un problème de localisation des dépôts, “Facility Location”. Ce problème consiste dans la pratique à déterminer l'emplacement des dépôts et à affecter les sites de production à un de ces dépôts. La figure 4.2 montre une solution du problème de gestion de stock pour un client avec la formulation de localisation des dépôts. La première ligne représente les

$T+1$ sites de production ; chaque site $l = 1, \cdots, T$ a une production unitaire $d_l = 1$ et le site 0 a une production de $d_0 = tinit$, il représente le stock initial. Le site $l$ est associé au remplissage de la période $l$. La seconde ligne représente les $T+1$ dépôts qui peuvent collecter la production des sites, le $T+1^{\text{ème}}$ dépôt représente le stock final. Le dépôt $t$ est associé à la collecte de la période $t$, il y a ouverture du dépôt $t$ s'il y a collecte en $t$.

La structure des solutions due à la politique "order up to level" impose le ramassage de périodes consécutives : si $t$ ramasse le remplissage de $l-1$, alors comme le stock doit être nul en $t$, les demandes des périodes $l$ jusqu'à $t$ doivent aussi être ramassées par $t$. Inversement, si $t$ ne ramasse pas le remplissage de $l$, alors $t$ ne doit pas ramasser le remplissage des périodes précédentes. Dans tous les cas, s'il y a collecte en $t \leq T$, alors cette collecte doit automatiquement ramasser la demande de la période $t$. De même, si le stock final est non nul, i.e. le dépôt $T+1$ est ouvert, alors la demande du site $T$ doit automatiquement être collectée par $T+1$.

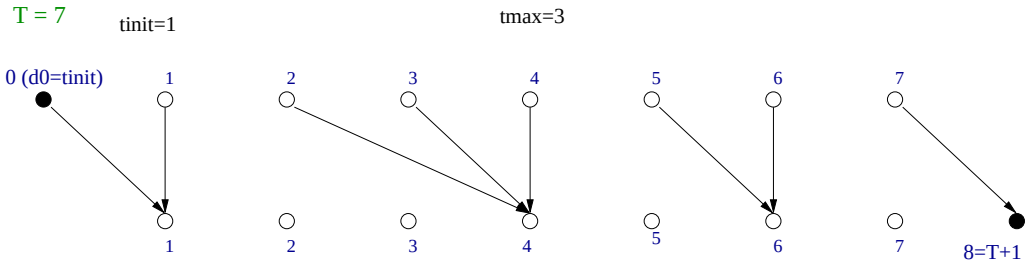



FIG. 4.2 – Un planning individuel avec la formulation de localisation des dépôts

Les variables sont définies comme suit, la capacité de stockage limitant les couples $(l,t)$ :
$w_{lt} = 1$ si le ramassage de la période $t$ couvre le remplissage de la période $l$ (i.e. le dépôt $t$ collecte la production de $l$), avec $1 \leq t \leq T+1$ et $\max(t - tmax + 1, 0) \leq l \leq \min(t, T)$, 0 sinon ;
$y_t = 1$ s'il y a un ramassage en période $t$ (ie si le dépôt $t$ est ouvert), 0 sinon ;
Soit $Cte = \sum_t (t + tinit)\gamma_t$ et $\beta'_t = \beta_t - \sum_{l=t}^{T} \gamma_l$, la formulation est la suivante :

$$[StockFL] \quad Cte + \min \sum_{t=1}^{T} (\pi_t y_t + \beta'_t \sum_{l=t-tmax+1}^{t} d_l w_{lt}) \tag{4.10}$$

$$\sum_{t=1}^{tmax-tinit} w_{0t} = 1 \tag{4.11}$$

$$\sum_{t=l}^{\min(l+tmax-1,T+1)} w_{lt} = 1 \qquad \forall\ 1 \leq l \leq T \tag{4.12}$$

$$w_{tt} = y_t \qquad \forall t = 1, \cdots, T \tag{4.13}$$

$$1 - w_{TT} = y_{T+1} \tag{4.14}$$

$$w_{lt} \geq w_{l-1\,t} \qquad \forall l \leq t \tag{4.15}$$

$$y_t \in \{0, 1\} \qquad \forall t \tag{4.16}$$

$$w_{lt} \geq 0 \qquad \forall l \leq t. \tag{4.17}$$

Les contraintes (4.11)-(4.12), (4.16), (4.17) et une variante des contraintes (4.13)-(4.14) : $w_{lt} \leq y_t \quad \forall l < t$ sont les contraintes habituelles du problème de localisation des dépôts. Les contraintes (4.11) et (4.14) restreignent le stock initial et le stock final. Les contraintes de continuité issues de la politique "order up to level" (4.15) et les égalités (4.13)-(4.14) renforcent la structure des solutions et rendent le problème facile.

**Remarque:** Avec les contraintes de continuité (4.15), la variante aux contraintes (4.13), i.e. $w_{lt} \leq y_t \quad \forall l < t$, peut être simplifiée en : $w_{tt} \leq y_t \quad \forall\ t$ ; en effet, $w_{lt} \leq \cdots \leq w_{t-1\,t} \leq w_{tt} \leq y_t$.

Les résultats classiques sont :

- la solution en $(w, y)$ peut être projetée sur l'espace $(x, y)$ ;
- si $y$ est entière alors le problème admet une solution en $w$ extrême entière.

L'hypothèse de la politique "order up to level" permet :

- de supprimer les variables $y$, la formulation obtenue est $[StockFLSimp]$ (donnée dans la section 4.1.3) ;
- d'établir que la relaxation linéaire donne une solution entière en $w$.

Ces résultats sont détaillés dans la section 4.1.3.

- **Formulation de plus court chemin**

Au vu de la structure des solutions imposée par la politique “order up to level”, le problème de gestion de stock peut aussi être reformulé comme un problème de plus court chemin, “Shortest Path Problem”, dans un graphe avec $T+2$ nœuds (voir la figure 4.3). Le nœud $t$ représente la période $t$, le nœud 0 le début de l'horizon (auquel on associe le stock initial) et le nœud $T+1$ la fin de l'horizon (auquel on associe le stock final). L'arc $(t, l)$ représente les périodes d'accumulation entre deux ramassages en $t$ et $l$, pour $l - tmax + 1 \leq t \leq l$, son coût est $s_{tl} = \pi_l + (l - t)\beta'_l$.

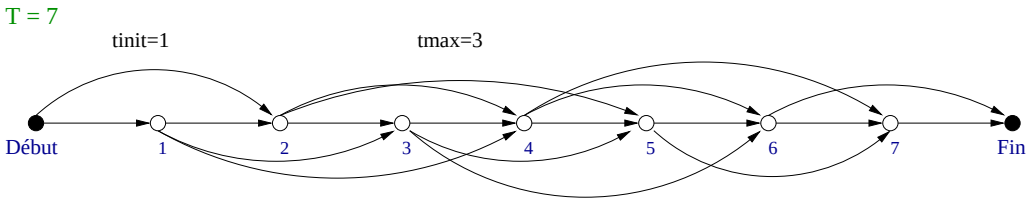



FIG. 4.3 – Graphe acyclique de la formulation en plus court chemin

Le planning individuel optimal est un plus court chemin de 0 à $T+1$. Les variables de décisions sont :
$\Phi_{tl} = 1$ si le client est ramassé en $l$, son dernier ramassage datant de $t$, 0 sinon ;
$\Phi_{0l} = 1$ si le premier ramassage a lieu en $l$, 0 sinon ;
$\Phi_{t,T+1} = 1$ si le dernier ramassage a lieu en $t$, 0 sinon.
Par conséquent le problème de gestion de stock est formulé comme suit :

$$[StockSP] \quad Cte + \min \sum_{t<l} s_{tl}\, \Phi_{tl} \tag{4.18}$$

$$\sum_{l=1}^{tmax-tinit} \Phi_{0l} = 1 \tag{4.19}$$

$$\sum_{l=\min(T-tinit,T)}^{T} \Phi_{l\,T+1} = 1 \tag{4.20}$$

$$\sum_{l=\max(0,t-tmax)}^{t-1} \Phi_{lt} = \sum_{l=t+1}^{\min(T+1,t+tmax)} \Phi_{tl} \quad \forall t \tag{4.21}$$

$$\Phi_{tl} \in \{0,1\} \quad \forall t < l. \tag{4.22}$$

**Observation 4.1.2** *La formulation* $[StockSP]$ *utilise* $T+2$ *contraintes et* $O(Ttmax)$ *variables.*

Les résultats, détaillés dans la section 4.1.4, sont :

- la solution $\Phi$ peut être projetée sur $(x, y)$ ;
- la relaxation linéaire donne une solution extrême entière ;
- ce problème peut être résolu par programme dynamique de complexité polynômiale.

• **Analyse comparative**

La formulation simplifiée de $[StockFL]$ et la formulation $[StockSP]$ sont de "bonnes" formulations (formulation compacte de l'enveloppe convexe des

solutions entières). La formulation $[StockSP]$ est préférable à la formulation simplifiée de $[StockFL]$ car il y a moins de contraintes pour un nombre de variables équivalent ; c'est donc $[StockSP]$ qui est choisie pour modéliser le problème de gestion de stock. Cependant on remarque une symétrie due au temps : le coût d'une solution peut ne pas changer si un saut de $l$ périodes est interchangé avec un saut de $t$ périodes. Par exemple une solution dans laquelle un client est collecté aux périodes 2, 4 et 7 peut être équivalente à une solution dans laquelle ce client est collecté aux périodes 2, 5 et 7 : un saut de 2 périodes a été interverti avec un saut de 3 périodes.

### 4.1.2 Modèles de lot sizing

La formulation $[StockLSsxy]$ définie en (4.1-4.9) suppose des demandes unitaires et rétrogrades, le stock et les quantités collectées sont bornés et le niveau de stock est mis à zéro à chaque collecte.

- **Solutions extrêmes**

La structure des solutions extrêmes est définie dans la propriété suivante.

**Propriété 4.1.3** *Si la variable $y$ est entière alors :*

1. *les solutions (x,s) extrêmes associées sont entières ;*
2. *$s\,y = 0$ ;*
3. *il y a unicité de la solution (x,s) extrême associée.*

En effet, le premier point est une observation classique du problème de lot sizing : à $y$ fixée et entière, $(x,s)$ est solution d'un problème de flot de coût minimum dans le graphe de la figure 4.1. Le deuxième point est dû à l'hypothèse de la politique "order up to level". Le troisième point utilise le deuxième point, i.e. $s_t y_t = 0 \quad \forall t$. Pour chaque $t$, il y aura soit $x_t = 0$ si $y_t = 0$, soit $s_t = 0$ si $y_t = 1$. D'après la contrainte (4.4), $x_1$ est fixé à une valeur entière (à 0 ou $tinit + 1$), puis de proche en proche en utilisant les contraintes (4.2) nous en déduisons les variables $x$ et $s$ non déjà fixées qui sont entières. □

- **Inégalités valides** $(k, l, I)$

Pour le problème général CCLS, une formulation compacte idéale (donnant l'enveloppe convexe des solutions entières) n'est pas connue dans l'espace des variables s,x,y. Cependant il existe un nombre exponentiel d'inégalités valides qui peuvent améliorer la formulation (Pochet et al. [73], 1995) : les inégalités $(k, l, S, I)$. Considérons l'intervalle $[k, l]$ et le stock entrant $s_{k-1}$. Notons $d_{kl}$, la

demande totale des périodes de $k$ à $l$. Soit $\eta_{kl} = \lceil \frac{d_{kl}}{C} \rceil$ le nombre minimum de fois que l'on doit produire pour satisfaire la demande dans $[k, l]$ si $s_{k-1} = 0$ ; et $\gamma_{kl} = d_{kl} - C(\eta_{kl} - 1)$ la valeur minimum de $s_{k-1}$ si l'on doit produire $\eta_{kl} - 1$ fois dans l'intervalle $(0 < \gamma_{kl} \leq C)$. Etant donné $S \subset \{k, \cdots, l\}$, choisissons $I \subset \{k, \cdots, l\}$ tel que pour chaque $i \in I$, soit $i+1 \in S$ soit $i = l$ et ordonnons les valeurs $\{\gamma_{ki}\}_{i \in I}$ telles que $0 = \gamma_{k[0]} \leq \gamma_{k[1]} \leq \gamma_{k[2]} \leq \cdots \leq \gamma_{k[|I|]}$ où $\{[1], [2], \cdots, [|I|]\}$ est une permutation de $I$. Alors les inégalités $(k, l, S, I)$ suivantes

$$s_{k-1} + \sum_{j \in \{k, \cdots, l\} \setminus S} x_j \geq \sum_{i=1}^{|I|} (\gamma_{k[i]} - \gamma_{k[i-1]})(\eta_{k[i]} - \sum_{j=k, j \in S}^{[i]} y_j)$$

sont valides pour $X^{CCLS}$ (l'ensemble des solutions du problème CCLS), mais on ne sait pas si la famille des inégalités $(k, l, S, I)$ suffit à décrire $conv(X^{CCLS})$.

Belvaux ([11], 1999) définit les inégalités $(k, l, I)$ pour le problème CCLSb (i.e. CCLS avec des demandes rétrogrades) :

$$s_{k-1} + \sum_{i=1}^{|I|} r_{[i]} \geq \sum_{i=1}^{|I|} (\gamma_{k[i]} - \gamma_{k[i-1]})(\eta_{k[i]} - \sum_{j=k}^{[i]} y_j) \tag{4.23}$$

où la variable $s_{k-1}$ correspond au stock en fin de période $k-1$ (stock venant de la période $k-1$ entrant en période $k$) et la variable $r_k$ représente le retard de la demande à la fin de la période $k$ (stock venant de la période $k+1$ entrant en période $k$). Belvaux obtient (4.23) en appliquant la "mixing procedure" de Günlük et Pochet aux inégalités suivantes :

$$s_{k-1} + r_l + C \sum_{\tau=k}^{l} y_\tau \geq d_{kl} \quad \forall 1 \leq k \leq l \leq T \tag{4.24}$$

Dans notre cas, la variable de stock est nulle, les demandes rétrogrades sont représentées par $s$, (4.24) devient $s_l + tmax \sum_{t=k}^{l} y_t \geq d_{kl} \quad \forall 1 \leq k \leq l \leq T$ (on somme les contraintes (4.2) de $k$ à $l$ et on utilise $s_{k-1} \geq 0$ et $x_t \leq tmax\, y_t$). De plus les demandes étant unitaires, $\eta$, $\gamma$ peuvent se déterminer facilement. D'abord, $d_{kl} = l - k + 1$ puis écrivons $l = k + n^1_{kl}\, tmax + n^2_{kl}$ avec $n^1_{kl}$ le quotient de $l - k$ par $tmax$ et $n^2_{kl} < tmax$ le reste. Ainsi :

$$\eta_{kl} = \lceil \frac{l - k + 1}{tmax} \rceil = \lceil \frac{n^1_{kl}\, tmax + n^2_{kl} + 1}{tmax} \rceil = n^1_{kl} + \lceil \frac{n^2_{kl} + 1}{tmax} \rceil = n^1_{kl} + 1$$

$$\gamma_{kl} = (l - k + 1) - tmax(n^1_{kl} + 1 - 1) = n^2_{kl} + 1$$
$$\text{et } \gamma_{k[0]} = n^2_{k[0]} + 1 = 0 \Rightarrow n^2_{k[0]} = -1$$

**Observation 4.1.4** *Les $(k, l, I)$ inégalités pour $1 \leq k \leq l \leq T$, $I \subseteq \{k, \cdots l\}$ :*

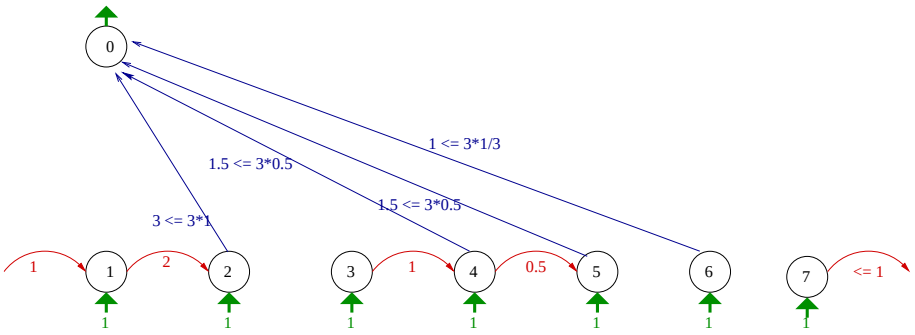
$$\sum_{i=1}^{|I|} s_{[i]} \geq \sum_{i=1}^{|I|} (n^2_{k[i]} - n^2_{k[i-1]})(n^1_{k[i]} + 1 - \sum_{j=k}^{[i]} y_j) \tag{4.25}$$

*sont valides pour $X^{StockLSsxy}$.*

Les inégalités $(k, l, I)$ coupent des solutions fractionnaires de $[StockLSsxy]$ comme le montre l'exemple 4.1.

Exemple 4.1 – Inégalités $(k, l, I)$

Soit la solution fractionnaire de la figure suivante :




Prenons l'intervalle $[k, k]$ et $I = \{k = [1]\}$, alors $n^1_{k[1]} = 0$, $n^2_{k[1]} = 0$ (et $n^2_{k[0]} = -1$). L'inégalité $s_k \geq 1(1 - y_k)$ coupe cette solution pour $k = 5, 6$ où $s_k = 0$ et $y_5 = 0.5$, $y_6 = \frac{1}{3}$.
Ensuite prenons $[k, l] = [3, 5]$ et $I = \{4 = [1]\}$, alors $n^1_{3[1]} = 0$, $n^2_{3[1]} = 1$ (et $n^2_{k[0]} = -1$). L'inégalité $s_4 \geq 2(1 - y_3 - y_4)$ coupe cette solution où $s_4 = 0.5$ et $y_3 = 0$, $y_4 = 0.5$.

- **Projection par élimination de variables**

Les variables $s$ peuvent être éliminées en utilisant les contraintes (4.2) :

$$s_t = t + t_{init} - \sum_{l=1}^{t} x_l$$

où $t + t_{init}$ est la demande totale des périodes de 1 à $t$. En posant $\beta'_t = \beta_t - \sum_{l=t}^{T} \gamma_l$ et $Cte = \sum_t (t + tinit)\gamma_t$, la formulation du problème de gestion de stock devient :

$$
\begin{aligned}
[StockLSxy] \quad Cte &+ \min \sum_t (\pi_t y_t + \beta'_t x_t) && && (4.26)\\
x_t &\leq tmax\, y_t && \forall t = 2, \cdots, T && (4.27)\\
x_1 &= (tinit + 1)\, y_1 && && (4.28)\\
t + t_{init} &\leq \sum_{l=1}^{t} x_l + (t_{max} - 1)\,(1 - y_t) && \forall t = 2, \cdots, T && (4.29)\\
T + t_{init} &\leq \sum_{l=1}^{T} x_l + t_{init}\,(1 - y_T) && && (4.30)\\
y_t &\in \{0, 1\} && \forall t && (4.31)\\
x_t &\geq 0 && \forall t && (4.32)\\
t + t_{init} &\geq \sum_{l=1}^{t} x_l && \forall t. && (4.33)
\end{aligned}
$$

**Observation 4.1.5** *$[StockLSxy]$ est la projection sur l'espace $(x, y)$ de $[StockLSsxy]$, elles sont par conséquent équivalentes.*

Alternativement, les contraintes (4.2) permettent d'éliminer $x$ :

$$x_t = s_{t-1} - s_t + 1$$

En posant $\gamma'_t = \gamma_t + \beta_{t+1} - \beta_t$ et $Cte = \sum_t \beta_t$, la formulation du problème de gestion de stock devient :

$$
\begin{aligned}
[StockLSsy] \; Cte &+ \min \sum_t (\pi_t y_t + \gamma'_t s_t) && && (4.34)\\
s_{t-1} + 1 &\leq s_t + tmaxy_t && \forall t = 2, \cdots, T && (4.35)\\
s_1 &= (tinit + 1)(1 - y_1) && && (4.36)\\
s_t &\leq (tmax - 1)\,(1 - y_t) && \forall t = 2, \cdots, T && (4.37)\\
s_T &\leq tinit\,(1 - y_T) && && (4.38)\\
y_t &\in \{0, 1\} && \forall t && (4.39)\\
s_t &\leq s_{t-1} + 1 && \forall t && (4.40)\\
s_t &\geq 0 && \forall t. && (4.41)
\end{aligned}
$$

**Observation 4.1.6** *$[StockLSsy]$ est la projection sur l'espace $(s, y)$ de $[StockLSsxy]$, elles sont par conséquent équivalentes.*

- **Contraintes de la politique order up to level**

Enfin, dans le cas de la politique d' "order up to level", la seule connaissance des $y$ permet de définir entièrement le planning individuel (i.e. les dates de collecte et les quantités ramassées) : les contraintes de gestion de stock peuvent être formulées avec seulement ces variables. En effet, un planning réalisable est tel qu'il doit y avoir un ramassage au moins sur tout intervalle de $tmax$ périodes consécutives. Les contraintes s'écrivent :

$$\sum_{l=1}^{tmax-tinit} y_l \geq 1 \qquad \text{(début)} \tag{4.42}$$

$$\sum_{l=T-tinit}^{T} y_l \geq 1 \qquad \text{(fin)} \tag{4.43}$$

$$\sum_{l=t}^{t+tmax-1} y_l \geq 1 \qquad \forall 1 \leq t \leq T - tmax + 1 \tag{4.44}$$

$$y_t \in \{0,1\} \quad \forall t. \tag{4.45}$$

Cependant, les variables $x$ ou $s$ se calculent avec des contraintes non linéaires :

$$x_1 = (tinit+1)y_1 \qquad s_1 = (tinit+1)(1-y_1)$$
$$x_t = (tinit + t - \sum_{\tau=1}^{t-1} x_\tau)y_t \qquad s_t = (tinit + t - \sum_{\tau=1}^{t-1} x_\tau)(1-y_t)$$

Ainsi, pour évaluer le coût du planning et avoir les informations sur la quantité à ramasser (pour coopérer avec le problème de tournées), les variables $x$ (ou $s$) et les contraintes associées doivent être gardées.

En revanche, les inégalités (4.42-4.44) améliorent la formulation $[StockLSxy]$ en coupant des solutions fractionnaires comme le montre l'exemple 4.2.

**Proposition 4.1.7**

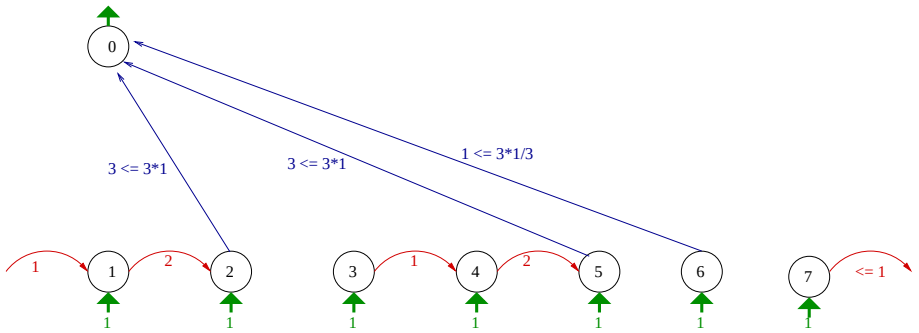
$$\sum_{l=t}^{t+tmax-1} y_l \geq 1 \quad \forall t = 1, \cdots, T+1-tmax \tag{4.46}$$

*est une inégalité valide pour* $X^{StockLSsxy}$.

Pour le début et la fin de l'horizon, les contraintes (4.46) deviennent les inégalités (4.42) et (4.43).

Exemple 4.2 – Inégalités valides (4.42-4.44)

Soit la solution fractionnaire pour $[StockLSxy]_{PL}$ de la figure suivante :




$y_6 + y_7 \geq 1$ coupe cette solution où $y_6 + y_7 = \frac{1}{3}$.

**Preuve:** En sommant les contraintes (4.2) sur une longueur de $tmax$, nous avons :

$$tmax = \sum_{l=t}^{t+tmax-1} x_l + s_{t+tmax-1} - s_{t-1}$$

Comme $x_l \leq tmax\, y_l$ et comme le stock est positif et borné par $tmax - 1$, alors $s_{t+tmax-1} - s_{t-1} \leq tmax - 1$ ; nous obtenons :

$$tmax \leq tmax \sum_{l=t}^{t+tmax-1} y_l + tmax - 1 \Rightarrow 1 \leq tmax \sum_{l=t}^{t+tmax-1} y_l$$

Finalement, en divisant par $tmax$ et en arrondissant à la partie entière supéroieure, nous obtenons les contraintes (4.46).

De la même manière, nous obtenons les inégalités (4.42) pour le début et (4.43) pour la fin de l'horizon en sommant sur $tmax - tinit$ et $tinit + 1$ termes respectivement. □

### 4.1.3 Modèles de localisation des dépôts

La formulation $[StockFL]$ définie en (4.10-4.17) est une reformulation de $[StockLSxy]$. La solution de $[StockFL]$ peut être projetée sur l'espace $(x, y)$ en utilisant $y_t = w_{tt}$ et $x_t = \sum_{l=t-t_{\max}+1}^{t} w_{tl}$.

- **Solutions extrêmes**

La structure des solutions extrêmes est définie dans la propriété suivante :

**Propriété 4.1.8** *Si $y$ est fixée et entière, alors le problème admet une solution PL extrême avec $w$ entière.*

En effet, lorsque les $w_{tt} = y_t$ sont fixés, il ne reste plus qu'un seul choix pour les autres variables $w_{lt}$ d'après les contraintes (4.15) (mise à zéro) et (4.12) (un seul est égal à 1) : les périodes $l$ non encore couvertes sont affectées à la collecte la plus proche $t$. □

**Remarque:** Les variables $y$ peuvent être éliminées du fait que $y_t = w_{tt}$ et $y_{T+1} = 1 - w_{TT}$. La formulation simplifiée avec $O(T\,tmax)$ variables et $O(T + T\,tmax)$ contraintes devient :

$$[StockFLSimp] \quad C \;+\; \min \sum_{t=1}^{T} \; (\pi_t w_{tt} + \beta'_t \sum_{l=t-tmax+1}^{t} d_l w_{lt}) \tag{4.47}$$

$$\sum_{t=1}^{tmax-tinit} w_{0t} \;=\; 1 \tag{4.48}$$

$$\sum_{t=\max(1,l)}^{\min(l+tmax-1,T+1)} w_{lt} \;=\; 1 \qquad \forall\ 1 \le l \le T \tag{4.49}$$

$$w_{lt} \;\ge\; w_{l-1\,t} \qquad \forall l \le t \tag{4.50}$$

$$w_{tt} \;\in\; \{0,1\} \qquad \forall t \tag{4.51}$$

$$w_{lt} \;\ge\; 0 \qquad \forall l < t. \tag{4.52}$$

**Observation 4.1.9** *La formulation* $[StockFLSimp]$ *nécessite* $O(T\,tmax)$ *variables et* $O(T + T\,tmax)$ *contraintes.*

**Remarque:** (Pochet [72], 2001)
Le problème de lot sizing sans capacité (ULS) peut être formulé comme un problème de localisation des dépôts, dans ce cas les variables $w_{lt} = 1$ si la production de la période $l$ (i.e. si le dépôt $l$ est ouvert, $y_l = 1$) couvre la demande de la période $t \ge l$, on peut aussi utiliser $z_{lt} = \frac{w_{lt}}{d_t}$ qui représente la fraction de la demande $d_t$ servie par la production de la période $l$. Krarup et Bilde, 1977, ont montré que la relaxation linéaire admet une solution $y$ entière optimale, due à la structure particulière de la fonction objectif (le coût de $w$ est nul dans les formulations $(x, y, w)$). Cependant, certains sommets du polyèdre peuvent encore être fractionnaires, mais en ajoutant les contraintes imposant la continuité de la solution $w_{lt} \le w_{l\,t+1} \quad \forall\ 1 \le l \le t \le T$ (les contraintes (4.15) pour notre cas) on obtient l'enveloppe convexe des solutions du ULS.

Un raisonnement similaire nous permet d'établir :

**Propriété 4.1.10** *La relaxation linéaire de la reformulation* $[StockFLSimp]$ *définie en (4.47-4.52) donne une solution entière au problème de gestion de stock d'un seul client.*

**Preuve:** Verweij et Wolsey [84] ont montré ce résultat pour le problème de lot sizing avec achat, vente et demandes rétrogrades. □

### 4.1.4 Modèles de plus court chemin

La formulation $[StockSP]$ définie en (4.18-4.22), est une reformulation de $[StockLSxy]$ exploitant la structure des solutions. La solution de $[StockSP]$ peut être projetée sur l'espace $(x, y)$ en utilisant les relations :

$$\begin{aligned} y_l &= \sum_{t<l} \Phi_{tl} \qquad \forall\, l \\ x_l &= \sum_{t<l} (l-t)\Phi_{tl} \qquad \forall l. \end{aligned}$$

- **Solutions extrêmes**

Les solutions extrêmes sont des solutions d'un problème de flot dont la matrice des coefficients des contraintes est totalement uni-modulaire.

**Propriété 4.1.11** *La reformulation de plus court chemin est une bonne formulation [4] : la relaxation linéaire donne une solution extrême entière (matrice TU).*

- **Résolution par programme dynamique**

Ce problème de plus court chemin peut être résolu par programme dynamique, posons :
$\Theta^t$ = le coût d'une politique optimale sur l'horizon $1, \cdots, t$ avec un ramassage en période $t$.
Il se calcule récursivement pour tout $t = 1, \cdots, T$. Si $t + tinit \leq tmax$, alors :

$$\Theta^t = \min \left\{ \pi_t + (t + tinit)\beta'_t;\ \min_l \{\Theta^l + \pi_t + (t-l)\beta'_t : 1 \leq l \leq t-1\} \right\}$$

sinon :

$$\Theta^t = \min_l \{\Theta^l + \pi_t + (t-l)\beta'_t : \max(1, t - tmax) \leq l \leq t-1\}$$

Le coût d'une politique optimale est :

$$\mathcal{O}pt = \min_{\min(T-tinit,T) \leq t \leq T} \{\Theta^t + \pi_t + (T-t+1)\beta'_t\}$$

Ce programme dynamique est un algorithme d'optimisation ayant une complexité polynômiale de $O(T \times tmax)$.

## 4.2 Solutions sur un horizon des temps restreint (scénario B)

Dans le scénario B, l'horizon est fini, on le note $T$. Le stock initial n'est pas donné mais doit être égal au stock final pour assurer la périodicité. La même démarche que celle de la section précédente peut être faite, à savoir que le problème de gestion de stock peut être formulé comme un problème de lot sizing (la formulation est notée $[StockCycleLSsxy]$). Ensuite ce problème peut être reformulé comme un problème de localisation des dépôts (la formulation est notée $[FLCycle]$) et comme un problème de flot (la formulation est notée $[CycleFlot]$). Dans cette section, seules les différences par rapport au scénario A sont indiquées. La non fixation du stock initial fait que :

- dans la formulation de lot sizing $[StockCycleLSsxy]$, $s_0$ est une variable égale à $s_T$ ; les variables $s$ ne peuvent plus être éliminées.
- les solutions extrêmes de la formulation de localisation des dépôts $[FLCycle]$ ne sont plus entières.
- le problème de plus court chemin devient un problème de flot cyclique de coût minimum $[CycleFlot]$ et la matrice des contraintes n'est plus TU. Cependant le programme dynamique pour le résoudre reste de complexité polynômiale.
- une symétrie supplémentaire due au temps apparaît : une solution peut être décalée d'une période sans en changer le coût.

Les formulations $[FLCycle]$ et $[CycleFlot]$ sont équivalentes et restent meilleures que la formulation $[StockCycleLSsxy]$. La formulation $[CycleFlot]$ a moins de contraintes que la formulation $[FLCycle]$. C'est donc $[CycleFlot]$ qui est retenue pour formuler le problème de gestion de stock.

- **Formulation de lot sizing**

Comme pour le problème de gestion de stock du scénario A vu en section 4.1.2, le problème de gestion de stock du scénario B pour un client se formule comme un problème de lot sizing où les demandes sont unitaires et rétrogrades, le stock et les quantités collectées sont bornés et le niveau de stock est mis à zéro à chaque collecte, à la différence que le stock initial n'est pas connu : $s_0$ est une variable. Elle est restreinte à prendre la même valeur que le stock final, i.e. $s_0 = s_T$ (sur la figure 4.4, un arc du nœud $T$ au nœud 1 associé à $s_T$ assure le cycle).

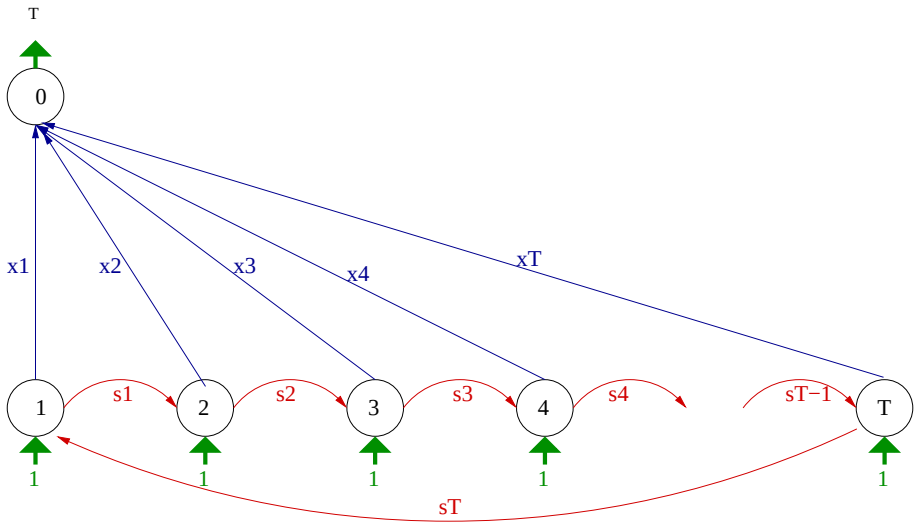



FIG. 4.4 – Modèle de planification : le stock initial est inconnu mais égal au stock final

La formulation du problème de planification cyclique $[StockCycleLSsxy]$ est similaire à la formulation $[StockLSsxy]$ définie en (4.1-4.9) sauf que :

- la contrainte (4.2) devient pour $t = 1$ : $s_T + 1 = x_1 + s_1$.
- les contraintes (4.3) et (4.5) sont valides pour tout $t$.
- les contraintes (4.4) et (4.6) sont supprimées.

La propriété 4.1.3 caractérisant les solutions extrêmes reste vraie car durant l'horizon $T$, il y aura au moins une collecte, posons la en $t_0$ et $s_{t_0} = 0$, puis $t_0$ devient la période de référence pour déterminer de proche en proche les valeurs des autres variables $x$ et $s$.

**Observation 4.2.1** *La formulation* $[StockCycleLSsxy]$ *utilise* $3T$ *variables et* $3T$ *contraintes.*

- **Inégalités valides**

La formulation en $(x, y, s)$ peut toujours être améliorée avec les inégalités $(k, l, I)$ (4.25), cependant l'intervalle $[k, l]$ peut être cyclique. En effet, à $k$ fixé, $l$ peut être plus grand que $k$, ou strictement plus petit. Dans le cas où $k \leq l$ rien ne change. Dans l'autre cas, $I \subseteq \{k, \cdots, T, 1, \cdots, l\}$ et $n^1_{k[i]}$, $n^2_{k[i]}$ sont respectivement le quotient et le reste de $[i] + T - k$ par $tmax$. Le même exemple 4.1 en ajoutant l'arc $(7, 1)$ peut être utilisé pour montrer une solution fractionnaire coupée par les inégalités $(k, l, I)$.

Les inégalités (4.46) avec $t$ cyclique (i.e. lorsque $l > T$ alors on redémarre avec $l = 1$) sont toujours valides, comme l'illustre l'exemple 4.3.

Exemple 4.3 – Inégalités valides (4.46) cyclique

Reprenons la figure de l'exemple 4.2 en ayant ajouté l'arc $(7, 1)$.
La coupe devient $y_6 + y_7 + y_1 \geq 1$.

Pour les mêmes raisons que pour le scenario A, il n'est pas possible d'éliminer les variables $x$ et $s$ car elles se calculent avec des contraintes non linéaires.

- **Elimination de variables**

Il n'est plus possible d'éliminer toutes les variables $s$ en utilisant les contraintes de flots (4.2) où $s_0 = s_T$ car :
(4.2) pour $t = 1$ $\Rightarrow s_T = x_1 + s_1 - 1$,
(4.2) pour $t = 2, \cdots, T$ $\Rightarrow s_t = t - 1 - \sum_{l=2}^{t} x_l + s_1, \forall t = 2, \cdots, T-1$
$\Rightarrow s_T = T - 1 - \sum_{l=2}^{T} x_l + s_1$,
Finalement, il reste $T-1$ équations pour $T$ variables $s$, la variable $s_1$ doit être gardée ; les autres peuvent être éliminées (notons que $\sum_{l=1}^{T} x_l = T$). Ainsi une solution fractionnaire en $x$ et $y$ peut avoir plusieurs solutions en $s$ possibles, selon la valeur de $s_1$ comme le montre l'exemple 4.4.

Exemple 4.4 – Plusieurs solutions $s$ possibles pour $y$ et $x$ fractionnaires

Soit la solution dans laquelle la demande est ramassée à chaque période, $x_t = 1, y_t = \frac{1}{tmax} \ \forall t$, alors $s_t = s_1$ avec $s_1$ variant dans $[0; \frac{tmax-1}{tmax}]$.

La projection sur l'espace $(x, y)$ ne peut donc pas se faire, il faut garder la variable $s_1$ (on peut projeter sur l'espace $(x, y, s_1)$).

Inversement les variables $x$ peuvent être éliminées :

$$\begin{aligned} x_1 &= s_T - s_1 + 1 \\ x_t &= s_{t-1} - s_t + 1 \ \ \forall t = 2, \cdots, T \end{aligned}$$

La formulation obtenue après suppression de la variable $x$, notée $[StockCycleLSsy]$, est la projection sur l'espace $(s, y)$ de $[StockCycleLSsxy]$, elles sont par conséquent équivalentes.

- **Formulation en localisation des dépôts**

Comme pour le cas du scénario A, le problème de gestion de stock du scénario B peut se formuler comme un problème de localisation de dépôt avec

ramassage de périodes consécutives. A la différence du cas du scénario A, il y a $T$ sites de production, $T$ dépôts et les cycles sont permis (voir la figure 4.5). Dans la figure 4.5, l'arc $(7,1)$ signifie que la collecte de la période 1 ramasse la demande de la dernière période du cycle précédent (i.e. la demande de la période $0 = T - T$ qui correspond à la période $T$ du cycle précédent) ; et que la demande de la période 7 sera ramassée à la première période du cycle suivant (i.e. en période $1 + T$).

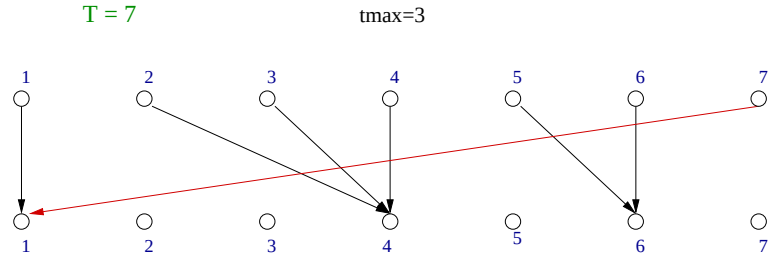



FIG. 4.5 – Un planning réalisable avec la formulation de localisation des dépôts

Pour formuler le problème, les mêmes variables $w_{lt}$ sont utilisées ($y$ est inutile vu que $y_t = w_{tt} \quad \forall t$), cependant $w_{lt}$, dénotant le fait que la production de $l$ est ramassée en $t$, est définie pour les couples $(l,t)$ tels que $1 \leq l \leq T$ et $l \leq t \leq \min(l + t_{\max} - 1, T)$ ou $T - tmax + 1 \leq l \leq T$ et $1 \leq t \leq l - T + tmax$ (la production de $l$ est ramassée en $T + t$). Le coût de l'arc $(l,t)$ est $s_{lt} = \beta_t + \sum_{k=l}^{t-1} \gamma_k$.

Une formulation similaire à $[StockFLSimp]$ définie en (4.47-4.52) peut être obtenue. La contrainte (4.48) est supprimée. Dans les contraintes (4.49) et (4.50) les indices cyclent, i.e. lorsque $t$ arrive à $T$ il repart de 1. L'objectif devient : $\min \sum_t (\pi_t w_{tt} + \sum_l s_{lt} w_{lt})$. La formulation obtenue est notée $[FLCycle]$.

**Observation 4.2.2** *La formulation* $[FLCycle]$ *implique* $O(Ttmax)$ *variables et* $O(T + Ttmax)$ *contraintes.*

Pour projeter la solution sur l'espace $(x,y)$, nous utilisons : $y_t = w_{tt}$ et $x_t = \sum_l w_{lt}$. Mais pour connaître $s$ étant donné une solution entière $y$ ($w_{tt}$ est remplacé par $y_t$), nous utilisons l'équation non linéaire suivant :

$$s_t = (((2 - y_{t-tmax+2})(1 - y_{t-tmax+3}) \cdots + 1)(1 - y_{t-1}) + 1)(1 - y_t) \quad (4.53)$$

La propriété 4.1.8 caractérisant les solutions extrêmes reste vraie en revanche si $tmax > 1$ la propriété d'intégralité 4.1.10 ne l'est plus comme le montre l'exemple 4.5. Par contre, si un arc retour $(l, t)$ avec $l > t$ est fixé à 1, les solutions extrêmes de $[FLCycle]$ sont entières car elles sont solutions du problème de localisation des dépôts en ayant fixé la situation initiale et finale ($d_0$ est le nombre d'arcs retours et $w_{tT+1} = 1$ s'il y a l'arc retour d'origine $t$).

Exemple 4.5 – La solution $PL$ de $[FLCycle]$ peut être fractionnaire

Soit un client tel que $tmax = 2$ et $\forall t,\ \gamma_t = 0, \pi_t = \pi, \beta_t = \beta$, alors les coûts des arcs du graphe sont tels que $s_{tt+1} = \beta$, $s_{tt} = \pi + \beta$.
La solution fractionnaire optimale a un coût de $3.5\pi + 7\beta = 0.5 * 7(\pi + 2\beta)$, elle consiste à prendre tous les arcs avec une valeur de 0.5, voir la figure 4.6.
La solution entière optimale a un coût de $4\pi + 7\beta = 3(\pi + 2\beta) + \pi + \beta$, voir la figure 4.7.
Remarquons qu'une planification sur $2T = 14$ périodes, mène à une solution entière de coût 2 fois supérieur à celui de la solution fractionnaire, i.e. $7\pi + 14\beta$, en visitant toutes les périodes impaires par exemple.

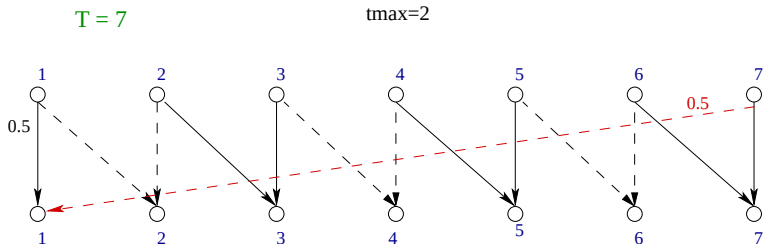



FIG. 4.6 – Exemple de solution fractionnaire pour $[FLCycle]$

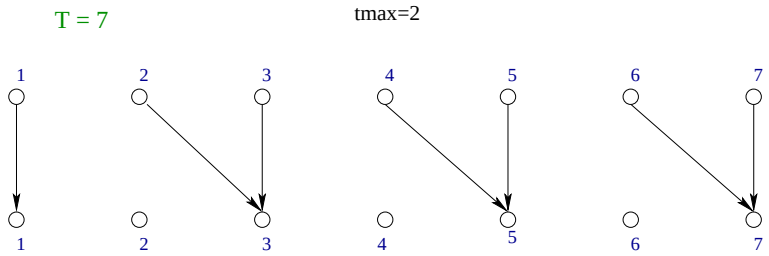



FIG. 4.7 – Exemple de solution entière pour $[FLCycle]$

- **Formulation de flot cyclique de coût minimum**

La structure des solutions permet toujours de formuler le problème de gestion de stock comme un problème de flot. Cette fois ci le problème de flot est défini dans un graphe avec $T$ nœuds (voir la figure 4.8) qui n'est plus acyclique comme le graphe de la figure 4.3. On obtient un problème flot cyclique de coût minimum.

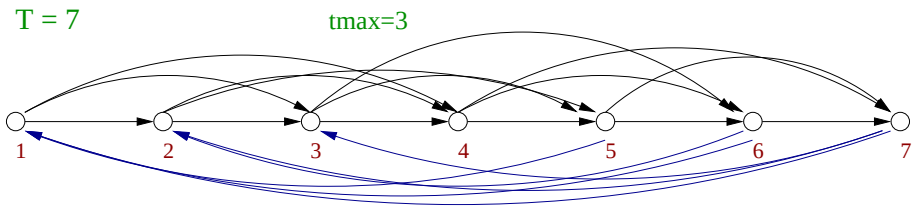



FIG. 4.8 – Graphe cyclique pour calculer un planning

L'arc $(t,k)$ représente toujours deux collectes en périodes $t$ et $k$ mais ces collectes ne se passent pas forcément toutes les deux durant l'horizon $T$. Si $t < k$, l'arc $(t,k)$ représente le fait que le client reçoit un service en $k$ alors que le ramassage précédent datait de $t$, $(t,k)$ est défini pour $t = 1, \cdots, T-1$ et $k = t+1, \cdots, \min\{t+tmax, T\}$. Si $t > k$, l'arc $(t,k)$ représente le fait que le client reçoit un service en $k$ alors que le ramassage précédent datait de $t-T$ (ou alors le futur ramassage se fait à la date $t+T$), $(k,t)$ est défini pour $k = 1, \cdots, tmax$ et $t = T - tmax + k, \cdots, T$. Sous l'hypothèse que $T$ est suffisamment grand (par exemple $T \geq 2\, tmax$), alors aucun nœud n'a d'arc retour entrant et d'arc retour sortant. Le coût de l'arc devient :

$$s_{tk} = \pi_k + (k-t)\beta_k + \sum_{l=t+1}^{k-1} (l-t)\gamma_l$$

Les contraintes (4.21)-(4.22) de $[StockSP]$ restent valides, (4.19)-(4.20) sont remplacées par :

$$\sum_{(t,k):k<t} \Phi_{tk} = 1 \tag{4.54}$$

Cette contrainte (4.54) peut être mise sous la forme "$\geq$" sans relaxer le problème. L'objectif devient : $\min \sum_{t,k} s_{tk}\, \Phi_{tk}$. La formulation obtenue, notée $[CycleFlot]$, est celle d'un problème de flot cyclique de coût minimum avec une contrainte de borne généralisée (4.54).

**Observation 4.2.3** *La formulation* $[CycleFlot]$ *implique* $O(Ttmax)$ *variables et* $T+1$ *contraintes.*

Pour projeter la solution $\Phi$ sur l'espace $(x, y)$, nous utilisons :

$$y_t = \sum_k \Phi_{kt} \ \forall\, t \tag{4.55}$$

$$x_t = \sum_{k;k<t} (t-k)\Phi_{kt} + \sum_{k;k>t} (t+T-k)\Phi_{kt} \ \forall t \tag{4.56}$$

Le calcul de $s$ dans le cas d'une solution entière $\Phi$ se fait à partir de l'équation non linéaire suivante ($\sum_k \Phi_{kt}$ est remplacé par $y_t$) :

$$s_t = (((2 - y_{t-tmax+2})(1 - y_{t-tmax+3}) \cdots + 1)(1 - y_{t-1}) + 1)(1 - y_t) \tag{4.57}$$

Avec l'ajout de la contrainte de borne généralisée (4.54), la matrice n'est plus TU, la propriété d'intégralité 4.1.11 n'est donc plus vraie si $tmax > 1$, comme le montre l'exemple 4.6. Par contre, si un arc retour $(k, t)$ avec $k > t$ est fixé à 1, les solutions extrêmes de $[CycleFlot]$ sont entières car elles sont solutions du problème de plus court chemin de $t$ à $k$.

Exemple 4.6 – La solution $PL$ de $[CycleFlot]$ peut être fractionnaire

Soit un client tel que $tmax = 2$ et $\forall t,\ \gamma_t = 0, \pi_t = \pi, \beta_t = \beta$, alors les coûts des arcs du graphe sont tels que $s_{tt+2} = \pi + 2\beta$, $s_{tt+1} = \pi + \beta$.
La solution fractionnaire optimale a un coût de $3.5\pi + 7\beta = 0.5 * 7(\pi + 2\beta)$, voir la figure 4.9.
La solution optimale a un coût de $4\pi + 7\beta = 3(\pi + 2\beta) + (\pi + \beta)$, voir la figure 4.10.
Remarquons qu'une planification sur $2T = 14$ périodes, mène à une solution entière de coût 2 fois supérieur à celui de la solution fractionnaire, i.e. $7\pi + 14\beta$, en visitant toutes les périodes paires par exemple.

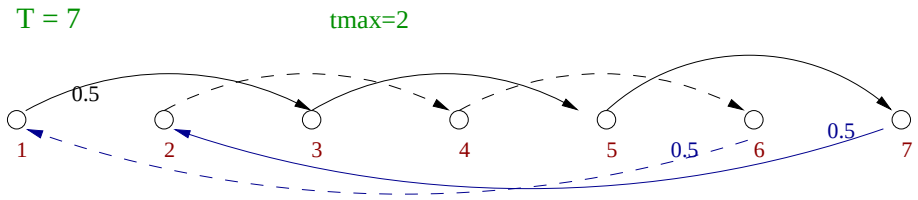



FIG. 4.9 – Exemple de solution fractionnaire pour $[CycleFlot]$

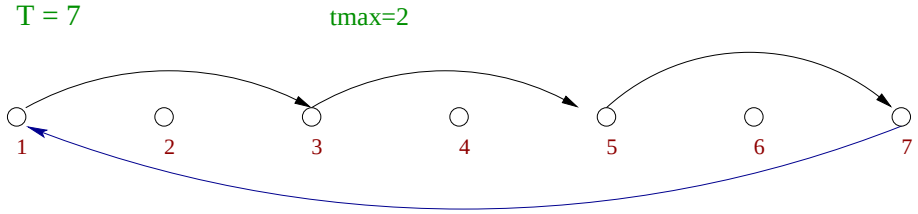



FIG. 4.10 – Exemple de solution entière pour $[CycleFlot]$

**Propriété 4.2.4** *Les formulations* $[CycleFlot]$ *et* $[FLCycle]$ *sont équivalentes.*

**Preuve:** En faisant le changement de variables $\Phi_{l-1\,t} = w_{lt} - w_{l-1t}$ dans la formulation $[CycleFlot]$, on retrouve la formulation $[FLCycle]$.
En effet, les contraintes de positivité $\Phi_{l-1\,t} \geq 0$ sont équivalentes aux contraintes de continuité $w_{lt} \geq w_{l-1t}$. La contrainte de borne généralisée $\sum_{(l,t):t<l} \Phi_{lt} = 1$ est équivalente à la contrainte de la couverture de la demande de la période 1, $\sum_l w_{1l} = 1$. Les contraintes de flot $\sum_{l=t-tmax}^{t-1} \Phi_{lt} = \sum_{l=t+1}^{t+tmax} \Phi_{tl}$ pour chaque période $t$ sont équivalentes à $w_{tt} = \sum_{l=t+1}^{t+tmax} w_{t+1\,l} - \sum_{l=t+1}^{t+tmax-1} w_{tl}$. Pour l'objectif, nous avons : $\sum_{lt}(\pi_t + (t-l)\beta_t + \sum_{k=l+1}^{t-1}(k-l)\gamma_k)\ \Phi_{lt} = \sum_t \pi_t w_{tt} + \sum_{lt}(((t-l+1)\beta_t + \sum_{k=l}^{t-1}(k-l+1)\gamma_k) - ((t-l)\beta_t + \sum_{k=l+1}^{t-1}(k-l)\gamma_k))w_{lt}$ et $(t-l+1)\beta_t + \sum_{k=l}^{t-1}(k-l+1)\gamma_k - ((t-l)\beta_t + \sum_{k=l+1}^{t-1}(k-l)\gamma_k)w_{lt} = (\beta_t + \sum_{k=l}^{t-1}\gamma_k)w_{lt}$. On retrouve ainsi la formulation $[FLCycle]$. □

**Propriété 4.2.5** *La formulation* $[CycleFlot]$ *est meilleure que la formulation* $[StockCycleLSsxy]$.

**Preuve:** Il existe des solutions fractionnaires de $[StockCycleLSsxy]$ qui ne sont pas solutions de $[CycleFlot]$ comme celle de l'exemple de la figure 4.11. En effet, les contraintes de flot ne sont pas respectées, pour le nœud 1, il y a $\frac{1}{2}$ d'unité de flot entrant ($\phi_{71} = y_1$) mais une unité de flot sortant ($\phi_{13} = y_3$).

En revanche, toutes les solutions fractionnaires de $[CycleFlot]$ sont aussi des solutions fractionnaires de $[StockCycleLSsxy]$. Les solutions extrêmes de

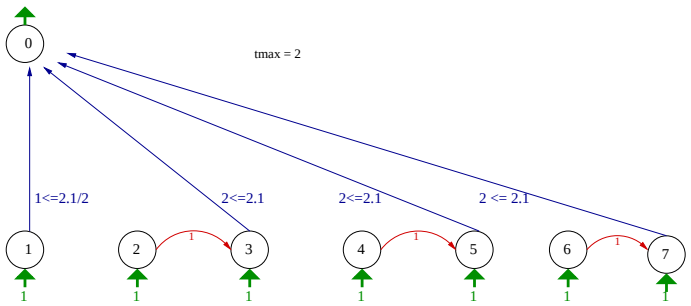



FIG. 4.11 – Exemple de solution fractionnaire pour $[StockCycleLSsxy]$

$[CycleFlot]$, $\phi^e$, sont une combinaison convexe de solutions de $[StockSP]$ (4.18-4.22), $\phi^e = \sum_f \lambda_f \phi^f$ avec $\sum_f \lambda_f = 1$. Les situations initiales et finales de $[StockSP]$ peuvent être différentes, elles sont déterminées par les arcs retour. On a au moins une solution de $[StockSP]$ par arc retour, le coefficient $\lambda_f$ associé à cette solution $\phi^f$ est égal au flot circulant sur les arcs la composant. Sur la figure 4.9, un flot est en trait continu, l'autre en pointillé, chacun ayant une valeur de 0.5. Pour le flot en trait continu, initialement il y a l'équivalent d'une période dans le stock, à la fin le stock est vide. Pour le flot en pointillé, initialement le stock est vide, à la fin il y a l'équivalent d'une période dans le stock.

Pour chacune des solutions $\phi_f$ de plus court chemin de $[StockSP]$, nous pouvons calculer la solution $(s^f, x^f, y^f)$ de $[StockLSsxy]$ (4.1-4.9) (le stock en fin de période n'est pas forcément inférieur au stock en début de période). La solution définie par $(\sum_f \lambda_f s^f, \sum_f \lambda_f x^f, \sum_f \lambda_f y^f)$ est une solution de $[StockCycleLSsxy]$. □

La formulation de flot $[CycleFlot]$ est préférable à la formulation de localisation des dépôts $[FLCycle]$. Elle implique moins de contraintes pour le même nombre de variables. Pour formuler le problème de gestion de stock du scénario B, la formulation compacte du flot cyclique de coût minimum est choisie. Cependant des symétries dues au temps apparaissent : comme précédemment pour le scénario A, un saut de $l$ périodes peut être interchangé avec un saut de $t$ périodes mais en plus on peut décaler la solution de $t$ périodes sans en changer son coût.

• **Résolution par programme dynamique**

Les deux reformulations $[FLCycle]$ et $[CycleFlot]$ n'ont pas permis de montrer que le problème était facile, mais le programme dynamique pouvant résoudre le problème de flot cyclique garde une complexité polynômiale. En effet, pour calculer le coût d'un planning optimal, on utilise le même programme dynamique que pour le cas où le stock initial est connu, mais en plus il faut itérer sur $tinit = 0, \cdots, tmax - 1$.
Soit $\Theta^t_{tinit}$ = le coût d'une politique optimale sur l'horizon $1, \cdots, t$ avec un ramassage en $t$ et un stock initial correspondant à $tinit$ périodes d'accumulation.
Le coût d'une politique optimale est : $\mathcal{O}pt = \min_{tinit}\{\Theta^{T-tinit}_{tinit}\}$

Ce programme dynamique est un algorithme d'optimisation ayant une complexité polynômiale de $O(T \times tmax^2)$, ce problème de gestion de stock cyclique reste donc facile.

## 4.3 Solutions sur un ensemble de périodicités restreint (scénario C)

Dans le scénario C, l'horizon est considéré comme infini. La donnée de l'ensemble $P$ des périodicités des routes restreint l'ensemble des solutions et borne implicitement l'horizon par $T = ppcm\{p \in P\}$. Avec cette politique restrictive, les routes sont utilisées avec une périodicité $p \in P$ et définissent ainsi un planning partiel (les variables de gestion de stock sont passées au sous-problème de tournées). Les contraintes de gestion de stock pour définir un planning réalisable ordonnent ces plannings partiels définis par les routes. Cette section ne donne pas la formulation du problème de gestion de stock à un seul client puisque le problème n'est plus indépendant de la construction des tournées. En lieu et place, on illustre la représentation d'un planning individuel réalisable.

D'abord, du point de vue du chauffeur/véhicule, le planning prévoit qu'il fasse telle route toutes les $p_1$ périodes et une autre route toutes les $p_2$ périodes, etc... Du point de vue client, il sait qu'il est collecté d'une quantité $\ell_1$ toutes les $p_1$ périodes et d'une quantité $\ell_2$ toutes les $p_2$ périodes, etc... De plus, une date de départ $s \leq p$ est définie pour chaque route permettant de connaître l'ordre des collectes. Ainsi chaque planning partiel $r$ est défini par le triplet $(\ell, p, s)^r$, où $\ell \leq \min(tmax_i, p)$ est la quantité collectée, $p \in P$ est la périodicité de la collecte et $s \leq p$ est la date de la première occurrence de la collecte. Le coût d'un planning partiel est un coût moyen par période, i.e c'est le coût lié à la route

qu'il génère, $c^r$, divisé par la périodicité $p$. Le nombre de plannings partiels est borné par $tmax \sum_{p \in P} p$ (on a $\sum_{p \in P} p$ couples $(p, s)$ possibles et au plus $tmax$ plannings partiels pour chaque couple $(p, s)$). Le planning individuel est réalisable si la combinaison $C$ des triplets $(\ell, p, s)^r$ permet de couvrir la demande de chaque client en chaque période $t$ de l'horizon $T$. Une condition nécessaire de réalisabilité est que $\sum_{r \in C} \ell^r \frac{T}{p^r} = T$, où $C$ est l'ensemble des plannings partiels choisis pour visiter le client. L'exemple 4.7 montre une combinaison de triplets (plannings partiels) réalisable et une irréalisable. Cet exemple nous servira à remarquer la symétrie dans la formulation et à visualiser une solution fractionnaire.

Exemple 4.7 – Combinaison de triplets pouvant donner un planning

Soit $tmax = 2$ et $P = \{1, 2, 3, 4\}$, alors $T = 12$. La combinaison de triplets $(\ell, p, s)^r$ :

- C1 $= (2, 4, 1)^{\Upsilon} + (1, 4, 2)^{\boxdot} + (1, 4, 3)^{\circledcirc}$ est réalisable.
- C2 $= (2, 4, 1)^{\Upsilon} + (1, 2, 2)^{\ominus}$ n'est pas réalisable.

Dans la table 4.1, nous pouvons visualiser sur l'horizon $T$ pour chaque combinaison, les périodes couvertes $t$ par chacun des triplets : $\Upsilon$, $\boxdot$, $\circledcirc$, $\ominus$ (i.e. les matrices $\delta$ de chaque triplet). Pour C1, toutes les périodes sont bien couvertes, pour C2 certaines ne le sont pas et d'autres le sont trop.

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1 | $\Upsilon$ | $\boxdot$ | $\circledcirc$ | $\Upsilon$ | $\Upsilon$ | $\boxdot$ | $\circledcirc$ | $\Upsilon$ | $\Upsilon$ | $\boxdot$ | $\circledcirc$ | $\Upsilon$ |
| C2 | $\Upsilon$ | $\ominus$ | | $\Upsilon\ominus$ | $\Upsilon$ | $\ominus$ | | $\Upsilon\ominus$ | $\Upsilon$ | $\ominus$ | | $\Upsilon\ominus$ |

TAB. 4.1 – Visualisation des combinaisons de triplets définissant des plannings

Les plannings représentés par les combinaisons C12 $= (2, 4, 2) + (1, 4, 3) + (1, 4, 4)$, C13 $= (2, 4, 3) + (1, 4, 4) + (1, 4, 1)$ et C14 $= (2, 4, 4) + (1, 4, 1) + (1, 4, 2)$ sont identiques au planning représenté par C1 décalé respectivement de 1, 2 et 3 périodes. Si dans le planning chaque triplet $(\ell, p, s)$ est décalé d'une ou plusieurs périodes la solution reste inchangée. De même, si dans chacune de ces combinaisons, on intervertit la date de départ des deux triplets $(1, 4, .)$ et $(1, 4, .)$ (un sous-ensemble de triplets est décalé), on obtient encore le même planning. La modélisation du scénario C souffre donc d'une symétrie par rapport au temps dans le choix de la date de départ.

Un planning fractionnaire symétrique consisterait à prendre $\frac{1}{4}$ fois chaque combinaison C1, C12, C13 et C14. Sa représentation est dans le tableau 4.2.

Chaque signe a une valeur de $\frac{1}{4}$. La demande de chaque période est bien couverte (chaque colonne a 4 signes). Le signe $\curlyvee$ représente le triplet $(2, 4, .)$ (quelque soit la date de départ) et le signe ◎ le triplet $(1, 4, .)$. Remarquons que cette solution est équivalente à une solution fractionnaire basée sur les triplets et non sur les combinaisons : chaque triplet $(2, 4, .)$ est pris $\frac{1}{4}$ de fois et chaque triplet $(1, 4, .)$ $\frac{1}{2}$ fois.

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1 - $(2, 4, 1)$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ |
| C1 - $(1, 4, 2)$ | | ◎ | | | | ◎ | | | | ◎ | | |
| C1 - $(1, 4, 3)$ | | | ◎ | | | | ◎ | | | | ◎ | |
| C12 - $(2, 4, 2)$ | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ | | |
| C12 - $(1, 4, 3)$ | | | ◎ | | | | ◎ | | | | ◎ | |
| C12 - $(1, 4, 4)$ | | | | ◎ | | | | ◎ | | | | ◎ |
| C13 - $(2, 4, 3)$ | | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ | |
| C13 - $(1, 4, 4)$ | | | | ◎ | | | | ◎ | | | | ◎ |
| C13 - $(1, 4, 1)$ | ◎ | | | | ◎ | | | | ◎ | | | |
| C14 - $(2, 4, 4)$ | | | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ | | | $\curlyvee$ | $\curlyvee$ |
| C14 - $(1, 4, 1)$ | ◎ | | | | ◎ | | | | ◎ | | | |
| C14 - $(1, 4, 2)$ | | ◎ | | | | ◎ | | | | ◎ | | |

TAB. 4.2 – Planning individuel fractionnaire

Le planning optimal est celui de coût minimum sur l'horizon $T$. Le coût d'un planning, représenté par la combinaison $C$, est $\sum_{r \in C} \frac{c^r}{p^r}$.

## 4.4 Analyse comparative des trois scénarios

Dans cette section, un exemple avec deux clients illustre les inconvénients de chaque scénario. Prenons un véhicule de capacité 100 ($V$ est limitatif et vaut 1). Le graphe des connexions entre les deux clients et le dépôt est donné à la figure 4.12. Le nœud 0 est le dépôt et les nœuds 1 et 2 sont les clients. Pour chaque client, nous avons la capacité de stockage $C$, le taux de consommation $d$, ce qui induit l'intervalle maximal entre 2 collectes $tmax$. Sur les arêtes est indiqué le temps entre les deux extrémités. Le coût des solutions est le coût des routes par période ($\alpha$ est fixé à 1).

Dans le cas où $T$ est fini et où la situation initiale est connue (scénario A), l'inconvénient du planning "opérationnel" est que la solution tactique dépend de la situation initiale donnée, comme le montre l'exemple 4.8.

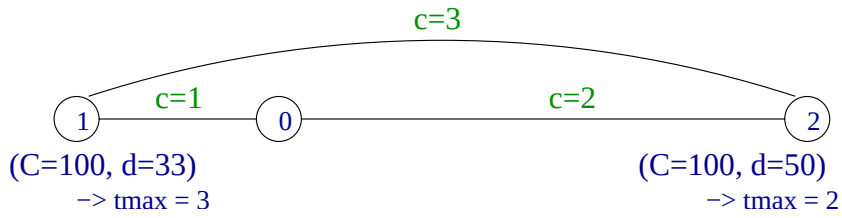
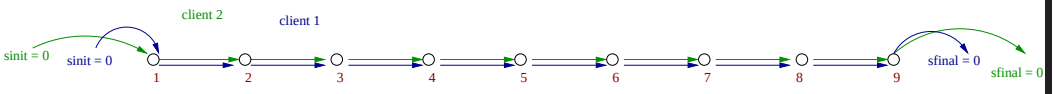



FIG. 4.12 – Graphe de connexion entre deux clients et le dépôt

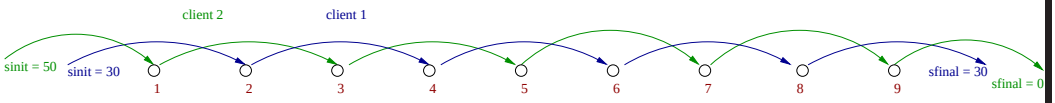
Exemple 4.8 – La solution du scénario A dépend de *tinit*

Soit $T = 9$. Si pour les 2 clients, le stock initial est :

- $tinit = 0$, le coût de la solution optimale est de **6 par période**.




- $tinit = 1$, il existe une meilleure solution de **3.11 par période**.




($sinit = tinit * d_i$ et $sfinal$ est le stock à la fin de la période 9.)

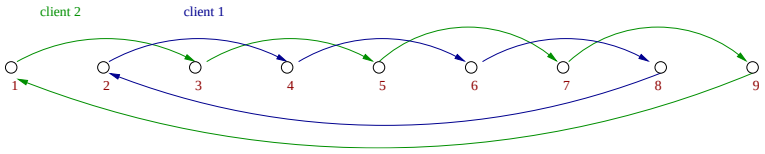
Pour éviter ceci, le stock initial devient une variable (scénario B). Dans ce cas, le planning est cyclique et la solution est plus "tactique", cependant $T$ agit sur son équilibre comme on peut le voir sur l'exemple 4.9.

Pour éviter que le choix de $T$ perturbe le résultat de l'optimisation, on peut à la place choisir $P$ (scénario C). Cette solution tactique représente mieux l'équilibre du système si $P$ est bien choisi, comme le montre l'exemple 4.10.

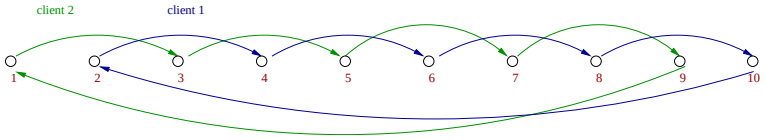
Exemple 4.9 – La solution du scénario B dépend de $T$

Si l'horizon des temps :

- $T = 9$, alors la solution optimale a un coût de **3.11 par période**.




- $T = 10$, alors il existe une meilleure solution de **3 par période**.




Exemple 4.10 – La solution du scénario C dépend de $P$

- Si $P = \{1, 2, 3\}$,
  alors $T = 6$ et la solution optimale a un coût de **3 par période** :
  - 1 est ramassé toutes les périodes paires.
  - 2 est ramassé toutes les périodes impaires.
- Si $P = \{1, 3\}$,
  alors $T = 3$ et la solution optimale a un coût de **6 par période** :
  - 1 et 2 sont ramassés toutes les périodes.

Pour conclure ce chapitre, dans le cas d'un horizon fini $T$ fixé explicitement (scénarios A et B de la table 1.3), le problème d'élaboration d'un planning individuel dicté par la gestion du stock du client est un problème facile. On peut le formuler comme un problème de flot, cependant dans le cas où le stock initial n'est pas connu (scénario B), la matrice associée n'est pas TU. Dans les trois scénarios, une symétrie due au temps apparaît, mais moins importante pour le scénario A. D'après l'exemple sur deux clients, le scénario C où l'ensemble des périodicités $P = \{1, \cdots, Pmax\}$ est choisi explicitement s'avère le mieux adapté à notre problématique.

# 5

# Formulation de l' "Inventory Routing Problem" tactique

Dans ce chapitre, le problème de l'IRP tactique est formulé en se basant sur la deuxième décomposition présentée à la section 2.3. Plusieurs modélisations sont possibles selon le choix effectué pour le problème de construction de tournées (étudié au chapitre 3) et pour le problème d'élaboration du planning dicté par la gestion des stocks (étudié au chapitre 4). Dans la décomposition de Dantzig-Wolfe choisie, le problème de construction de tournées constitue le sous-problème. La route opérationnelle de la table 1.1 est un problème de plus court chemin $[ESPPRC]$ défini en (3.7-3.18) et le cluster homogène de la table 1.1 est un problème de sac à dos $[KP]$ défini en (3.30-3.36). En revanche, le problème de planification tactique est formulé explicitement dans le problème maître. Dans le cas des scénarios A et B de la table 1.3, la formulation du problème d'élaboration d'un planning individuel est respectivement le problème de flot $[StockFL]$ défini en (4.18-4.22) et $[CycleFlot]$ défini en (4.21-4.22) et (4.54). Dans le cas du scénario C de la table 1.3, les contraintes d'élaboration d'un planning individuel sont les contraintes (1.23) et s'expriment à l'aide d'une matrice indicatrice $\delta$ qui est calculée à partir des valeurs des variables du sous-problème de tournées.

La section 5.1 est consacrée à la formulation du scénario B où la solution est restreinte à être périodique sur un horizon des temps fini. La formulation du scénario A où la solution est opérationnelle peut facilement être déduite en adaptant la formulation du scénario B. Le cas du scénario C où l'horizon est infini mais la périodicité des routes doit être choisie dans un ensemble restreint

est formulé dans la section 5.2. Ensuite ces deux formulations ayant comme sous-problème le modèle de route opérationnelle sont comparées numériquement sur des petits exemples dans la section 5.3. Les modèles de tournées (route opérationnelle ou cluster de points à visiter) sont comparés dans la section 5.4 dans le cas du scénario C.

Les observations déduites de ces expériences numériques sont :

- en choisissant $T$ suffisamment grand (expérimentalement on note que $T \geq 2\,tmax$ suffit) pour le scénario B, et $P = \{1, \cdots, tmax\}$ pour le scénario C, les deux formulations des sections 5.1 et 5.2 donnent la même borne duale.
- Si $P = \{1, \cdots, tmax\}$ pour le scénario C et $T$ est fixé égal au *ppcm* des périodicités de $P$ pour le scénario B, alors l'espace des solutions du scénario C est inclus dans l'espace des solutions du scénario B. La formulation du scénario C permet de trouver la borne duale plus rapidement. De plus, les solutions entières obtenues par l'heuristique primale sont typiquement plus simples à mettre en pratique.
- le problème de tournées modélisant des clusters homogènes permet de traiter des données de plus grande taille.

Ainsi nous développerons dans la suite de la thèse le modèle du scénario C avec un sous-problème de tournées basé sur les clusters homogènes. La dernière section 5.5 introduit notre batterie de tests sur la base desquels nous évaluerons nos algorithmes de résolution.

## 5.1 Modélisation du scénario B

Dans le cas du scénario B de la table 1.3, l'horizon $T$ est fixé et le stock initial n'est pas connu mais doit être égal au stock final. Les contraintes de gestion de stock pour chaque client sont celles de la formulation de flot $[CycleFlot]$ donnée dans la section 4.2. Elles appartiennent au maître. Ces contraintes utilisent la variable $\Phi_{ikt}$ indiquant si le client $i$ est collecté à la période $t$, sa dernière collecte datant de la période $k$.

Le sous-problème issu de la décomposition de Dantzig-Wolfe est le problème de tournées qui consiste à générer une route opérationnelle ou un cluster homogène $q$ de coût minimum qui est utilisé à une période $t$. Soit $Q = \cup_t (c^q, \phi^q, z^q)_{q \in Q(t)}$ l'ensemble des solutions pour chaque période $t$ du sous-

problème de tournées $[ESPPRC]$ ou $[KP]$. Rappelons que nous utilisons abusivement le mot route pour désigner une tournée opérationnelle ou un cluster de points de visite. Pour une route $q$, $c^q$ dénote son coût. Les variables $\phi^q_{i\ell}$ indiquent pour chaque client $i$ visité dans la route $q$ la quantité collectée $\ell$ ; les variables $z$ décrivent la tournée ou le cluster correspondant (i.e. les arêtes faisant partie de la route ou de l'étoile représentant le cluster).

Notons $\lambda_{qt} = 1$ la variable indiquant si la route $q \in Q(t)$ est choisie pour la date $t$, zéro sinon ; et $Vmax$ le nombre de véhicules nécessaires. $Vmax$ est borné par $V$. Si la variable de gestion de stock $\Phi_{ikt}$ est égale à 1, alors la quantité qui doit être collectée à la période $t$ est $t-k$. Le lien entre les variables de gestion de stock $\Phi$ et les variables du sous-problème de tournées $\phi$ est donné par la relation :

$$\Phi_{ikt} = \sum_{q \in Q(t)} \phi^q_{i\ell} \lambda_{qt} \;\; \forall\, i,k,t, \;\; \text{tel que } \ell \; = \begin{cases} t-k & \text{si } k < t \\ t+T-k & \text{sinon} \end{cases}$$

Le problème global (maître) peut se formuler comme suit :

$$[FormB] \quad Z^B_{PLE} = \min \; Vmax \; + \alpha \sum_{t,q \in Q(t)} c_q \lambda_{qt} \tag{5.1}$$

$$\sum_{\ell} \sum_{q \in Q(t)} \phi^q_{i\ell} \lambda_{qt} \;=\; \sum_{k} \sum_{q \in Q(t)} \phi^q_{i(k-t)} \lambda_{qk} \qquad \forall i,t \quad (\beta_{it}) \tag{5.2}$$

$$\sum_{t,\ell : t-\ell \le 0} \sum_{q \in Q(t)} \phi^q_{i\ell} \lambda_{qt} \;=\; 1 \qquad \forall i \quad (\nu_i) \tag{5.3}$$

$$\sum_{q \in Q(t)} \lambda_{qt} \;\le\; Vmax \qquad \forall\, t \quad (\sigma_t) \tag{5.4}$$

$$\lambda_{qt} \;\in\; \{0,1\} \qquad \forall\, q,t \tag{5.5}$$

$$Vmax \;\in\; I\!N. \tag{5.6}$$

(5.2) et (5.3) sont les contraintes de $[CycleFlot]$ exprimées à l'aide des colonnes $q$. Les égalités (5.3) peuvent être relaxées en $\geq$ pour plus de stabilité dans la résolution (les variables duales $\nu$ sont ainsi limitées à être positives). Les inégalités (5.4) définissent le nombre de véhicules utilisés. Des contraintes de convexité limitant le nombre de colonnes pouvant être prises peuvent être ajoutées :

$$\sum_{q} \lambda_{qt} \;\le\; V \quad \forall t \tag{5.7}$$

Si le nombre de véhicules $V$ est limitatif, une restriction supplémentaire sur tous les plannings peut aider : à chaque période, la quantité ramassée doit être

inférieure à la capacité totale des véhicules :

$$\sum_i d_i \sum_q \ell \phi_{i\ell}^q \lambda_{qt} \quad \leq \quad WV \quad \forall t$$

Dans notre cas, ces contraintes sont typiquement non contraignantes.

Cette formulation met en jeu un grand nombre de variables, $O(|Q|T)$, la relaxation linéaire est donc résolue par génération de colonnes (méthode présentée dans la section 2.1). Un problème maître restreint au sous ensemble $\overline{Q}$ de colonnes générées jusqu'à présent est résolu à l'optimalité, la solution duale $(\beta, \nu, \sigma)$ est utilisée pour tester l'optimalité pour le problème non restreint. Pour chaque période $t$, le coût réduit minimum $\overline{c_t}(\beta, \nu, \sigma)$ des colonnes $q \in Q(t)$, est calculé en résolvant un sous-problème de tournées (problème de pricing), s'il est négatif alors la colonnes générée $q^*$ est ajoutée au maître, $\overline{Q(t)} = \overline{Q(t)} \cup \{q^*\}$. Si aucune colonne n'est ajoutée, la solution courante est $PL$ optimale.

**Observation 5.1.1** *Variables duales et coût réduit*

1. *$NT + N + T$ variables duales sont passées au problème de pricing. $\beta_{it}$ correspond à un coût ou une récompense (selon que $\beta$ est positif ou négatif) pour visiter le client $i$ à la période $t$, $\nu_i$ le coût de visite du client $i$ et $\sigma_t$ le coût d'utilisation d'un véhicule à la période $t$.*
2. *A chaque itération, $T$ problèmes de pricing ([ESPPRC] défini en (3.7-3.18) ou [KP] défini en (3.30-3.36)) doivent être résolus. Pour le problème de pricing $t$, notons par $\pi_{i\ell}$ la récompense de visite associée au client $i$ pour la collecte de $\ell$ périodes, alors :*

$$\pi_{i\ell} = \begin{cases} \beta_{it} - \beta_{ik} & \text{avec } \ell = t - k, \text{ si } k < t \\ \nu_i + \beta_{it} - \beta_{ik} & \text{avec } \ell = t + T - k, \text{ si } k > t \end{cases}$$

*Les problèmes de pricing [ESPPRC] et [KP] deviennent respectivement :*

$$[ESPPRC] \quad \zeta_t(\beta, \nu) = \min\{\alpha \sum_{i,j} c_{ij} z_{ij} + \sum_{i,\ell} (\alpha f_i - \pi_{i\ell}) \phi_{i\ell}, (\phi, z) \in X^{ESPPRC}\}$$

$$[KP] \quad \zeta_t(\beta, \nu) = -\max\{\sum_{i,\ell} \pi_{i\ell} \phi_{i\ell} - \alpha \sum_k c'_k z_{kk} - \alpha \sum_{ki} c'_{ki} z_{ki}, (\phi, z) \in X^{KP}\}$$

*où $X^{ESPPRC}$ et $X^{KP}$ sont définis respectivement en (3.8-3.18) et en (3.31-3.36). Le coût réduit de la colonne optimale $q^* \in Q(t)$ est donc :*

$$\overline{c_t}(\beta, \nu, \sigma) = \zeta_t(\beta, \nu) + \sigma_t$$

*La colonne $q^*$ générée par le problème de pricing $t$ est valide pour les autres périodes.*

3. *La borne lagrangienne* (2.13) *est obtenue en relaxant d'une manière lagrangienne les contraintes (5.2-5.4) et prend la forme :*

$$L(\beta,\nu,\sigma) = (1 - \sum_t \sigma_t)Vmax + \sum_i \nu_i + V\sum_t \overline{c_t}(\beta,\nu,\sigma)$$

## 5.2 Modélisation du scénario C

Dans le cas du scénario C de la table 1.3, l'horizon est théoriquement infini mais on restreint l'espace des solutions en fixant l'ensemble des périodicités $P$. Un cycle $T = ppcm\{p \in P\}$ est ainsi défini. Le sous-problème issu de la décomposition de Dantzig-Wolfe consiste à générer une route opérationnelle ou un cluster homogène $q$ de coût minimum qui est utilisé toutes les $p^q$ périodes, sa première occurrence étant à la date $s^q \leq p^q$. La solution du problème de tournées est une route périodique. En fixant dans le problème de tournées la périodicité $p \in P \subseteq \{1, \cdots, Pmax\}$ et la date de départ $s \leq p$, on retrouve les problèmes $ESPPRC$ ou $KP$ où la quantité collectée pour chaque client est limitée par la périodicité : $\ell \leq \min(tmax_i, p^q)$. Par conséquent, pour obtenir la solution optimale $q^*$ du problème de pricing, il faut itérer sur les différentes périodicités $p$ et les différentes dates de départ $s$ et pour chacune d'elle calculer la solution $(\phi, z)$ de $[ESPPRC]$ ou de $[KP]$. Ainsi, le nombre de problèmes de tournées résolu est $\sum_{p\in P} p$. L'algorithme est présenté dans la table 5.1.

1. Initialisation : $\overline{c} = 0$ et $q^* = (0, 0, 0, 0)$.
2. Pour $p \in P$ faire :
   (a) pour $s = 1, \cdots, p$
      i. Calculer la solution $(\phi, z)^{p,s}$ de coût réduit $\zeta_{ps}$ minimum de $[ESPPRC]$ ou du $[KP]$
      ii. Si $\overline{c} > \zeta_{ps}$ alors $\overline{c} = \zeta_{ps}$ et $q^* = (p, s, (\phi, z)^{p,s})$.
3. Retourne $\overline{c}$ et $q^*$.

TAB. 5.1 – Procédure de pricing dans le cas du scénario C

Les contraintes de gestion de stock (2.27) imposent un ordonnancement de ces routes périodiques de telle sorte que le stock produit à chaque période $t$ du

cycle $T$ soit collecté par une route. Pour chaque route $(p^q, s^q, \phi^q, z^q)$, les valeurs $\delta^q$ indiquant les périodes d'utilisation du véhicule et les demandes couvertes pour chaque client sont calculées (la matrice $\delta$ est introduite à la section 1.2). La route $q$ utilise un véhicule à la période $t$ si $t$ est de la forme $kp^q + s^q$ avec $k \in I\!N$, ainsi :

$$\delta^q_{0t} = \begin{cases} 1 & \text{si la route } q \text{ utilise un véhicule à la période } t \text{ ie si } \frac{t-s^q}{p^q} \in I\!N \\ 0 & \text{sinon} \end{cases}$$

Supposons que la route $q$ ramasse chez le client $i$ le contenu de $\ell$ périodes (ie $\phi^q_{i\ell} = 1$). Les demandes couvertes par la route $q$ sont celles des périodes $t$ telles que $t \in \{s^q - \ell + 1, \cdots, s^q - 1, s^q, s^q - \ell + 1 + p^q, \cdots, s^q - 1 + p^q, s^q + p^q, \cdots\}$. La demande de la période $t$ est couverte s'il existe $\tau \in \{0, \cdots, \ell - 1\}$ tel que $t = kp^q + s^q - \tau$ avec $k \in Z\!\!Z$, ainsi :

$$\delta^q_{it} = \begin{cases} 1 & \text{si le stock du client } i \text{ produit à la période } t \text{ est collecté par la route } q \\ & \text{ie si } \sum_\ell \phi^q_{i\ell} = 1 \text{ et } \exists \tau \in \{0, \cdots, \sum_\ell \ell \phi^q_{i\ell} - 1\} \text{ tel que } \frac{t+\tau-s^q}{p^q} \in Z\!\!Z \\ 0 & \text{sinon} \end{cases}$$

Avec le scénario C, le problème global prend la forme :

$$[FormC] \quad Z^C_{PLE} = \min \; Vmax + \alpha \sum_{q \in Q} \frac{c^q}{p^q} \lambda_q \tag{5.8}$$

$$\sum_q \delta^q_{it} \lambda_q \geq 1 \qquad \forall i, t \quad (\beta_{it}) \tag{5.9}$$

$$\sum_q \delta^q_{0t} \lambda_q \leq Vmax \qquad \forall t \quad (\sigma_t) \tag{5.10}$$

$$\lambda_q \in \{0, 1\} \qquad \forall\, q \tag{5.11}$$

$$Vmax \in I\!N. \tag{5.12}$$

Pour chaque client $i$, les contraintes (5.9) imposent que le stock produit à la période $t$ soit collecté par une route. Ces contraintes ont été mises sous la forme "$\geq$" au lieu de "$=$", sans que cela induise une relaxation. Les contraintes (5.10) définissent le nombre de véhicules nécessaires. Un véhicule pouvant faire au plus $Pmax$ routes, la contrainte de convexité peut être ajoutée :

$$\sum_q \lambda_q \leq Pmax\, V \tag{5.13}$$

Néanmoins, cette contrainte est typiquement non active et ne servira que pour le calcul de la borne Lagrangienne.

Cette formulation $[FormC]$ est aussi appelée formulation discrète due à la discrétisation du temps sur le cycle $T$. Le nombre de variables est $O(|Q|)$, la relaxation linéaire est donc résolue par génération de colonnes.

**Observation 5.2.1** *Variables duales et coût réduit*

1. *$NT + T$ variables duales sont passées au problème de pricing. $\beta_{it}$ correspond au coût de visite du client $i$ à la période $t$ et $\sigma_t$ le coût d'utilisation d'un véhicule à la période $t$.*

2. *A chaque itération sont résolus $\sum_{p\in P} p$ problèmes de pricing ([ESPPRC] défini en (3.7-3.18) ou [KP] défini en (3.30-3.36)). A $(p, s)$ fixé, les valeurs duales $\pi_{i\ell}$ sont calculées et correspondent à la somme des récompenses collectées $\beta_{it}$ sur tout le cycle $T$. A chaque occurrence de la route, soit aux périodes $t$ telles que $\frac{t-s}{p} \in I\!N$, les récompenses collectées sont $\beta_{it}, \cdots, \beta_{i\,t-\ell+1}$. Ainsi*

$$\pi_{i\ell} = \sum_{\tau=0}^{\ell-1} \sum_{t, \frac{t+\tau-s}{p}\in Z\!\!Z} \beta_{it}$$

*Les problèmes [ESPPRC] et [KP] deviennent respectivement :*

$$[ESPPRC] \quad \zeta_{p,s}(\beta) = \min\{\frac{\alpha}{p}\sum_{i,j} c_{ij}z_{ij} + \sum_{i,\ell}(\frac{\alpha}{p}f_i - \pi_{i\ell})\phi_{i\ell}, (\phi, z) \in X^{ESPPRC}\}$$

$$[KP] \quad \zeta_{p,s}(\beta) = -\max\{\sum_{i\ell}\pi_{i\ell}\phi_{i\ell} - \frac{\alpha}{p}(\sum_{k} c'_k z_{kk} + \sum_{ki} c'_{ki} z_{ki}), (\phi, z) \in X^{KP}\}$$

*où $X^{ESPPRC}$ et $X^{KP}$ sont définis respectivement en (3.8-3.18) et en (3.31-3.36). Le coût réduit de la colonne optimale $q^* \in Q$ est donc :*

$$\overline{c}(\beta, \sigma) = \min_{p\in P, s\leq p}\{(\zeta_{p,s}(\beta) + \sum_{t, \frac{t-s}{p}\in I\!N} \sigma_t)\}$$

*La colonne $q^*$ générée avec une date $s$ est valide pour les autres dates de départ.*

3. *La borne lagrangienne (2.13) est obtenue en relaxant d'une manière lagrangienne les contraintes (5.9-5.10) et prend la forme :*

$$L(\beta, \sigma) = (1 - \sum_t \sigma_t)Vmax + \sum_{i,t}\beta_{it} + Pmax\, V\, \overline{c}(\beta, \sigma)$$

## 5.3 Comparaison numérique des modélisations des scénarios B et C

Nous comparons les solutions numériques de deux modélisations : $[FormB]$ définie en (5.1-5.6) et $[FormC]$ définie en (5.8-5.12). Pour ces tests, le problème de pricing est un plus court chemin élémentaire avec contraintes de ressources résolu avec le code de Feillet et al. [42] (présenté dans le section 3.1). Dans les exemples et tests qui suivent, $\alpha$ est fixé à 1 et $V$ est limitatif. Ces tests ont été effectués sur 5 petits jeux de données aléatoires dont les caractéristiques sont résumées dans le tableau 5.2. Les instances comportent N clients tel que $C_i = 100$ et $tmax_i \in \{1, 2, 3\}$, V véhicules de capacité (cap) et G=V signifie que le garage et le vidage se situent au même endroit.

| Nom | N | V(cap) | G=V |
|---|---|---|---|
| Stat0 | 6 | 1(250) | non |
| Stat1 | 8 | 2(300) | oui |
| Stat2 | 10 | 2(300) | oui |
| Stat3 | 8 | 2(250) | non |
| Stat4 | 12 | 2(400) | non |

TAB. 5.2 – Caractéristiques de 5 petits fichiers tests aléatoires

Pour $[FormC]$, $P = \{1, 2, 3\}$ et $T = 6$, la formulation comporte $6(N + 1)$ contraintes et à chaque itération il y a 6 appels à l'oracle. Pour $[FormB]$, deux longueurs d'horizon $T$ sont essayées, $T = 6 = 2\,tmax$, $T = 9 = 3\,tmax$, la formulation comporte $T(N + 1) + T \geq 6(N + 1)$ contraintes et à chaque itération il y a $T \geq 6$ appels à l'oracle.

Comme dans la pratique le nombre de clients ayant des périodicités de 4 à 6 n'est pas négligeable, un autre problème test est généré avec $tmax_i \in \{1, 2, 3, 5, 6\}$. Cette instance comporte $N = 12$ clients tels que $C_i = 200$ et $V = 2$ véhicules de capacité 600, le garage est différent du vidage. Pour $[FormC]$, $P = \{1, 2, 3, 4, 5, 6\}$ et $T = 60$, la formulation comporte 780 contraintes et à chaque itération 21 appels à l'oracle sont faits. Pour $[FormB]$, trois longueurs d'horizon $T$ sont essayées, $T = 18 = 3tmax$, $T = 21$ et $T = 60$, la formulation comporte respectivement 246, 283 et 792 contraintes et à chaque itération il y a $T$ appels à l'oracle.

Les résultats sont résumés dans le tableau 5.3 pour la moyenne des 5 ins-

tances ayant $tmax_i \in \{1,2,3\}$ et dans le tableau 5.4 pour l'instance avec $tmax_i \in \{1,2,3,5,6\}$. Les colonnes "Bornes" sont respectivement la borne duale sur l'horizon $T$ (notée DB), et la borne duale par période (notée DB/T). Les "Compteurs" sont respectivement le nombre d'appels au problème de pricing (noté SpSol), et le nombre de colonnes générées (noté Col). Les "Temps", exprimés en tic ($\frac{1}{100}$ de secondes), sont respectivement le temps passé dans l'oracle ESPPRC (noté ESPPRC), dans le problème de pricing (ce temps inclut en plus le temps de préparation avant l'appel, le temps de l'enregistrement de la colonne retournée) (noté SpSol), le temps passé à résoudre le maître restreint (noté RM), et le temps total (noté Total).

| | Bornes | | Compteurs | | Temps (tic) | | | |
|---|---|---|---|---|---|---|---|---|
| Form | DB | DB/T | SpSol | Col | ESPPRC | SpSol | RM | Total |
| $[FormB]$ , T=6 | 1349.02 | 224.837 | 488 | 440 | 24191 | 24388 | 147 | 4m8s81t |
| $[FormB]$ , T=9 | 2023.54 | 224.837 | 857 | 753 | 33122 | 33430 | 272 | 5m44s68t |
| $[FormC]$ , T=6 | 1349.02 | 224.837 | 55 | 52 | 10631 | 10652 | 32 | 1m47s53t |

TAB. 5.3 – Solution du maître $PL$ de $[FormB]$ et $[FormC]$ sur les instances de 5.2

| | Bornes | | Compteurs | | Temps (tic) | | | |
|---|---|---|---|---|---|---|---|---|
| Form | DB | DB/T | SpSol | Col | ESPPRC | SpSol | RM | Total |
| $[FormB]$ , $T = 18$ | 5078.55 | 282.142 | 1539 | 1496 | 539137 | 540689 | 639 | 1h30m48s24t |
| $[FormB]$ , $T = 21$ | 5924.97 | 282.142 | 1810 | 1718 | 633190 | 635103 | 893 | 1h46m46s76t |
| $[FormB]$ , $T = 60$ | Stop au bout de 10h | | | | | | | |
| $[FormC]$ , $T = 60$ | 16928.5 | 282.142 | 347 | 345 | 1566564 | 1567445 | 3867 | 4h24m39s36t |

TAB. 5.4 – Solution du maître $PL$ de $[FormB]$ et $[FormC]$ sur une instance lorsque $tmax_i \in \{1,2,3,5,6\}$

La première observation déduite de ces expériences numériques sur la solution du maître $PL$ est :

**Observation 5.3.1** *Lorsque $T$ est choisi suffisamment grand ($T \geq 2\,tmax$ dans nos tests) pour le scénario B, et $P = \{1, \cdots, tmax\}$ pour le scénario C, les bornes duales par période sont identiques :*

$$\frac{DB}{T} = \frac{Z^B_{PL}}{T} = \frac{Z^C_{PL}}{T}$$

Cette observation nous laisse à penser que la solution $PL$ optimale, quelque soit la formulation, atteint le même équilibre (la meilleure périodicité de collecte

pour chaque client et les meilleures routes pour chaque période) en utilisant des combinaisons convexes de solutions valides pour chacune de ces formulations.

Du point de vue temps de calcul pour obtenir cette borne duale, il est difficile de dire quelle formulation est la meilleure, le nombre de contraintes et le nombre d'appels à l'oracle dépendant des données. Dans le cas du tableau 5.4, $[FormB]$ lorsque $T = 18$ a moins de contraintes et d'appels à l'oracle et est la plus rapide. En revanche dans la cas du tableau 5.3, c'est $[FormC]$ qui est la plus rapide. Il est plus juste de comparer $[FormB]$ lorsque $T = 6$ et $[FormC]$ lorsque $P = \{1, 2, 3\}$ car elles ont le même nombre d'appels à l'oracle à chaque itération et un nombre équivalent de contraintes. $[FormB]$ a appelé 488 fois l'ESPPRC et $[FormC]$ $6 * 55 = 330$ fois l'ESPPRC. Dans ce cas, le temps de résolution du maître $PL$ est moins important avec $[FormC]$ et moins de colonnes sont nécessaires, en effet une colonne dans $[FormC]$ représente $\frac{T}{p}$ colonnes dans $[FormB]$.

Avant de déterminer quelle formulation $[FormB]$ ou $[FormC]$ semble la plus adaptée pour déterminer la borne duale, regardons les solutions entières des formulations $[FormB]$ et $[FormC]$ qui sont obtenues à l'aide d'une heuristique d'arrondi (présentée dans la section 7.3) basée sur la solution $PL$. Ces solutions entières peuvent être bien différentes.

Dans le cas du tableau 5.4, l'horizon $T = 18$ de $[FormB]$ empêche les périodicités égales à 4 et 5 (18 n'étant pas un multiple de 4 et 5) alors qu'avec $T = 60$, toutes les périodicités $p \in \{1, 2, 3, 5, 6\}$ sont permises. Ainsi de meilleures solutions ont plus de chance d'être trouvées avec $T = 60$ qu'avec $T = 18$ ; mais cela entraîne une augmentation du temps de résolution importante. Dans le cas où $T$ est grand, $[FormC]$ est plus adaptée pour obtenir la borne duale. Cependant, lorsque $P = \{1, 2, 3, 4, 5, 6\}$ pour le scénario C et $T = 60$ pour le scénario B (la longueur choisie pour l'horizon $T$ pour le scénario B correspond à la longueur du cycle induit par $P$ pour le scénario C), les routes dans la solution périodique du scénario B peuvent avoir une périodicité de $10, 12, 15, 20, 30, 60$ : l'espace des solutions du scénario B contient l'espace des solutions du scénario C. Dans ce cas, il peut donc exister des solutions entières de $[FormB]$ qui ont un coût inférieur à la solution optimale de $[FormC]$, mais ces solutions peuvent être moins intéressantes d'un point de vue pratique, les routes pouvant en effet différer à chaque période, comme on peut le voir sur l'exemple 5.1. L'observation déduite de ces expériences numériques est donc :

**Observation 5.3.2** *Si l'ensemble de périodicités $P$ pour le scénario C est tel que $P = \{1, \cdots, tmax\}$, et la longueur de l'horizon $T$ pour le scénario B est choisie égale à la longueur maximale du cycle de régénération du scénario C, i.e. $T = ppcm(p \in P)$, alors l'espace des solutions du scénario B contient l'espace des solutions du scénario C. Néanmoins, la borne duale est trouvée plus rapidement avec $[FormC]$. De plus, les solutions entières obtenues sont typiquement plus pratiques.*

Exemple 5.1 – Illustration du fait que la solution du scénario C est plus pratique

Prenons dans le scénario C, $P = \{1, 2, 3\}$ et dans le scénario B, $T = 6$. La table 5.5 et les figures 5.1, 5.2 permettent de visualiser les solutions obtenues pour les scénarios C et B sur l'instance Stat0. Dans la table 5.5, un signe dans la case $(i, t)$ signifie que la route associée au signe couvre la demande de la période $t$ du client $i$. Les figures 5.1 et 5.2 présentent respectivement les routes du scénario C et B.

- La solution optimale du scénario C a un coût de 83.8 par périodes (l'optimalité est prouvée à l'aide du branchement disjonctif de la section 6.4.2), voir la figure 5.1 et la partie gauche de la table 5.5 (♈ est la route avec $s = 2$ et ✓ avec $s = 1$).
- Il existe une solution du scénario B de coût 79.133 par période, voir la figure 5.2 et la partie droite de la table 5.5 (✓ est la route pour $t = 1$, ♈ pour $t = 2$, ♑ pour $t = 3$, ◎ pour $t = 4$, ⊡ pour $t = 5$, ⊖ pour $t = 6$). Remarquons que si la périodicité 6 était permise, cette solution pourrait être obtenue par la formulation du scénario C.

| | Scénario C | | | | | | Scénario B | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta$ | t1 | t2 | t3 | t4 | t5 | t6 | t1 | t2 | t3 | t4 | t5 | t6 |
| i2 | ♈ | ♈ | ♈ | ♈ | ♈ | ♈ | ♈ | ♈ | ◎ | ◎ | ⊡ | ♈ |
| i3 | ♈ | ♈ | ♈ | ♈ | ♈ | ♈ | ♈ | ♈ | ♑ | ⊖ | ⊖ | ⊖ |
| i4 | ✓ | ♈ | ✓ | ♈ | ✓ | ♈ | ✓ | ◎ | ◎ | ◎ | ⊖ | ⊖ |
| i5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ♈ | ♈ | ◎ | ◎ | ⊖ | ⊖ |
| i6 | ♈ | ♈ | ♈ | ♈ | ♈ | ♈ | ✓ | ♑ | ♑ | ⊡ | ⊡ | ✓ |
| i7 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ♑ | ♑ | ⊡ | ⊡ | ✓ |

TAB. 5.5 – Table des demandes couvertes des solutions des scénarios C et B

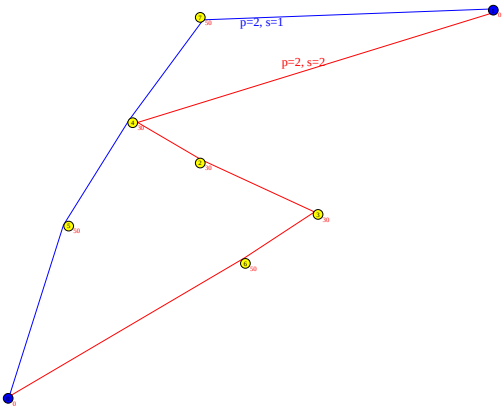



FIG. 5.1 – Solution du scénario C

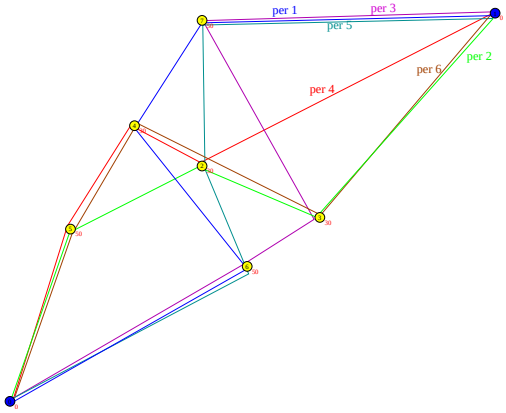



FIG. 5.2 – Solution du scénario B

En conclusion, la modélisation $[FormC]$ semble mieux adaptée à notre problème, du point de vue temps de calcul pour trouver la solution $PL$ optimale et aussi du point de vue type de solution entière obtenue. Cependant, pour pouvoir résoudre des problèmes de plus grande taille, il va falloir chercher à accélérer le temps d'obtention de la solution $PL$ optimale. Une première accélération peut être due au choix de modélisation du sous-problème de transport (vu au chapitre 3) : ESPPRC ou KP.

## 5.4 Comparaison numérique des modèles de tournées

Les deux modélisations de la table 1.1 pour le problème de tournées sont comparées avec la formulation $[FormC]$ définie en (5.8-5.12). L'oracle KP utilise le programme dynamique de Pisinger [71] pour résoudre le problème de sac-à-dos à choix multiple lorsque le centre est fixé. Pour diminuer le nombre d'appels à l'oracle, deux stratégies basées sur la génération de plusieurs colonnes à chaque itération sont essayées :

- dans le cas de l'oracle ESPPRC, le programme dynamique peut produire plusieurs chemins. Nous retournons les 5 meilleures colonnes en plus de la colonne optimale (option notée MC5) ;
- comme une colonne est valide pour toutes les dates de départ, alors les colonnes retournées sont enregistrées pour les autres dates (option all).

Les résultats sur la moyenne des 5 mêmes instances définies à la table 5.2 sont résumés dans le tableau 5.6. Le temps passé dans l'oracle KP est au moins 50

fois moins important que le temps passé dans l'oracle ESPPRC.

| | Bounds | | Counters | | Timers (tic) | | | |
|---|---|---|---|---|---|---|---|---|
| Nom | DB | DB/T | SpSol | Col | Oracle | SpSol | RM | Main |
| moy ESPPRC (MC5+all) | 1349.02 | 224.837 | 9 | 151 | 1279 | 1444 | 12 | 15s13t |
| moy ESPPRC 1col | 1349.02 | 224.837 | 55 | 52 | 10631 | 10652 | 32 | 1m47s53t |
| moy KP (all) | 1330.26 | 221.711 | 30 | 71 | 26 | 79 | 28 | 1s75t |
| moy KP 1col | 1330.26 | 221.711 | 61 | 59 | 55 | 90 | 42 | 2s13t |

TAB. 5.6 – Solutions du maître $PL$ avec les oracles ESPPRC et KP sur les instances de 5.2

Comme précédemment, des tests sont effectués avec $P = \{1, 2, 3, 4, 5, 6\}$, sur l'instance avec $N = 12$ et sur une plus grande avec $N = 24$ clients ayant une capacité de 100 et $V = 3$ véhicules de capacité 600. Les résultats sont rapportés dans le tableau 5.7.

| | Bornes | | Compteurs | | Temps (tic) | | | |
|---|---|---|---|---|---|---|---|---|
| Param | DB | DB/T | SpSol | Col | Oracle | SpSol | RM | Total |
| N = 12 | | | | | | | | |
| ESPPRC-1 col | 16928.5 | 282.142 | 347 | 345 | 1566564 | 1567445 | 3867 | 4h24m39s36t |
| ESPPRC-MC5+all | 16928.5 | 282.142 | 10 | 1051 | 26315 | 74228 | 414 | 12m58s4t |
| KP-1 col | 16863.1 | 281.052 | 2366 | 2364 | 3947 | 10642 | 48276 | 25m55s44t |
| KP-all | 16863.1 | 281.052 | 162 | 858 | 189 | 26714 | 1643 | 6m17s21t |
| N = 24 | | | | | | | | |
| ESPPRC-MC5+all | STOP au bout de 1h, toujours dans le premier oracle | | | | | | | |
| KP-all | 80.8349 | 1.34725 | 163 | 915 | 178 | 80645 | 3212 | 17m44s68t |

TAB. 5.7 – Solutions du maître $PL$ avec les oracles ESPPRC et KP sur deux instances plus grandes

En moyenne sur les 6 instances résolues par les deux oracles, 440 colonnes sont générées pour l'oracle KP contre 100 pour l'oracle ESPPRC, par contre le temps passé dans l'oracle KP est nettement inférieur. Pour l'instance où $N = 12$, les stratégies "MC5+all" ont permis de diviser le temps passé dans l'oracle ESPPRC par 59, mais ce temps reste plus de 100 fois supérieur au temps passé dans l'oracle KP avec la stratégie "all". Pour ces 6 instances, le nombre moyen de clients par véhicule est de l'ordre de 3 alors que pour l'instance où $N = 24$ ce nombre passe à 6. Dans ce cas, l'oracle ESPPRC ne permet plus d'obtenir une solution en temps raisonnable. L'observation déduite de ces expériences numériques est :

**Observation 5.4.1** *Le problème de tournées modélisant des clusters homogènes permet de traiter des données de plus grande taille.*

On remarque que la borne duale est du même ordre de grandeur. Cependant dans ces instances, il y a peu de clients par véhicule et le coût du sac-à-dos sur un petit ensemble de clients est très proche du coût d'un chemin sur cet ensemble (et est même égal lorsqu'il y a 2 clients). Comme les deux oracles ne visent pas le même objectif, malgré les coûts proches, la solution entière obtenue peut être différente, comme l'illustre l'exemple 5.2 ci-dessous.

Exemple 5.2 – Solution de stat0 selon le modèle de tournées

- Rappelons que la solution optimale avec l'oracle ESPPRC a un coût de 83.8 par périodes, voir la figure 5.1.
- Cette solution traduite en clusters, en choisissant comme centres le sommet 4 pour la route de la date $s = 1$ et le sommet 3 pour la route de la date $s = 2$, garde le même coût.
  Cependant, avec l'objectif des clusters, il existe une meilleure solution de coût $83,65$ par périodes, voir la figure 5.3.

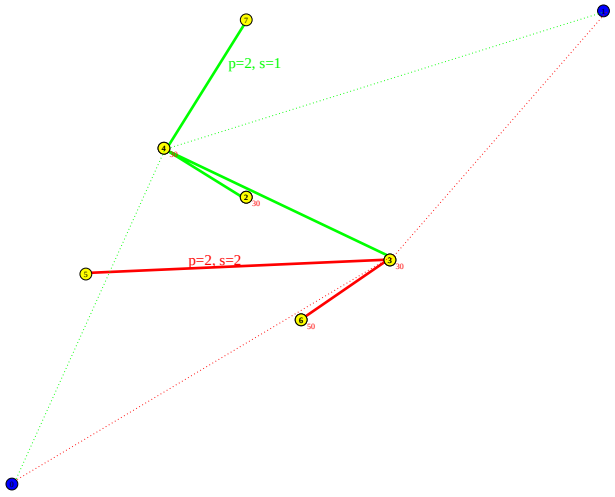



FIG. 5.3 – Solution avec fréquence du scénario C et oracle KP

Pour conclure ces sections, la formulation $[FormC]$ avec l'oracle KP répond le mieux à notre problématique du niveau tactique. C'est cette modélisation qui est utilisée pour obtenir des solutions duales et primales par la suite.

## 5.5 Base de tests

Pour mettre au point notre algorithme, une base de tests est générée comportant 5 instances extraites du jeu de données réelles représenté à la figure 1.1 et basé sur huit communes des Charentes : Pons, Archiac, Jonzac, Saint-Genis, Mirambeau, Montendre, Montlieu, Montguyon. Ensuite $P$ est composé des périodicités les plus fréquentes, i.e $P = \{1, 2, 3, 4, 5, 6\}$ et $T = 60$. Pour certaines communes, seules les bornes ayant $tmax \leq 6$ sont considérées. Ces 5 instances ont en moyenne 60 clients, le groupe est noté S60. Deux autres instances de taille plus importante sont issues de la base. Ces extraits sont définis dans le tableau 5.8. Pour que le coût d'une route soit inférieur au coût d'utilisation d'un véhicule, dans l'objectif $\alpha$ est fixé à 0.5

| Nom | N | tmax |
|---|---|---|
| Groupe S60 | | |
| Jonzac | 39 | 14 |
| Nord (Pons, Archiac) | 66 | 14 |
| Ouest (Saint-Genis, Mirambeau) | 65 | 14 |
| SudP6 (Montendre, Montlieu, Montguyon) | 65 | 6 |
| NordJonzacP6 (Nord+Jonzac) | 67 | 6 |
| GrandNord (Nord+Jonzac+Ouest) - GN | 172 | 14 |
| Probleme1 (un peu partout) - Pb1 | 157 | 6 |

TAB. 5.8 – Caractéristiques des extraits de la base Charentes

De plus, pour élargir la base de tests, 10 instances aléatoires de taille $N = 100$ se rapprochant des caractéristiques des problèmes réels sont générées. Ces instances font partie du groupe AL. Les coordonnées des bornes, du garage et du vidage sont générées aléatoirement selon les trois répartitions possibles : urbain, rural, mixte. Pour la génération des coordonnées, des zones sont définies, chaque zone étant pondérée d'une probabilité de $\frac{pr}{20}$, avec $pr$ différent si c'est une borne ou le garage-vidage, voir la figure 5.4. Les temps de parcours sont proportionnels à la distance euclidienne. Chaque borne a une capacité de $3m^3$, leur intervalle maximum entre deux collectes est généré selon les probabilités de $\frac{1}{10}$ pour $tmax = 1, 2$ et de $\frac{2}{10}$ pour $tmax = 3, 4, 5, 6$. Les véhicules ont une capacité de $30m^3$. Le groupe AL contient 4 instances de type mixte (voir la figure 5.6), 4 de type rural (voir la figure 5.7) et 2 de type urbain (voir la figure 5.5).

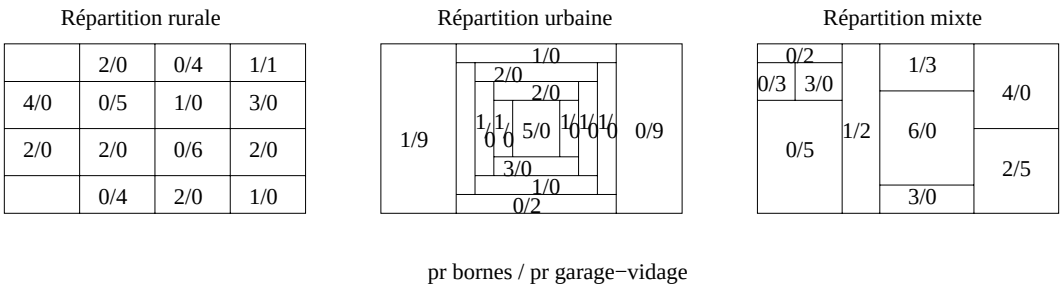



FIG. 5.4 – Probabilités des zones

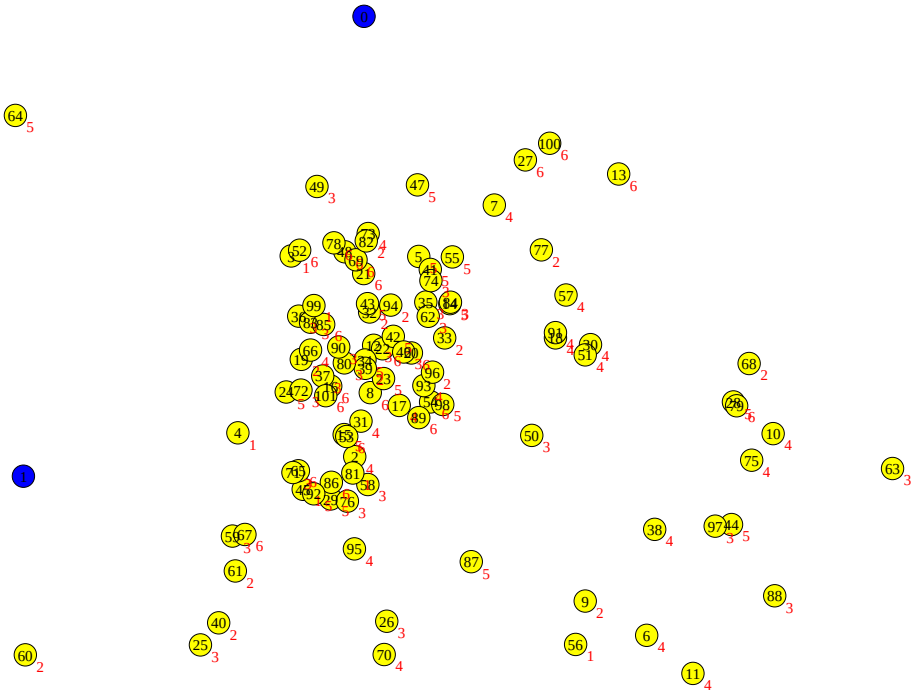


FIG. 5.5 – Exemple de répartition des points clients en zone urbaine

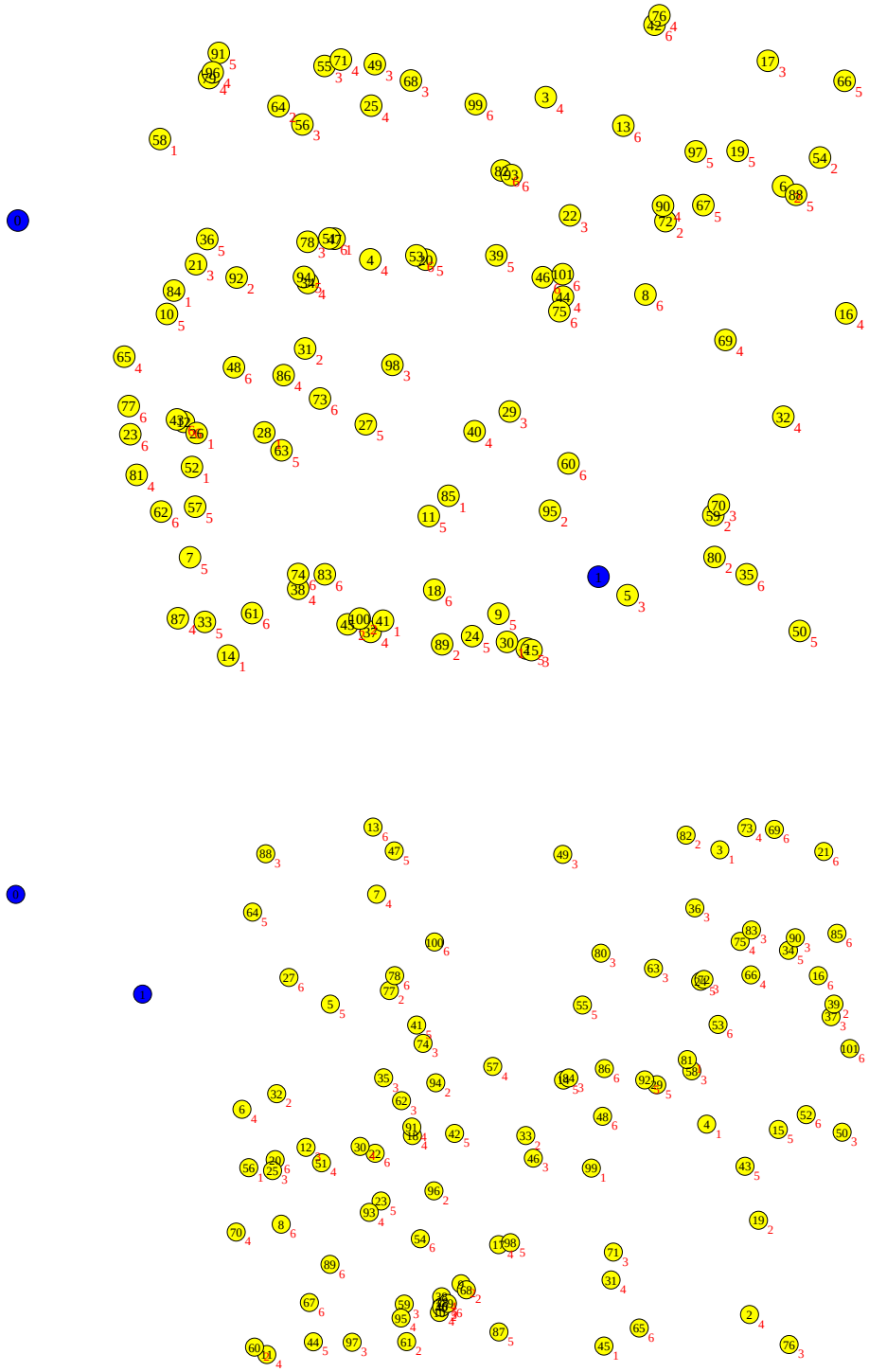


FIG. 5.6 – Exemples de répartition des points clients en zone mixte

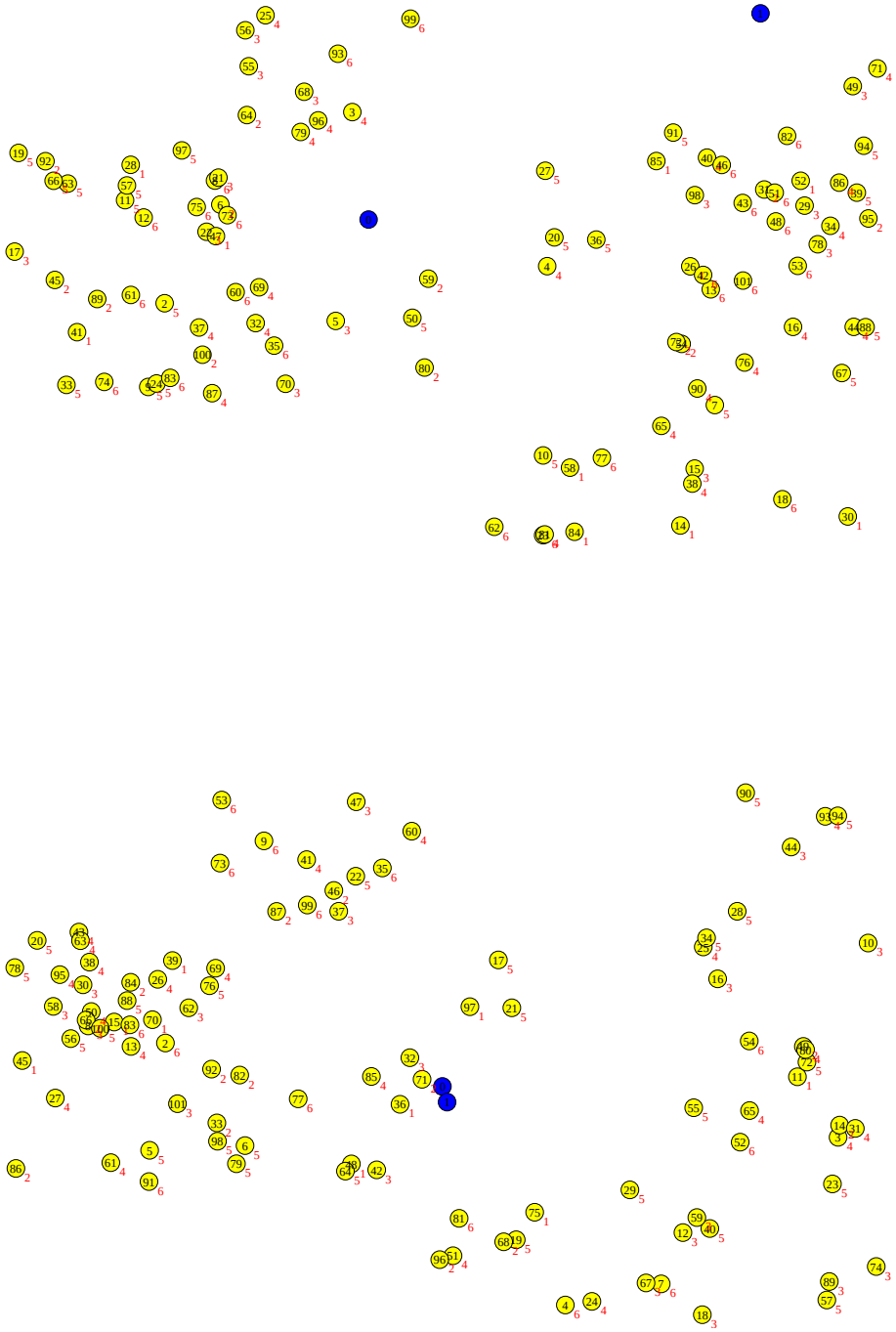


FIG. 5.7 – Exemples de répartition des points clients en zone rurale

# 6

# Bornes duales obtenues par la méthode de Branch-and-Price-and-Cut

Ce chapitre est consacré à la méthode exacte de Branch-and-Price-and-Cut. Une version tronquée de cet algorithme nous permet d'obtenir des bornes inférieures. La meilleure borne obtenue est utilisée afin d'évaluer nos heuristiques primales du chapitre 7.

La formulation discrète $[FormC]$ donnée en section 5.2 par (5.8-5.12), souffre d'une symétrie dans le choix des dates de départ pour chaque route. Pour éviter cet inconvénient, une modélisation agrégée est proposée ici. Elle est obtenue en agrégeant les périodes. Elle modélise un comportement moyen. Techniquement parlant, cela correspond à une relaxation d'état dans l'espace des solutions du sous-problème. Cette technique dite de "state space relaxation" est décrite dans [7]. Toutes les colonnes qui diffèrent seulement par leur date de départ sont considérées comme une seule et même colonne ; cela revient à agréger plusieurs solutions du sous-problème en une seule. Nous montrons que les modèles, discret et agrégé, ont la même solution $PL$ optimale mais une solution $PLE$ optimale différente.

Pour améliorer la qualité de la borne duale fournie par la solution $PL$, nous générons des coupes. On utilise une procédure d'arrondi pour les programmes entiers basée sur des fonctions superadditives (ceci est présenté en section 6.4.1). Ensuite, une procédure de Branch-and-Bound tronquée peut encore améliorer la borne duale : le branchement est effectué seulement sur la variable du maître qui indique le nombre de véhicules nécessaires (voir section

6.4.2). Etant donnée la structure de notre objectif qui est principalement de déterminer la taille de la flotte, ce branchement a un impact important sur la borne. L'amélioration de la borne duale observée en ajoutant des coupes est petite (moins de 2% dans nos tests numériques), par contre l'amélioration obtenue à travers le branchement partiel peut être plus importante (cela dépend de l'instance, elle varie de moins de 1 % à plus de 15%).

Un autre branchement disjonctif basé sur le regroupement de périodicités par classe est proposé. Ce branchement s'appuie sur le fait qu'un véhicule ne peut pas effectuer des routes de périodicités "incompatibles" entre elles. Par exemple, dans le cas où $P = \{1, 2, 3, 4, 5, 6\}$, la périodicité la plus utilisée est 6. Elle n'est pas compatible avec la périodicité 5, i.e. un véhicule utilisé pour une route de périodicité 6 ne peut pas se rendre disponible pour une route de périodicité 5, comme cela est illustré dans l'exemple 6.1. La route de périodicité 6, notée P6, commence à la date $s = 1$ et la route de périodicité 5, notée P5, commence à la date $s = 5$. Ces routes doivent toutes les deux être effectuées à la période $t25$. Pour tout choix de date de départ pour P5, il existe une période $t$ pendant laquelle ces deux routes doivent être effectuées, et par conséquent, couvrir ces deux routes demande deux véhicules distincts. Ce branchement est testé dans le cas où l'espace est restreint à l'ensemble $P = \{1, 2, 3\}$. L'amélioration apportée par ce branchement sur les tests numériques est négligeable (moins de 0.01%).

Exemple 6.1 – Incompatibilité des routes P5 et P6

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t1<br>P6 | t2 | t3 | t4 | t5<br>P5 | t6 | t7<br>P6 | t8 | t9 | t10<br>P5 | t11 | t12 | t13<br>P6 | t14 | t15<br>P5 |
| t16 | t17 | t18 | t19<br>P6 | t20<br>P5 | t21 | t22 | t23 | t24 | t25<br>P6-P5 | t26 | t27 | t28 | t29 | t30 |

## 6.1 Modèle agrégé

Les symétries dont souffre le problème sont identifiées dans la section 2.2. La première symétrie liée à l'utilisation de véhicules identiques disparaît naturellement avec la formulation de Dantzig-Wolfe où une colonne route est valide pour tous les véhicules. La deuxième symétrie est la symétrie en $t$ dans le choix des dates de départ. Les solutions symétriques diffèrent par un changement de dates de départ pour certaines colonnes de la solution.

Cette symétrie rend difficile la résolution du problème $[FormC]$. Les variables duales $\beta_{it}$ associées à la contrainte (5.9) n'ont pas une réelle signification économique et sont plus instables ; le nombre de colonnes devant être générées devient vite important. Ensuite l'heuristique d'arrondi permettant d'obtenir des solutions entières peut devenir peu efficace en tombant sur des solutions équivalentes à différentes passes en fixant la même route périodique mais en prenant une date de départ différente. Enfin le tout a une répercussion sur le temps de calcul.

Notre but premier dans la recherche de l'élimination de la symétrie en $t$ est de diminuer le nombre de variables et contraintes pour une génération de colonnes plus stables et obtenir la solution linéaire plus rapidement. Pour ce faire, les périodes sont agrégées, le modèle résultant modélise un comportement moyen. Pratiquement, nous mettons en œuvre une technique de type "state space relaxation" dans l'espace des colonnes : toutes les colonnes $q$ différant seulement par la date de départ sont agrégées en une seule colonne $r$. L'espace des colonnes est projeté comme suit :

$$\{(c^q, \phi^q, p^q, s^q)\}_{q \in Q} \quad \underrightarrow{proj} \quad \{(c^r, \phi^r, p^r)\}_{r \in R}.$$

A chaque colonne $r$ est associé l'ensemble $Q(r)$ des colonnes $q$ qui se projettent sur $r$ (la cardinalité de $Q(r)$ est $p^r$) :

$$Q(r) = \{q : c^q = c^r, \phi^q = \phi^r, p^q = p^r, s^q \in \{1, \cdots, p^r\}\}.$$

La section 6.1.1 présente la formulation agrégée, notée $[FormAgr]$, ainsi que les modifications apportées au problème de pricing. Un résultat important est donné dans la section 6.1.2 : les formulations discrète et agrégée ont la même solution $PL$ optimale. Des exemples montrent que ces deux formulations ne sont pas équivalentes du point de vue des solutions entières. Du point de vue entier, la formulation agrégée définit une relaxation du problème original. Du point de vue continu, la formulation agrégée définit une reformulation de la relaxation linéaire de la formulation discrète. A travers les tests numériques de la section 6.1.3, nous pouvons voir que la solution $PL$ optimale est obtenue beaucoup plus rapidement avec la formulation agrégée, moins de colonnes nécessitent d'être générées. Par conséquent, la formulation agrégée est utilisée pour calculer les bornes duales. Cependant, la formulation discrète reste utile pour calculer les bornes primales (comme cela est présenté dans le chapitre 7).

### 6.1.1 Problème maître et sous-problème

Soit $\{(c^r, \phi^r, p^r)\}_{r \in R}$ l'ensemble des solutions du problème de tournées indiquant les quantités collectées pour chaque client et la périodicité de la route. A chaque colonne agrégée $r$ est associée la variable $\lambda_r$ qui représente le nombre de fois qu'un véhicule collecte le cluster $r$ avec une périodicité $p^r$, $\lambda_r$ est égale au nombre de dates de départ associées au cluster, i.e.

$$\lambda_r = \sum_{q \in Q(r)} \lambda_q.$$

L'agrégation consiste à sommer les variables et les contraintes sur les périodes du cycle $T$ et définit un comportement moyen si on divise par $T$. Le problème maître dans l'espace des variables $\lambda_r$ associées aux colonnes agrégées prend la forme :

$$[FormAgrC] \quad Z^{AgC}_{PLE} = \min \quad Vmoy + \quad \alpha \sum_{r \in R} \frac{c^r}{p^r} \lambda_r \tag{6.1}$$

$$\sum_{r \in R} \frac{\ell}{p^r} \phi^r_{i\ell} \lambda_r \quad \geq \quad 1 \qquad \forall i \qquad (\pi_i) \tag{6.2}$$

$$\sum_{r \in R} \frac{1}{p^r} \lambda_r \quad \leq \quad Vmoy \qquad (\sigma) \tag{6.3}$$

$$\lambda_r \quad \in \quad I\!N \qquad \forall\, r \tag{6.4}$$

$$Vmoy \quad \in \quad I\!N. \tag{6.5}$$

La variable $Vmoy$ correspond au nombre moyen de véhicules utilisés par période. L'objectif (6.1) minimise le nombre moyen de véhicules utilisés et le coût moyen des tournées par période. Les contraintes (6.2) garantissent pour chaque client que la production d'une période est couverte par les routes en moyenne, chaque route ramassant une fraction de cette demande. La contrainte (6.3) définit le nombre moyen de véhicules utilisés. Notons qu'en multipliant par $T$, on peut amener les coefficients dans les contraintes à être entiers.

**Proposition 6.1.1** *$[FormAgr]$ définie en (6.1-6.5) est une formulation agrégée de $[FormC]$ définie en (5.8-5.12) avec $Vmoy \leq Vmax$.*

**Preuve:** La somme sur $t$ des contraintes (5.9) et (5.10) donne (6.2) et (6.3) multipliées par T. Notons que $\sum_t \sum_{q \in Q} \delta^q_{.t} \lambda_q = \sum_{q \in Q} (\sum_t \delta^q_{.t}) \lambda_q$ et $\sum_{q \in Q} \lambda_q = \sum_{r, q \in Q(r)} \lambda_q = \sum_r \lambda_r$. Pour toutes les contraintes, la somme sur $t$ du membre de gauche s'écrit :

$$\sum_t \sum_{q \in Q} \delta^q_{.t} \lambda_q = \sum_{r, q \in Q(r)} (\sum_t \delta^q_{.t}) \lambda_q$$

Commençons par sommer (5.10) : $\sum_t \delta^q_{0t}$ représente le nombre de fois qu'une route $q$ de périodicité $p^q$ utilise un véhicule, i.e. le nombre de périodes du cycle $T$ où ce véhicule est utilisé, ce nombre est indépendant de la date de départ et vaut $\frac{T}{p^q}$. Par conséquent les contraintes (5.10) deviennent :

$$\sum_{r,q\in Q(r)} (\sum_t \delta^q_{0t})\lambda_q = \sum_{r,q\in Q(r)} \frac{T}{p^q}\lambda_q = \sum_r \frac{T}{p^r}\lambda_r \leq T\ Vmax$$

En divisant par $T$ on obtient : $\sum_r \frac{1}{p^r}\lambda_r \leq Vmax$.
Dans une solution continue, $Vmoy = \sum_r \frac{1}{p^r}\lambda_r$ donc $Vmoy \leq Vmax$.
Dans une solution entière, $Vmoy = \lceil \sum_r \frac{1}{p^r}\lambda_r \rceil = \lceil \sum_{r,q\in Q(r)} \frac{\sum_t \delta^q_{0t}}{T}\lambda_q \rceil$. Or $Vmax = max_t(\sum_q \delta^q_{0t}\lambda_q)$ donc $Vmax \geq \sum_q \frac{\sum_t \delta^q_{0t}}{T}\lambda_q$. Comme $Vmax$ est entier, $Vmax \geq \lceil \sum_{r,q\in Q(r)} \frac{\sum_t \delta^q_{0t}}{T}\lambda_q \rceil$, d'où $Vmoy \leq Vmax$. Finalement on retrouve la contrainte (6.3) où $Vmoy \leq Vmax$.

Sommons sur $t$ les contraintes (5.9) : $\sum_t \delta^q_{it}$ représente le nombre de périodes du cycle de régénération $T$ couvertes par la route $q$ de périodicité $p^q$. Dans la route $q$, parmi les $\phi^q_{i\ell}$ tels que $\sum_\ell \phi^q_{i\ell} > 0$, un seul prend la valeur 1, soit $\phi^q_{i\ell^*} = 1$. Ainsi à chaque utilisation, la route $q$ couvre $\ell^*$ périodes et elle est utilisée $\frac{T}{p^q}$ fois ; donc $\sum_t \delta^q_{it} = \frac{T}{p^q}\ell^*\phi^q_{i\ell^*} = T\sum_\ell \frac{\ell}{p^q}\phi^q_{i\ell} = T\sum_\ell \frac{\ell}{p^r}\phi^r_{i\ell}$. Par conséquent, les contraintes (5.9) deviennent :

$$\sum_{r,q\in Q(r)} (\sum_t \delta^q_{it})\lambda_q = \sum_{r,q\in Q(r)} T(\sum_\ell \frac{\ell}{p^q}\phi^q_{i\ell})\lambda_q = \sum_r T(\sum_\ell \frac{\ell}{p^r}\phi^r_{i\ell})\lambda_r \leq T$$

En divisant par $T$ on retrouve les contraintes (6.2). □

L'objectif du problème de pricing (3.30-3.36) à $p$ fixé se simplifie :

$$\zeta_p(\pi) = -\max\{\sum_{i\ell} \ell\pi_i\phi_{i\ell} - \alpha(\sum_{ki}(c'_{kk}z_{kk} + c'_{ki}z_{ki})), (\phi, z) \in X^{KP}\}.$$

$Pmax$ problèmes KP sont résolus à chaque appel. Le coût réduit de la colonne optimale $r^*$ est :

$$\overline{c}(\pi, \sigma) = \min_{p\in P} \frac{1}{p}(\zeta_p(\pi) + \sigma)$$

La borne lagrangienne (2.13) est :

$$L(\pi, \sigma) = (1 - \sigma)Vmoy + \sum_i \pi_i + Pmax\ V\,\overline{c}(\pi, \sigma)$$

### 6.1.2 Comparaison des formulations discrète et agrégée

Dans cette section, les deux formulations $[FormC]$ et $[FormAgr]$ sont comparées. D'abord nous montrons qu'il y a une correspondance des solutions $PL$ de ces deux formulations (comme cela est illustré dans l'exemple 6.2), mais cette correspondance disparaît au niveau des solutions $PLE$.

Exemple 6.2 – Solutions de $[FormAgr]$ et $[FormC]$

Soient deux routes. La route A collecte l'équivalent de 2 périodes de production chez le client 1 toutes les 2 périodes. La route B collecte l'équivalent de 3 périodes de production chez le client 2 toutes les 3 périodes. Par la suite, nous noterons les routes comme suit :

- la route A : $0-1(2)-0$ avec $p_A = 2$.
- la route B : $0-2(3)-0$ avec $p_B = 3$.

où le chiffre entre paranthèses dénote la quantité collectée. Les solutions des deux formulations sont représentées ci-dessous. Pour $[FormAgr]$, ces deux routes sont prises avec une valeur de 1. Pour $[FormC]$, $p$ colonnes sont associées à chaque route, une pour chaque date de départ. La valeur de chacune de ces colonnes est $\frac{1}{p}$. Dans le tableau, nous avons indiqué avec la valeur de la colonne les périodes d'utilisation des véhicules. ✓signifie que la demande du client est collectée de la valeur de la colonne à cette période par cette route.

| $[FormAgr]$ | $[FormC]$ | | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|---|---|
| $\lambda_{rA} = 1$ | $\lambda_{qA}$ | $s_A = 1$ | $\frac{1}{2}$ | ✓ | $\frac{1}{2}$ | ✓ | $\frac{1}{2}$ | ✓ |
| | | $s_A = 2$ | ✓ | $\frac{1}{2}$ | ✓ | $\frac{1}{2}$ | ✓ | $\frac{1}{2}$ |
| $\lambda_{rB} = 1$ | $\lambda_{qB}$ | $s_B = 1$ | $\frac{1}{3}$ | ✓ | ✓ | $\frac{1}{3}$ | ✓ | ✓ |
| | | $s_B = 2$ | ✓ | $\frac{1}{3}$ | ✓ | ✓ | $\frac{1}{3}$ | ✓ |
| | | $s_B = 3$ | ✓ | ✓ | $\frac{1}{3}$ | ✓ | ✓ | $\frac{1}{3}$ |

**Proposition 6.1.2** *Toute solution $PL$ de la formulation $[FormAgr]$ définie en (6.1-6.5) peut être transformée en une solution $PL$ symétrique de la formulation $[FormC]$ définie en (5.8-5.12) de même coût.*

**Preuve:** La colonne $r$ de $[FormAgr]$ est désagrégée en $p^r$ colonnes $q$ pour $[FormC]$, une colonne $q$ pour chaque date de départ possible $s \in \{1, \cdots, p^r\}$. Soit une solution $\{\hat{\lambda}_r\}_{r \in R}$ de $[FormAgr]$, la solution correspondante de $[FormC]$ est obtenue en posant $\hat{\lambda}_q = \frac{1}{p^r}\hat{\lambda}_r$ pour tout $q \in Q(r)$. Le coût des tournées est le même que dans la solution de $[FormAgr]$ puisque $\frac{c^q}{p^q} = \frac{c^r}{p^r}$ et $\hat{\lambda}_r = \sum_{q \in Q(r)} \hat{\lambda}_q$.

Notons que la route $r$ est équitablement partagée entre les $p^r$ périodes, par conséquent toutes les périodes sont identiques : ainsi à chaque période $t$, il y a le même nombre de véhicules utilisés et $\hat{V}moy = \hat{V}max$.

Nous avons donc montré que toute solution $PL$ de $[FormAgr]$ peut être transformée en une solution $PL$ symétrique de $[FormC]$ de même coût.□

**Proposition 6.1.3** *Toute solution $PL$ de la formulation $[FormC]$ définie en (5.8-5.12) peut être transformée en une solution $PL$ de la formulation $[FormAgr]$ définie en (6.1-6.5) de coût inférieur ou égal.*

**Preuve:** Soit une solution $\{\hat{\lambda}_q\}_{q\in Q}$ de $[FormC]$. Nous agrégeons les différentes colonnes $q$ : $q \in Q(r)$ en une colonne $r$ de valeur $\hat{\lambda}_r = \sum_{q\in Q(r)} \hat{\lambda}_q$. Le coût des tournées est le même dans les deux solutions de $[FormAgr]$ et $[FormC]$. Or $\hat{V}moy = \sum_r \frac{1}{p^r}\hat{\lambda}_r \leq \hat{V}max$, par conséquent le coût de la solution $[FormAgr]$ est inférieur ou égal à celui de $[FormC]$.

Nous avons donc montré que toute solution $PL$ de $[FormC]$ peut être transformée en une solution $PL$ de $[FormAgr]$ de coût inférieur ou égal.□

Notons qu'une solution $PL$ de $[FormC]$, $\hat{\lambda}$, telle que sa valeur est strictement supérieure à la valeur de la solution optimale de $[FormAgr]$, i.e. $\hat{Z}^C_{PL} > Z^{AgC*}_{PL}$, ne peut pas être optimale. En effet, la proposition 6.1.2 nous dit que la solution agrégée de $[FormAgr]$ peut se désagréger en une solution de même valeur pour $[FormC]$ qui serait donc meilleure que la solution $\hat{\lambda}$. Une conséquence de ces deux propositions est donc le corollaire suivant :

**Corollaire 6.1.4** *La solution $PL$ optimale de la formulation $[FormAgr]$ définie en (6.1-6.5) peut être transformée en une solution $PL$ optimale de la formulation $[FormC]$ définie en (5.8-5.12) de même coût $Z^{C*}_{PL} = Z^{AgC*}_{PL}$ et vice versa.*

**Proposition 6.1.5** *A l'optimalité, il existe une solution duale $(\pi, \sigma)$ de la formulation $[FormAgr]$ définie en (6.1-6.5) et une solution duale $(\hat{\pi}, \hat{\sigma})$ de la formulation $[FormC]$ définie en (5.8-5.12) qui vérifie :*

$$\forall t \quad \hat{\sigma}_t = \frac{\sigma}{T} \quad et \quad \hat{\pi}_{it} = \frac{\pi_i}{T}$$

**Preuve:** D'après le corollaire 6.1.4, les solutions $PL$ optimales de $[FormAgr]$ et $[FormC]$ ont même coût, les solutions duales de ces deux formulations ont donc aussi même coût.

D'après la proposition 6.1.2 il existe une solution primale $PL$ symétrique pour $[FormC]$, la solution duale associée vérifie : $\hat{\sigma}_t = \sigma'$ et $\hat{\pi}_{it} = \pi'_i \quad \forall t$.

Soit $\pi_i$ et $\sigma$ la solution duale optimale de $[FormAgr]$. Prenons pour $[FormC]$ la solution duale définie par : $\hat{\sigma}_t = \sigma$ et $\hat{\pi}_{it} = \pi_i \quad \forall t$. Cette solution duale a le même coût que la solution primale $PL$ optimale, elle est donc optimale pour $[FormC]$. □

D'après la proposition 6.1.2, nous avons :

**Observation 6.1.6** *Toute solution entière de la formulation* $[FormAgr]$ *définie en (6.1-6.5) peut toujours être transformée en une solution PL symétrique de la formulation* $[FormC]$ *définie en (5.8-5.12).*

Dans l'exemple 6.2, une solution entière de $[FormAgr]$ et transformée en une solution fractionnaire de $[FormC]$. Comme la valeur optimale $Vmoy^*$ de la solution $PLE$ fournit une borne inférieure sur le nombre de véhicules utilisés, si la contrainte $Vmax \geq Vmoy^*$ est ajoutée, la solution symétrique de $[FormC]$ a le même coût que la solution entière de $[FormAgr]$.

**Observation 6.1.7** *Les solutions entières de la formulation* $[FormAgr]$ *définie en (6.1-6.5) ne se traduisent pas forcément en solutions entières réalisables de même coût dans la formulation* $[FormC]$ *définie en (5.8-5.12).*

Une conséquence de cette observation est que la solution entière optimale de $[FormC]$ n'est pas forcément la solution entière optimale de $[FormAgr]$. En fait, $[FormAgr]$ dans sa version entière est la formulation d'une relaxation du problème dans laquelle on modélise un comportement moyen.

Les trois exemples 6.3, 6.4 et 6.5 illustrent des cas de solutions agrégées entières mais irréalisables pour $[FormC]$.

Exemple 6.3 – Un planning individuel n'est pas réalisable

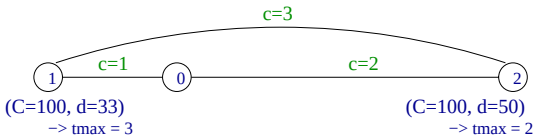
Prenons le cas d'un client avec $tmax = 3$ qui est visité par 2 routes :
- $1*$ la route A : $0 - 1(3) - \cdots - 0$ avec $p_A = 6$
- $1*$ la route B : $0 - 1(2) - \cdots - 0$ avec $p_B = 4$

Mais en pratique, aucun planning construit avec ces 2 routes n'est possible pour ce client : certaines périodes sont couvertes par les 2 routes, d'autres par aucune.
Pour représenter les plannings partiels de chaque route, prenons comme dates de départ $s_A = 6$ et $s_B = 2$ :

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta^A_{it}$ | | | | ✓ | ✓ | A | | | | ✓ | ✓ | A |
| $\delta^B_{it}$ | ✓ | B | | | ✓ | B | | | ✓ | B | | |

Exemple 6.4 – Le nombre de véhicules utilisés est sous estimé

Cet exemple est basé sur la figure ci-dessous et la capacité du véhicule considéré est $W = 100$.




La solution entière optimale consiste à prendre :

- $1*$ la route A : $0-1(2)-0$ avec $p_A = 2$ et $c_A = 2$
- $1*$ la route B : $0-2(3)-0$ avec $p_B = 3$ et $c_B = 4$

Seulement 1 véhicule est comptabilisé à chaque période, $\frac{1}{2} + \frac{1}{3} \leq 1$. Le coût de la solution est $1 + \alpha * (2\frac{1}{2} + 4\frac{1}{3}) = 1 + \alpha(1 + \frac{4}{3})$.

Mais en pratique avec ces 2 routes et 1 véhicule, aucune planification n'est possible : à certaines périodes 2 véhicules sont nécessaires.

Pour représenter l'utilisation du véhicule, prenons comme dates de départ $s_A = 1$ et $s_B = 2$ :

| | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|
| $\delta^A_{0t}$ | A | | A | | A | |
| $\delta^B_{0t}$ | | B | | | B | |

Une solution réalisable est :

- soit de prendre 1 véhicule faisant la route A et la route $C : 0-2(2)-0$ avec $p_C = 2$ et $c_C = 4$, son coût étant $1 + \alpha * (2\frac{1}{2} + 4\frac{1}{2}) = 1 + 3\alpha$ ;
- soit de prendre 2 véhicules, l'un faisant la route A, l'autre la route B, de coût $2 + \alpha(1 + \frac{4}{3})$.

Exemple 6.5 – Les plannings individuels de deux clients ne sont pas réalisables

Prenons le cas de deux clients avec respectivement $tmax = 4$ et 5 qui sont visités par 2 routes :

- $1*$ la route A : $0-1(5)-2(4)-\cdots-0$ avec $p_A = 6$
- $1*$ la route B : $0-1(1)-2(2)-\cdots-0$ avec $p_B = 6$

Mais en pratique, aucun planning utilisant ces deux routes n'est possible pour ces deux clients. Soit $s_A = 1$, alors $s_B = 2$ mais le planning du client 2 n'est pas réalisable, ou $s_B = 3$ mais le planning du client 1 n'est pas réalisable.

| | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|
| $\delta^A_{1t}$ | A | | ✓ | ✓ | ✓ | ✓ |
| $\delta^A_{2t}$ | A | | | ✓ | ✓ | ✓ |
| $\delta^B_{1t}$ | | B | | | | |
| $\delta^B_{2t}$ | ✓ | B | | | | |

### 6.1.3 Comparaison numérique des deux formulations

L'implémentation de l'algorithme de Branch-and-Price nécessaire à la résolution de $[FormAgr]$ ou $[FormC]$ est celle du code générique Bapcod [83] ; le maître fait appel à la librairie XPress-MP [87]. Les tests sont effectués sur un bi-processeur Xeon 3GHz avec 2Go de mémoire. Les abréviations utilisées dans les tableaux des résultats sont données dans la table 6.1.

- Nom = le nom de l'instance ou du groupe.
- Bornes :
  INC = le coût d'une première solution entière.
  rootPB = le coût de la meilleure solution entière sur le cycle $T$ à la racine.
  bestPB = le coût de la meilleure solution entière connue sur le cycle $T$.
  rootDB = la borne duale sur le cycle $T$ à la racine.
  bestDB = la meilleure borne duale sur le cycle $T$ obtenue par l'algorithme.
  am = l'amélioration relative de la borne duale (ou primale) par rapport à rootDB (ou rootPB).
  dev = l'écart relatif entre la meilleure borne primale et la meilleure borne duale connue (déviation à l'optimum).
- Compteurs :
  V = le nombre de véhicules utilisés dans la solution.
  SpSol = le nombre de problèmes de pricing appelés.
  Col = le nombre de colonnes générées.
- Temps en secondes :
  MCKP = le temps passé dans l'oracle.
  SpSol = le temps passé à résoudre le problème de pricing (préparation pour le ou les appels à l'oracle, temps de l'oracle, enregistrement de la solution).
  RM = le temps passé à résoudre le problème maître restreint.
  CP = le temps passé dans la procédure des plans coupants (comprend le temps passé dans la routine de séparation, dans la résolution du maître et dans le problème de pricing).
  Ni = le temps passé dans le nœud i.
  Total = le temps total de l'algorithme.

TAB. 6.1 – Abréviations utilisées dans les tableaux des résultats

Pour comparer le temps d'obtention de la solution $PL$ optimale de ces deux formulations $[FormC]$ et $[FormAgr]$, des tests sont effectués dans le cas où $\alpha = 1$ et $V$ est limitatif. Il y a 3 batteries de tests, avec en moyenne 8, 27 et 100 clients. Les 5 instances du premier groupe sont celles présentées dans le tableau 5.2, l'ensemble des périodicités est $P = \{1, 2, 3\}$. Les 10 instances du deuxième groupe sont des extraits de l'instance Problème1 et les 6 instances du troisième groupe sont des instances aléatoires ; pour ces 16 dernières instances l'ensemble des périodicités est $P = \{1, 2, 3, 4, 5, 6\}$. Pour $[FormC]$, la colonne retournée par le problème de pricing est enregistrée pour toutes ces dates de départ. Les résultats du tableau 6.2 montrent que $[FormAgr]$ permet d'obtenir la solution $PL$ optimale plus rapidement avec un gain en temps et en nombre de colonnes générées (lorsqu'il y a une colonne ajoutée dans $[FormAgr]$, $Pmax$ colonnes peuvent être ajoutées dans $[FormC]$). Ainsi des problèmes de taille plus importante peuvent être résolus.

| | Form Discrète | | Form Agrégée | |
|---|---|---|---|---|
| N-V (nb instances) | nbCol | Temps | nbCol | Temps |
| 8-2 (5inst) | 71 | 1s75t | 20 | 25t |
| 27-3 (10inst) | - | >1h | 123 | 5s |
| 100-5 (6inst) | - | - | 701 | 3m52s |

TAB. 6.2 – Comparaison numérique pour l'obtention de la solution $PL$ optimale

Ayant la même solution $PL$ que la formulation discrète, la formulation agrégée est utilisée pour calculer les bornes duales plus rapidement. Dans la suite de ce chapitre, les résultats sont donnés pour la formulation agrégée, chacun des résultats pouvant être traduit dans la formulation discrète. Cependant, comme les solutions entières de la formulation agrégée ne sont pas forcément réalisables, la formulation discrète reste utile pour calculer des bornes primales (voir le chapitre 7). Avant d'aborder les techniques pour améliorer la borne duale, les sections suivantes décrivent l'initialisation du maître puis les stratégies utilisées pour la génération de colonnes.

## 6.2 Initialisation du maître

Le problème maître est initialisé avec une colonne artificielle associée à chaque contrainte (6.2). La colonne associée à la contrainte $i$ est le vecteur $e_i$, i.e. la $i^{\text{ème}}$ colonne de la matrice identité. Son coût est $\pi_i^{init}$, une estimation de la variable duale $\pi_i$. L'interprétation économique de $\pi_i$ est le coût moyen par période de collecte du client $i$. Comme à l'optimalité de la relaxation linéaire,

$\sum_i \pi_i = Vmax^* + \sum_r c^r \lambda_r^*$, le coût moyen par période de collecte du client $i$ comprend un coût fixe d'utilisation des véhicules. L'estimation de ces variables $\pi$ se fait à l'aide d'une heuristique duale.

Dans une solution "idéale", chaque client $i$ est collecté à sa fréquence optimale $tmax_i$, un véhicule est rempli à pleine capacité et un cluster est formé de clients très proches l'un de l'autre. Une solution plus réaliste regroupe rarement ces conditions. Par exemple le cluster visité par le véhicule n'est pas forcément constitué des clients les plus proches les uns des autres, car ces clients ne nécessitent pas forcément la même fréquence de visite.

Le nombre de clients dans un cluster est le nombre de clients qu'un véhicule peut contenir, notons par $N_v$ ce nombre approché par $\lceil \frac{W}{\sum_i C_i/N} \rceil$. Dans le coût d'un cluster, il y a l'aller/retour au garage-vidage et des coûts de connexions entre les clients. Le coût de l'aller/retour au garage-vidage peut être partagé entre les $N_v$ clients, chaque client $i$ prend en charge $\frac{c_{0i}+c_{i1}}{N_v}$ de ce coût. Ensuite les coûts de connexions sont définis par rapport à un centre, un client parmi les $N_v$. La visite de chaque client $i$ entraîne l'ajout d'une arête ; le client $i$ a plus de chance d'être connecté à un de ses voisins les plus proches, donc un parmi les $N_v$ voisins. Cependant, il est peu probable que chaque client $i$ soit toujours avec ces $N_v$ voisins les plus proches à chaque visite, $2\,N_v$ voisins peuvent être considérés pour estimer le coût de connexion. Finalement pour chaque client $i$, une liste des voisins $LV_i$ est définie et ordonnée par $c_{ij}$ croissant. $LV_i$ est tronquée en ne gardant que les $2\,N_v$ voisins les plus proches. L'estimation du coût de visite de $i$ comprend donc son coût de collecte $f_i$, une partie du coût aller/retour au garage-vidage $\frac{c_{0i}+c_{i1}}{N_v}$ et un coût représentant un coût de connexion à un autre client voisin $\frac{\sum_{j \in LV_i} c_{ij}}{|LV_i|}$ :

$$coutVisit_i = f_i + \frac{c_{0i} + c_{i1}}{N_v} + \frac{\sum_{j \in LV_i} c_{ij}}{|LV_i|}$$

La fréquence de collecte du client $i$ est définie par $per_i = max\{p \in P, p \leq tmax_i\}$, ainsi son coût de visite par période est : $\frac{coutVisit_i}{per_i}$.

Le coût fixe d'utilisation des véhicules est basé sur une estimation du nombre de véhicules nécessaires. Pour chaque périodicité $p$, le nombre minimum de clusters à couvrir est comptabilisé, soit $NbMinRoute_p = \lfloor \frac{NbClient(per_i == p)}{N_v} \rfloor$ ce nombre ($NbClient(per_i == p)$ est le nombre de clients ayant une fréquence de collecte $per_i$ égale à la périodicité $p$). Un véhicule peut couvrir

$p$ clusters, le nombre de véhicules à réserver pour chaque périodicité $p$ est $reserv(p) = \lceil \frac{NbMinRoute_p}{p} \rceil$. Ainsi $Vmax$ est estimée égale à $\sum_p reserv(p)$. Finalement,

$$\pi_i^{init} = \frac{\sum_p reserv(p)}{N} + \alpha \frac{coutVisit_i}{per_i}$$

Dans nos tests, ces valeurs initiales optimistes sont multipliées par 1.2 pour donner des valeurs duales plus réalistes.

## 6.3 Stratégies pour la génération de colonnes

Notre procédure de pricing consiste à générer une route périodique. Dans cette section, nous décrivons cette procédure en détail et les différentes stratégies de résolution qui peuvent être utilisées. L'efficacité de ces stratégies est testée numériquement. Dans la section 6.3.1, une observation sur le type de solution $PL$ obtenue est donnée. Nous illustrons cette observation dans l'exemple 6.6 ci-dessous.

Exemple 6.6 – Solution continue optimale observée

Les solutions $PL$ de $[FormAgr]$ sont de la forme :
- $1*$ la route A : $0 - 1(6) - 2(3) - \cdots - 0$ avec $p_A = 6$
- $1*$ la route B : $0 - 2(1) - 3(2) - 4(3) - \cdots - 0$ avec $p_B = 3$
- $\frac{1}{6}*$ la route C : $0 - 2(3) - \cdots - 0$ avec $p_C = 3$
- $\frac{1}{3}*$ la route D : $0 - 3(2) - \cdots - 0$ avec $p_D = 2$

Pour visualiser ces quatres routes, prenons comme date de départ $s = 1$ :

| | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|
| $\delta^A_{1t}$ | A | ✓ | ✓ | ✓ | ✓ | ✓ |
| $\delta^A_{2t}$ | A | | | | ✓ | ✓ |
| $\delta^B_{2t}$ | B | | | B | | |
| $\delta^B_{3t}$ | B | | ✓ | B | | ✓ |
| $\delta^B_{4t}$ | B | ✓ | ✓ | B | ✓ | ✓ |
| $\delta^C_{2t}$ | $\frac{1}{6}C$ | ✓ | ✓ | $\frac{1}{6}C$ | ✓ | ✓ |
| $\delta^D_{3t}$ | $\frac{1}{3}D$ | ✓ | $\frac{1}{3}D$ | ✓ | $\frac{1}{3}D$ | ✓ |

### 6.3.1 Propriétés des solutions

Nous faisons l'hypothèse que $P$ est de la forme $P = \{1, \cdots, Pmax\}$. Nous supposons que le problème maître est résolu par l'algorithme du simplex. Alors dans toutes les colonnes optimales (retournées au maître), il y a au moins un client tel que la quantité collectée est égale à la périodicité de la route,

autrement dit la route générée va satisfaire tous les besoins en collecte de ce client de référence (voir l'exemple 6.6). On en déduit :

**Observation 6.3.1** *Sous l'hypothèse* $P = \{1, \cdots, Pmax\}$*, il existe une solution* $PL$ *optimale telle que :*

$$\forall r \in R : \lambda_r > 0, \exists i, \ell \text{ pour lesquels } \phi^r_{i\ell} = 1 \text{ et } \frac{\ell}{p^r} = 1$$

**Preuve:** Nous allons montrer que chaque colonne $r$ retournée par le problème de pricing vérifie "$\exists i, \ell$ pour lesquels $\phi^r_{i\ell} = 1$ et $\frac{\ell}{p^r} = 1$". Par conséquent la solution $PL$ vérifie cette observation.
Soit la périodicité $p1$ fixée. L'oracle $[KP]$ retourne la solution optimale $(\phi, z)^{r1}$ de coût $\zeta_{p1}(\pi)$. Le coût réduit de la colonne $r_1 = (p1, (\phi, z)^{r1})$ est $\overline{c^{r_1}}(\pi, \sigma) = \frac{1}{p_1}(\zeta_{p1}(\pi) + \sigma)$.
Si pour tout $i$, tel que $\ell$ vérifie $\phi^{r1}_{i\ell} = 1$ on a $\ell < p^{r1}$, alors $(\phi, z)^{r1}$ est aussi une solution optimale du $[KP]$ pour les périodicités $p$ telles que $p1 > p \geq \ell max$ où $\ell max = max(\ell : \phi^{r1}_{i\ell} = 1)$ (dans le problème $[KP]$, des variables non optimales sont mises à 0, i.e. $\phi^{r1}_{i\ell} = 0$ pour $p < \ell \leq p^1$). Pour chaque valeur de $p$, la colonne associée est $r_p = (p, (\phi, z)^{r1})$ de coût réduit $\overline{c^{r_p}}(\pi, \sigma) = \frac{1}{p}(\zeta_{p1}(\pi) + \sigma)$. Parmi ces colonnes, la colonne ayant le plus petit coût réduit est celle ayant la plus petite périodicité possible, i.e. $p = \ell max$, alors $\overline{c^{r_{\ell max}}} = \frac{1}{\ell max}(\zeta_{p1}(\pi) + \sigma)$. Toutes les colonnes optimales retournées $r$ vérifient donc :

$$\exists i, \ell \text{ pour lesquels } \phi^r_{i\ell} = 1 \text{ et } \frac{\ell}{p^r} = 1. \ \square$$

**Remarque:** Comme dans chaque cluster de la solution $PL$, au moins un client est totalement couvert, alors ce cluster ne peut pas être utilisé plus d'une fois : $\lambda_r \in [0, 1] \ \forall r$.

**Remarque:** L'observation 6.3.1 n'est plus vérifiée dans la cas de la solution $PL$ si $P \subset \{1, \cdots, tmax\}$ et dans le cas de la solution entière.

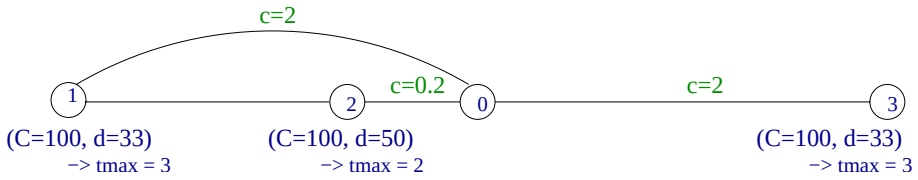
En effet, si $P \subset \{1, \cdots, tmax\}$, d'après la preuve de l'observation 6.3.1, le coût réduit minimum de la colonne (optimale) est pour la plus petite périodicité $p \in P$, mais cette fois-ci la périodicité $p$ peut être plus grande que $\ell max$ car $\ell max$ n'appartient pas forcément à $P$. Pour le cas de la solution entière voir l'exemple 6.7.

**Remarque:** L'observation 6.3.1 peut ne plus être vérifiée si des coupes sont ajoutées.

En effet, les coefficients dans l'objectif du problème de pricing (3.30-3.36) peuvent dépendre des périodicités.

Exemple 6.7 – La solution entière ne vérifie pas l'observation 6.3.1

Prenons l'exemple de la figure ci-dessous.




La capacité du véhicule est $W = 150$. Fixons $\alpha$ à 1.

La solution entière optimale, de coût $1 + \frac{8}{3} + \frac{0.4}{3}$, consiste à prendre :

- $1*$ la route A : $0 - 1(3) - 2(1) - 0$ avec $p_A = 3$ et $c_A = 4$
- $1*$ la route B : $0 - 2(2) - 0$ avec $p_B = 3$ et $c_B = 0.4$
- $1*$ la route C : $0 - 3(3) - 0$ avec $p_C = 3$ et $c_C = 4$

Pour visualiser ces trois routes, prenons comme date de départ $s = 1$ :

| | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|
| $\delta^A_{1t}$ | A | ✓ | ✓ | A | ✓ | ✓ |
| $\delta^A_{2t}$ | A | | ✓ | A | | ✓ |
| $\delta^B_{2t}$ | B | | ✓ | B | | ✓ |
| $\delta^C_{3t}$ | C | ✓ | ✓ | C | ✓ | ✓ |

Si l'observation 6.3.1 doit être vérifiée, la meilleure solution consiste à prendre les routes A et C, la route B avec une périodicité de 2 au lieu de 3 et $\lceil \frac{2}{3} + \frac{1}{2} \rceil = 2$ véhicules. Dans ce cas, la route A ne visite plus le client 2 mais son coût ne change pas. Le coût de cette solution est $2 + \frac{8}{3} + \frac{0.4}{2}$.

### 6.3.2 Problème de pricing

Le problème de pricing génère une route qui est répétée toutes les $p \in P$ périodes. A $p$ fixé, le problème se modélise comme un problème de sac-à-dos $[KP]$ défini en (3.30-3.36). Nous avons vu à la section 3.2.2 (l'observation 3.2.1), qu'une fois le centre $k$ fixé, le problème de construction d'une route se réduit en un problème de sac-à-dos à choix multiple (MCKP). Pour obtenir la solution du problème de pricing, on doit donc itérer sur les périodicités et les centres possibles et résoudre un problème MCKP à chaque itération.

Au pire des cas, on résout $(|P|N)$ sous-problèmes $[MCKP]$ (défini par (3.38-3.40)). La valeur de la solution optimale, $\phi$, du MCKP à $p$ et $k$ fixés est notée $\zeta_{p,k}(\pi)$ :

$$\zeta_{p,k}(\pi) = \max\{\sum_{i\ell}(\ell\pi_i - \alpha c'_{ki})\phi_{i\ell}, \phi \in X^{MCKP}\} \tag{6.6}$$

où la notation $c'_{ki}$ est définie en (3.29) et $X^{MCKP}$ en (3.38-3.40). La colonne $r$ obtenue est $(p, c, \phi)$ avec $c = c'_k + \sum_{i\ell} c'_{ki}\phi_{i\ell}$ de coût réduit $\overline{c^r} = \frac{1}{p}(\sigma + \alpha c'_k - \zeta_{p,k}(\pi))$. Le coût réduit de la colonne optimale $r^*$ est :

$$\overline{c}(\pi, \sigma) = \min_{p\in P, k\in N} \frac{1}{p}(\sigma + \alpha c'_k - \zeta_{p,k}(\pi)) \tag{6.7}$$

Pour diminuer le nombre d'appels à l'oracle MCKP et le nombre de variables, un préprocessing est effectué. En effet, le nombre d'itérations en $k$ peut être diminué : si $\pi_k \leq 0$, alors le sommet $k$ ne sera pas choisi comme centre dans une solution optimale ($y_k = 0$) car le profit associé au sommet $k$ est négatif quelque soit la quantité collectée. Puis à $k$ fixé, la taille de la classe $N_i = \{0, \cdots, \ell^i\}$ peut être réduite : si $\ell\pi_i \leq \alpha c'_{ki}$ alors l'article $\ell^i$ ne sera pas dans le sac-à-dos optimal ($\phi_{i\ell} = 0$) puisqu'il est dominé par l'article nul (son profit étant négatif).

Lors de la résolution d'un MCKP, on peut introduire une "cut off value" représentant le coût réduit de la meilleure colonne obtenue précédemment en ayant résolu d'autres MCKP avec d'autres paires $(p, k)$. Après le calcul de la solution $PL$ optimale du problème de pricing MCKP, un test par borne est effectué pour éventuellement ne pas rechercher la solution entière optimale. Soit $\overline{c}$ le meilleur coût réduit obtenu jusqu'à maintenant et $\zeta^{PL}_{p,k}(\pi)$ le coût de la solution $PL$. Si $\frac{1}{p}(\sigma + \alpha c'_k - \zeta^{PL}_{p,k}(\pi)) > \overline{c}$, alors il n'est pas nécessaire de rechercher la solution entière optimale puisqu'elle ne donnera pas une colonne de coût réduit meilleur.

On peut aussi accélérer le temps de résolution à l'aide d'une autre technique : l'algorithme s'arrête lorsqu'une colonne de coût réduit négatif est trouvée. Si aucune n'est trouvée alors l'optimalité est prouvée. Dans ce cas, l'ordre de résolution des MCKP est important. Pour augmenter nos chances d'identifier des colonnes de coût réduit négatif tôt dans le processus, nous faisons l'itération des périodicités $p$ de la plus grande $Pmax$ à la plus petite car la solution $PL$ du maître a tendance à prendre les colonnes avec de grandes périodicités. Ensuite l'itération des centres $k$ se fait par $\frac{profit(k)}{attract(k)}$ décroissant avec $profit(k)$ le profit maximum que peut apporter l'article $k$, et $attract(k)$ une estimation du coût d'un cluster, avec $k$ comme centre, calculée comme suit : soit la liste des $N_v$ plus proches voisins de $k$, $LV_k$, ordonnée par $c'_{ki}$ croissant, alors $attract(k) = \frac{c_{0k}+c_{k1}+\sum_{i\in LV_k} c'_{ki}}{N_v+2}$.

Enfin, lorsque l'oracle s'arrête à la première colonne de coût réduit négatif trouvée, une post-optimisation est effectuée sur cette colonne pour améliorer son coût réduit. La colonne retournée $r$ peut être améliorée en recherchant le meilleur centre possible $k'$, car le cluster avec un centre non optimisé ne sera jamais dans la solution optimale. La différence de coût réduit pour changer le centre de $k$ à $k'$ est $d(k,k') = \frac{\alpha}{p}(c'_{k'} - c'_k + \sum_{i\ell}(c'_{k'i} - c'_{ki})\phi_{i\ell})$. On itère sur tous les centres potentiels $k' \neq k$ tels que $k'$ appartient au cluster, et $k' = argmin_{k'\neq k,\, \sum_\ell \phi_{k'\ell}=1}\{d(k,k') < 0\}$. D'après l'observation 6.3.1, le coût réduit de cette colonne peut encore être amélioré en prenant comme périodicité $p$ la plus petite possible, i.e $p = \ell max = \max\{\ell, \phi_{i\ell} = 1\}$. Ceci revient à multiplier son coût réduit $\overline{c^r}$ par $\frac{p}{\ell max}$, puis $p$ est posé égal à $\ell max$.

L'algorithme de résolution du problème de pricing est dans la table 6.3. Le paramètre *StopNRC* est vrai si l'on choisit de s'arrêter à la première colonne de coût réduit négatif.

1. Initialisation : $\overline{c}(\pi,\sigma) = 0$ et $r^* = (0,0,0)$.
2. Pour $p = Pmax, \cdots, 1$ faire :
   (a) si $p \notin P$, on passe la valeur suivante de $p$.
   (b) Si *StopNRC* et $\overline{c}(\pi,\sigma) < 0$, STOP.
   (c) Pour $k \in \{2, \cdots, N\}$ ordonné par $\frac{profit(k)}{attract(k)}$ décroissant faire
      i. Preprocessing : si $\pi_k \leq 0$, alors on passe au $k$ suivant.
      ii. Preprocessing : pour chaque $i$, $\ell \leq min(tmax_i, p)$, si $\ell\pi_i \leq c'_{ki}$ alors $\phi_{i\ell} = 0$.
      iii. Résolution $PL$ du MCKP, soit $\zeta^{PL}_{p,k}(\pi)$ son coût.
      iv. Test par borne : si $\frac{1}{p}(\sigma + \alpha c'_k - \zeta^{PL}_{p,k}(\pi)) > \overline{c}(\pi,\sigma)$ alors on passe la valeur suivante de $k$.
      v. Recherche de la solution entière ($\phi$) du MCKP de coût $\zeta_{p,k}(\pi)$, obtention de la colonne $r = (c,\phi,p)$ de coût réduit $\overline{c^r}$.
      vi. Recherche du meilleur centre pour $r$.
      vii. $p = \ell max$
      viii. Si $\overline{c}(\pi,\sigma) > \overline{c^r}$ alors $\overline{c}(\pi,\sigma) = \overline{c^r}$ et $r^* = r$.
3. Retourne $\overline{c}(\pi,\sigma)$ et $r^*$.

TAB. 6.3 – Procédure de pricing

Dans le tableau 6.4, nous comparons numériquement les performances de la procédure de pricing selon qu'on retourne la colonne de meilleur coût réduit ($StopNRC$ est à faux), ou qu'on stoppe à la première colonne de coût réduit négatif ($StopNRC$ est à vrai). Les comparaisons sont effectuées sur la base des problèmes tests présentée à la section 5.5. Les abréviations utilisées dans le tableau 6.4 sont celles de la table 6.1.

| | Bornes | Compteurs | | Temps | | | |
|---|---|---|---|---|---|---|---|
| Nom | rootDB | SpSol | Col | MCKP | SpSol | RM | Total |
| Stratégie1 : Recherche de la colonne de meilleur coût réduit | | | | | | | |
| moy AL | 325.674 | 1449 | 1449 | 236 | 274 | 10 | 4m52s (292) |
| moy S60 | 224.826 | 782 | 781 | 67 | 80 | 2 | 1m26s (86) |
| GN | 517.348 | 2839 | 2839 | 1411 | 1547 | 53 | 27m10s (1630) |
| Pb1 | 601 | 1811 | 1811 | 646 | 720 | 15 | 12m30s (750) |
| Stratégie2 : Stop dès qu'on trouve une colonne de coût réduit négatif | | | | | | | |
| moy AL | 325.674 | 3073 | 3072 | 13 | 40 | 30 | 1m26s (86) |
| moy S60 | 224.826 | 1532 | 1531 | 10 | 19 | 7 | 31s (31) |
| GN | 517.348 | 6467 | 6465 | 180 | 284 | 161 | 8m22s (502) |
| Pb1 | 601 | 3083 | 3082 | 73 | 113 | 27 | 2m44s (164) |

TAB. 6.4 – Comparaison numérique des stratégies pour le problème de pricing

La deuxième stratégie permet un gain en temps assez important par rapport à la première stratégie bien que plus de colonnes soient générées. Ce gain en temps est celui passé à résoudre l'oracle. Par la suite, lorsque cela n'est pas mentionné, le problème de pricing est résolu avec la deuxième stratégie.

## 6.4 Amélioration de la borne duale

Les solutions entières de $[FormAgr]$ non réalisables des exemples 6.3 à 6.5 une fois traduites dans la formulation $[FormC]$, en appliquant la transformation de la preuve de la proposition 6.1.2, donnent une solution fractionnaire (voir l'exemple 6.2). L'ajout de coupes ou de contraintes de branchement pourraient peut être les éliminer, mais nous n'en avons pas. Nous pouvons aussi couper des solutions fractionnaires de $[FormAgr]$. Les exemples 6.8 à 6.10 sont des exemples caractéristiques de solutions fractionnaires rencontrées dans la solution $PL$ de $[FormAgr]$ : les routes sélectionnées pour couvrir les collectes d'un client sont incompatibles.

Eliminer de telles solutions permettrait d'améliorer la borne duale. Pour ne pas réintroduire les dates de départ, les contraintes que nous cherchons à ajouter sont des contraintes agrégées valides pour les deux formulations. Dans la

section 6.4.1, des coupes sont dérivées des contraintes (6.2). Ensuite, une procédure de Branch-and-Bound tronquée peut encore améliorer la borne duale. Dans la section 6.4.2, des contraintes de branchement basées sur l'utilisation des véhicules sont testées.

Exemple 6.8 – Premier type de solution fractionnaire

Soit $tmax = 5$, la couverture du client 1 se fait en prenant :

- $1.2*$ la route A : $0 - 1(5) - \cdots - 0$ avec $p_A = 6$

Cette route peut être prise au plus une fois dans une solution entière. Si $\lambda_{rA} > 1$, alors cela signifie que la route est prise pour deux dates de départ. Prenons par exemple $s_A = 1$ et 2 :

| | | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|---|
| $\delta^A_{1t}$ | $s_A = 1$ | A | | ✓ | ✓ | ✓ | ✓ |
| | $s_A = 2$ | ✓ | A | | ✓ | ✓ | ✓ |

Soit maintenant $tmax = 2$, la couverture du client 1 se fait en prenant :

- $2*$ la route B : $0 - 1(2) - 2(3) - \cdots - 0$ avec $p_B = 6$
- $\frac{5}{6}*$ la route C : $0 - 1(2) - 3(5) - \cdots - 0$ avec $p_C = 5$

La route C ne peut jamais servir pour compléter les collectes de la route B car ces routes sont incompatibles (voir exemple 6.1). Représentons les 12 premières périodes en prenant $s_B = 1$ et 4 (d'après le client 2) et $s_C = 5$.

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta^B_{1t}$ | B | | ✓ | B | | ✓ | B | | ✓ | B | | ✓ |
| $\delta^C_{1t}$ | | | | ✓ | C | | | | ✓ | C | | |

Exemple 6.9 – Deuxième type de solution fractionnaire

Soit maintenant $tmax = 4$, la couverture du client 1 se fait en prenant :

- $1*$ la route A : $0 - 1(4) - 2(6) - \cdots - 0$ avec $p_A = 6$
- $\frac{1}{3}*$ la route B : $0 - 1(3) - 3(6) - \cdots - 0$ avec $p_B = 6$
- $1*$ la route C : $0 - 1(1) - 4(6) - \cdots - 0$ avec $p_C = 6$

Dans une solution entière, la route B ne peut jamais servir pour compléter les routes A et C, elle couvrira des périodes déjà couvertes par A.

| | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|
| $\delta^A_{1t}$ | A | | | ✓ | ✓ | ✓ |
| $\delta^B_{1t}$ | ✓ | ✓ | B | | | |
| $\delta^C_{1t}$ | | C | | | | |

Exemple 6.10 – Troisième type de solution fractionnaire

Soit maintenant $tmax = 6$, la couverture du client 1 se fait en prenant :

- $\frac{1}{3}*$ la route A : $0 - 1(3) - 2(3) - \cdots - 0$ avec $p_A = 3$
- $\frac{1}{3}*$ la route B : $0 - 1(5) - 3(5) - \cdots - 0$ avec $p_B = 5$
- $\frac{1}{3}*$ la route C : $0 - 1(6) - 4(6) - \cdots - 0$ avec $p_C = 6$

Dans une solution entière, aucune de ces 3 routes ne peut être combinée avec une autre, chacune couvrant totalement la demande du client. Seules les 12 premières périodes sont représentées.

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta^A_{1t}$ | A | ✓ | ✓ | A | ✓ | ✓ | A | ✓ | ✓ | A | ✓ | ✓ |
| $\delta^B_{1t}$ | B | ✓ | ✓ | ✓ | ✓ | B | ✓ | ✓ | ✓ | ✓ | B | ✓ |
| $\delta^C_{1t}$ | C | ✓ | ✓ | ✓ | ✓ | ✓ | C | ✓ | ✓ | ✓ | ✓ | ✓ |

### 6.4.1 Ajout de coupes

Pour alléger les notations dans les inégalités présentées ci-dessous, utilisons une variable agrégée du problème original défini en (1.22-1.35) représentant pour chaque client $i$ la quantité $\ell$ ramassée et la périodicité $p$ de cette collecte (i.e. la variable $\sum_s x_{i\ell ps}$), notons cette variable

$$x^i_{\ell p} = \sum_{r:p^r=p} \phi^r_{i\ell}\lambda_r.$$

Dans la première solution fractionnaire de l'exemple 6.8, la route $A$ ne peut pas être prise plus d'une fois et pour la compléter il est nécessaire d'en prendre une autre. Une inégalité valide qui coupe cette solution fractionnaire où $x^1_{56} = 1.2$ est :

$$2x^i_{\ell=p} + x^i_{\ell\neq p} \geq 2.$$

Cette inégalité signifie que pour couvrir deux périodes quelconques consécutives, il faut soit une route qui couvre toutes les périodes, soit deux routes qui couvrent seulement une partie des périodes. Elle est obtenue en multipliant les inégalités (6.2) par $(1+\epsilon)$ puis en arrondissant chaque terme à l'entier supérieur (on retrouve une coupe de Gomory). Or les contraintes (6.2) sont initialement une égalité, i.e. $x^i_{\ell=p} + \frac{\ell}{p}x^i_{\ell\neq p} = 1$, si on retranche à l'inégalité ci-dessus $x^i_{\ell=p} + \frac{\ell}{p}x^i_{\ell\neq p} = 1$, alors on obtient :

$$x^i_{\ell=p} + (1 - \frac{\ell}{p})x^i_{\ell\neq p} \geq 1.$$

Une généralisation de cette inégalité est obtenue en utilisant la proposition 2.1.2 avec comme fonction $F : \mathbb{R} \to \mathbb{R}$ définie par $F(d) = \lfloor d \rfloor$ :

**Proposition 6.4.1** *Pour tout $h = 1, \cdots, T-2$ et $i \in N$ tel que $tmax_i > 1$,*

$$\sum_{r:\, h\ell \bmod p=0} \phi_{i\ell}^r \lambda_r + \sum_{r:\, h\ell \bmod p\neq 0} \left(\left\lceil \frac{h\ell}{p} \right\rceil - \frac{h\ell}{p}\right)\phi_{i\ell}^r \lambda_r \geq 1 \tag{6.8}$$

*est une inégalité valide pour $[FormAgr]$ définie en (6.1-6.5).*

**Preuve:** Soit $h \in I\!N$ et $0 < \epsilon < \frac{1}{p}$. On multiplie les contraintes (6.2) par $(-(h+\epsilon))$ puis on applique 2.17 avec $F(d) = \lfloor d \rfloor$ ; l'inégalité valide est :

$$\sum_r \left\lfloor -\frac{(h+\epsilon)\ell}{p} \right\rfloor \phi_{i\ell}^r \lambda_r \leq \lfloor -(h+\epsilon) \rfloor \tag{6.9}$$

Or $\lfloor -g \rfloor = -\lceil g \rceil$, l'inégalité (6.9) devient :

$$\sum_r \left\lceil \frac{(h+\epsilon)\ell}{p} \right\rceil \phi_{i\ell}^r \lambda_r \geq \lceil h+\epsilon \rceil \tag{6.10}$$

Calculons le second membre puis le coefficient de chacune des variables dans l'inégalité (6.10) :

1. $\lceil h+\epsilon \rceil = h+1$
2. Notons que dans le cas où
   - $h\ell \bmod p = 0$, ie $\frac{h\ell}{p} \in I\!N$, $\lceil \frac{(h+\epsilon)\ell}{p} \rceil = \frac{h\ell}{p} + 1$ ;
   - $h\ell \bmod p \neq 0$, $\lceil \frac{(h+\epsilon)\ell}{p} \rceil = \lceil \frac{h\ell}{p} \rceil$.

En utilisant $\sum_r h\frac{\ell}{p}\phi_{i\ell}^r \lambda_r = h$, l'inégalité (6.10) se réduit en (6.8), et ceci quelque soit $h$. □

**Observation 6.4.2** *Si $tmax_i = 1$ alors les inégalités (6.2) sont plus fortes que les inégalités (6.8).*

En effet si $h \bmod p \neq 0$, $\lceil \frac{h}{p} \rceil - \frac{h}{p} \geq \frac{1}{p}$.

**Observation 6.4.3** *En restreignant $h$ à prendre les valeurs de 1 à $T-2$, on génère toutes les inégalités de type (6.8) qui puissent être obtenue.*

En effet, la suite $u_h^{\frac{\ell}{p}} = \lceil \frac{h\ell}{p} \rceil - \frac{h\ell}{p}$ avec $\frac{\ell}{p}$ fixé représente le coefficient devant la variable $\sum_{r:p^r=p} \phi_{i\ell}\lambda_r$ des différentes inégalités (6.8) lorsque $h\ell \bmod p \neq 0$. Dans le tableau 6.5, on peut visualiser les premiers coefficients de chaque suite $u_h$ dans le cas où $P = \{1,2,3,4,5,6\}$, † signifie que dans (6.8) le coefficient est de 1 (car $h\ell \bmod p = 0$). La ligne $h$ de ce tableau représente les coefficients de l'inégalité $h$. Chaque suite $u_h^{\frac{\ell}{p}}$ avec $\frac{\ell}{p}$ fixé est $p$ périodique, donc l'ensemble des

suites (pour les différentes valeurs de $\frac{\ell}{p}$ avec $p \in P$) sont $T$ périodiques ; ainsi on peut restreindre $h$ à prendre les valeurs de 1 à $T$.
Comme $T = ppcm(p \in P)$, dans (6.8) pour $h = T$ tous les coefficients sont de 1, les inégalités (6.2) sont donc plus fortes que (6.8).
Dans (6.8) pour $h = T-1$, lorsque $(T-1)\ell \bmod p \neq 0$, $\left\lceil \frac{(T-1)\ell}{p} \right\rceil = \frac{T\ell}{p} - \left\lfloor \frac{\ell}{p} \right\rfloor = \frac{T\ell}{p}$, donc $\left\lceil \frac{(T-1)\ell}{p} \right\rceil - \frac{(T-1)\ell}{p} = \frac{\ell}{p}$ et on retrouve le coefficient des inégalités (6.2).
On considère donc qu'en faisant varier $h$ de 1 à $T-2$, on génère tous les jeux de coefficients différents qu'on peut obtenir avec des inégalités de type (6.8).

| h \ $\frac{\ell}{p}$ | 1/2 | 1/3 | 2/3 | 1/4 | 2/4 | 3/4 | 1/5 | 2/5 | 3/5 | 4/5 | 1/6 | 2/6 | 3/6 | 4/6 | 5/6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1/2 | 2/3 | 1/3 | 3/4 | 2/4 | 1/4 | 4/5 | 3/5 | 2/5 | 1/5 | 5/6 | 4/6 | 3/6 | 2/6 | 1/6 |
| 2 | † | 1/3 | 2/3 | 2/4 | † | 2/4 | 3/5 | 1/5 | 4/5 | 2/5 | 4/6 | 2/6 | † | 4/6 | 2/6 |
| 3 | 1/2 | † | † | 1/4 | 2/4 | 3/4 | 2/5 | 4/5 | 1/5 | 3/5 | 3/6 | † | 3/6 | † | 3/6 |
| 4 | † | 2/3 | 1/3 | † | † | † | 1/5 | 2/5 | 3/5 | 4/5 | 2/6 | 4/6 | † | 2/6 | 4/6 |
| 5 | 1/2 | 1/3 | 2/3 | 3/4 | 2/4 | 1/4 | † | † | † | † | 1/6 | 2/6 | 3/6 | 4/6 | 5/6 |
| 6 | † | † | † | 2/4 | † | 2/4 | 4/5 | 3/5 | 2/5 | 1/5 | † | † | † | † | † |
| 7 | 1/2 | 2/3 | 1/3 | 1/4 | 2/4 | 3/4 | 3/5 | 1/5 | 4/5 | 2/5 | 5/6 | 4/6 | 3/6 | 2/6 | 1/6 |

TAB. 6.5 – Les premiers coefficients de $u_h$

**Remarque:** Lorsqu'on regarde les coefficients du tableau 6.5, on voit que le rôle des "gros" ramassages, i.e. tels que $\frac{\ell}{p} > \frac{1}{2}$ (par exemple $\frac{5}{6}$ et $\frac{3}{5}$) est échangé avec celui des "petits" (dans cet exemple $\frac{1}{6}$ et $\frac{2}{5}$), i.e. lorsque le coefficient devant le "gros" ramassage diminue, celui des "petits" augmente. Ainsi pour couvrir la demande du client, il n'est plus intéressant de prendre le "gros" ramassage $\frac{p}{\ell}$ fois avec $1 < \frac{p}{\ell} < 2$ (dans cet exemple $\frac{6}{5}$ et $\frac{5}{3}$ fois) car il existe une valeur de $h$ pour laquelle l'inégalité n'est pas vérifiée (dans cet exemple pour $h = 1$) ; le plus facile pour compléter est de prendre un "petit" ramassage compatible. Nous remarquons aussi que les combinaisons entre les "petits" ramassages (par exemple $2 * \frac{2}{5} + \frac{1}{5}$) sont toujours possibles.

Les inégalités (6.8) coupent aussi la deuxième solution fractionnaire de l'exemple 6.8 où $x^1_{26} = 2$ et $x^1_{25} = \frac{5}{6}$. En effet prenons $h = 2$, alors on doit avoir

$$\frac{2}{6}x^1_{26} + \frac{1}{5}x^1_{25} \geq 1 \quad \text{or} \quad \frac{2}{6}x^1_{26} + \frac{1}{5}x^1_{25} = \frac{5}{6}$$

En revanche la solution fractionnaire de l'exemple 6.9 où $x^1_{46} = 1$, $x^1_{36} = \frac{1}{3}$ et $x^1_{16} = 1$ n'est pas coupée. Dans ce cas, il suffit de vérifier seulement les 4 premières contraintes ($u_h^{\frac{\ell}{6}}$ est 6 périodique), énumérées ici avec seulement les variables non nulles :

$$
\begin{array}{lrcl}
h=1 \rightarrow & \frac{2}{6}x^1_{46}+\frac{3}{6}x^1_{36}+\frac{5}{6}x^1_{16} & = \frac{8}{6} & \geq 1 \\
h=2 \rightarrow & \frac{4}{6}x^1_{46}+1x^1_{36}+\frac{4}{6}x^1_{16} & = \frac{10}{6} & \geq 1 \\
h=3 \rightarrow & 1x^1_{46}+\frac{3}{6}x^1_{36}+\frac{3}{6}x^1_{16} & = \frac{10}{6} & \geq 1 \\
h=4 \rightarrow & \frac{2}{6}x^1_{46}+1x^1_{36}+\frac{2}{6}x^1_{16} & = 1 & \geq 1
\end{array} \tag{6.11}
$$

Si dans l'inégalité (6.11), le coefficient devant $x^1_{36}$ était de $\frac{3}{6}$ (coefficient dans les inégalités (6.2)) alors l'inégalité pour $h=4$ serait toujours valide et la solution serait coupée (cela consisterait à remplacer les † par les coefficients des contraintes (6.2) dans le tableau 6.5, et avec ces coefficients les combinaisons réalisables resteraient réalisables).

Nous avons donc cherché à améliorer les inégalités (6.8) en utilisant dans la proposition 2.1.2 la fonction $F : I\!R \rightarrow I\!R$ définie par $F_\gamma(d) = \lfloor d \rfloor + \frac{(f_d-\gamma)^+}{1-\gamma}$ avec $0 \leq \gamma = b - \lfloor b \rfloor < 1$ où $b$ est le membre de droite (i.e. le $\gamma$ utilisé pour générer des inégalités valides pour les ensembles entiers mixtes [68]) :

**Proposition 6.4.4** *Pour tout $h = 1, \cdots, T-2$ et $i \in N$ tel que $tmax_i > 1$,*

$$
\sum_{r:\ h\ell \mod p=0} \frac{\ell}{p}\phi^r_{i\ell}\lambda_r + \sum_{r:\ h\ell \mod p\neq 0} \left(\left\lceil \frac{h\ell}{p} \right\rceil - \frac{h\ell}{p}\right)\phi^r_{i\ell}\lambda_r \geq 1 \tag{6.12}
$$

*est une inégalité valide pour $[FormAgr]$ définie en (6.1-6.5).*

**Preuve:** Soit $h \in I\!N$ et $0 < \epsilon < \frac{1}{p}$. On multiplie les contraintes (6.2) par $(-(h+\epsilon))$ puis on applique 2.17 avec $F_\gamma(d) = \lfloor d \rfloor + \frac{(d-\lfloor d \rfloor-\gamma)^+}{1-\gamma}$ où $0 \leq \gamma = -(h+\epsilon) - \lfloor -(h+\epsilon) \rfloor = 1-\epsilon < 1$ ; l'inégalité valide est :

$$
\sum_r F_{1-\epsilon}\left(-\frac{(h+\epsilon)\ell}{p}\right)\phi^r_{i\ell}\lambda_r \leq F_{1-\epsilon}(-(h+\epsilon)) \tag{6.13}
$$

Soit $d > 0$, alors

$$
F_{1-\epsilon}(-d) = \lfloor -d \rfloor + \frac{(-d - \lfloor -d \rfloor - (1-\epsilon))^+}{1-(1-\epsilon)} = -\lceil d \rceil + \frac{(-(d - \lceil d \rceil + (1-\epsilon)))^+}{\epsilon}
$$

Soit $(g)^- = \min(g, 0)$, alors $(-g)^+ = \max(0, -g) = -\min(0, g) = -(g)^-$. Ainsi $F_{1-\epsilon}(-d) = -(\lceil d \rceil + \frac{(d - \lceil d \rceil + (1-\epsilon))^-}{\epsilon})$. (6.13) devient donc :

$$\sum_r (\left\lceil \frac{(h+\epsilon)\ell}{p} \right\rceil + \frac{(\frac{(h+\epsilon)\ell}{p} - \lceil \frac{(h+\epsilon)\ell}{p} \rceil + 1 - \epsilon)^-}{\epsilon}) \phi^r_{i\ell} \lambda_r \geq \lceil h + \epsilon \rceil \qquad (6.14)$$

Calculons le second membre puis le coefficient de chacune des variables dans l'inégalité (6.14) :

1. $\lceil h + \epsilon \rceil = h + 1$
2. Notons que dans le cas où
   - $h\ell \mod p = 0$, ie $\frac{h\ell}{p} \in I\!N$
   $\lceil \frac{(h+\epsilon)\ell}{p} \rceil = \frac{h\ell}{p} + 1$
   $(\frac{(h+\epsilon)\ell}{p} - \lceil \frac{(h+\epsilon)\ell}{p} \rceil + 1 - \epsilon)^- = (\frac{\epsilon\ell}{p} - \epsilon)^- = \epsilon(\frac{\ell}{p} - 1)$ car $\frac{\ell}{p} \leq 1$.
   Ainsi $\lceil \frac{(h+\epsilon)\ell}{p} \rceil + \frac{(\frac{(h+\epsilon)\ell}{p} - \lceil \frac{(h+\epsilon)\ell}{p} \rceil + 1 - \epsilon)^-}{\epsilon} = \frac{h\ell}{p} + \frac{\ell}{p}$.
   - $h\ell \mod p \neq 0$
   $\lceil \frac{(h+\epsilon)\ell}{p} \rceil = \lceil \frac{h\ell}{p} \rceil$
   $\frac{(h+\epsilon)\ell}{p} - \lceil \frac{(h+\epsilon)\ell}{p} \rceil = \frac{h\ell}{p} - \lceil \frac{h\ell}{p} \rceil + \frac{\epsilon\ell}{p}$ notons que $\lceil \frac{h\ell}{p} \rceil - \frac{h\ell}{p} \leq 1 - \frac{1}{p}$ et $\frac{\epsilon\ell}{p} > 0$,
   donc $(\frac{(h+\epsilon)\ell}{p} - \lceil \frac{(h+\epsilon)\ell}{p} \rceil + 1 - \epsilon)^- = 0$.
   Ainsi $\lceil \frac{(h+\epsilon)\ell}{p} \rceil + \frac{(\frac{(h+\epsilon)\ell}{p} - \lceil \frac{(h+\epsilon)\ell}{p} \rceil + 1 - \epsilon)^-}{\epsilon} = \lceil \frac{h\ell}{p} \rceil = \frac{h\ell}{p} + \lceil \frac{h\ell}{p} \rceil - \frac{h\ell}{p}$.

En utilisant $\sum_r h\frac{\ell}{p}\phi^r_{i\ell}\lambda_r = h$, l'inégalité (6.14) se réduit en (6.12). □

Les observations 6.4.2 et 6.4.3 restent vraies : si $tmax_i = 1$ alors les contraintes (6.2) restent toujours plus fortes que les inégalités (6.12) et on peut restreindre $h$ à prendre les valeurs de 1 à $T-2$. Dans le tableau 6.5, cette fois ci, † signifie que le coefficient de (6.12) est identique à celui de (6.2), i.e. est égal à $\frac{\ell}{p}$.

Les inégalités (6.12) coupent les solutions fractionnaires des exemples 6.8 et 6.9, mais pas celle de l'exemple 6.10. Dans cet exemple, dans toutes les inégalités le coefficient devant ces ramassages reste le même, égal à 1 car lui seul suffit. De même, la solution entière de l'exemple 6.3 où $x_{24} = x_{36} = 1$ n'est pas coupée, car le coefficient devant ces variables est toujours égal à $\frac{1}{2}$, pouvant être combiné avec lui même. Enfin, ces inégalités (6.12) ne prennent pas en compte les incompatibilités entre les clients.

**Routine de séparation** :
Il y a un nombre polynômial d'inégalités, la plus violée peut donc être trouvée en les énumérant toutes. A chaque fois, $O(NT)$ inégalités sont passées

en revue. Cependant on peut écourter la procédure de séparation en s'arrêtant lorsqu'une inégalité violée par le point fractionnaire courant est identifiée. Dans ce cadre, l'ordre dans lequel est mené l'énumération est important. On observe en pratique que les inégalités correspondant aux petites valeurs de $h$ sont les plus violées. De même, les inégalités avec $i$ tel que $tmax_i$ est grand mais non égal à $Pmax$ sont plus violées que celles avec $tmax_i$ petit ou égal à $Pmax$. En commençant l'énumération par $h$ petit et $tmax$ grand (mais non égal à $Pmax$), la routine de séparation a plus de chance de trouver rapidement une inégalité violée. Dans nos tests sur un ensemble de 10 instances (5 instances du groupe AL et les 5 instances du groupe S60), lorsque l'inégalité retournée est la première trouvée versus la plus violée, le temps passé dans l'algorithme des plans coupants est divisé par 3. La routine de séparation est donnée dans la table 6.6. Si cette routine échoue à trouver une inégalité violée, alors c'est qu'il n'en existe plus.

- Pour $h = 1, \cdots, T-2$
  - Pour $i$ rangé par ordre $tmax_i = Pmax - 1, \cdots, 2, Pmax$
    1. si l'inégalité $(i,h)$ est déjà dans le maître, passé au $i$ suivant.
    2. calculer la violation
    $$\upsilon = 1 - \Big( \sum_{r:\, h\ell \bmod p = 0} \frac{\ell}{p} \phi^r_{i\ell} \lambda_r + \sum_{r:\, h\ell \bmod p \neq 0} \Big( \Big\lceil \frac{h\ell}{p} \Big\rceil - \frac{h\ell}{p} \Big) \phi^r_{i\ell} \lambda_r \Big)$$
    3. si $\upsilon > 0$, ajouter l'inégalité $(i,h)$ au maître, STOP.

TAB. 6.6 – Routine de séparation exacte

L'objectif du problème $[MCKP]$ est modifié par l'ajout des coupes. Considérons la coupe $(i,h)$, posons $\xi_{ih}$ la variable duale associée et $coef_{\phi_{i\ell}}(i,h)$ le coefficient de $\phi_{i\ell}$, alors $\zeta_{p,k}$ devient

$$\zeta_{p,k}(\pi,\xi) \;=\; \max\{\sum_{i\ell}(\ell\pi_i - \alpha\overline{c_{ki}} + p\sum_{h} coef_{\phi_{i\ell}}(i,h)\xi_{ih})\phi_{i\ell}, \phi \in X^{MCKP}\}$$

et le coût réduit de la colonne optimale garde la même forme donnée en (6.7). Pour que le problème maître reste réalisable avec l'ajout des coupes, une variable artificielle globale est utilisée, son coût est une estimation de la solution et est égal à $\sum_i \pi_i^{init}$.

Le tableau 6.7 indique les temps de calcul pour 3 stratégies :

1. une coupe est ajoutée par appel de la routine de séparation ;
2. 10 coupes au plus sont ajoutées ;
3. 10 coupes au plus sont ajoutées et dans l'algorithme des plans coupants, le problème de pricing n'est résolu de manière exacte que toutes les 20 itérations. Cette stratégie permet de retourner de meilleures colonnes régulièrement, par exemple avec de plus petites périodicités et peut faciliter la résolution du maître devenu plus difficile.

C'est avec la troisième stratégie que le temps de calcul est le plus petit, mais il reste important, plus de la moitié du temps est passé dans la procédure des plans coupants (séparation plus réoptimisation du maître).

| | Compteurs | | Temps | | | | |
|---|---|---|---|---|---|---|---|
| Nom | SpSol | Col | MCKP | SpSol | RM | CP | Total |
| Stratégie 1 : 1 coupe/iter | | | | | | | |
| moy AL | 9462 | 9371 | 380 | 488 | 549 | 1330 | 24m45s (1485) |
| moyS60 | 3603 | 3546 | 103 | 130 | 68 | 224 | 4m48s (288) |
| GN | 12559 | 12415 | 932 | 1168 | 943 | 2405 | 51m12s (3072) |
| PB1 | 10693 | 10414 | 1395 | 1601 | 848 | 3856 | 1h11m57s (4317) |
| Stratégie 2 : 10 coupes/iter | | | | | | | |
| moy AL | 9622 | 9609 | 301 | 406 | 531 | 992 | 20m31s (1231) |
| moy S60 | 3582 | 3574 | 86 | 112 | 68 | 180 | 4m13s (253) |
| GN | 13687 | 13663 | 829 | 1083 | 1003 | 1866 | 44m53s (2693) |
| Pb1 | 10157 | 10115 | 1275 | 1465 | 686 | 2534 | 51m9s (3069) |
| Stratégie 3 : 10 coupes/iter, pricing exacte dans l'algorithme CP toutes les 20 iter | | | | | | | |
| moy AL | 8624 | 8611 | 276 | 370 | 424 | 851 | 17m49s (1069) |
| moy S60 | 3072 | 3065 | 66 | 89 | 50 | 143 | 3m27s (207) |
| GN | 11798 | 11770 | 752 | 970 | 759 | 1619 | 39m1s (2341) |
| Pb1 | 9163 | 9123 | 1018 | 1191 | 556 | 2001 | 41m15s (2475) |

TAB. 6.7 – Comparaison numérique de différentes stratégies d'implémentation de la génération de coupes

Le tableau 6.8 montre les bornes duales obtenues après l'ajout des coupes "rootDB" et l'amélioration obtenue par rapport à notre borne initiale "initDB" pour les 2 groupes d'instances AL et S60 et les deux plus grandes instances GN et Pb1. L'amélioration de la borne duale est assez faible, elle est de 1.03% en moyenne sur les 17 instances. Les colonnes "nbCoupe" et "nbIV" indiquent respectivement le nombre de coupes ajoutées et le nombre d'inégalités valides (6.12) (ce nombre est égal à $58 \sum_{i \in N: tmax_i > 1} \mathbb{1}_{\{tmax_i > 1\}}$ où $\mathbb{1}_{\{tmax_i > 1\}} = 1$ si $tmax_i > 1$, sinon 0). En moyenne sur les 17 instances, 7.63% des inégalités valides sont ajoutées à la formulation $[FormAgr]$.

| | Bornes | | | Compteurs | |
|---|---|---|---|---|---|
| Nom | initDB | rootDB | am | nbCoupe | nbIV |
| moy AL | 325.674 | 329.684 | 1.23 | 382 | 5220 |
| moy S60 | 224.826 | 226.405 | 0.7 | 256 | 3108 |
| GN | 517.348 | 521.238 | 0.75 | 560 | 9280 |
| Pb1 | 601 | 606.301 | 0.88 | 800 | 8526 |

TAB. 6.8 – Borne duale obtenue avec l'ajout des coupes

Finalement, même si les coupes ont un faible impact sur la borne duale, elles éliminent bien certaines solutions et on observe que la solution du $PL$ change de structure. Typiquement dans la solution $PL$ la périodicité des colonnes est 6, avec les coupes les périodicités 4 et 5 apparaissent, voir même les périodicités 1, 2 ou 3 pour certaines instances. La diversité des périodicités dans la solution du $PL$ pourrait peut être permettre l'obtention de meilleurs solutions entières par l'heuristique d'arrondi. Ensuite la solution est plus fractionnaire : sur un ensemble de 10 instances (5 instances du groupe AL et les instances du groupe S60), initialement 90% des colonnes $r$ telles que $\lambda_r > 0$ ont une valeur inférieure à 0.5 et 58% inférieure à 0.2 ; après l'ajout des coupes les pourcentages deviennent respectivement 96% et 75%. La section 7.3 montre que la solution entière obtenue en appliquant l'heuristique d'arrondi sur la solution $PL$ après l'ajout de coupes n'est pas meilleure que celle obtenue sur la solution sans les coupes.

### 6.4.2 Branchement sur les véhicules

- **Branchement sur Vmoy**

Comme la variable $Vmoy$ indiquant le nombre de véhicules utilisés doit être entière, un branchement naturel est celui sur cette variable. Soit $Vmoy = v$ fractionnaire, alors les contraintes de branchement sont :

$$\text{(Nœud 1)} \qquad Vmoy \leq \lfloor v \rfloor \qquad \text{ou} \qquad Vmoy \geq \lceil v \rceil \qquad \text{(Nœud 2)} \tag{6.15}$$

Ce branchement a d'autant plus d'influence sur la borne duale que le coût d'utilisation d'un véhicule est grand par rapport au coût des routes, ce qui est notre cas en ayant choisi $\alpha = 0.5$. De plus, comme le but du $PL$ est principalement de minimiser le nombre de véhicules utilisés, le nœud 1 a de grande chance d'être irréalisable et dans le nœud 2, $Vmoy$ a de bonne chance d'être entière.

Au nœud 1, la contrainte sur le nombre de véhicules est très contraignante (que le nœud soit réalisable ou irréalisable). La solution de la racine n'est plus

réalisable, les variables artificielles de la phase 1 et 2 combinées sont à nouveau dans la solution $PL$. Pour les éliminer, beaucoup d'itérations sont nécessaires. Lorsque la solution $PL$ optimale courante contient des variables artificielles, leur coût est multiplié (par 1.8 dans nos tests) et le processus de génération de colonnes recommence. Si le nombre d'augmentations atteint le paramètre nbMaxMajPenalite, une phase 1 pure est appelée. Dans le cas où le nœud 1 est irréalisable, la phase 1 permet de le détecter. Dans le cas contraire, passer par une phase 1 pure est moins efficace car les colonnes générées ne sont pas optimisées sur leur coût réel.

Au nœud 2, la solution de la racine en prenant $Vmoy = \lceil v \rceil$ est réalisable, la réoptimisation porte sur les routes utilisées et on observe qu'avec notre objectif il n'est pas intéressant de prendre plus de $\lceil v \rceil$ véhicules. Ainsi après l'ajout de la contrainte disjonctive (6.15), $Vmoy$ est typiquement entier. On observe en pratique que le nombre de nœuds dans notre arbre de Branch-and-Bound est 2.

Dans le tableau 6.9, les temps de calcul sont comparés sur quelques instances en ayant choisi nbMaxMajPenalite à 50 et à 5. Lorsque nbMaxMajPenalite est à 50, une borne supérieure sur la valeur de la solution, INC, est calculée (c'est la solution d'une heuristique vue à la section 7.1) pour aider à tronquer le nœud plus tôt (lorsque la borne lagrangienne (2.14) devient supérieure à INC). Dans tous les cas, les temps de calculs sont plus long avec nbMaxMajPenalite = 50 qu'avec nbMaxMajPenalite = 5, cette limite de 50 n'est d'ailleurs jamais atteinte.

| | Bornes | | | Compteurs | | Temps | | | |
|---|---|---|---|---|---|---|---|---|---|
| Nom | INC | rootDB | bestDB | SpSol | Col | MCKP | SpSol | RM | Total |
| nbMaxMajPenalite = 50 | | | | | | | | | |
| 3.mixte | 515.698 | 352.035 | 399.935 | 6027 | 6009 | 97 | 150 | 80 | 18m25s (1105) |
| 0.rural | 555.312 | 340.863 | 379.663 | 5571 | 5573 | 73 | 122 | 69 | 14m25s (865) |
| Nord | 324.939 | 228.393 | 273.331 | 2712 | 2708 | 57 | 75 | 14 | 4m21s (261) |
| GrandNord | 725.371 | 517.348 | 554.55 | 13308 | 13307 | 2307 | 2525 | 533 | 2h3m36s (7416) |
| nbMaxMajPenalite = 5 | | | | | | | | | |
| 3.mixte | 1e+12 | 352.035 | 399.935 | 3332 | 3326 | 18 | 47 | 30 | 2m44s (164) |
| 0.rural | 1e+12 | 340.863 | 379.663 | 2950 | 2945 | 16 | 41 | 20 | 3m20s (200) |
| Nord | 1e+12 | 228.393 | 273.331 | 2030 | 2024 | 21 | 35 | 8 | 2m24s (144) |
| GrandNord | 1e+12 | 517.348 | 554.55 | 7758 | 7753 | 289 | 415 | 212 | 27m4s (1624) |

TAB. 6.9 – Comparaison numérique de l'influence du paramètre nbMaxMajPenalite dans le cas où on branche sur Vmoy

Le tableau 6.10 indique les résultats obtenus en prenant nbMaxMajPenalite = 5. Le nœud 1 est irréalisable pour toutes les instances et la variable

$Vmoy$ est entière au nœud 2. Plus de la moitié du temps de calcul est passé à prouver l'irréalisabilité du nœud 1, par contre la réoptimisation au nœud 2 est rapide. En moyenne, la borne duale s'est trouvée améliorée de 11.2% sur les 17 instances. Cependant, cette amélioration dépend beaucoup de l'instance comme le montre le tableau récapitulatif 6.11. L'amélioration est plus importante si dans la solution $PL$, $Vmoy = \lfloor v \rfloor + \varepsilon$ (par exemple pour l'instance 3.urbain $Vmoy_{PL} = 3.20167$, et pour Nord $Vmoy_{PL} = 2.24913$) ; elle l'est beaucoup moins si $Vmoy = \lceil v \rceil - \varepsilon$ (par exemple pour l'instance 6.mixte $Vmoy_{PL} = 2.99667$, et pour SudP6 $Vmoy_{PL} = 2.96576$). Enfin, sur les deux plus grandes instances l'amélioration est moins importante car dans la solution $PL$, le rapport entre le coût d'utilisation des véhicules et le coût des routes est de l'ordre de 1.5, alors qu'il est de 2 pour les autres instances (le paramètre $\alpha$ dans (6.1) est fixé à 0.5 pour toutes les instances).

| | Bornes | | | Compteurs | | Temps | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nom | rootDB | bestDB | am | SpSol | Col | MCKP | SpSol | RM | N0 | N1 | N2 | Total |
| moy AL | 325.674 | 362.363 | 11.26 | 3600 | 3595 | 23 | 40 | 35 | 85 | 199 | 4 | 4m48s (288) |
| moy S60 | 224.826 | 252.551 | 12.33 | 2221 | 2215 | 25 | 32 | 10 | 30 | 120 | 25 | 2m55s (175) |
| GN | 517.348 | 554.55 | 8363 | 7.19 | 8355 | 404 | 483 | 235 | 492 | 1213 | 254 | 32m40s (1960) |
| Pb1 | 601 | 654.692 | 8.96 | 4861 | 4854 | 238 | 274 | 79 | 164 | 461 | 110 | 12m15s (735) |

TAB. 6.10 – Borne duale obtenue après le branchement sur Vmoy et répartition du temps de calcul entre les différents nœuds

Les différentes bornes duales obtenues pour chaque instance avec $P = \{1, 2, 3, 4, 5, 6\}$ sont présentées dans le tableau 6.11 : on y compare la borne à la racine (notée rootDB), la borne obtenue après l'ajout des coupes (notée cutDB), et celle obtenue après le branchement sur $Vmoy$ (notée brDB) et après l'ajout des coupes et le branchement (notée (cut+br)DB). Pour cette dernière borne, comme le nœud 1 est irréalisable pour toutes nos instances et que la limite de nbMaxMajPenalite à 5 n'a jamais été atteinte dans un autre nœud, nbMaxMajPenalite a été mis à 3. Deux stratégies sont testées sur quelques instances, les résultats sont dans le tableau 6.12 (les bornes duales obtenues par ces deux stratégies sont identiques). Dans la première stratégie, les coupes sont ajoutées à chaque nœud (i.e. à la racine et au nœud 2). Dans la deuxième stratégie, les coupes sont ajoutées une fois que la variable $Vmoy$ est entière, c'est à dire au nœud 2 uniquement. Que des coupes soient ajoutées ou non, nous obtenons la même valeur entière pour $Vmoy$ après le branchement (par exemple pour l'instance 3.urbain $Vmoy^* = 4$, et pour les instances 6.mixte,

Nord et SudP6 $Vmoy^* = 3$). Lorsque les coupes sont ajoutées à la racine et au nœud 2, plus de la moitié du temps est passée dans le nœud 1. Lorsque les coupes sont ajoutées après le branchement sur la variable $Vmoy$, le temps est passé principalement dans la procédure des plans coupants. Pour la suite, la borne duale ((cut+br)DB) est enregistrée dans un fichier et le calcul de la déviation à l'optimum des solutions entières se fera ainsi par rapport à cette borne.

| Nom | rootDB | cutDB | am | brDB | am | (cut+br)DB | am |
|---|---|---|---|---|---|---|---|
| 0.mixte | 314.143 | 317.386 | 1.03 | 352.943 | 12.35 | 356.032 | 13.33 |
| 1.mixte | 328.873 | 332.77 | 1.18 | 367.191 | 11.65 | 370.7 | 12.71 |
| 3.mixte | 352.035 | 356.743 | 1.34 | 399.935 | 13.6 | 404.399 | 14.87 |
| 6.mixte | 282.816 | 287.142 | 1.53 | 283.016 | 0.07 | 287.151 | 1.53 |
| 0.rural | 340.863 | 345.081 | 1.24 | 379.663 | 11.38 | 383.675 | 12.56 |
| 1.rural | 339.115 | 342.518 | 1 | 377.915 | 11.44 | 381.168 | 12.40 |
| 2.rural | 334.671 | 340.56 | 1.75 | 369.836 | 10.51 | 375.024 | 12.05 |
| 9.rural | 324.412 | 329.012 | 1.42 | 366.612 | 13 | 370.888 | 14.32 |
| 0.urbain | 339.183 | 342.575 | 1 | 377.983 | 11.43 | 381.276 | 12.41 |
| 3.urbain | 300.635 | 303.051 | 0.8 | 348.535 | 15.93 | 350.913 | 16.72 |
| moy AL | 325.674 | 329.684 | 1.23 | 362.363 | 11.26 | 366.123 | 12.29 |
| Jonzac | 157.355 | 157.95 | 0.38 | 182.161 | 15.76 | 182.725 | 16.22 |
| Nord | 228.393 | 230.657 | 0.99 | 273.331 | 19.67 | 275.626 | 20.68 |
| Ouest | 147.989 | 149.924 | 1.31 | 175.405 | 18.52 | 177.235 | 19.76 |
| SudP6 | 268.391 | 269.593 | 0.45 | 270.156 | 0.66 | 271.066 | 0.99 |
| NordJonzacP6 | 322.004 | 323.9 | 0.59 | 361.705 | 12.33 | 363.761 | 12.97 |
| moy S60 | 224.836 | 226.405 | 0.7 | 252.551 | 12.33 | 254.083 | 14.12 |
| GrandNord | 517.348 | 521.238 | 0.75 | 554.55 | 7.19 | 558.452 | 7.94 |
| Probleme1 | 601 | 606.301 | 0.88 | 654.692 | 8.96 | 659.839 | 9.79 |
| moy 17 inst | 323.486 | 326.837 | 1.02 | 358.566 | 11.2 | 361.761 | 12.42 |

TAB. 6.11 – Récapitulatifs des différentes bornes duales

| | Compteurs | | Temps | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Nom | SpSol | Col | MCKP | SpSol | RM | CP | N0 | N1 | N2 | Total |
| Les coupes sont ajoutées à chaque nœud | | | | | | | | | | |
| 3.mixte | 9727 | 9700 | 214 | 318 | 526 | 1008 | 829 | 4861 | 1279 | 1h56m10s (6970) |
| 0.rural | 11694 | 11675 | 401 | 545 | 737 | 927 | 846 | 9513 | 1297 | 3h14m18s (11658) |
| Nord | 5620 | 5610 | 102 | 145 | 151 | 215 | 201 | 1878 | 373 | 40m53s (2453) |
| GrandNord | 15164 | 15141 | 1226 | 1519 | 2197 | 1760 | 2545 | 8307 | 2451 | 3h41m44s (13304) |
| Les coupes sont ajoutées après le branchement sur $Vmoy$ | | | | | | | | | | |
| 3.mixte | 7350 | 7326 | 80 | 125 | 249 | 1995 | 85 | 30 | 2089 | 36m45s (2205) |
| 0.rural | 6982 | 6970 | 67 | 109 | 181 | 1531 | 82 | 78 | 1598 | 29m18s (1758) |
| Nord | 3676 | 3669 | 36 | 51 | 42 | 420 | 24 | 71 | 446 | 9m2s (542) |
| GrandNord | 13411 | 13388 | 832 | 974 | 573 | 5380 | 514 | 1614 | 5729 | 2h10m59s (7859) |

TAB. 6.12 – Comparaison numérique de différentes stratégies pour l'ajout des coupes et le branchement sur quelques instances

- **Disjonction**

Considérons la solution présentée dans l'exemple 6.4. Dans cet exemple le nombre de véhicules nécessaires est sous estimé. Dans la solution entière agrégée, en moyenne 1 véhicule est utilisé, mais dans une solution réalisable 2 véhicules sont nécessaires, un véhicule effectuant la route de périodicité 2, et l'autre la route de périodicité 3. En effet, si un véhicule effectue une route de périodicité 2, il ne pourra par la suite faire que des routes de périodicités paires $p \in GP_2 = \{2, 4, 6, 8, 10, 12, \cdots\}$. Le véhicule effectuant une route de périodicité 3, ne pourra faire que des routes de périodicités $p$ multiples de 3, $p \in GP_3 = \{3, 6, 9, 12, \cdots\}$. Ces classes $GP_2$ et $GP_3$ sont en ce sens "incompatibles" : un véhicule utilisé pour une classe ne peut pas être disponible pour l'autre classe. Le nombre de véhicules utilisés au total doit être égal à la somme des nombres de véhicules utilisés pour chaque classe. Cependant on constate que certaines périodicités, comme 6 et 12, sont dans la classe $GP_2$ et $GP_3$. Pour les routes ayant ces périodicités, on ne peut pas dire à l'avance par quel véhicule elles seront effectuées, un de la classe $GP_2$ ou $GP_3$ ? Ainsi, on peut seulement comptabiliser le nombre minimum de véhicules nécessaires pour effectuer les routes de chaque sous-classe disjointe ne contenant pas ces périodicités. Cette sous-classe contient les puissances du nombre de base, soit $SP_2 = \{2, 4, 8, \cdots\}$ et $SP_3 = \{3, 9, \cdots\}$. Cependant, les véhicules utilisés pour la périodicité 6 ne sont pas comptabilisés puisqu'ils peuvent être combinés avec des périodicités 2 ou 3. En se basant sur les classes, des contraintes sont ajoutées et des branchements sur les variables associées sont effectués pour renforcer les disjonctions entre les véhicules de différentes sous-classes $SP_p$.

Notons $PP$ l'ensemble comprenant la périodicité 1 et l'ensemble des périodicités de $P$ qui sont des nombres premiers, par exemple si $P = \{1, 2, 3, 4, 5, 6\}$ alors $PP = \{1, 2, 3, 5\}$.

**Définition 6.4.5** *Pour chaque $p \in PP$, définissons l'ensemble de ses puissances $SP_p$ et celui des ses multiples $GP_p$ :*

$$SP_p = \{\tau \in P, \exists k > 0, \tau = p^k\}$$

$$GP_p = \{\tau \in P, \exists k > 0, \tau = k * p\}$$

**Remarque:** $SP_p \subset GP_p$, $\cap_p SP_p = \emptyset$, $\cup_p GP_p = P$.

**Proposition 6.4.6** *Dans toutes les solutions réalisables de la formulation $[FormC]$ définie en (5.8-5.12), un véhicule peut être affecté à une seule classe $SP_p$ avec $p \in PP$.*

**Preuve:** Supposons un véhicule affecté à deux classes $SP_{pp_1}$ et $SP_{pp_2}$.

Si $pp_1 = 1$, alors il est évident que le véhicule ne pourra pas effectuer les routes ayant une périodicité appartenant à $SP_{pp_2}$ vu qu'il est utilisé toutes les périodes.

Soit $pp_1 > 1$ et $pp_2 > 1$. Remarquons que tout élément de $SP_{pp_1}$ et tout élément de $SP_{pp_2}$ sont premiers entre eux (i.e. leur plus grand commun diviseur, noté pgcd, est 1). Prenons comme périodicités $p1 \in SP_{pp_1}$ et $p2 \in SP_{pp_2}$ et les dates de départ pour ces routes de périodicités $p1$ et $p2$ sont respectivement $s1 = 1$ et $1 < s2 \leq p2$. Le cycle de régénération du véhicule effectuant ces deux routes est $h = ppcm(p1, p1) = p1p2$, i.e. son planning est $h$ périodique.

Si l'on montre qu' il existe une infinité de couples $(k1, k2) \in \mathbb{Z}^2$ tels que

$$k1p1 + 1 = k2p2 + s2 \Leftrightarrow k1p1 - k2p2 = s2 - 1 \tag{6.16}$$

alors on aura montré qu'il existe des dates $t = k1p1 + 1 = k2p2 + s2$ (une infinité) telles qu' à chacune de ces dates le véhicule devra effectuer ces deux routes. Comme son planning est $h$ périodique, il y aura au moins une date dans ce planning durant laquelle le véhicule fera ces 2 routes, ce qui est impossible.

L'équation de la forme $ax + by = c$ avec $a, b, c \in \mathbb{Z}$ est une équation diophantienne d'inconnues $x$ et $y$. L'ensemble de ses solutions $S$ est infini si et seulement si le pgcd de $a$ et $b$ divise $c$, ce qui est toujours le cas si le pgcd est 1. Alors dans ce cas $S = \{(c\, u_0 + k\, b, c\, v_0 - k\, a), k \in \mathbb{Z}\}$ où $(u_0, v_0) \in \mathbb{Z}^2$ est une solution de l'identité de Bezout $ax + by = 1$. Par conséquent, l'équation (6.16) admet une infinité de solutions $(k1, k2) \in \mathbb{Z}^2$. □

Notons par $v_p$ la variable indiquant le nombre minimum de véhicules nécessaires pour couvrir les routes de périodicités appartenant à $SP_p$. La valeur de $v_p$ peut aussi être comprise comme le nombre de véhicules qu'il faut réserver pour les routes de la classe des périodicités associées à $p$. Pour comptabiliser ces véhicules, on ajoute les contraintes suivantes au modèle $[FormAgr]$ :

$$\sum_{r, p^r \in SP_p} \frac{1}{p^r} \lambda_r \leq v_p \qquad \forall p \in PP \tag{6.17}$$

$$\sum_{p \in PP} v_p \leq Vmoy \tag{6.18}$$

$$v_p \in \mathbb{N} \qquad \forall p \in PP \tag{6.19}$$

Les contraintes (6.17) définissent le nombre minimum de véhicules nécessaires pour chaque classe et la contrainte (6.18) définit le nombre moyen de véhicules

nécessaires par rapport aux nombres de véhicules utilisés pour chaque classe. Les contraintes (6.17) ne sont pas une égalité car la partie de gauche peut être fractionnaire et la solution réalisable (par exemple s'il y a 3 routes de périodicités 2, la partie de gauche est égale à 1.5 mais $v_2 = 2$).

Un branchement supplémentaire se fait sur la variable $v_p = \hat{v}_p$ fractionnaire. Cette contrainte de branchement peut être exprimée directement sur les variables $\lambda_r$. En effet, si l'on réserve seulement $\lfloor \hat{v}_p \rfloor$ véhicules pour la classe fractionnaire $p$, alors il faut limiter le nombre de routes qui ont une périodicité appartenant à la classe $SP_p$. Par contre si on réserve $\lceil \hat{v}_p \rceil$ véhicules pour la classe fractionnaire $p$, alors il faut limiter le nombre de routes qui ne peuvent pas être combinées avec une route ayant une périodicité appartenant à la classe $SP_p$, c'est à dire limiter le nombre de routes qui ont une périodicité n'appartenant pas à la classe $GP_p$. Les contraintes de branchement sont :

$$v_p \leq \lfloor \hat{v}_p \rfloor \quad \Leftrightarrow \quad \sum_{r, p^r \in SP_p} \frac{1}{p^r} \lambda_r \leq \lfloor \hat{v}_p \rfloor$$

$$v_p \geq \lceil \hat{v}_p \rceil \quad \Leftrightarrow \quad \sum_{r, p^r \notin GP_p} \frac{1}{p^r} \lambda_r \leq Vmoy - \lceil \hat{v}_p \rceil.$$

En observant les solutions $PL$ de $[FormAgr]$ obtenues dans nos tests numériques, nous constatons que la périodicité la plus fréquente est 6 et par conséquent les variables $v_p$ pour $p$ appartenant à l'ensemble $PP$ sont souvent égales à zéro. L'ajout des coupes modifie les solutions $PL$ en diversifiant les périodicités et dans ce cas les variables $v_p$ ne sont plus égales à zéro (en particulier les variables $v_2$ et $v_5$ qui comptabilisent les routes de périodicités 4 et 5 sont strictement positives). Cependant, nous constatons que la somme des variables $v_p$ est inférieure à la valeur de $Vmoy$, i.e. que la solution $PL$ vérifie la contrainte (6.18). Dans le cas où l'ensemble des périodicités est égal à $P = \{1, 2, 3, 4, 5, 6\}$, ces contraintes (6.17-6.19) ne sont pas utiles, cela est dû au fait que les routes de périodicité 6 ne sont pas comptabilisées.

Pour tester l'impact de ces contraintes sur la solution $PL$, nous avons restreint les périodicités à prendre les valeurs dans l'ensemble $P = \{1, 2, 3\}$. Dans ce cas, la longueur du cycle $T$ est égale à 6. Les 3 périodicités appartiennent à l'ensemble $PP$, i.e $P = PP$, et pour tout $p \in PP$, on a $SP_p = GP_p = \{p\}$ (toutes les routes sont donc comptabilisées). Il suffit alors de définir $v_2$ car $v_1$ est exactement égal à $\sum_{r, p^r \in SP_1} \lambda_r$ et les véhicules res-

tants, i.e. $Vmoy - v_1 - v_2$, peuvent être utilisés par les routes de périodicité 3 ($\sum_{r,p^r \in SP_3} \frac{1}{3}\lambda_r \leq Vmoy - v_1 - v_2$). Le branchement sur la variable $Vmoy$ est effectué avant le branchement sur la variable $v_2$. Après le branchement sur la variable $v_2$, un autre branchement sur la variable $Vmoy$ peut éventuellement être nécessaire. Le parcours de l'arbre est fait en largeur.

Le branchement sur la variable $v_2$ peut être combiné avec l'ajout de coupes. Les inégalités (6.12) avec $h = 1, 2, 3, 4$ sont toujours valides mais sur les 4 inégalités trois sont équivalentes à la contrainte (6.2) comme on peut le voir dans le tableau 6.5. Finalement, l'inégalité valide intéressante pour chaque client est :

$$\sum_{r,p} \phi^r_{ip}\lambda_r + \sum_{r,p=2} \frac{1}{2}\phi^r_{i1}\lambda_r + \sum_{r,p=3} \frac{2}{3}\phi^r_{i1}\lambda_r + \sum_{r,p=3} \frac{1}{3}\phi^r_{i2}\lambda_r + \geq 1$$

Les résultats avec ce branchement disjonctif sont dans le tableau 6.13. Les coupes sont ajoutées à chaque nœud. La borne duale obtenue à la racine est notée rootDB ; la borne obtenue après le premier branchement sur $Vmoy$ est notée DB-P1 (à une profondeur de 1 de l'arbre) ; la borne obtenue après le branchement sur $v_2$ et puis à nouveau sur $Vmoy$ est notée DB-P$2^+$ (à une profondeur de 2 ou plus). La case vide DB-P$2^+$ signifie que $v_2$ était entière après le branchement sur $Vmoy$. Le branchement disjonctif apporte une amélioration pour 4 instances sur 17 seulement et cette amélioration reste minime (moins de 0.01% par rapport à la borne DB-P1). Dans le tableau 6.13, nous avons aussi indiqué la meilleure borne duale obtenue sur l'espace moins restreint $P = \{1, 2, 3, 4, 5, 6\}$. Les bornes duales sont données sur le cycle $T$, pour les comparer, il faut donc multiplier DB-P1 par 10. En restreignant l'espace à $P = \{1, 2, 3\}$, l'augmentation de la borne duale DB-P1 par rapport à bestDB n'est pas négligeable : elle est de 4.07% en moyenne pour les 17 instances.

**Remarque:** Des disjonctions sont aussi possibles au niveau des clients, en effet dans toute solution réalisable, un client peut être visité par des routes dont les périodicités appartiennent à seulement un $GP_p$. Cependant si un client n'est visité que par des routes de périodicités multiples de nombres premiers (par exemple $6 = 3 * 2$), il n'est affecté à aucun groupe $GP_p$ (comme on peut le voir sur l'exemple 6.11). De plus, ces disjonctions ne prennent pas en compte la quantité collectée et n'empêchent pas les solutions de l'exemple 6.3 sur l'irréalisabilité du planning d'un client (4 et 6 appartiennent à $GP_2$).

| | $P = \{1,2,3\}$ | | | $P = \{1,2,3,4,5,6\}$ |
|---|---|---|---|---|
| Nom | rootDB | DB-P1 | DB-P2$^+$ | bestDB |
| 0.mixte | 32.9504 | 36.7793 | 36.7793 | 356.032 |
| 1.mixte | 34.5784 | 38.3632 | 38.3632 | 370.7 |
| 3.mixte | 36.9462 | 41.6586 | 41.6586 | 404.399 |
| 6.mixte | 29.8283 | 35.7805 | | 287.151 |
| 0.rural | 35.6907 | 39.5138 | 39.5138 | 383.675 |
| 1.rural | 35.499 | 39.3198 | 39.3198 | 381.168 |
| 2.rural | 35.5731 | 38.9044 | 38.9127 | 375.024 |
| 9.rural | 34.1297 | 38.3031 | 38.3031 | 370.888 |
| 0.urbain | 35.4799 | 39.3156 | 39.3156 | 381.275 |
| 3.urbain | 31.4012 | 36.1602 | 36.1602 | 350.913 |
| moy AL | 34.2077 | 38.4099 | 38.4107 | 366.123 |
| Jonzac | 16.5227 | 19.0091 | 19.0115 | 182.725 |
| Nord | 23.8778 | 28.3715 | | 275.626 |
| Ouest | 16.124 | 18.8734 | | 177.235 |
| SudP6 | 27.7649 | 27.91 | 27.9242 | 271.066 |
| NordJonzacP6 | 32.7402 | 36.7236 | 36.7237 | 363.761 |
| moy S60 | 23.4059 | 26.1775 | 26.1809 | 254.083 |
| GrandNord | 54.3959 | 58.1405 | | 558.452 |
| Probleme1 | 62.4976 | 67.8358 | 67.8358 | 659.839 |

TAB. 6.13 – Borne duale obtenue avec les coupes, le branchement sur $Vmoy$ et le branchement disjonctif lorsque les périodicités sont restreintes à prendre leur valeur dans $P = \{1,2,3\}$

## 6.5 Conclusion

Dans ce chapitre une formulation agrégée pour éviter la symétrie dans le choix des dates de départ est proposée. Cette formulation est obtenue en utilisant la technique de "state space relaxation" et modélise en comportement moyen. De plus, elle fournit la même solution $PL$ que la formulation discrète. La formulation agrégée permet de calculer une borne duale pour des instances de grande taille. L'ajout de coupes basées sur des fonctions superadditives n'a pas un apport véritable bien que la solution du $PL$ change de structure, elle permet une amélioration peu significative de la borne duale. Une amélioration plus franche est obtenue en branchant sur la variable $Vmoy$. La meilleure borne duale obtenue en combinant l'ajout des coupes et le branchement est enregistrée pour permettre d'évaluer les heuristiques du chapitre 7.

Exemple 6.11 – Les classes pour un client

Soit la solution entière :

- $1*$ la route A : $0-1(1)-2(2)-\cdots-0$ avec $p_A = 2$
- $1*$ la route B : $0-1(1)-3(3)-\cdots-0$ avec $p_B = 3$
- $1*$ la route C : $0-1(1)-4(6)-\cdots-0$ avec $p_C = 6$

Représentons les plannings partiels de chaque route en prenant $s_A = 1$, $s_B = 2$ et $s_C = 4$.

| | t1 | t2 | t3 | t4 | t5 | t6 |
|---|---|---|---|---|---|---|
| $\delta^A_{1t}$ | A | | A | | A | |
| $\delta^A_{2t}$ | A | ✓ | A | ✓ | A | ✓ |
| $\delta^B_{1t}$ | | B | | | B | |
| $\delta^B_{3t}$ | ✓ | B | ✓ | ✓ | B | ✓ |
| $\delta^C_{1t}$ | | | | C | | |
| $\delta^C_{4t}$ | ✓ | ✓ | ✓ | C | ✓ | ✓ |

Cette solution est non réalisable, à cause des collectes du client 1 par des routes de périodicités 2, 3 et 6 : ces périodicités appartiennent à plusieurs classes, $GP_2$ et $GP_3$. Pour les clients 2 et 3, les routes les visitant n'appartiennent qu'à une seule classe : $GP_2$ pour le client 2 et $GP_3$ pour le client 3. Par contre, pour le client 4, on ne sait pas à quelle classe appartient la route le visitant, $GP_2$ ou $GP_3$ ?

# 7

# Bornes primales obtenues par des heuristiques basées sur l'approche de décomposition

Dans ce chapitre, nous présentons notre implémentation des heuristiques de la section 2.4. La première heuristique présentée (en section 7.1) est une heuristique gloutonne propre à notre application. Cette heuristique est guidée par la solution duale de la formulation agrégée $[FormAgr]$ définie en (6.1-6.5). Une autre manière d'obtenir une solution entière consiste à résoudre la formulation discrète $[FormC]$ définie en (5.8-5.12) en nombre entier restreinte à un sous ensemble de colonnes. Cette heuristique est présentée en section 7.2. Ensuite, nous présentons en section 7.3 une heuristique d'arrondi avec diverses variantes concernant le choix de la solution $PL$ de départ et du critère de fixation. Une heuristique de recherche locale basée sur l'échange de colonnes dans la solution est présentée en section 7.4. Nous développons ces deux dernières heuristiques en utilisant à la fois la formulation $[FormAgr]$ pour la solution $PL$ (primale ou duale) et la formulation $[FormC]$ pour la solution entière.

Les expérimentations sur notre base de problèmes tests de la section 5.5 permettent de comparer ces heuristiques en terme de la qualité des solutions et de temps de calcul. L'heuristique gloutonne est rapide (moins de 5s) mais les solutions obtenues ont une déviation à l'optimum estimée à 30% en moyenne. Le temps de résolution du maître restreint en nombre entier est trop important pour permettre de trouver des solutions entières aux problèmes de taille réelle (la symétrie de la formulation $[FormC]$ est largement responsable de cet état de fait). L'heuristique d'arrondi appelée sur la solution $PL$ optimale de la racine et celle du nœud 2 est la variante la plus efficace : les solutions sont obtenues

en moins de 1h30 pour les deux plus grandes instances et ont une déviation à l'optimum estimée à 11.66% en moyenne. Cette déviation à l'optimum est estimée à 9.67% en moyenne pour les solutions obtenues quand on restreint les périodicités à prendre les valeurs dans $P = \{1, 2, 3\}$. L'heuristique de recherche locale est appelée sur les solutions obtenues par l'heuristique d'arrondi à la racine. Elle produit des solutions améliorées de moins de 1% à plus de 15% (par rapport aux solutions trouvées à la racine par l'heuristique d'arrondi), en 1h pour les deux plus grandes instances. La déviation à l'optimum est estimée à 11.2158% en moyenne.

## 7.1 Heuristique initiale de type glouton

L'heuristique gloutonne est spécifique à notre problème. Il s'agit d'insérer les clients 1 à 1 dans la solution partielle en privilégiant l'insertion de toute la demande du client dans une même route ; et si ce n'est pas possible alors la demande du client est insérée dans plusieurs routes. Une nouvelle route est créée à chaque fois que l'insertion de la demande du client n'est possible dans aucune des routes existantes ou lorsque le coût de son insertion est supérieur au coût de création. Le coût de création est augmenté d'une pénalité pour dissuader ces créations. Voici comment sont définis les coûts d'insertion et de création :

1. soit le client $i$ est inséré dans une route existante de périodicité $p$ et de centre $k$. Alors son coût d'insertion est $\frac{\alpha}{p} c'_{ki}$ avec $c'_{ki}$ le coût de connexion défini en (3.29).
2. soit le client est défini comme centre d'une nouvelle route de périodicité $p$. Alors le coût de création de cette route est $\frac{f_i + \frac{attract(i) + c_{0i} + c_{i1}}{3}}{p}$ où $attract(i)$ est défini en section 6.2 ; si après la création $Vmax$ augmente de 1 alors 1 est ajouté au coût de création.

A la fin de l'algorithme, les centres des clusters sont réoptimisés.

Les solutions obtenues dépendent de l'ordre d'insertion des clients. Trois critères de priorité sont considérés. Ils sont basés sur les valeurs duales $\pi$ représentant les coûts moyens de collecte par période et le vecteur $attract$ qui représente l'attractivité des clients à devenir centre (un client attractif est un client qui a beaucoup de voisins proches et a des chances d'être un centre). Un score est attribué à chaque client pour définir sa priorité. Chaque critère permet d'ordonner les clients par score décroissant. Le score attribué au client

$i$ est respectivement pour les trois critères : $tmax_i \pi_i$ (i.e. l'estimation du coût de collecte du client $i$ à une période donnée), $\frac{\pi_i}{attract(i)}$ (un client $i$ sera en début de liste si son coût moyen de collecte est élevé et/ou s'il est attractif, i.e. il peut définir un bon centre) et $\frac{tmax_i \pi_i}{attract(i)}$. L'heuristique peut être appelée avec l'estimation des variables duales $\pi_{init}$ présentée en section 6.2 ou après la résolution du maître $PL$ avec la solution duale optimale $\pi^*$ de $[FormAgr]$.

Les résultats de cette heuristique sont donnés dans le tableau 7.1, les abréviations sont celles de la table 6.1. Cette heuristique met quelques secondes pour trouver une solution mais la qualité d'approximation est faible comme on peut l'estimer par comparaison à une borne duale : la déviation à l'optimum est supérieure à 15% pour 13 instances sur 17. Les solutions obtenues avec $\pi_i^*$ sont en moyenne meilleures et utilisent moins de véhicules. D'ailleurs pour 15 instances le nombre de véhicules utilisés ne peut pas être amélioré par nos autres heuristiques (voir le tableau 7.12). Notre principal objectif est de minimiser le nombre de véhicules utilisés. Cette heuristique appelée après la résolution du maître $PL$ répond bien à cet objectif. Cependant, nous avons décidé de prendre aussi en considération le coût des routes pour ne pas obtenir des solutions ayant le nombre de véhicules nécessaires minimum mais dont les routes seraient de mauvaise qualité (i.e. des routes très coûteuses en temps). En comparant les solutions obtenues par cette heuristique et les solutions obtenues par les autres heuristiques (voir le tableau 7.12), nous voyons bien que pour un même nombre de véhicules, le coût associé aux routes peut être très différent, et mauvais dans le cas de cette heuristique gloutonne. Cette heuristique gloutonne peut être utile pour déterminer une borne supérieure sur le nombre de véhicules nécessaires.

Les figures 7.1 et 7.2 illustrent deux solutions obtenues par l'heuristique gloutonne avec $\pi^{init}$. Sur chaque figure, nous pouvons visualiser les clusters obtenus pour chaque date de départ $s$. Un cluster est associé à une couleur et est représenté par son ensemble de points clients le constituant (le centre n'est pas précisé). La périodicité $p$ et la charge $W$ de chaque cluster sont notées dans la légende. Les plannings sont respectivement de périodicité 3 et 6. La figure 7.1 illustre la meilleure solution obtenue, pour l'instance Jonzac : bien qu'il y ait des bornes avec $tmax = 14$, une solution n'ayant que des routes de périodicités 3 est à 4.9% de l'optimum. Pour cette même instance, la solution obtenue avec $\pi_i^*$ utilise le même nombre de véhicules mais sa déviation à l'optimum est de 8.67% : il y a 7 clusters de périodicité 6 et un de périodicité 2. La figure 7.2 montre une solution avec une déviation à l'optimum proche de 40%, pour

| | Bornes | | | Compteurs | Bornes | | Compteurs |
|---|---|---|---|---|---|---|---|
| Nom | INC | bestDB | dev | V | INC | dev | V |
| | appel avec $\pi_i^{init}$ | | | | et avec $\pi_i^*$ | | |
| 0.mixte | 428.332 | 356.032 | 20.3072 | 4 | 423.92 | 19.0679 | 4 |
| 1.mixte | 521.793 | 370.7 | 40.7588 | 5 | 440.307 | 18.7771 | 4 |
| 3.mixte | 515.698 | 404.399 | 25.5221 | 4 | 492.844 | 21.8708 | 4 |
| 6.mixte | 445.639 | 287.151 | 55.1923 | 4 | 434.71 | 51.3873 | 4 |
| 0.rural | 555.312 | 383.675 | 44.7349 | 5 | 518.886 | 35.2412 | 4 |
| 1.rural | 496.525 | 381.168 | 30.7888 | 4 | 500.7 | 31.3594 | 4 |
| 2.rural | 608.575 | 375.024 | 62.2746 | 4 | 543.165 | 44.8347 | 4 |
| 9.rural | 479.34 | 370.888 | 29.2412 | 4 | 501.552 | 35.23 | 4 |
| 0.urbain | 531.381 | 381.276 | 39.3691 | 5 | 511.584 | 34.1768 | 5 |
| 3.urbain | 410.046 | 350.913 | 16.8512 | 4 | 385.067 | 9.7329 | 4 |
| moy AL | 499.264 | 366.123 | 36.504 | 4.3 | 475.274 | 30.1678 | 4.1 |
| Jonzac | 191.714 | 182.725 | 4.9193 | 2 | 198.568 | 8.67056 | 2 |
| Nord | 324.939 | 275.626 | 17.8911 | 3 | 325.766 | 18.1912 | 3 |
| Ouest | 201.933 | 177.235 | 13.9351 | 2 | 200.573 | 13.1677 | 2 |
| SudP6 | 382.09 | 271.066 | 40.9583 | 4 | 364.282 | 34.3886 | 4 |
| NordJonzacP6 | 398.946 | 363.761 | 9.67246 | 4 | 416.731 | 14.5617 | 4 |
| moy S60 | 299.924 | 254.083 | 17.4753 | 3 | 301.184 | 17.7959 | 3 |
| GrandNord | 725.371 | 558.452 | 29.8896 | 7 | 662.91 | 18.705 | 6 |
| Probleme1 | 927.312 | 659.839 | 40.5361 | 7 | 980.227 | 48.5555 | 7 |

TAB. 7.1 – Solutions primales obtenues avec l'heuristique gloutonne

l'instance 0.rural. Il y a un déséquilibre dans l'utilisation des véhicules : à la date $s = 4$, il existe une route avec un seul client, alors qu'à la date $s = 3$ seulement 3 véhicules sont utilisés.

## 7.2 Résolution du maître restreint entier

Pour avoir une solution entière réalisable, le maître restreint discret est résolu en nombre entier. Il utilise les variables $\lambda_{rs}$ qui sont égales à 1 si la colonne $r$ est prise avec la date de départ $s$, 0 sinon. De même que pour la formulation $[FormC]$, une matrice $\delta$ associée à chaque colonne $(r, s)$ peut être calculée. La formulation du maître restreint entier est la suivante :

$$[FormPLE] \quad \min \quad Vmax \quad +\alpha \sum_{r \in \overline{R}, s} \frac{c^r}{p^r} \lambda_{rs} \tag{7.1}$$

$$\sum_{r \in \overline{R}, s} \delta_{it}^{rs} \lambda_{rs} \quad \geq \quad 1 \qquad \forall i, t \tag{7.2}$$

$$\sum_{r \in \overline{R}, s} \delta_{0t}^{rs} \lambda_{rs} \quad \leq \quad Vmax \qquad \forall t \tag{7.3}$$

$$\lambda_{rs} \quad \in \quad \{0, 1\} \qquad \forall\, r, s \tag{7.4}$$

$$Vmax \quad \in \quad I\!N. \tag{7.5}$$

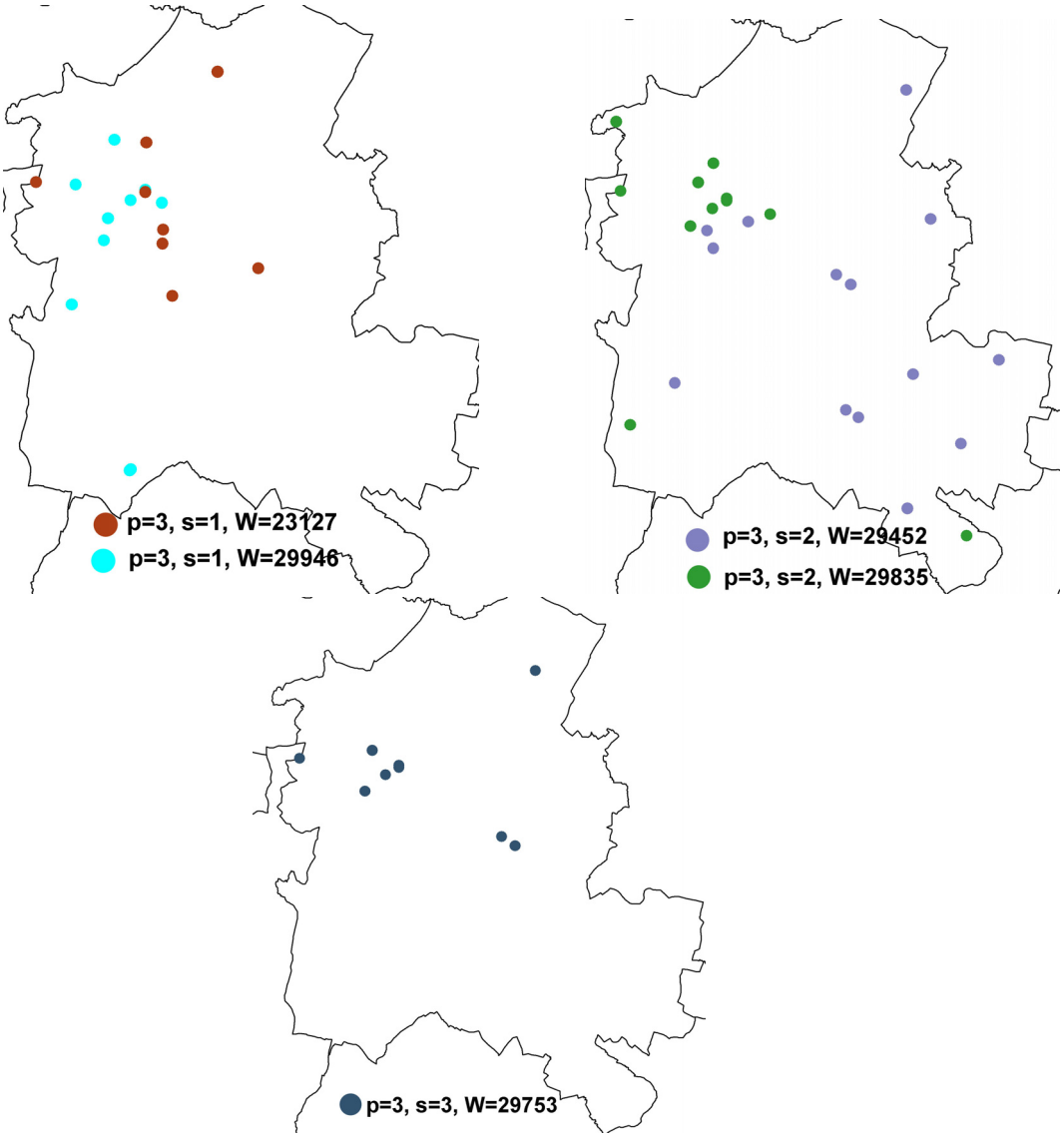
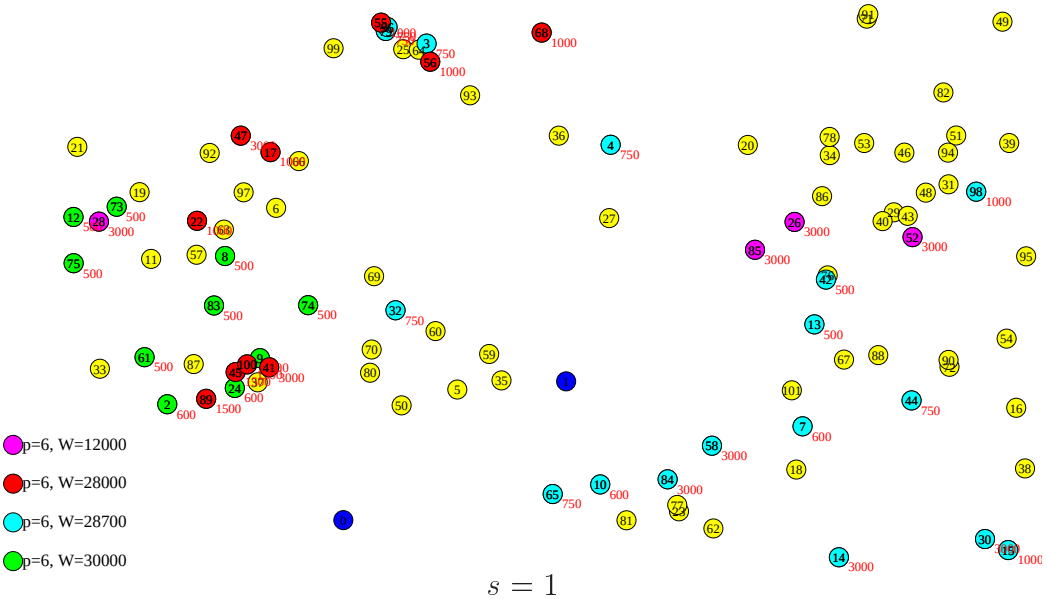
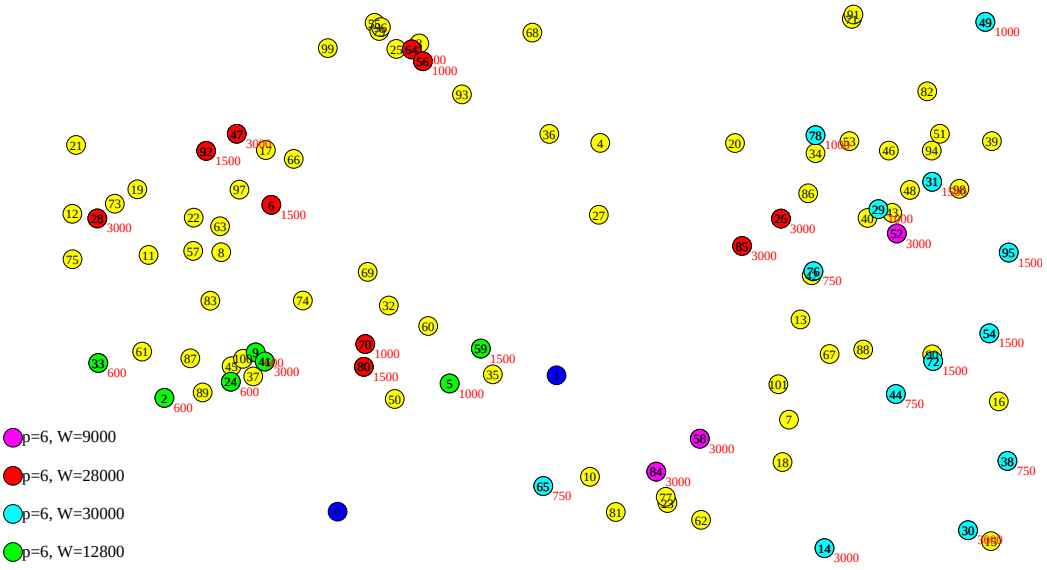


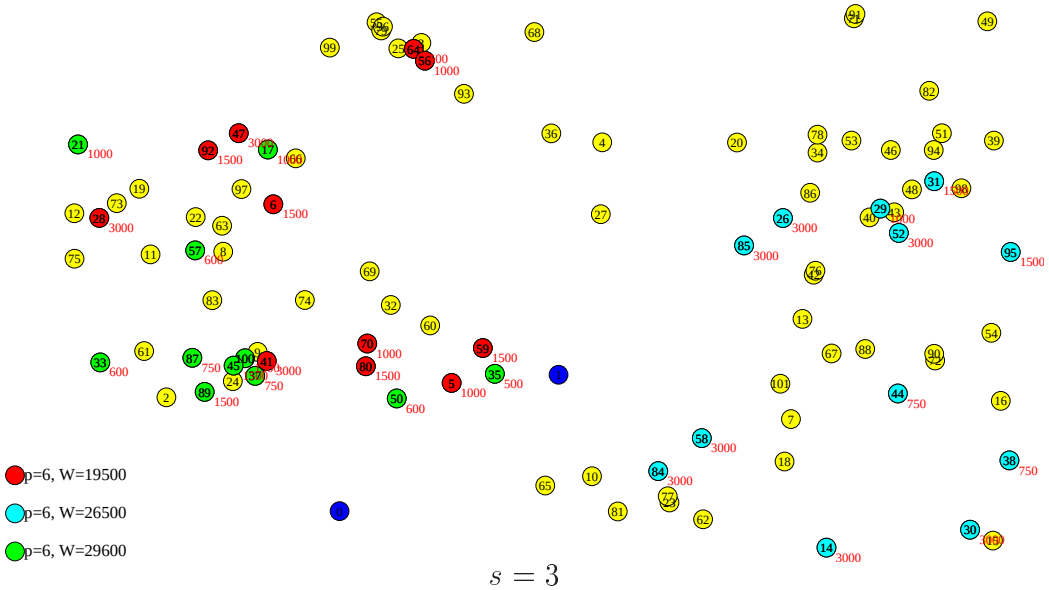

FIG. 7.1 – Solution de Jonzac obtenue par l'heuristique gloutonne avec $\pi_i^{init}$


p=6, W=12000
p=6, W=28000
p=6, W=28700
p=6, W=30000

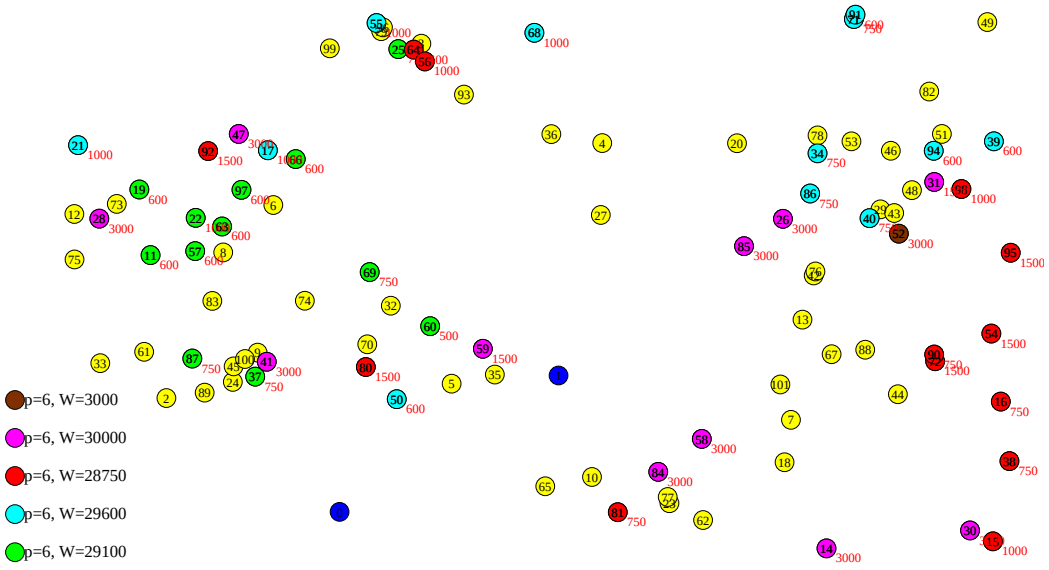
$s = 1$


p=6, W=9000
p=6, W=28000
p=6, W=30000
p=6, W=12800

$s = 2$


p=6, W=19500
p=6, W=26500
p=6, W=29600

$s = 3$


p=6, W=3000
p=6, W=30000
p=6, W=28750
p=6, W=29600
p=6, W=29100

$s = 4$

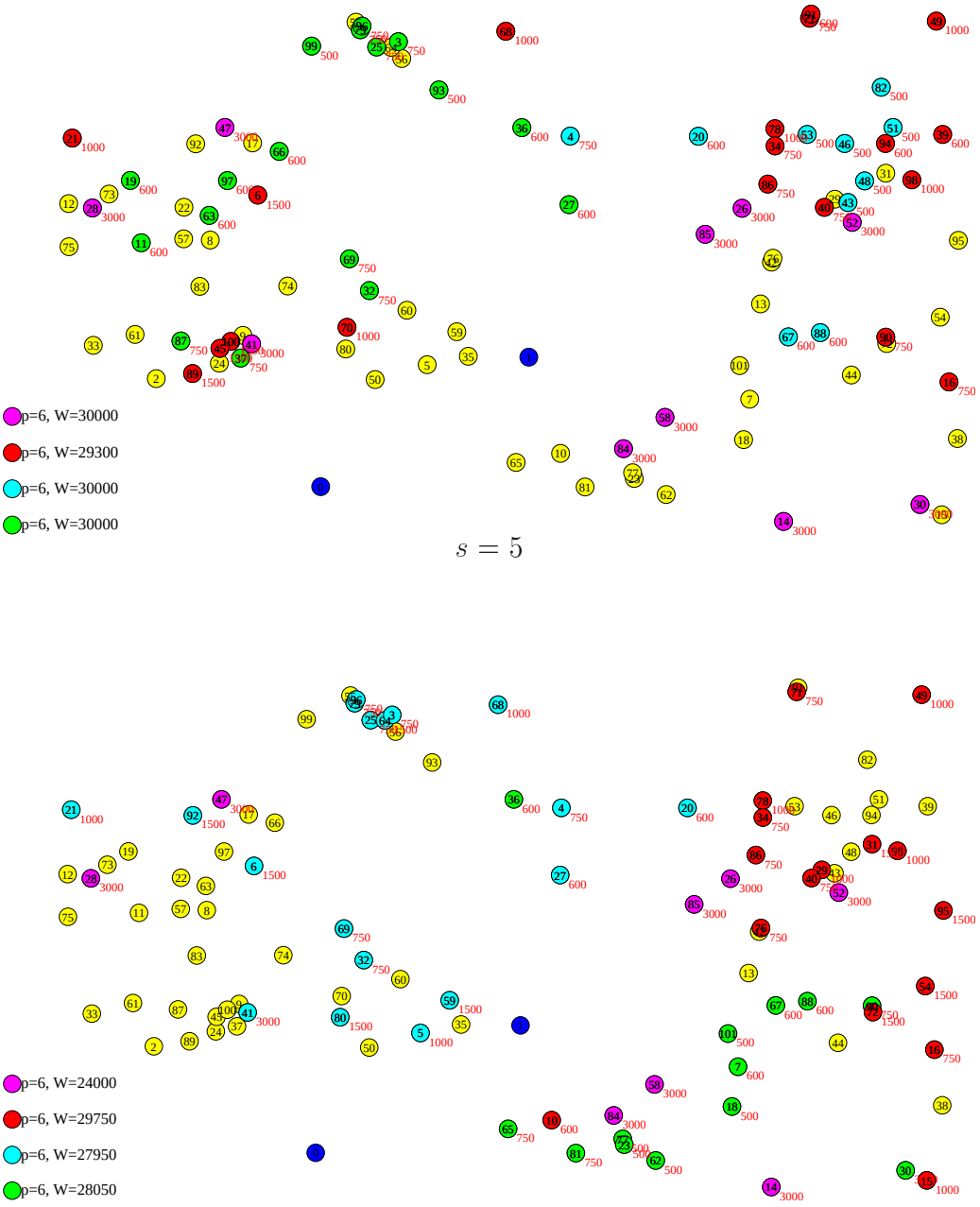



FIG. 7.2 – Solution de 0.rural obtenue par l'heuristique gloutonne avec $\pi_i^{init}$

L'ensemble restreint des colonnes, noté $\overline{R}$, est constitué de toutes les colonnes qui ont été générées durant la résolution du maître $PL$. Dans la contrainte (7.2), la demande de la période $t$ peut être couverte par plusieurs routes. Ce sur-couvrement des demandes permet d'avoir une solution réalisable. En effet, le nombre de véhicules n'étant pas fixé, on peut choisir autant de colonnes que l'on veut, et ainsi choisir pour chaque client un ensemble de colonnes couvrant totalement sa demande. Au nœud racine, les colonnes générées au cours de la résolution du maître $PL$ sans les coupes ont pour la plupart une périodicité égale à $Pmax$, ainsi il est peu probable que la demande d'un client soit couverte par des routes de périodicités incompatibles (par exemple par une route de périodicité 5 et une de périodicité 6). Un post-processing sera nécessaire pour enlever les sur-couvrements.

Pour casser une partie de la symétrie dans le choix de la date de départ $s$, une visite pour un client de référence est fixé à la date $s = 1$. Le client de référence, noté $iref$, est choisi parmi les clients qui ne nécessitent pas une visite toutes les périodes, i.e. $tmax_{iref} > 1$. Ensuite pour espérer que la fixation d'une de ses visites puisse toucher un grand nombre de clients, nous choisissons un client de référence sociable, c'est à dire un client qui tendance à être visité par des routes contenant des clients totalement différents. Nous fixons donc le client qui se trouve, parmi les colonnes de $\overline{R}$, avec le plus de clients différents. Fixer la date de départ $s = 1$ revient à ajouter la contrainte :

$$\sum_{r} \delta^{r1}_{iref\ 1} \lambda_{r1} = 1$$

Ensuite, pour obtenir plus rapidement une solution, l'arbre de Branch-and-Bound est parcouru en profondeur d'abord. Le temps de résolution du maître entier est limité.

Le but étant de résoudre une instance de grande taille, nous avons testé cette heuristique sur les instances aléatoires du groupe AL et l'instance Probleme1. Dans nos tests, nous avons limité le temps de résolution à 2h ; mais aucune solution n'a été trouvée pour ces instances dans cette limite de temps. Nous n'avons donc pas pu visualiser le type de solutions obtenues. Est ce que beaucoup de clients sont sur-couverts ? Comment est le sur-couvrement ? Cette heuristique ne permet pas de trouver des solutions pour notre problème. La principale difficulté provient de la symétrie. La solution linéaire optimale (agrégée et surtout discrète) est très fractionnaire : pour Probleme1, sur les 134 colonnes agrégées étant prises avec une valeur strictement positive, il y en

a 88% avec une valeur inférieure à 0.5 dont 57% avec une valeur inférieure à 0.2. Pour espérer avoir des solutions entières, il faudrait casser la symétrie plus franchement en faisant par exemple plus de fixations (lesquelles ?).

## 7.3 Heuristique d'arrondi

L'heuristique d'arrondi de la table 2.2 de la section 2.4.3 consiste à arrondir itérativement des variables $\lambda$ de la solution $PL$ agrégée pour construire une solution discrète réalisable. A chaque étape de l'heuristique d'arrondi, une solution partielle de $[FormC]$ est connue. A la solution partielle sont associés le vecteur $U$ qui indique, pour chaque période $t$, le nombre de véhicules utilisés et la matrice $D$ qui indique, pour chaque client $i$ et chaque période $t$, si la demande est collectée ou non. La solution partielle est définie par un ensemble de variables $\hat{\lambda}_{rs}$ déjà fixées à leur valeur entière et $U_t = \sum_{rs} \delta_{0t}^{rs} \hat{\lambda}_{rs}$, $D_{it} = \sum_{rs} \delta_{it}^{rs} \hat{\lambda}_{rs}$. Afin de construire une solution discrète entière réalisable, deux mécanismes sont mis en place.

**Mécanismes pour être sûr que la solution entière soit réalisable :**

**1.** Dans le premier mécanisme on ne génère pour la formulation agrégée que des colonnes compatibles avec la solution partielle discrète. Ainsi, en fixant une colonne à la fois, on est sûr que cette colonne fixée peut compléter la solution partielle. Pour ce faire, une liste de triplets $L_i = \{(\ell max, p, s)^r\}$ est tenue à jour pour chaque client $i$ dont la demande est partiellement couverte, $\ell max$ est la quantité maximale qui peut être collectée chez ce client $i$ par une route de périodicité $p$ et commençant à la date $s$. Chaque triplet représente une route possible $r$ pour collecter le complément de $i$, la quantité collectée chez le client dans cette route doit être inférieure à $\ell max$. Cette liste est construite par énumération exhaustive. Initialement pour chaque paire $(p \in P, 1 \leq s \leq p)$, on calcule la quantité maximale, $\ell max$, que peut collecter la route $r$ de périodicité $p$ commençant à la date $s$ (voir l'exemple 7.1). Si $\ell max \geq 1$, alors le triplet $(\ell max, p, s)$ est inséré dans la liste $L_i$. Pour la mise à jour de la liste $L_i$, on recalcule la quantité maximale, $\ell max$, pour chacun des triplets de cette liste (si $\ell max$ tombe à zéro, alors le triplet est enlevé de $L_i$).

Les 3 premières étapes de l'algorithme RH de la table 2.2 doivent donc être adaptées pour construire une solution primale à $[FormC]$ en se basant sur $[FormAgr]$. Elles sont basées sur la connaissance des listes $L_i$.

Exemple 7.1 – Initialisation et mise à jour de $L_i$

Prenons $P = \{1, 2, 3, 4, 6\}$. Considérons un client tel que $tmax_1 = 3$. Fixons comme première colonne : $0 - 1(1) - \cdots - 0$ avec $p = 4$. La date de départ choisie est 1 et on obtient la solution partielle suivante :

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta_{1t}$ | ✓ | | | | ✓ | | | | ✓ | | | |

Le client 1 étant partiellement visité, une liste comprenant les triplets compatibles est créée. Parmi tous les couples $(p \in P, 1 \leq s \leq p)$, les couples $(p, s)$ tels que $p$ est impair et les couples $(2, 1)$, $(4, 1)$, $(6, 1)$, $(6, 3)$, $(6, 5)$ sont éliminés car une de leur occurrence, i.e. à une date $t = s + kp$ avec $k \in \{0, \cdots, \lfloor \frac{T-s}{p} \rfloor\}$, tombe sur une croix ✓ (on trouve $\ell max = 0$). Pour les autres couples $(p, s)$, $\ell max$ est calculé en comptant pour chaque occurrence de la route le nombre de cases vides consécutives, $\ell_k$, alors $\ell max = \min\{tmax_i, p, \min_k(\ell_k)\}$. Par exemple :

- pour le couple $(2, 2)$ : aux dates $t = 2, 6, 10$, le nombre de cases vides est 1, aux dates $t = 4, 8, 12$, le nombre de cases vides est 2. Alors $\ell max = \min\{3, 2, \min(1, 2)\} = 1$ ;
- pour le couple $(4, 3)$ : aux dates $t = 3, 7, 11$, le nombre de cases vides est 2. Alors $\ell max = \min\{3, 4, \min(2)\} = 2$ ;
- pour le couple $(6, 4)$ : à la date $t = 4$, le nombre de cases vides est 3, à la date $t = 10$, le nombre de cases vides est 1. Alors $\ell max = \min\{3, 6, \min(3, 1)\} = 1$ ;

La liste du client 1 est donc :

$$L_1 = \{(1, 2, 2), (1, 4, 2), (2, 4, 3), (3, 4, 4), (1, 6, 2), (1, 6, 4), (1, 6, 6)\}.$$

Maintenant fixons la colonne : $0 - 1(2) - \cdots - 0$ avec $p = 4$. Les dates de départ possibles sont $s = 3$ et $s = 4$. Prenons par exemple $s = 3$, la solution partielle devient :

| | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta_{1t}$ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | |

Les couples $(1, 2, 2), (1, 4, 2), (2, 4, 3), (1, 6, 2), (1, 6, 4), (1, 6, 6)$ de $L_1$ sont éliminés car la nouvelle valeur de $\ell max$ est 0.
Pour le couple $(3, 4, 4)$, $\ell max$ vaut désormais 1.
La liste devient : $L_1 = \{(1, 4, 4)\}$.

Essentiellement, vérifier la compatibilité des colonnes générées avec la solution partielle implique de tester toutes les dates de départ dans la procédure de pricing. De plus, à chaque fixation d'une colonne de $[FormAgr]$,

on doit choisir une date de départ. Les adaptations sont présentées dans la table 7.2.

| |
|---|
| **Etape 0 :** Dans la résolution du maître continu, on ne génère que des colonnes compatibles. Dans la procédure de pricing 6.3, une fois que la périodicité $p$ est fixée, on itère sur les différentes dates de départ $s$ et avant chaque appel à l'oracle associé à la périodicité $p$ et à la date $s$, $\phi_{i\ell}$ est mise à 0 si $(\ell \leq \ell max, p, s) \notin L_i$. |
| **Etape 1 :** Une date de départ doit être choisie pour la colonne fixée, i.e une variable $\lambda_{rs}$ est fixée à 1 (il existe toujours une date départ car les colonnes du maître sont compatibles). |
| **Etape 2 :** Les listes de triplets $L_i$ sont mises à jour et les colonnes ne respectant pas ces listes sont éliminées. |

TAB. 7.2 – Adaptations des étapes de l'heuristique d'arrondi pour construire une solution entière discrète à partir de la solution continue agrégée

**2.** Le deuxième mécanisme mis en place consiste à vérifier si une solution entière à $[FormAgr]$ est une solution réalisable de $[FormC]$. Signalons d'abord qu'une solution de $[FormAgr]$ est rarement entière pour des instances de taille raisonnable. Par contre, si une partie de la solution est déjà fixée, alors les variables $\hat{\lambda}$ du maître $[FormAgr]$ peuvent être entières. Dans ce cas, même si $Vmoy$ est fractionnaire, une solution entière peut être trouvée avec $Vmax \geq \lceil Vmoy \rceil$. Cependant il faut vérifier que la solution $\hat{\lambda}$ complète bien la solution partielle (i.e. que la solution finale soit bien réalisable). On le vérifie en affectant des dates de départ à chacune des colonnes $r$. Le maître discret restreint à l'ensemble des colonnes de $\hat{R}$ est résolu en nombre entier où $\hat{R}$ est l'ensemble des colonnes de la solution agrégée $\hat{\lambda}$. A chacune des ces colonnes $r \in \hat{R}$ et chacune des dates de départ possibles $s$, associons la variable $\lambda_{rs}$ et la matrice indicatrice $\delta^{rs}$ (définies en section 7.2). La solution agrégée $\hat{\lambda}$ est réalisable si le problème $[FormReal]$ ci-dessous est réalisable.

$$[FormReal] \quad \min \quad Vmax \quad +\alpha \sum_{r \in \hat{R}\, s} \frac{c^r}{p^r}\, \lambda_{rs} \tag{7.6}$$

$$\sum_{rs} \delta^{rs}_{it} \lambda_{rs} \quad = \quad 1 - D_{it} \qquad \forall i, t \tag{7.7}$$

$$U_t + \sum_{rs} \delta^{rs}_{0t} \lambda_{rs} \quad \leq \quad Vmax \qquad \forall t \tag{7.8}$$

$$\lambda_{rs} \in \{0,1\} \quad \forall\, r,s \tag{7.9}$$
$$Vmax \in I\!N. \tag{7.10}$$

La connaissance de la solution partielle casse en partie la symétrie et le nombre de colonnes dans $\hat{R}$ est dans la pratique assez petit ($|\hat{R}| \leq 10$), $[FormReal]$ est résolu rapidement (quelques secondes au plus). De plus $[FormReal]$ n'est résolu que si on a l'espoir d'améliorer une solution connue. Soit $Z_{PLE}^{AgC}$ le coût de la solution agrégée comprenant le coût de la solution partielle (le coût associé à $Vmoy$ est compris dans $Z_{PLE}^{AgC}$ et $Vmoy$ peut être fractionnaire) et $INC$ le coût de la meilleure solution discrète connue, si $Z_{PLE}^{AgC} + \lceil Vmoy \rceil - Vmoy \geq INC$ alors $[FormReal]$ n'est pas résolu.

**Différentes variantes de l'heuristiques d'arrondi :**

L'heuristique d'arrondi est appelée sur la solution $PL$ optimale. A l'étape 0 de l'algorithme RH de la table 2.2, le nombre de colonnes qui peuvent être générées durant la réoptimisation du maître continu est limité (ce paramètre est noté nbMaxCgRH et vaut 300 dans nos tests). A l'étape 3 de l'algorithme RH de la table 2.2, lorsque les critères d'arrêt sont atteints, on appelle à nouveau l'heuristique sur la même solution $PL$ mais la première colonne fixée est différente (i.e. on fait une nouvelle passe). Dans nos tests, nous avons appelé l'heuristique 3 fois sur la même solution (i.e. on fait 3 passes).

Le maître $PL$ agrégé est réalisable car le nombre de véhicules n'est pas limité. Néanmoins, notre heuristique ne trouve pas de solution réalisable si, après nbMaxCgRH itérations, les colonnes artificielles font toujours partie de la solution $PL$. Ne pouvant pas générer de nouvelles colonnes, une phase 1 pure conclut que le maître $PL$ est irréalisable.

Des variantes de l'heuristique sont essayées, elles diffèrent par :

1. le critère de sélection de la variable $\lambda_{rs}$ fixée ;
2. le paramètre nbMaxCgRH ;
3. la solution de départ (la solution $PL$ optimale avant ou après ajout des coupes, ou plusieurs solutions $PL$).

Pour l'étape 1 de l'algorithme RH de la table 2.2, plusieurs critères de fixation sont testés. Remarquons que la valeur primale d'une colonne $\lambda_r$ ne représente pas son attractivité. En effet $\lambda_r \in I\!N$ mais $\lambda_{rs} \in \{0,1\}$, donc une

colonne vérifiant l'observation 6.3.1 ayant une valeur de 1 peut être plus attractive qu'une colonne ne vérifiant pas 6.3.1 mais ayant une valeur de 1.1. Le choix de la route et de la date de départ est dicté par un classement des colonnes de la solution $PL$ par ordre croissant du "score" attribué à chacune d'elles.

Le score d'une colonne $r$ représente le coût qu'elle engendre $coutCol_r$ divisé par la satisfaction de contraintes $satContr_r$ et dépend de la date $s$ à laquelle on la fixe. Le coût engendré par la colonne $r$ est le coût associé à cette colonne dans la solution, i.e. $\alpha\frac{c^r}{p^r}$, plus la répercussion sur la valeur de $Vmax$, i.e. $\delta_{rs}$ est ajouté à $\alpha\frac{c^r}{p^r}$ ($\delta_{rs} = 1$ si la variable $Vmax = \max_t\{U_t\}$ doit augmenter de 1 après cette fixation, 0 sinon). Ainsi :

$$coutCol_r = \delta_{rs} + \alpha\frac{c^r}{p^r}$$

Le critère de satisfaction des contraintes peut être évalué de trois façons.

1. toutes les contraintes (6.2) et (6.3) sont prises en compte,

$$satContr_r^1 = \frac{\sum_i \ell\phi_{i\ell}}{p^r} \times p^r = \sum_i \ell\phi_{i\ell}$$

2. seulement les contraintes (6.2) sont prises en compte,

$$satContr_r^2 = \frac{\sum_i \ell\phi_{i\ell}}{p^r}$$

3. aucune contrainte n'est prise en compte, $satContr_r^3 = 1$.

Le critère coût réduit est fait de différences tandis que le critère score et le critère de satisfaction des contraintes sont faits de rapports.

La colonne à fixer peut donc être celle ayant le plus petit score $\frac{coutCol_r}{sc_r}$ où $sc_r \in \{satContr_r^1, satContr_r^2, satContr_r^3\}$ (ces trois critères sont respectivement notés C1, C2, C3). La valeur $\lambda_r$ peut aussi être utilisée. On fixe alors la colonne pour laquelle le score divisé par $min(1, \lambda_r)$ est minimum (on ajoute la lettre $L$ à la notation du critère). Pour comparer ces critères par rapport à un critère basique, on implémente aussi une variante où la colonne peut être choisie avec une probabilité $\lambda_r$ si $\lambda_r > 0.1$ (le critère aléatoire).

Ensuite il faut choisir la date de départ $s$. La date de départ de la première colonne est fixée à 1. Pour les autres colonnes, la date peut être choisie selon

un critère de répartition de la charge entre les périodes (le choix de ce critère est noté S1). Il s'agit de définir un ensemble attractif de dates, et la date de la colonne $r$ est choisie aléatoirement parmi cet ensemble attractif. Les dates $s$ candidates pour appartenir à l'ensemble attractif sont les dates pour lesquelles $\delta_{rs} = 0$, i.e. il n'est pas utile de prendre un autre véhicule (si pour toutes les dates $\delta_{rs} = 1$, alors elles sont toutes candidates). Parmi ces candidates, seules les dates ayant une charge minimum sont insérées dans l'ensemble attractif. La charge d'une date, $charge_s$, mesure le rapport entre la quantité des demandes des différentes périodes d'occurrence de la route restant à couvrir et la capacité restantes des véhicules. Soit $\hat{R}$ les colonnes de la solution partielle, $charge_s$ correspond à la somme, sur les périodes d'occurrence de la route (i.e. aux périodes $t$ telles que $\delta_{0t}^{rs} = 1$), de la quantité des demandes non encore couvertes $(\sum_i d_i(1 - D_{it}))$ divisée par la capacité restante des véhicules $((V - U_t)W$ où $V$ est une borne supérieure sur le nombre de véhicules nécessaires),

$$charge_s = \sum_{t,\delta_{0t}^{rs}=1} \frac{\sum_i d_i(1 - D_{it})}{(V - U_t)W}.$$

Pour tester l'efficacité de ce critère, nous pouvons le comparer avec le critère basique qui choisit la date de départ aléatoirement parmi toutes les dates possibles. Le récapitulatif des critères de fixation est dans la table 7.3.

Fixation de la colonne $\overline{r}$ :

$$\begin{cases} \text{C0 :} & \text{aléatoirement,} \\ \text{C1 :} & \\ \text{C2 :} & \overline{r} = argmin_r\{\frac{coutCol_r}{sc_r}\} \text{ avec } sc_r = \begin{cases} satContr_r^1, \\ satContr_r^2, \\ satContr_r^3, \end{cases} \\ \text{C3 :} & \\ \text{C1L :} & \\ \text{C2L :} & \overline{r} = argmin_r\{\frac{coutCol_r}{sc_r \ \min(1,\lambda_r)}\} \text{ avec } sc_r = \begin{cases} satContr_r^1, \\ satContr_r^2, \\ satContr_r^3. \end{cases} \\ \text{C3L :} & \end{cases}$$

Fixation de la date de départ :

$$\begin{cases} \text{S0 :} & \text{aléatoirement,} \\ \text{S1 :} & \text{selon la charge des périodes.} \end{cases}$$

TAB. 7.3 – Récapitulatif des différents critères de fixation

La qualité des solutions primales obtenues par les différents critères est comparée dans le tableau 7.4. Les temps d'obtention de ces solutions sont similaires. Pour les instances des deux groupes AL et S60 et les deux grandes

instances GN et Pb1, la meilleure solution est mise en gras. Aucun critère ne domine systématiquement les autres. Cependant, on observe que les critères C0 et C3 (voir la table 7.3) sont nettement moins efficaces que C1 et C2. Avec le critère C2, on a plus de difficultés pour trouver des solutions. Finalement, le critère retenu est C1. Parmi les 3 tests avec C1, celui qui utilise le moins de véhicules est C1S1. Les tests suivants se font avec le critère C1S1. Les temps de calculs et le nombre de colonnes générées avec ce critère sont indiqués dans le tableau 7.5 (pour les abréviations, voir la table 6.1). Par rapport au tableau 6.4, le nombre de colonnes générées est multiplié par 5 au maximum alors que le temps peut être multiplié par 12 (pour Pb1) : la majeure partie du temps est passée dans l'heuristique où beaucoup de colonnes sont générées. Le tableau 7.5 sert de tableau témoin, les variantes de l'heuristique d'arrondi sont comparées à ce tableau. A chaque fois, les meilleures solutions sont mises en gras.

| | moy AL | | | moy S60 | | | GN | | | Pb1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PB | dev | V | PB | dev | V | PB | dev | V | PB | dev | V |
| C0S0 | 515.29 | 41.43 | 5.5 | 347.303 | 38.31 | 4 | 658.055 | 17.83 | 7 | 1e+12 | | |
| C1S0 | 428.476 | 17.35 | 4.2 | 313.81 | 27.35 | 3.4 | 601.598 | 7.73 | 6 | 732.679 | 11.04 | 7 |
| C1S1 | 416.056 | 14.12 | 4 | 297.29 | 18.39 | 3.2 | 596.963 | 6.89 | 6 | **731.48** | **10.86** | **7** |
| C1LS1 | **412.033** | **12.99** | **4** | 297.994 | 19.03 | 3.2 | 597.025 | 6.91 | 6 | 793.558 | 20.26 | 8 |
| C2S1 | Pas de solution pour 3/10 | | | 311.56 | 22.48 | 3.4 | 595.701 | 6.67 | 6 | 1e+12 | | |
| C2LS1 | 412.827 | 13.19 | 4 | **285.277** | **14.28** | **3** | **594.851** | **6.52** | **6** | 1e+12 | | |
| C3S1 | 485.862 | 33.26 | 5 | 327.11 | 30.55 | 3.6 | 658.437 | 17.90 | 7 | 801.914 | 21.53 | 8 |
| C3LS1 | 445.731 | 22.16 | 4.5 | 308.898 | 25.15 | 3.4 | 664.579 | 19.00 | 7 | 798.515 | 21.02 | 8 |

TAB. 7.4 – Comparaison de la qualité des solutions primales discrètes obtenues par les différents critères de fixation

| | Bornes | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Nom | bestPB | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| moy AL | 416.056 | 14.1216 | 4 | 15064 | 14989 | 172 | 358 | 137 | 522 | 10m5s (605) |
| moy S60 | 297.29 | 18.3989 | 3.2 | 5636 | 5556 | 105 | 160 | 26 | 181 | 3m33s (213) |
| GN | 596.963 | 6.89603 | 6 | 20689 | 20527 | 1291 | 1722 | 341 | 1753 | 37m56s (2276) |
| Pb1 | 731.48 | 10.8573 | 7 | 15096 | 15047 | 1744 | 2122 | 257 | 2364 | 42m3s (2523) |

TAB. 7.5 – Temps de calcul de l'obtention de la solution primale discrète par le critère témoin C1S1

Le test suivant étudie l'influence sur l'heuristique du paramètre limitant le nombre d'itérations pendant la réoptimisation, nbMaxCgRH est mis à 100 au lieu de 300. Les résultats sont rapportés dans le tableau 7.6. Il est clair que l'heuristique avec nbMaxCgRH à 100 est plus rapide que l'heuristique avec

nbMaxCgRH à 300 (les temps sont au moins divisés par 1.8), mais il y a une perte de la qualité de la solution (avec nbMaxCgRH à 100, les déviations à l'optimum sont plus importantes et plus de véhicules sont utilisés en moyenne pour S60). Pour les tests suivants, MaxCgIterRh reste à 300.

| | Bornes | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Nom | bestPB | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| Temoin : nbMaxCgRH = 300 | | | | | | | | | | |
| moy AL | **416.056** | **14.1216** | **4** | 15064 | 14989 | 172 | 358 | 137 | 522 | 10m5s (605) |
| moy S60 | **297.29** | **18.3989** | **3.2** | 5636 | 5556 | 105 | 160 | 26 | 181 | 3m33s (213) |
| GN | **596.963** | **6.89603** | **6** | 20689 | 20527 | 1291 | 1722 | 341 | 1753 | 37m56s (2276) |
| Pb1 | **731.48** | **10.8573** | **7** | 15096 | 15047 | 1744 | 2122 | 257 | 2364 | 42m3s (2523) |
| nbMaxCgRH = 100 | | | | | | | | | | |
| moy AL | 419.508 | 15.0366 | 4 | 8727 | 8699 | 43 | 145 | 75 | 188 | 4m41s (281) |
| moy S60 | 313.933 | 27.2 | 3.4 | 4245 | 4189 | 36 | 70 | 16 | 74 | 1m47s (107) |
| GN | 608.674 | 8.99312 | 6 | 15250 | 15201 | 516 | 820 | 280 | 731 | 20m58s (1258) |
| Pb1 | 757.34 | 14.7765 | 7 | 7500 | 7477 | 265 | 412 | 82 | 392 | 9m23s (563) |

TAB. 7.6 – Comparaison de la solution primale discrète obtenue en faisant varier le paramètre nbMaxCgRH

Les trois tests suivants concernent la solution de départ. Dans le premier test, l'heuristique est appelée toutes les 200 itérations si la solution $PL$ est réalisable (i.e. si elle ne contient pas de colonnes artificielles). Ainsi différentes solutions de départ sont essayées. Les résultats sont rapportés dans le tableau 7.7. Pour l'instance GN, l'heuristique n'est appelée que sur la solution $PL$ optimale, les solutions intermédiaires contenaient les variables artificielles. Pour les autres instances, le temps est multiplié par 1.6 au moins et les solutions obtenues utilisent plus de véhicules en moyenne. Par contre, pour Pb1, la déviation à l'optimum a diminué. Dans les tests qui suivent, l'heuristique est appelée seulement sur la solution $PL$ optimale.

Dans le test qui suit, l'heuristique est appelée sur la solution $PL$ après l'ajout des coupes vues en section 6.4.1. Ces coupes ont un faible impact sur la borne duale, cependant elles modifient la solution $PL$ en diversifiant les périodicités utilisées, mais la solution devient plus fractionnaire. Les résultats sont dans le tableau 7.8. Le nombre de colonnes générées est 1.39 fois plus grand que le nombre de colonnes générées dans le test témoin, les temps de calcul sont multipliés par au moins 3 et les déviations à l'optimum sont plus importantes. Plus de la moitié du temps est passé dans l'heuristique. En comparaison avec les résultats témoins, l'heuristique avec les coupes a besoin de générer plus de colonnes, cela est dû en partie au maître devenu plus difficile à résoudre.

| | Bornes | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Nom | bestPB | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| Temoin : Appel sur la solution $PL$ optimale | | | | | | | | | | |
| moy AL | **416.056** | **14.1216** | **4** | 15064 | 14989 | 172 | 358 | 137 | 522 | 10m5s(605) |
| moy S60 | **297.29** | **18.3989** | **3.2** | 5636 | 5556 | 105 | 160 | 26 | 181 | 3m33s(213) |
| GN | **596.963** | **6.89603** | **6** | 20689 | 20527 | 1291 | 1722 | 341 | 1753 | 37m56s(2276) |
| Pb1 | 731.48 | 10.8573 | 7 | 15096 | 15047 | 1744 | 2122 | 257 | 2364 | 42m3s (2523) |
| Appel toutes les 200 itérations | | | | | | | | | | |
| moy AL | 417.112 | 14.3053 | 4.1 | 24643 | 24500 | 294 | 607 | 343 | 1127 | 19m57s(1197) |
| moy S60 | 310.246 | 25.8825 | 3.4 | 8663 | 8480 | 167 | 252 | 49 | 330 | 5m57s(357) |
| GN | **596.963** | **6.89603** | **6** | 20689 | 20527 | 1291 | 1722 | 341 | 1753 | 37m56s(2276) |
| Pb1 | **703.295** | **6.58584** | **7** | 33946 | 33709 | 4290 | 5106 | 842 | 6185 | 1h45m49s(6349) |

TAB. 7.7 – Comparaison de la solution primale discrète obtenue en appelant l'heuristique d'arrondi sur plusieurs solutions $PL$

| | Bornes | | Compteurs | | | Temps | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Nom | bestPB | dev | V | SpSol | Col | MCKP | SpSol | RM | CP | RH | Total |
| Temoin : Appel sur la solution $PL$ optimale | | | | | | | | | | | |
| moy AL | **416.056** | **14.1216** | **4** | 15064 | 14989 | 172 | 358 | 137 | | 522 | 10m5s(605) |
| moy S60 | **297.29** | **18.3989** | **3.2** | 5636 | 5556 | 105 | 160 | 26 | | 181 | 3m33s(213) |
| GN | **596.963** | **6.89603** | **6** | 20689 | 20527 | 1291 | 1722 | 341 | | 1753 | 37m56s(2276) |
| Pb1 | **731.48** | **10.8573** | **7** | 15096 | 15047 | 1744 | 2122 | 257 | | 2364 | 42m3(2523) |
| Appel sur la solution $PL$ après l'ajout des coupes | | | | | | | | | | | |
| moy AL | 481.493 | 32.5501 | 4.9 | 21027 | 20965 | 828 | 1166 | 1026 | 1075 | 1493 | 46m23s(2783) |
| moy S60 | 300.623 | 19.8666 | 3.2 | 8737 | 8670 | 405 | 506 | 198 | 205 | 589 | 14m46s(886) |
| GN | 597.77 | 7.04053 | 6 | 31839 | 31748 | 2510 | 3366 | 2398 | 1536 | 4627 | 1h54m32s(6872) |
| Pb1 | 830.392 | 25.8477 | 8 | 34090 | 33982 | 2743 | 3627 | 3557 | 1919 | 6388 | 2h25m23s(8723) |

TAB. 7.8 – Comparaison de la solution primale discrète obtenue en appelant l'heuristique d'arrondi après l'ajout des coupes

Le dernier test consiste à appeler l'heuristique d'arrondi après le branchement sur la variable $Vmoy$ (mais sans l'ajout de coupes). L'heuristique est appelée à la racine et au nœud 2 (comme le nœud 1 est irréalisable, il n'est pas exploré). Les résultats obtenus sont repris dans le tableau 7.9 [1]. On constate d'abord que pour le même nombre de colonnes générées, le temps de calcul dans le cas où l'heuristique est appelée à différents nœuds est moins important

[1]Signalons que dans chaque groupe d'instances, les tests sont effectués en série et un changement dans la résolution du fichier précédent peut faire que l'oracle ne retourne pas la même colonne (on constate que plusieurs colonnes ont le même coût réduit, une seule d'entre elles est retournée). Dans l'algorithme de la table 3.3, $\lambda$ est la médiane de $M$ pentes $\lambda_i$. Pisinger choisit ces $M = 15$ pentes aléatoirement, l'ordre des articles peut ensuite changer. La solution $PL$ de la racine peut donc avoir des colonnes différentes ; c'est pourquoi, rootPB= $299.61 \neq 297.29$ pour moy S60

que dans la cas où l'heuristique est appelée après l'ajout des coupes. De plus, l'heuristique a été appelée deux fois au lieu d'une. Les solutions de la racine se sont trouvées améliorées en moyenne sur les 17 instances de 3.11%. Cette variante donne des solutions primales avec un nombre de véhicules inférieur au nombre de véhicules constaté dans les tests précédents.

| | Bornes | | | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nom | rootPB | bestPB | am | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| Appel sur la solution $PL$ après l'ajout des coupes | | | | | | | | | | | | |
| moy AL | 481.493 | | | 32.55 | 4.9 | 21027 | 20965 | 828 | 1166 | 1075 | 1493 | 46m23s(2783) |
| moy S60 | 300.623 | | | 19.86 | 3.2 | 8737 | 8670 | 405 | 506 | 198 | 589 | 14m46s(886) |
| GN | 597.77 | | | 7.040 | 6 | 31839 | 31748 | 2510 | 3366 | 2398 | 4627 | 1h54m32s(6872) |
| Pb1 | 830.392 | | | 25.84 | 8 | 34090 | 33982 | 2743 | 3627 | 3557 | 6388 | 2h25m23s(8723) |
| Appel à la racine et après le branchement sur $Vmoy$ | | | | | | | | | | | | |
| moy AL | 420.544 | **409.399** | **2.72** | **12.29** | **4** | 23481 | 23269 | 287 | 582 | 298 | 990 | 18m22s(1102) |
| moy S60 | 299.61 | **285.056** | **4.89** | **11.75** | **3** | 8300 | 8107 | 140 | 222 | 49 | 288 | 6m10s(370) |
| GN | 596.963 | **589.582** | **1.25** | **5.574** | **6** | 32261 | 31871 | 2511 | 3241 | 761 | 3914 | 1h22m2s(4922) |
| Pb1 | **731.48** | 731.48 | **0** | **10.85** | **7** | 32239 | 31908 | 3324 | 4083 | 652 | 4925 | 1h30m16s(5416) |

TAB. 7.9 – Comparaison de la solution primale discrète obtenue en appelant l'heuristique d'arrondi à la racine et après le branchement sur $Vmoy$

Les solutions entières d'un espace plus restreint sont aussi des solutions valides pour l'espace plus large (l'espace des solutions actuelles est défini par l'ensemble $P = \{1, 2, 3, 4, 5, 6\}$). Deux restrictions sont possibles.

1. Dans le cluster nous forçons la quantité collectée chez chaque client à être égale à la périodicité, i.e. $\ell = p \; \forall i$. Cette politique est appelé la politique de partition fixe (notée PF) rencontrée dans la littérature.
2. Nous limitons l'espace en imposant aux périodicités de prendre leur valeur dans l'ensemble $P = \{1, 2, 3\}$.

L'heuristique d'arrondi appelée à la racine et après le branchement sur Vmoy permet de trouver des solutions pour ces deux politiques. Dans le cas de la politique PF, comme le problème est très restreint, la colonne de meilleur coût réduit est retournée à chaque itération. Dans le cas où $P = \{1, 2, 3\}$, d'après les tests sur le branchement disjonctif effectués à la section 6.4.2, on a constaté que le nœud 1 est irréalisable, il n'est donc pas exploré. Les résultats de ces tests sont dans les tableaux 7.10 et 7.11 respectivement.

Les solutions obtenues avec la politique PF font partie des moins bonnes solutions trouvées. La déviation à l'optimum est estimée à 26.49% en moyenne et le nombre de véhicules nécessaires est important. La seule exception est la

solution trouvée pour l'instance Pb1 qui est à 6.78% de l'optimum. D'ailleurs dans cette solution, le nombre de véhicules utilisés est le plus petit rencontré au cours des différents tests. Le temps de résolution est en revanche plus rapide que dans le cas non restreint, d'autant plus que du temps est passé dans le nœud 1 (ce nœud est réalisable seulement pour l'instance 6.mixte). Une fois encore, le branchement a permis d'améliorer la solution obtenue à la racine de 2.76% en moyenne sur les 17 instances. Les bornes duales de la politique PF obtenues sont respectivement de 397.47 pour les instances du groupe AL, de 267.376 pour les instances du groupe S60, de 581.008 pour l'intance GN et de 691.481 pour l'intance Pb1. Dans ce cas, la déviation à l'optimum reste importante, elle est estimée à 15.45% pour les instances du groupe AL, à 26.16% pour les instances du groupe S60, à 12.64% pour l'intance GN et à 1.89% pour l'intance Pb1.

| | Bornes | | | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nom | rootPB | bestPB | am | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| $P = \{1,2,3,4,5,6\}$ | | | | | | | | | | | | |
| moy AL | 420.544 | **409.399** | **2.72** | **12.29** | **4** | 23481 | 23269 | 287 | 582 | 298 | 990 | 18m22s(1102) |
| moy S60 | 299.61 | **285.056** | **4.89** | **11.75** | **3** | 8300 | 8107 | 140 | 222 | 49 | 288 | 6m10s(370) |
| GN | 596.963 | **589.582** | **1.25** | **5.574** | **6** | 32261 | 31871 | 2511 | 3241 | 761 | 3914 | 1h22m2s(4922) |
| Pb1 | 731.48 | 731.48 | 0 | 10.85 | 7 | 32239 | 31908 | 3324 | 4083 | 652 | 4925 | 1h30m16s(5416) |
| Politique de la Partition Fixe | | | | | | | | | | | | |
| moy AL | 477.386 | 458.881 | 4.03 | 25.54 | 4.8 | 6293 | 6212 | 341 | 417 | 81 | 330 | 21m18s(1278) |
| moy S60 | 338.371 | 337.326 | 0.31 | 34.19 | 3.8 | 3144 | 3089 | 129 | 158 | 31 | 241 | 11m33s(693) |
| GN | 674.061 | 654.461 | 2.99 | 17.19 | 7 | 12023 | 11899 | 2889 | 3177 | 284 | 1468 | 1h37m6s(5826) |
| Pb1 | 719.876 | **704.579** | **2.17** | **6.780** | **7** | 8785 | 8658 | 996 | 1166 | 158 | 814 | 39m55s(2395) |

TAB. 7.10 – Comparaison de la solution primale discrète obtenue en appliquant la Politique de Partition Fixe

La restriction de l'espace à $P = \{1,2,3\}$ permet d'obtenir de bonnes solutions. La déviation à l'optimum est estimée à 9.67% en moyenne (alors que lorsque $P = \{1,2,3,4,5,6\}$ la déviation à l'optimum est de 11.66% en moyenne) et le nombre de véhicules utilisés dans la solution est le nombre minimum connu. Nous remarquons que pour les instances du groupe S60, la moyenne des déviations à l'optimum est plus petite lorsque $P = \{1,2,3,4,5,6\}$ que lorsque $P = \{1,2,3\}$, par contre pour la moyenne du coût des solutions c'est l'inverse. Dans les deux cas, les valeurs des solutions sont similaires. Le temps nécessaire pour obtenir ces solutions est inférieur à celui du cas non restreint, ce temps est divisé par 2 (ou plus pour l'instance PB1). Le branchement améliore un peu la borne primale de la racine. Cette amélioration est de 1% en moyenne sur les 17 instances. En utilisant la meilleure borne duale obtenue

pour cette politique où $P = \{1, 2, 3\}$ (ces bornes duales sont indiquées dans le tableau 6.13), la déviation à l'optimum de ces solutions est estimée à 4.12% pour les instances du groupe AL, à 8.53% pour les instances du groupe S60, à 3.63% pour l'instance GN et à 2.24% pour l'instance Pb1.

| | Bornes | | | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nom | rootPB | bestPB | am | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| $P = \{1, 2, 3, 4, 5, 6\}$ | | | | | | | | | | | | |
| moy AL | 420.544 | 409.399 | 2.72 | 12.29 | 4 | 23481 | 23269 | 287 | 582 | 298 | 990 | 18m22s(1102) |
| moy S60 | 299.61 | 285.056 | 4.89 | **11.75** | **3** | 8300 | 8107 | 140 | 222 | 49 | 288 | 6m10s(370) |
| GN | 596.963 | **589.582** | **1.25** | **5.574** | **6** | 32261 | 31871 | 2511 | 3241 | 761 | 3914 | 1h22m2s(4922) |
| Pb1 | 731.48 | 731.48 | 0 | 10.85 | 7 | 32239 | 31908 | 3324 | 4083 | 652 | 4925 | 1h30m16s(5416) |
| $P = \{1, 2, 3\}$ | | | | | | | | | | | | |
| moy AL | 404.096 | **399.945** | **1.04** | **9.237** | **4** | 13035 | 12956 | 68 | 126 | 171 | 281 | 8m56s(536) |
| moy S60 | 286.362 | **284.155** | **0.77** | 11.83 | **3** | 5219 | 5144 | 18 | 34 | 41 | 72 | 3m1s(181) |
| GN | 607.103 | 602.505 | 0.76 | 7.888 | 6 | 27542 | 27426 | 382 | 582 | 1476 | 843 | 51m39s(3099) |
| Pb1 | 707.523 | **693.564** | **2.01** | **5.111** | **7** | 16863 | 16731 | 239 | 348 | 347 | 598 | 19m32s(1172) |

TAB. 7.11 – Comparaison de la solution primale discrète obtenue en restreignant l'espace des périodicités, $P = \{1, 2, 3\}$

Dans le tableau 7.12, la meilleure borne primale obtenue au cours de ces différents tests est notée pour chaque instance, en précisant la variante utilisée. En moyenne, les meilleures solutions sont obtenues en restreignant l'espace à $P = \{1, 2, 3\}$. Ces solutions sont obtenues en moins de 1h et la déviation à l'optimum est de 9.67% en moyenne. Dans le cas sans restriction où $P = \{1, 2, 3, 4, 5, 6\}$, les meilleures solutions sont obtenues en moins de 1h30 avec la variante où :

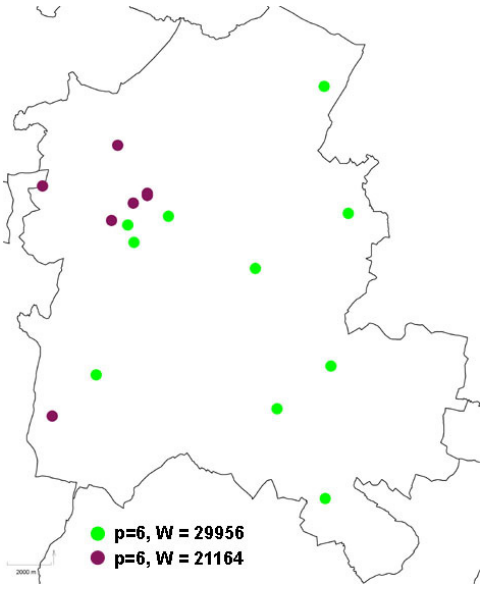
- le critère de fixation est C1S1 (défini dans la table 7.3),
- MaxCgIter = 300 et il y a 3 passes,
- l'heuristique d'arrondi est appelée sur la solution $PL$ optimale (sans coupes) de la racine et du nœud 2.

Dans ce cas, la déviation à l'optimum est de 11.66% en moyenne. Pour pratiquement toutes les instances, des solutions avec une déviation à l'optimum inférieure à 10% sont trouvées. Seule l'instance 6.mixte a une déviation à l'optimum supérieure à 25%. La meilleure solution de 6.mixte est la seule solution pour laquelle le nombre de véhicules utilisés dans la solution agrégée est inférieur au nombre de véhicules utilisés dans la solution discrète, i.e. $Vmax = Vmoy^* + 1$.

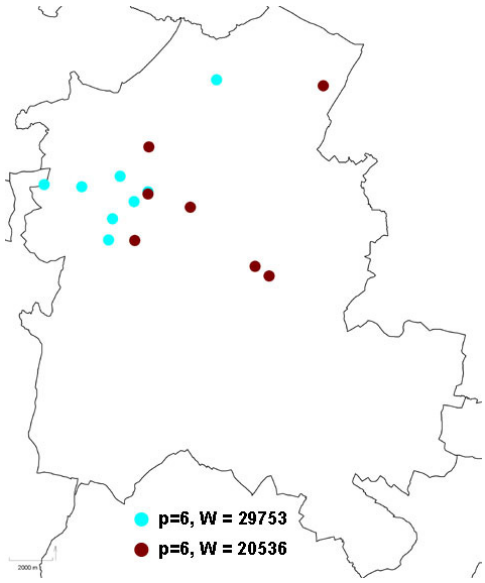
| Nom | bestPB | dev | V | variante |
|---|---|---|---|---|
| 0.mixte | 379.243 | 6.51936 | 4 | C1S1-P123 |
| 1.mixte | 399.318 | 7.71999 | 4 | C2S1 |
| 3.mixte | 428.159 | 5.87539 | 4 | C1S1-P123 |
| 6.mixte | 367.268 | 27.9007 | 4 | C1S1-P123 |
| 0.rural | 414.525 | 8.04066 | 4 | C1S1-P123 |
| 1.rural | 413.777 | 8.55507 | 4 | C2S1 |
| 2.rural | 411.321 | 9.67858 | 4 | C1S1-P123 |
| 9.rural | 396.108 | 6.7999 | 4 | C1S1-P123 |
| 0.urbain | 407.787 | 6.95323 | 4 | C1S1-P123 |
| 3.urbain | 372.651 | 6.1947 | 4 | C1S1-P123 |
| Jonzac | 192.707 | 5.46307 | 2 | C1S1+Br |
| Nord | 290.442 | 5.37558 | 3 | C0S0 |
| Ouest | 188.574 | 6.39775 | 2 | C1S1 |
| SudP6 | 282.753 | 4.31141 | 3 | C2LS1 |
| NordJonzacP6 | 381.92 | 4.99205 | 4 | C2S1-PF |
| GrandNord | 589.582 | 5.57432 | 6 | C1S1+Br |
| Probleme1 | 693.564 | 5.11109 | 7 | C1S1-P123 |
| moy 17 inst | 388.866 | 7.73311 | 3.94 | |

TAB. 7.12 – Récapitulatifs des bornes primales obtenues par l'heuristique d'arrondi

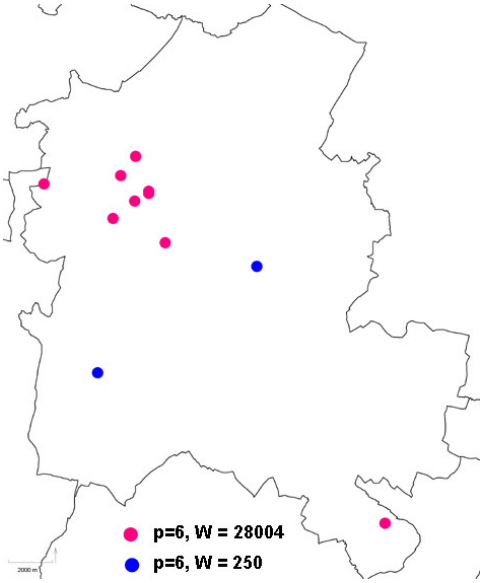
Les figures 7.3 et 7.4 illustrent les meilleures solutions obtenues pour les deux instances Jonzac et 0.rural. On constate que les clusters de le figure 7.4 sont plus homogènes que ceux de la figure 7.2. De plus, le nombre de véhicules par période est mieux équilibré, mais la charge ne l'est pas forcément.



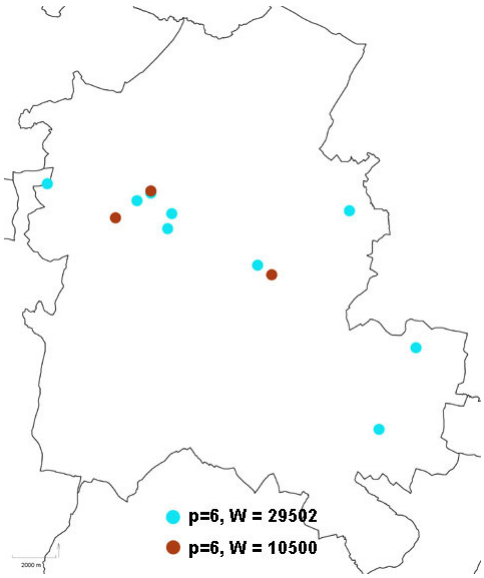

$s = 1$



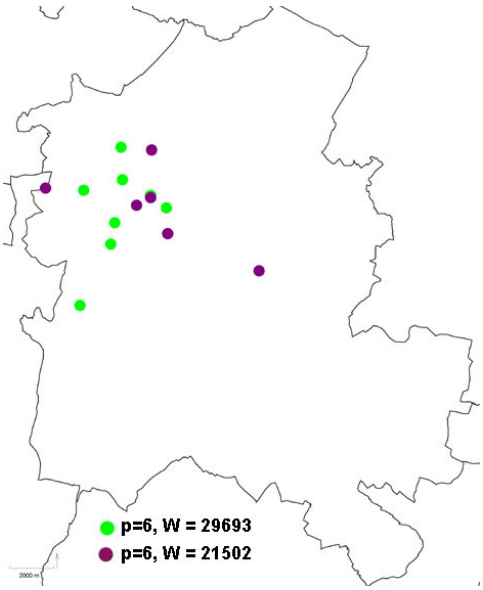

$s = 2$



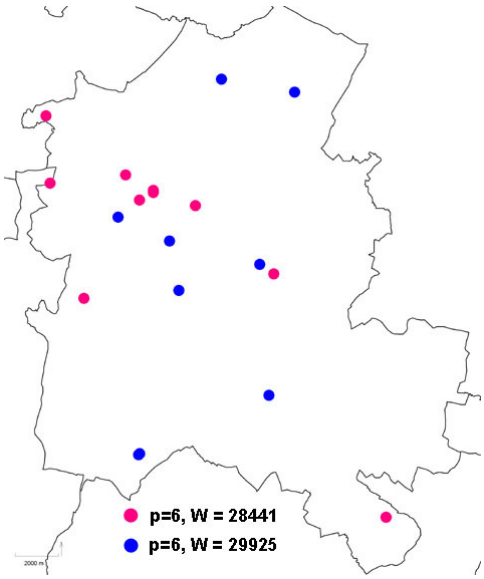

$s = 3$




$s = 4$

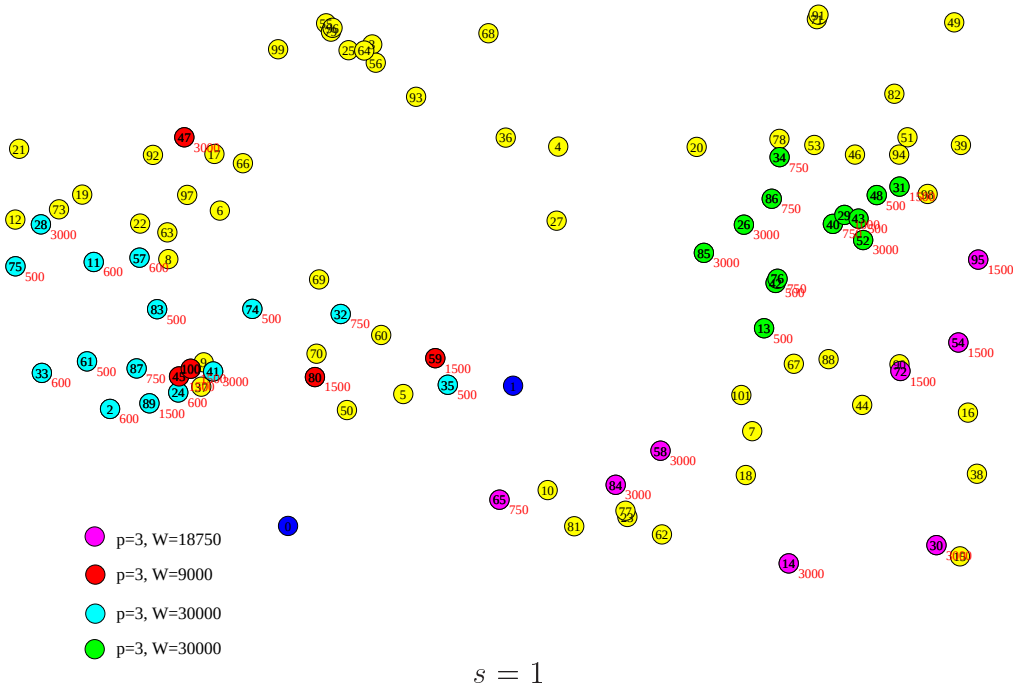



$s = 5$

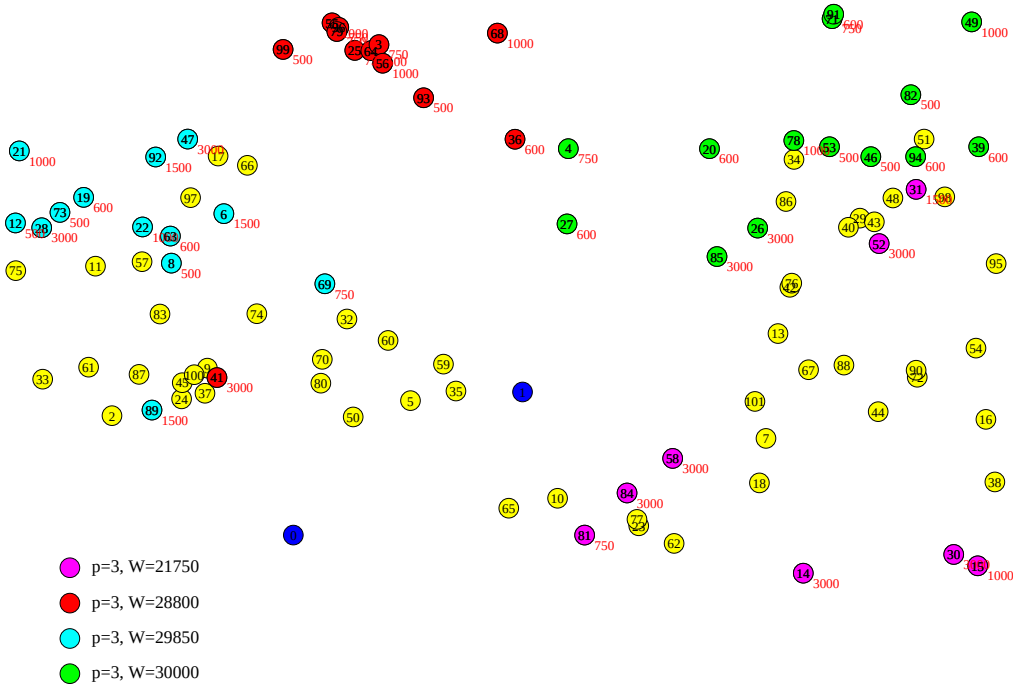



$s = 6$

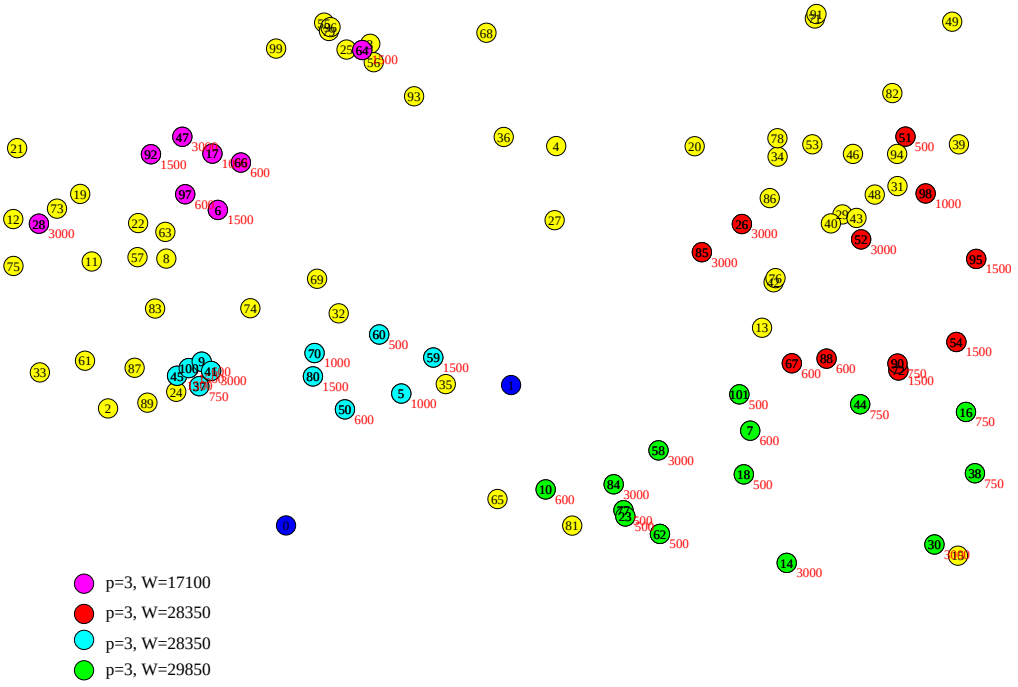
FIG. 7.3 – Solution de Jonzac obtenue par l'heuristique d'arrondi avec la variante C1S1+Br


p=3, W=18750
p=3, W=9000
p=3, W=30000
p=3, W=30000

$s = 1$


p=3, W=21750
p=3, W=28800
p=3, W=29850
p=3, W=30000

$s = 2$




$s = 3$

FIG. 7.4 – Solution de 0.rural obtenue par l'heuristique d'arrondi avec la variante C1S1+Br sur l'espace restreint $P = \{1, 2, 3\}$

Les figures 7.5 et 7.6 illustrent les solutions obtenues par la politique PF sur les instances Jonzac et 1.mixte. Les clients appartiennent à un seul cluster, nous avons donc mis tous les clusters sur la même figure. Dans la légende, nous précisons la date de départ de chacun des clusters. Sur ces deux figures, les clusters obtenus sont plus ou moins homogènes. Dans les solutions obtenues précédemment, la longueur du cycle de régénération $H$ était typiquement égale à 6 (voir 3 dans le cas de la restriction $P = \{1, 2, 3\}$). Dans le cas de la politique de la partition fixe, les périodicités 1, 4, 5 apparaissent dans la solution. Le cycle de régénération est plus long que pour les solutions moins restreintes, il est égal à $H = 12$ pour l'instance Jonzac et $H = 60$ pour l'instance 1.mixte.

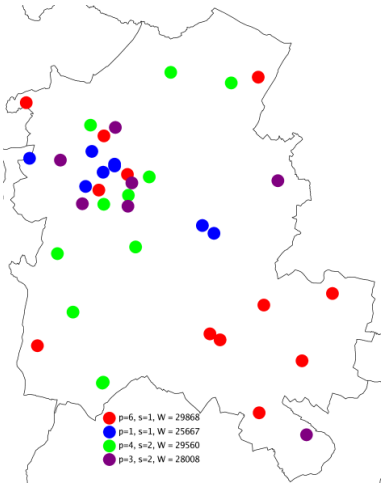



FIG. 7.5 – Solution de Jonzac obtenue par l'heuristique d'arrondi avec la variante C1S1+Br sur l'espace restreint de la politique de partition fixe

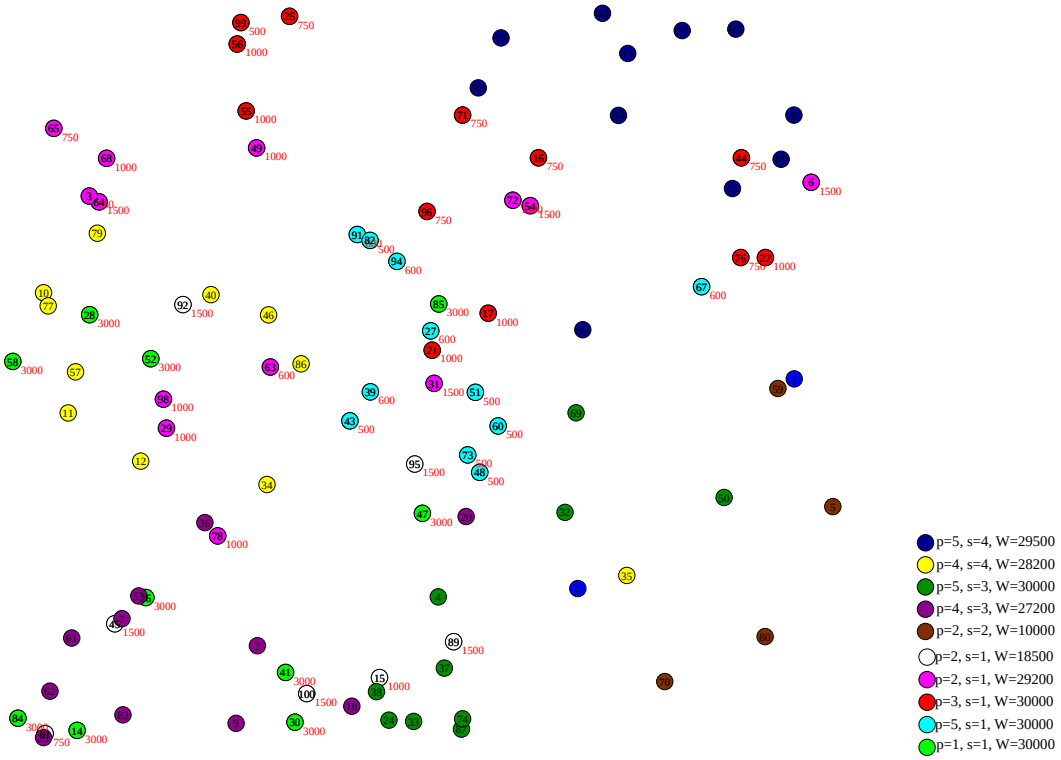



FIG. 7.6 – Solution de 1.mixte obtenue par l'heuristique d'arrondi avec la variante C1S1+Br sur l'espace restreint de la politique de partition fixe

## 7.4 Recherche locale

Notre heuristique suit les grandes lignes de l'algorithme de recherche locale présenté dans la table 2.4 de la section 2.4.4. Notre solution initiale est la solution obtenue par l'heuristique d'arrondi de la section 7.3 à une passe donnée. Une solution $\hat{\lambda}$ est caractérisée par l'ensemble des colonnes $Q(\hat{\lambda}) = \{q \in Q, \hat{\lambda}_q > 0\}$. Une solution voisine $\lambda'$ se caractérise par le fait que $Q(\hat{\lambda})$ et $Q(\lambda')$ ne diffèrent que par un nombre limité $K$ de colonnes, i.e. $Q^{ech} = (Q(\hat{\lambda}) \backslash Q(\lambda')) \cup (Q(\lambda') \backslash Q(\hat{\lambda}))$ et $0 < |Q^{ech}| \leq K$. Le voisinage de $\hat{\lambda}$ est noté $N(\hat{\lambda})$.

L'algorithme pour obtenir une solution voisine $\lambda' \in N(\hat{\lambda})$ est donné dans la table 7.13. Il s'agit de définir dans un premier temps un sous-ensemble de colonnes de $Q(\hat{\lambda})$ qui vont être supprimées, $Q^{suppr} \subset Q(\hat{\lambda})$ et $0 < |Q^{suppr}| \leq k$ ($k$ est fixé ou non). Après suppression dans la solution $\hat{\lambda}$ des colonnes de $Q^{suppr}$, on obtient une solution partielle dans laquelle des demandes de certains clients ne sont plus couvertes. Le maître agrégé associé à la solution partielle et le problème de pricing sont alors mis à jour. La solution partielle est complétée en appelant l'heuristique d'arrondi. Avant l'appel de l'heuristique d'arrondi, une première colonne est fixée. Comme à partir d'une même solution $\hat{\lambda}$, plusieurs solutions voisines $\lambda' \in N(\hat{\lambda})$ peuvent être construites, on contrôle la fixation de cette colonne pour ne pas obtenir des solutions identiques $\lambda' \in N(\hat{\lambda})$. Le contrôle se fait en enregistrant les premières colonnes fixées lors de la construction des solutions $\lambda' \in N(\hat{\lambda})$ dans un ensemble $Q^{prem}$. Par conséquent, la colonne fixée est choisie parmi l'ensemble de toutes les colonnes générées compatibles privé des colonnes de $Q^{suppr}$ et $Q^{prem}$. Le critère de fixation utilisé est le critère C1S1 de l'heuristique d'arrondi (défini dans la table 7.3). La solution obtenue est une solution voisine $\lambda'$.

Dans l'heuristique d'arrondi de la table 2.2, la qualité des solutions obtenues dépend du choix des colonnes fixées. Ce choix est basé sur un "score" attribué à chacune des colonnes de la solution $PL$. Par analogie, la qualité de la solution obtenue $\lambda'$ dépend des colonnes supprimées à l'étape 0 de l'algorithme de la table 7.13. Le choix des colonnes à supprimer, $q \in Q^{suppr}$, est dicté par un classement des colonnes de la solution $\hat{\lambda}$ par ordre croissant du "score" attribué à chacune d'elle. La détermination de l'ensemble $Q^{suppr}$ est spécifique à notre application et se fait en trois étapes. L'algorithme pour déterminer $Q^{suppr}$ est donné dans la table 7.14.

***Input :*** Soit $\hat{\lambda}$ une solution primale discrète caractérisée par $Q(\hat{\lambda})$.

***Etape 0 :*** Le sous-ensemble $Q^{suppr} \subset Q(\hat{\lambda})$ tel que $0 < |Q^{supprr}| \leq k$ est déterminé. On supprime de la solution $\hat{\lambda}$ les colonnes de $Q^{suppr}$, on obtient une solution partielle.

***Etape 1 :*** Le maître agrégé associé à la solution partielle et le problème de pricing sont mis à jour. Certains clients ont une partie de la demande qui n'est plus couverte. Pour ces clients, les contraintes (6.2) sont réintroduites en ajustant le membre de droite.

***Etape 2 :*** La colonne $q \notin Q^{suppr} \cup Q^{prem}$ minimisant le critère C1S1 est fixée à 1 ; puis $Q^{prem} = Q^{prem} \cup \{q\}$.

***Etape 3 :*** La solution partielle est complétée à l'aide de l'heuristique d'arrondi de la table 2.2 pris à partir de l'étape 2.

***Output :*** La solution voisine obtenue $\lambda'$ est retournée.

TAB. 7.13 – Algorithme pour obtenir une solution voisine $\lambda'$ de la solution initiale $\hat{\lambda}$

Dans une première étape, les très "mauvaises" colonnes de $\hat{\lambda}$ sont supprimées. Une colonne est jugée très mauvaise si sa charge, $charge^r$, est inférieure à $\frac{1}{3}$ de la capacité du véhicule. Si la solution $\hat{\lambda}$ ne comporte pas de très mauvaises colonnes, alors la première colonne à supprimer est choisie aléatoirement (toutes les colonnes ont le même score égale à 1).

Dans les deux étapes suivantes, les colonnes $r$ à supprimer sont choisies selon un score dépendant de leur charge, $charge^r$, leur distance par rapport aux colonnes de $Q^{suppr}$, $dist^r$, et le nombre de fois qu'elles ont déjà été supprimées durant les itérations précédentes, $nbSuppr^r$. On choisit des colonnes qui sont peu chargées, qui ont des clients proches de ceux des colonnes appartenant à $Q^{suppr}$ et on pénalise les colonnes fréquemment supprimées. La distance entre la colonne $r$ et les colonnes de $Q^{suppr}$ est égale à la somme des distances entre les clients de la colonne $r$ et tous les clients des colonnes $q \in Q^{suppr}$ telles que $p^r = p^q$ (des échanges sont possibles entre ces colonnes si elles ont même périodicité), cette somme est ensuite divisée par le nombre de clients dans la colonne $r$. En résumé, le score d'une colonne est : $\frac{dist^r}{W - charge^r} * (1 + \frac{nbSuppr^r}{2})$, on supprime les colonnes ayant un score minimum.

Plus particulièrement, dans la deuxième étape, on supprime les colonnes ayant les mêmes caractéristiques que les colonnes supprimées à la première étape, i.e. ayant les mêmes périodicités et dates de départ. Le but de ce choix est de permettre des échanges entre les clusters utilisés aux mêmes périodes. Les colonnes supprimées à la première étape vont libérer un véhicule à certaines périodes. Pour libérer complètement l'utilisation d'un véhicule, les colonnes supprimées à la troisième étape sont celles qui vont permettre cette libération. Le but est de permettre des échanges de périodes de collecte pour certains clients.

***Etape 1 :*** Détermination de l'ensemble $R_0$ composé des très mauvaises colonnes.
Soit la colonne $r_0$, si $charge^{r_0} < \frac{1}{3}W$, alors $R_0 = R_0 \cup \{r_0\}$. Si $R_0 = \emptyset$, alors une colonne $r_0$, choisie aléatoirement, est ajoutée à $R_0$.

***Etape 2 :*** Détermination de l'ensemble $R_1$ composé des colonnes ayant les mêmes caractéristiques que les colonnes de $R_0$.
Pour chaque paire $(p, s)$ telle qu'il existe $r_0 \in R_0 : (p, s) = (p^{r_0}, s^{r_0})$, on définit un ensemble $R_{ps}$ contenant les colonnes de périodicité $p$ et de date de départ $s$ candidates pour appartenir à l'ensemble $R_1$. Chaque colonne de la solution $\hat{\lambda}$ n'appartenant pas à $R_0$ est insérée dans l'ensemble $R_{ps}$ associé à la colonne (s'il existe) et son score est calculé. Pour chaque ensemble $R_{ps}$, on choisit la colonne $r_1$ de score minimum et $R_1 = R_1 \cup \{r_1\}$.

***Etape 3 :*** Détermination de l'ensemble $R_2$ composé des colonnes permettant de libérer un véhicule.
Pour chaque paire $(p, s)$ qui permet de libérer un véhicule (déjà en partie libéré par les colonnes de $R_0$), on définit un ensemble $R_{ps}$ contenant les colonnes de périodicité $p$ et de date de départ $s$ candidates pour appartenir à l'ensemble $R_2$. Chaque colonne de la solution $\hat{\lambda}$ n'appartenant pas à $R_0 \cup R_1$ est insérée dans l'ensemble $R_{ps}$ associé à la colonne (s'il existe), et son score est calculé. Pour chaque ensemble $R_{ps}$, on choisit la colonne $r_2$ de score minimum et $R_2 = R_2 \cup \{r_2\}$.

***Output :*** $Q^{suppr} = R_0 \cup R_1 \cup R_2$.

TAB. 7.14 – Algorithme pour déterminer l'ensemble des colonnes à supprimer, $Q^{suppr}$

Pour tester l'efficacité du critère de la table 7.14, nous le comparons avec un critère basique qui consiste à supprimer aléatoirement $k$ colonnes de la solution $\hat{\lambda}$, par exemple $k = Pmax = 6$. Une colonne $r$ n'est pas considérée comme une "bonne" colonne si son coût, $\frac{c^r}{p^r}$, est grand et sa charge, $charge^r$, petite. Par conséquent, la probabilité associée à une colonne $r$ est proportionnelle à $\frac{c^r}{p^r\ charge^r}$.

Le voisinage d'une solution, $N(\hat{\lambda})$, est de grande taille. Il est donc exploré heuristiquement. La recherche d'une solution voisine s'arrête lorsqu'une meilleure solution $\lambda^*$ est trouvée. Pour limiter les temps de calculs, nous tronquons la recherche d'une solution : nous testons au maximum NBI solutions voisines $\lambda' \in N(\hat{\lambda})$ (NBI = 10 dans nos tests). L'algorithme d'exploration du voisinage de la solution $\hat{\lambda}$ est dans la table 7.15.

***Input :*** Soit $\hat{\lambda}$ une solution primale discrète.

***Etape 1 :*** $\lambda^* = \hat{\lambda}$.

***Etape 2 :*** Pendant NBI itérations faire :

1. Construire une solution $\lambda' \in N(\hat{\lambda})$ (en utilisant l'algorithme de la table 7.13).
2. Si le coût de la solution $\lambda'$ est meilleure que le coût de la solution $\lambda^*$ alors $\lambda^* = \lambda'$, STOP.

***Output :*** La meilleure solution voisine trouvée $\lambda^*$ est retournée.

TAB. 7.15 – Algorithme d'exploration du voisinage de la solution $\hat{\lambda}$

Les résultats de l'exploration du voisinage de la première solution trouvée par l'heuristique d'arrondi sont dans le tableau 7.16. La colonne "FirstPB" indique le coût de cette première solution et "bestPB" le coût de la meilleure solution voisine trouvée. Dans la colonne "E/R" est indiqué le nombre d'échecs (E.), i.e. lorsque (bestPB=FirstPB), et le nombre d'itérations nécessaires à l'étape 2 de l'algorithme de la table 7.15 pour trouver une meilleure solution (R.), (lorsqu'il y a échec, R=NBI=10). Sur notre base de problèmes tests, l'amélioration relative du coût de la première solution est de 2.11% en moyenne dans le cas d'un choix aléatoire de $k = 6 = Pmax$ colonnes à supprimer ; et de 7.43% en moyenne dans le cas d'un choix déterministe. L'amélioration moyenne la plus importante est obtenue avec le choix déterministe. De plus, il n'y a eu aucun échec. C'est donc ce critère qui est utilisé pour la construction de $Q^{suppr}$

dans l'algorithme de recherche locale.

| | Bornes | | | | Compteurs | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Nom | bestPB | FirstPB | am | E/R | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| Voisinage stochastique, $k = 6 = Pmax$ | | | | | | | | | | | |
| moy AL | 449.133 | 456.34 | 1.60 | E3/R6.6 | 11122 | 11006 | 77 | 192 | 58 | 309 | 6m33s (393) |
| moy S60 | 298.714 | 301.547 | 4.25 | E2/R7 | 6738 | 6599 | 55 | 107 | 17 | 160 | 3m11s (191) |
| GN | 602.916 | 602.916 | 0 | E1/R10 | 19324 | 19160 | 570 | 907 | 234 | 1053 | 25m47s (1547) |
| Pb1 | 737.087 | 739.574 | 0.34 | E0/R2 | 15268 | 15201 | 974 | 1318 | 157 | 1468 | 27m17s (1637) |
| Voisinage déterministe | | | | | | | | | | | |
| moy AL | 416.652 | 451.775 | 8.43 | E0/R1.9 | 8541 | 8481 | 68 | 165 | 58 | 200 | 4m47s (287) |
| moy S60 | 314.142 | 340.136 | 8.27 | E0/R2.4 | 3950 | 3858 | 42 | 77 | 14 | 89 | 1m58s (118) |
| GN | 600.872 | 602.916 | 0.34 | E0/R6 | 14073 | 13907 | 539 | 834 | 225 | 794 | 21m28s (1288) |
| Pb1 | 737.627 | 739.574 | 0.26 | E0/R1 | 14817 | 14756 | 973 | 1310 | 158 | 1434 | 26m43s (1603) |

TAB. 7.16 – Comparaison de la solution obtenue par exploration du voisinage d'une solution

L'algorithme de recherche locale LS est celui de la table 2.4. L'étape (1a) de LS fait appel à l'algorithme de l'exploration du voisinage de la table 7.15. Pour diversifier la recherche locale, l'algorithme LS est appelé sur différentes solutions $\hat{\lambda}$ de départ, ces solutions sont obtenues par les différentes passes de l'heuristique d'arrondi (i.e. l'heuristique RH est appelée plusieurs fois sur la même solution $PL$). Dans nos tests, il y a 3 passes. Le recherche locale s'arrête soit lorsque "la" meilleure solution voisine est moins bonne que la solution de départ, soit lorsque le nombre maximum d'itérations dans l'heuristique d'arrondi est atteint (i.e lorsque le nombre maximum de colonnes qu'on peut fixer est atteint). Ce paramètre vaut 500 dans nos tests.

Les résultats de la recherche locale sont dans le tableau 7.17. La déviation à l'optimum est de 11.2158% en moyenne sur les 17 instances, elle est inférieure à la déviation à l'optimum obtenue par la meilleure variante de l'heuristique d'arrondi. Cette heuristique a permis d'améliorer légèrement les meilleures solutions pour 3 instances :

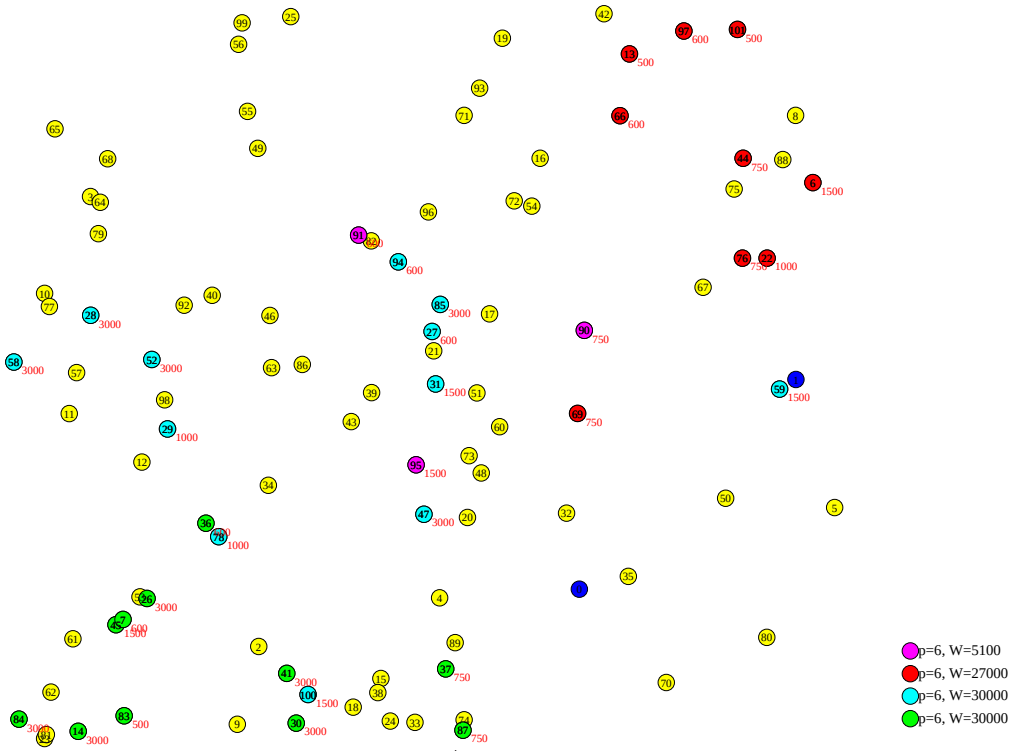
- 1.mixte : bestPB = 398.617 ; dev = 7.53101 ; V = 4
- 1.rural : bestPB = 413.507 ; dev = 8.4843 ; V = 4
- Ouest : bestPB = 188.318 ; dev = 6.25319 ; V = 2

L'amélioration pour ces 3 instances est respectivement de 0.77%, 0.06% et 0.14%. Le temps de résolution de l'algorithme avec l'heuristique de recherche locale est plus long que le temps de résolution de l'algorithme avec l'heuristique d'arrondi appelée à la racine et après le branchement sur $Vmoy$ pour les petites instances, en revanche pour les deux grandes instances, il est plus court.

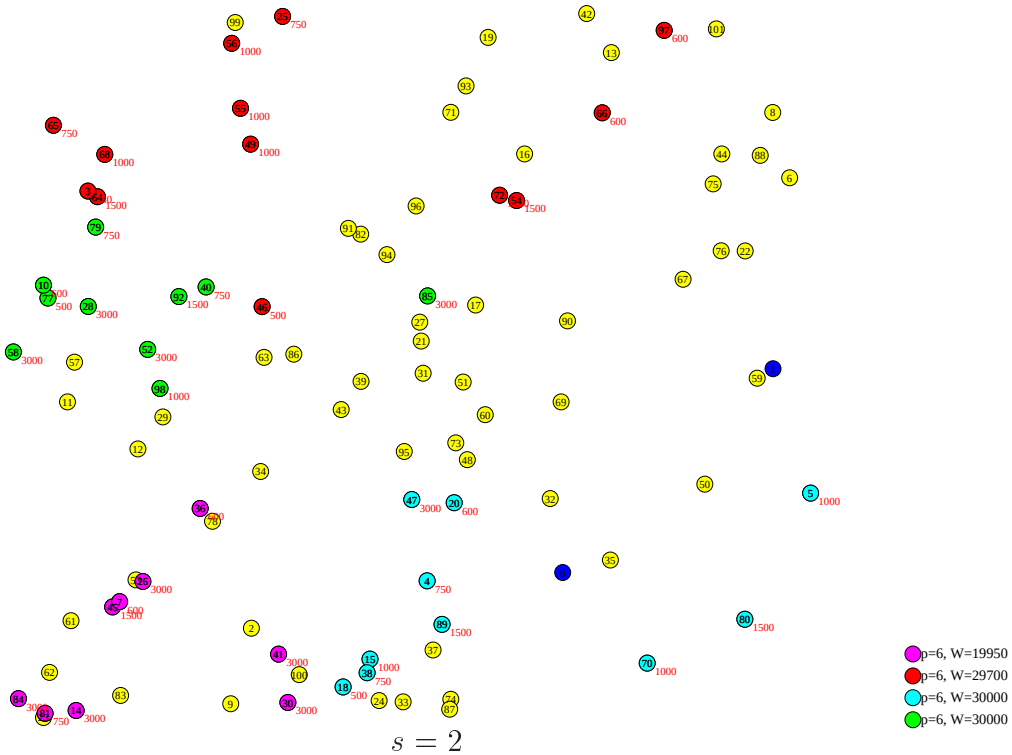
Cette heuristique est plus adaptée pour la résolution de grandes instances pour lesquelles la résolution $PL$ d'un nœud prend plus de temps que l'exploration du voisinage de plusieurs solutions. La figure 7.7 illustre la solution obtenue de 1.mixte. Cette solution est 6 périodique.

| | Bornes | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Nom | bestPB | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| Heuristique d'arrondi appelée à la racine et après le branchement sur $Vmoy$ | | | | | | | | | | |
| moy AL | 409.399 | 12.29 | 4 | 23481 | 23269 | 287 | 582 | 298 | 990 | 18m22s (1102) |
| moy S60 | **285.056** | 11.75 | **3** | 8300 | 8107 | 140 | 222 | 49 | 288 | 6m10s (370) |
| GN | **589.582** | **5.574** | **6** | 32261 | 31871 | 2511 | 3241 | 761 | 3914 | 1h22m2s (4922) |
| Pb1 | 731.48 | 10.85 | 7 | 32239 | 31908 | 3324 | 4083 | 652 | 4925 | 1h30m16s (5416) |
| Heuristique de recherche locale | | | | | | | | | | |
| moy AL | **407.892** | **11.8858** | **4** | 33882 | 32765 | 286 | 599 | 342 | 2675 | 46m4s1 (2764) |
| moy S60 | 285.18 | **11.7374** | **3** | 13908 | 12765 | 188 | 296 | 70 | 776 | 13m27s (807) |
| GN | 590.48 | 5.73506 | 6 | 27356 | 26345 | 1303 | 1784 | 384 | 3151 | 1h0m50s (3650) |
| Pb1 | **708.594** | **7.38898** | **7** | 25857 | 25207 | 1550 | 2045 | 297 | 3120 | 54m50s (3290) |

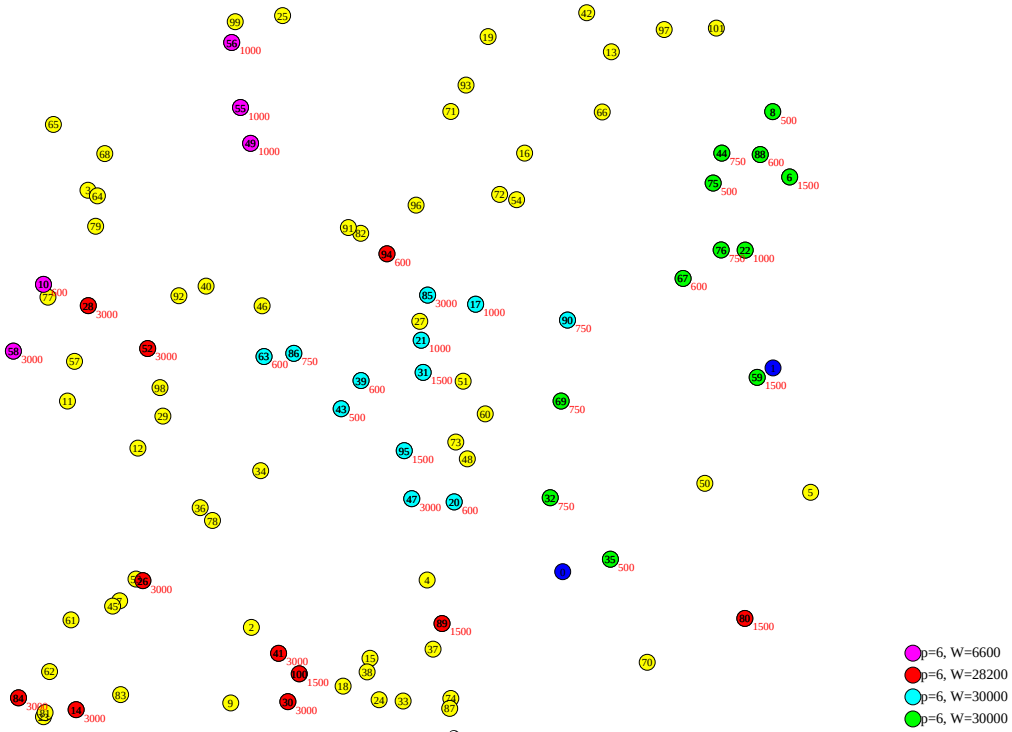
TAB. 7.17 – Comparaison de la solution primale discrète obtenue par l'heuristique de recherche locale


p=6, W=5100
p=6, W=27000
p=6, W=30000
p=6, W=30000

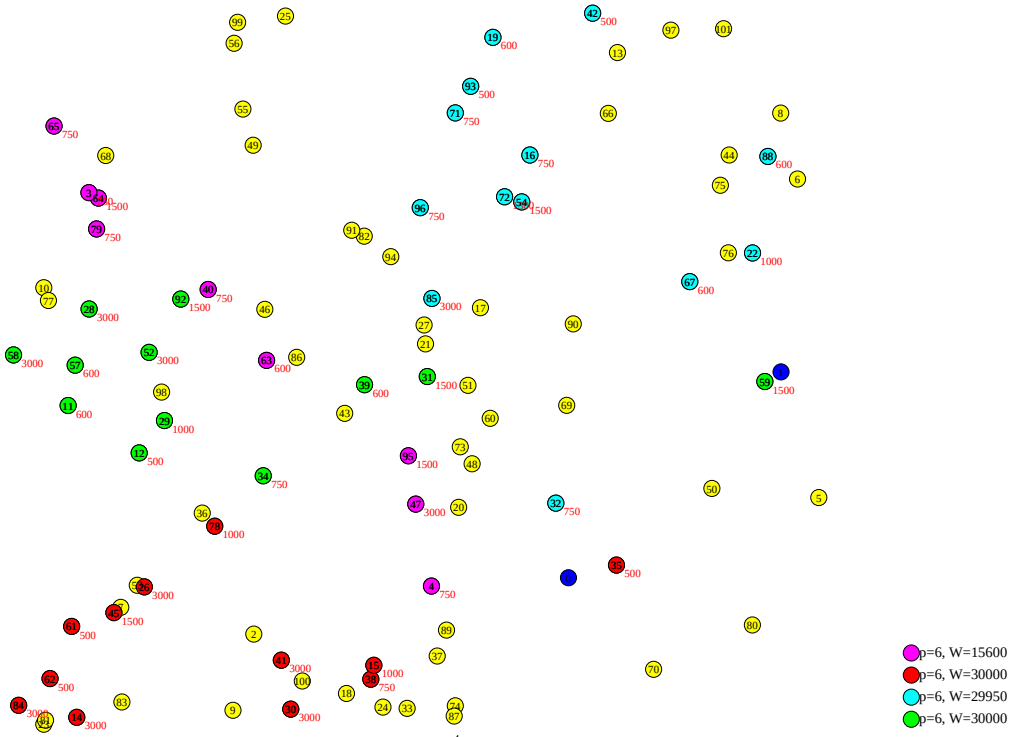
$s = 1$


p=6, W=19950
p=6, W=29700
p=6, W=30000
p=6, W=30000

$s = 2$


p=6, W=6600
p=6, W=28200
p=6, W=30000
p=6, W=30000

$s = 3$


p=6, W=15600
p=6, W=30000
p=6, W=29950
p=6, W=30000

$s = 4$

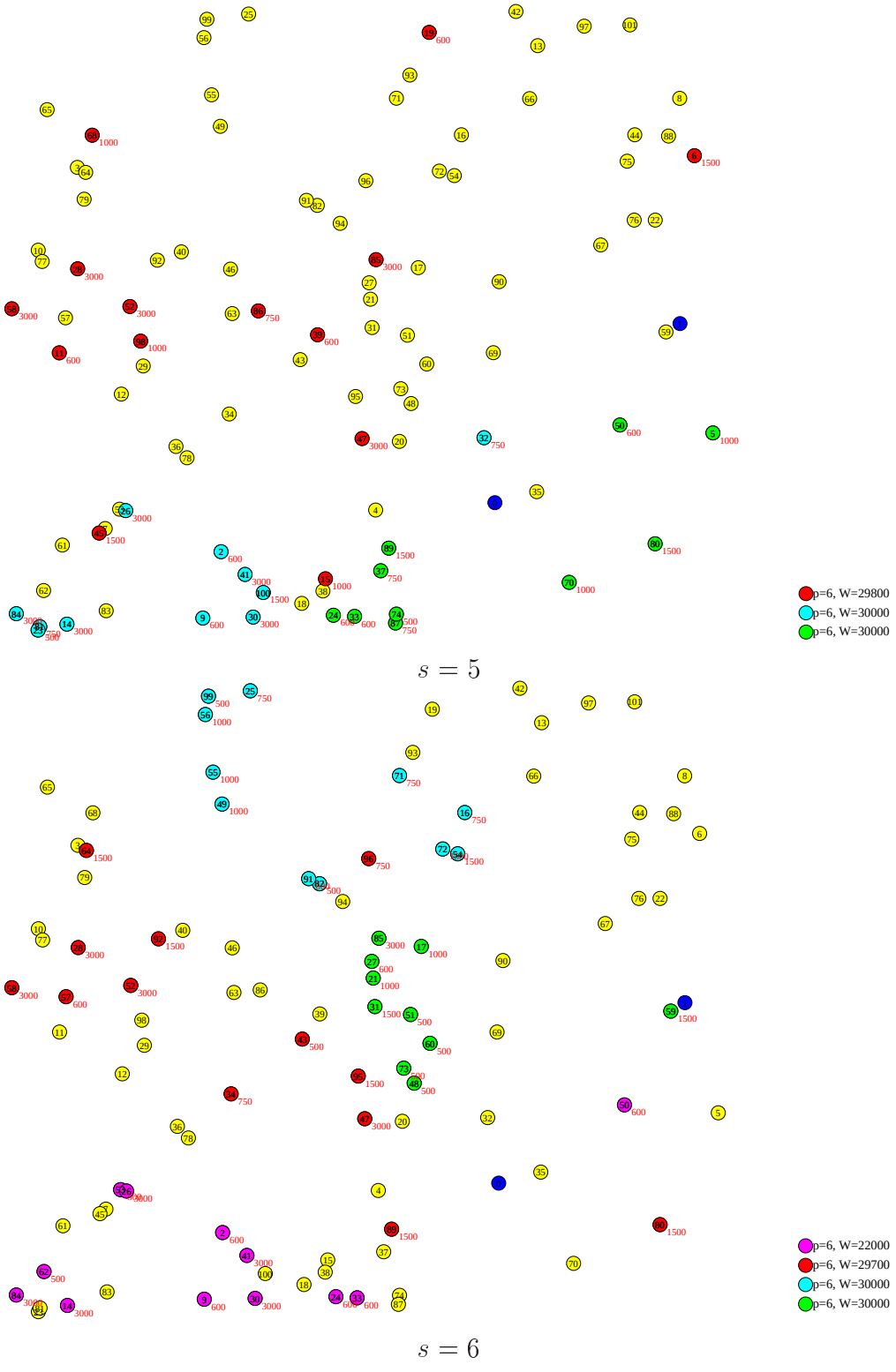



FIG. 7.7 – Solution de 1.mixte obtenue par l'heuristique de recherche locale

## 7.5 Conclusion

L'heuristique gloutonne permet d'obtenir une première solution rapidement et de connaître une borne supérieure sur le nombre de véhicules nécessaires. Cependant les solutions obtenues ne sont pas de bonne qualité du point de vue des routes. Les meilleures solutions sont obtenues par l'heuristique d'arrondi. La variante la plus performante appelle cette heuristique sur la solution $PL$ optimale (sans les coupes) de la racine et du nœud 2 (après le branchement sur $Vmoy$).

Le temps de résolution en nombre entier du maître restreint à un sous ensemble de colonnes ne permet pas d'obtenir des solutions. La symétrie dans les choix des dates de départ en est principalement responsable. Une méthode hybride qui fixe un certain nombre de colonnes à l'aide de l'heuristique d'arrondi avant de résoudre le maître restreint en nombre entier pourrait peut être permettre d'obtenir de bonnes solutions.

Les solutions obtenues par l'heuristique d'arrondi peuvent être améliorées par une heuristique de recherche locale. La recherche locale s'appuie d'ailleurs sur l'heuristique d'arrondi pour compléter la solution partielle obtenue après avoir supprimer quelques colonnes. Pour que les temps de calculs restent raisonnables, l'heuristique de recherche locale est appelée seulement sur les solutions obtenues à la racine.

Parmi les heuristiques décrites dans la section 2.4, nous n'avons pas testé l'heuristique gloutonne de la table 2.1 qui itérativement génère une colonne puis la fixe. Le principal avantage de cette heuristique est son temps de calcul : à chaque étape de fixation, une seule colonne est générée. Pour l'heuristique d'arrondi, avant chaque étape de fixation, beaucoup de colonnes peuvent être générées (le nombre de colonnes générées sera plus important avant les premières fixations qu'avant les dernières), cela prend plus de temps. En combinant cette heuristique gloutonne avec les heuristiques de ce chapitre, des gains en temps de calcul pourraient être fait (par exemple en faisant quelques itérations de l'heuristique gloutonne), mais peut être au détriment de la qualité des solutions obtenues.

# 8

# Résolution du problème industriel

Ce chapitre est consacré aux résultats obtenus sur les données réelles de l'application industrielle présentées sur la figure 1.1. Les bornes primales obtenues à l'aide des heuristiques d'arrondi et de recherche locale du chapitre 7 sont comparées à la borne duale obtenue en ajoutant à la formulation $[FormAgr]$ les coupes et contraintes de branchement vues au chapitre 6.

Les périodicités des routes sont restreintes à prendre leur valeur dans l'ensemble $P = \{1, 2, 3, 4, 5, 6\}$. Deux politiques restrictives sont essayées. Dans la première, la restriction se fait sur l'ensemble des périodicités $P = \{1, 2, 3\}$. La deuxième restriction est la politique de partition fixe dans laquelle l'ensemble des clients est partitionné en sous ensembles disjoints ; chaque sous ensemble est servi séparément et indépendamment des autres et lorsqu'un client d'un sous ensemble est livré, tous les autres le sont. Ainsi chaque route prend en charge toute la production des clients qu'elle visite (i.e. chaque client est collecté par une seule route, on impose "$\ell = p$").

Dans une première section 8.1, les caractéristiques plus détaillées du problème sont données ainsi que la méthode utilisée pour construire les routes . La deuxième section 8.2 est consacrée aux résultats en terme de la borne duale et la troisième 8.3 en terme des bornes primales.

## 8.1 Caractéristiques du problème

Le problème industriel de la figure 1.1 concerne la collecte de points de recyclage de type verre dans la région des Charentes. Il comprend 260 bornes ($N = 260$). La répartition des bornes est plutôt de type mixte. Pour la planification tactique, l'unité de temps est la semaine (dans le cas d'une région urbaine, la période de base pourrait être le jour). La périodicité de collecte des bornes (notée P.) varie de toutes les semaines à toutes les 6 semaines, voir même toutes les 12 semaines et exceptionnellement 14. Sur les 260 bornes, 169 (soit 65% des bornes) ont une périodicité de collecte inférieure ou égale à 6 semaines. Il peut donc être intéressant de travailler avec seulement les périodicités $P = \{1, \cdots, 6\}$ plutôt que $P = \{1, \cdots, 6, 7, \cdots, 12\}$, d'autant plus que dans le premier cas le cycle de régénération de longueur maximum est de $T = 60$ semaines alors que dans le deuxième cas il est de $T = 27720$ semaines (plus de 530 ans) ! Ensuite, du point de vue pratique, l'utilisateur peut plus facilement travailler avec des routes ayant une périodicité de 6 plutôt que de 12 semaines.

La capacité d'un véhicule est de $30m^3$, toutes les bornes ont la même capacité de stockage, $3.5m^3$. Dans la pratique, le nombre moyen de bornes qu'un véhicule peut collecter est égal à 10 (les bornes sont collectées en moyenne à 85% de leur capacité). En prenant comme périodicité maximale $Pmax = 6$ dans la planification tactique, les bornes ayant une périodicité de collecte strictement supérieure à 6 seront toujours vidées avant d'être pleines, leur capacité de stockage se voit donc réduite à $Pmax\ d_i < 3.5m^3$.

Une particularité des données est l'expression du temps en décimal : $1j = 1$, $1h = \frac{1}{24} = 0.041666$ et $5min = \frac{5}{24*60} = 0.003472$ (temps de collecte d'une borne). En prenant le paramètre de l'objectif $\alpha$ égal à 5, le coût associé au nombre de véhicules nécessaires (constaté à $\frac{2}{3}$ environ du coût de la solution dans nos tests) reste plus important que le coût des routes mais ce dernier n'est pas négligeable. Comme les données réelles ne sont pas symétriques, pour le planning tactique, le temps entre deux points est calculé comme la moyenne des temps aller-retour entre ces deux points.

Dans la pratique, il y a deux flux à collecter : le verre et le papier. Deux véhicules sont disponibles chaque jour, soit un véhicule par flux. Cependant un véhicule peut faire plusieurs rotations, c'est à dire qu'il part du garage, collecte

des bornes, va au vidage puis repart collecter des bornes, etc ... Il se trouve que chaque jour ouvrable (du lundi au vendredi), chaque véhicule peut faire jusqu'à deux rotations. Nous considérons donc que nous avons 10 véhicules par semaine.

La solution actuellement pratiquée par l'industriel est construite en se basant principalement sur les fréquences de collecte. Il observe que les tournées avec une périodicité paire (i.e. toutes les 2, 4, ... périodes) sont identiques ou quasi identiques. Sur la semaine, 10 rotations pour la collecte de verre sont effectuées en moyenne (mais il peut y avoir des semaines creuses ou fortes, cela dépend des fréquences de collecte, des saisons).

## 8.2 Borne duale

La borne duale permettant d'évaluer la qualité de nos solutions primales, est celle obtenue avec les coupes (6.12) et le branchement sur la variable $Vmoy$ (6.15). Pour le calcul de cette borne, nous avons utilisé la variante de l'algorithme la plus performante, c'est à dire que les coupes sont ajoutées une fois que $Vmoy$ est à sa valeur entière (voir le chapitre 6).

Sur l'instance industrielle, la borne duale sur le cycle $T = 60$ est égale à 838.17. A la racine $Vmoy$ vaut 8.63008, après le branchement (et les coupes), $Vmoy$ vaut 9 (le coût fixe associé à l'utilisation des véhicules est donc de $9 * 60 = 540$ et représente $\frac{2}{3}$ environ de la borne duale). Le nœud 1 est irréalisable. En tout, 612 coupes sur 13630 ont été ajoutées, soit 4.49% des coupes. La borne duale à la racine vaut 816.81, les coupes et le branchement ont permis une amélioration de 2.62%. La borne duale après le branchement et avant l'ajout des coupes est égale à 832.54, les coupes ont amélioré cette borne de 0.67%.

Il a fallu 8h34m pour obtenir cette borne duale. La plus grande partie du temps est passée dans la procédure des plans coupants qui a nécessité 5h21m. Les temps passés au nœud racine, à prouver l'irréalisabilité du nœud 1 et à la réoptimisation du nœud 2 sont respectivement de 23m, 2h31m et 18m.

Il y a eu 25018 appels au problème de pricing et 24985 colonnes générées. 54m sont passées dans la procédure de pricing (dont 48m dans l'oracle). La résolution du maître restreint a nécessité 34m.

## 8.3 Bornes primales

Pour connaître rapidement une première solution, nous avons appelé l'heuristique gloutonne présentée en section 7.1. La version avec l'estimation des valeurs $\pi_i^{init}$ fournit une solution ayant un coût de 1191.62 et 10 véhicules sont utilisés. La déviation à l'optimum de cette solution est de 42.17%. Une meilleure solution est obtenue en utilisant les valeurs optimales de la solution duale $\pi_i^*$, son coût est de 1176.71 et 9 véhicules sont utilisés. La déviation à l'optimum égale à 40.39% reste importante mais la connaissance de cette solution nous permet de savoir qu'il existe des solutions où le nombre de véhicules utilisés est égal à $Vmoy^* = 9$. Le coût fixe associé à l'utilisation des véhicules est donc de $9 * 60 = 540$. Plus de la moitié du coût de cette solution est dû aux coûts des routes.

Ensuite nous avons utilisé la variante la plus performante de l'heuristique d'arrondi vue en section 7.3, c'est à dire la variante où le critère de fixation est C1S1 (défini dans la table 7.3) et où l'heuristique est appelée sur la solution $PL$ optimale (sans les coupes) de la racine et du nœud 2. Nous avons essayé d'améliorer la solution obtenue à la racine avec l'heuristique de recherche locale vue en section 7.4. Le récapitulatif des bornes primales obtenues est dans le tableau 8.1. La colonne "Heur" spécifie l'algorithme utilisé pour trouver la borne primale. Pour toutes ces solutions, le coût égal au nombre de véhicules utilisés représente environ les $\frac{2}{3}$ du coût total.

| | Bornes | | Compteurs | | | Temps | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Heur | bestPB | dev | V | SpSol | Col | MCKP | SpSol | RM | RH | Total |
| $P = \{1,2,3,4,5,6\}$ | | | | | | | | | | |
| RH | 915.735 | 9.25414 | 9 | 56046 | 55460 | 9670 | 11588 | 2461 | 13643 | 4h29m5s (16145) |
| LS | 912.981 | 8.9255 | 9 | 49161 | 48278 | 6540 | 7905 | 1455 | 11817 | 3h39m56s (13196) |
| $P = \{1,2,3\}$ | | | | | | | | | | |
| RH | 907.965 | 8.3270 | 9 | 39342 | 39119 | 1111 | 1520 | 2860 | 2277 | 1h48m57s (6537) |
| LS | **906.773** | **8.18486** | **9** | 41635 | 40826 | 1036 | 1426 | 2496 | 7487 | 2h53m9s (10389) |
| PF | | | | | | | | | | |
| RH | 1039.83 | 24.059 | 11 | 13059 | 12751 | 6175 | 6655 | 471 | 3432 | 2h21m57s (8517) |

TAB. 8.1 – Bornes Primales obtenues sur l'instance industrielle

Lorsque l'ensemble des périodicités est $P = \{1,2,3,4,5,6\}$, nous obtenons une solution primale de coût 915.735 et dans laquelle 9 véhicules sont utilisés. La déviation à l'optimum est de 9.25%. Cette solution a été obtenue à la racine, les branchements ne l'ont pas améliorée. Il a fallu 4h29m pour obtenir

cette solution (dont 2h08m passées au nœud racine et 2h21m au nœud 2). 3h47m sont consacrées à l'heuristique d'arrondi. La recherche locale a permis d'améliorer la solution de 0.30%, le meilleur coût est de 912.981 et la déviation à l'optimum est de 8.92%. Toujours 9 véhicules sont utilisés. Il a fallu 3h40m pour obtenir cette solution. 3h17m sont passées dans l'heuristique.

Les détails des routes de la solution, i.e. leur périodicité $p$, leur date de départ $s$, leurs caractéristiques (le nombre de bornes $b$, la charge $W$ en $m^3$, la distance $D$ en $km$ et le temps de parcours $C$ en $j.h.m.s$) sont dans le tableau 8.2. Une colonne de ce tableau représente une semaine ; une ligne représente l'utilisation d'un véhicule. La dernière ligne est la caractéristique de la semaine, i.e. la somme des caractéristiques des routes.

Toutes les routes ont une périodicité égale à 6, la solution est donc 6−périodique et tous les plannings individuels sont 6−périodiques. On peut visualiser ces routes sur les figures de 8.1 à 8.6 (une figure par date de départ). Chaque route est utilisée une seule fois. Les clusters sont dans l'ensemble assez homogènes, ils ont une forme un peu allongée due à notre structure de coût qui favorise la collecte de bornes sur le chemin allant au garage-vidage. Nous remarquons que, sur presque chaque figure, au moins un cluster est "éclaté" (C399829, C400269, C400701, C401132), i.e. il collecte deux bornes se trouvant aux extrémités de la région. La distance et le temps de parcours de ces routes est assez important. Sur les 54 routes, 50 routes ont une charge supérieure à $28000m^3$ et 2 ont une charge inférieure à $15000m^3$. Une route peut être chargée mais avoir une petite distance de collecte si le cluster est compact (C4011850). Les routes ne sont pas équilibrées, en revanche la charge d'une semaine l'est.

| p | Caract | $s=1$ | $s=2$ | $s=3$ | $s=4$ | $s=5$ | $s=6$ |
|---|---|---|---|---|---|---|---|
| 6 | Tour | C399691 | C400121 | C400547 | C400978 | C401412 | C401833 |
| | $b$ | 11 | 10 | 10 | 9 | 12 | 9 |
| | $W$ | 29990 | 29920 | 29908 | 29758 | 29944 | 29938 |
| | $D$ | 81.05 | 59.48 | 51.85 | 20.34 | 53.47 | 40.83 |
| | $C$ | 3.52.05 | 3.03.57 | 2.48.41 | 1.40.40 | 3.01.56 | 2.21.39 |
| 6 | Tour | C399709 | C400139 | C400569 | C400998 | C401429 | C401850 |
| | $b$ | 10 | 10 | 14 | 12 | 9 | 9 |
| | $W$ | 29889 | 29952 | 29999 | 30000 | 30000 | 29750 |
| | $D$ | 69.52 | 53.82 | 67.46 | 89.60 | 57.53 | 18.56 |
| | $C$ | 3.24.02 | 2.52.38 | 3.39.54 | 4.14.11 | 2.55.02 | 1.37.06 |
| 6 | Tour | C399726 | C400156 | C400587 | C401015 | C401447 | C401868 |
| | $b$ | 9 | 9 | 10 | 9 | 10 | 10 |
| | $W$ | 30753 | 29914 | 29926 | 29918 | 29959 | 29993 |
| | $D$ | 60.57 | 28.08 | 79.43 | 57.39 | 45.75 | 73.60 |
| | $C$ | 3.01.07 | 1.56.08 | 3.43.50 | 2.54.46 | 2.36.29 | 3.32.11 |
| 6 | Tour | C399744 | C400173 | C400608 | C401036 | C401466 | C401886 |
| | $b$ | 10 | 9 | 13 | 13 | 11 | 10 |
| | $W$ | 29898 | 29868 | 29989 | 29985 | 29400 | 29923 |
| | $D$ | 53.83 | 85.58 | 74.99 | 98.59 | 67.34 | 88.95 |
| | $C$ | 2.52.39 | 3.51.02 | 3.49.58 | 4.39.10 | 3.24.40 | 4.02.11 |
| 6 | Tour | C399762 | C400194 | C400626 | C401056 | C401485 | C401906 |
| | $b$ | 10 | 13 | 10 | 12 | 11 | 10 |
| | $W$ | 29951 | 29989 | 29754 | 30000 | 29954 | 29913 |
| | $D$ | 86.23 | 87.58 | 81.35 | 127.51 | 103.67 | 145.71 |
| | $C$ | 3.57.39 | 4.15.08 | 3.47.41 | 5.30.00 | 4.37.19 | 6.36.23 |
| 6 | Tour | C399779 | C400214 | C400645 | C401076 | C401507 | C401926 |
| | $b$ | 9 | 12 | 11 | 12 | 14 | 12 |
| | $W$ | 29754 | 29952 | 29971 | 29900 | 29990 | 29495 |
| | $D$ | 56.09 | 106.71 | 95.36 | 66.88 | 118.45 | 57.30 |
| | $C$ | 2.52.10 | 4.40.24 | 4.20.42 | 3.28.45 | 5.21.53 | 3.09.35 |
| 6 | Tour | C399802 | C400233 | C400663 | C401095 | C401525 | C401945 |
| | $b$ | 15 | 11 | 10 | 11 | 10 | 11 |
| | $W$ | 29870 | 29878 | 29752 | 29873 | 29928 | 29984 |
| | $D$ | 107.54 | 104.00 | 63.68 | 109.9 | 103.50 | 95.17 |
| | $C$ | 5.05.45 | 4.37.59 | 3.12.21 | 4.49.47 | 4.31.58 | 4.20.20 |
| 6 | Tour | C399829 | C400253 | C400681 | C401113 | C401543 | C401955 |
| | $b$ | 19 | 12 | 10 | 10 | 10 | 2 |
| | $W$ | 29928 | 29868 | 29750 | 28876 | 28176 | 3500 |
| | $D$ | 187.66 | 109.08 | 90.19 | 99.30 | 91.22 | 50.56 |
| | $C$ | 8.05.17 | 4.53.20 | 4.05.22 | 4.23.35 | 4.07.25 | 2.06.07 |
| 6 | Tour | C399843 | C400269 | C400701 | C401132 | C401556 | C401973 |
| | $b$ | 6 | 8 | 12 | 11 | 5 | 10 |
| | $W$ | 22168 | 20917 | 28762 | 28701 | 13125 | 29109 |
| | $D$ | 83.51 | 167.72 | 181.65 | 158.80 | 86.39 | 104.99 |
| | $C$ | 3.32.0 | 6.30.25 | 7.18.17 | 6.27.35 | 3.32.46 | 4.34.58 |
| | $b$ | 99 | 94 | 100 | 99 | 92 | 85 |
| | $W$ | 261301 | 260258 | 267811 | 267011 | 250476 | 241605 |
| | $D$ | 786.00 | 802.10 | 785.97 | 829.32 | 727.31 | 675.68 |
| | $C$ | 1.12.41.50 | 1.12.49.02 | 1.12.46.46 | 1.14.08.28 | 1.10.09.28 | 1.07.51.13 |

TAB. 8.2 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3, 4, 5, 6\}$. Caractéristiques des routes utilisées.

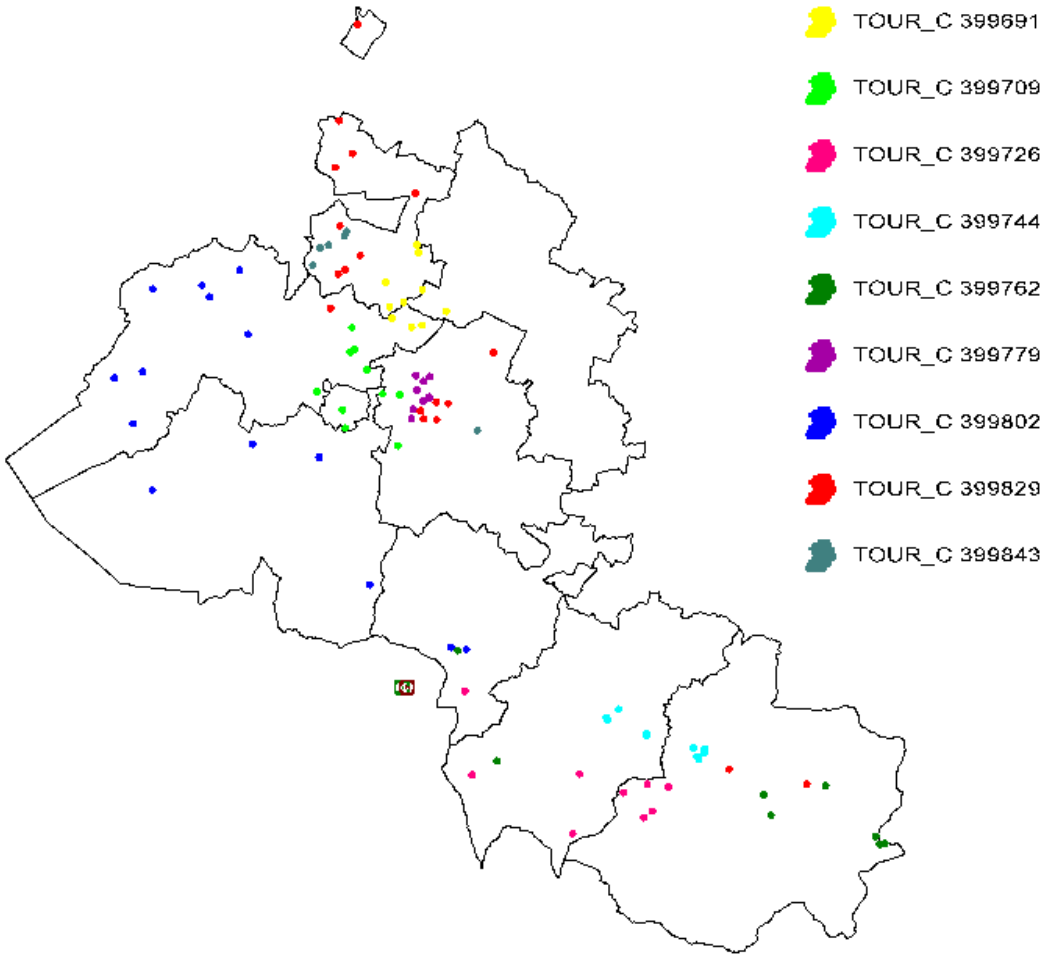



FIG. 8.1 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3, 4, 5, 6\}$. Visualisation des routes démarrant à la date $s = 1$

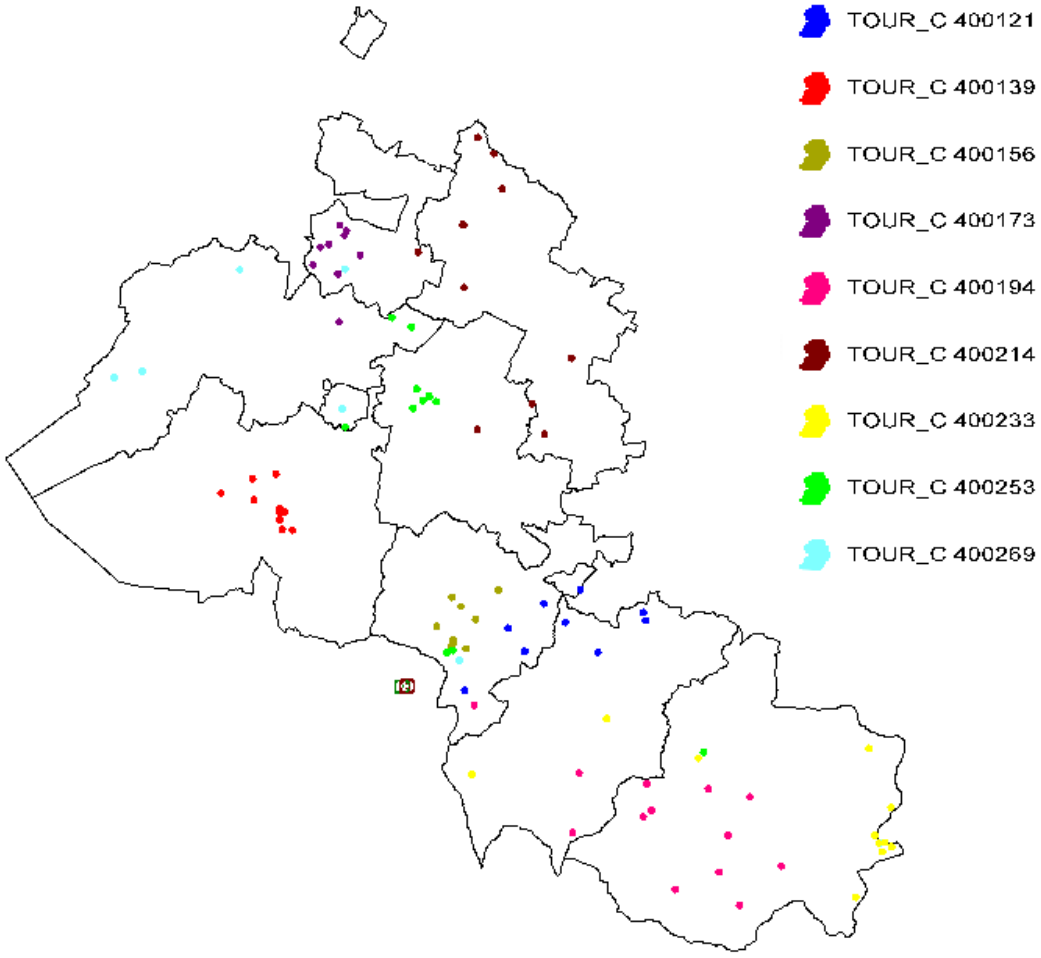



FIG. 8.2 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3, 4, 5, 6\}$. Visualisation des routes démarrant à la date $s = 2$

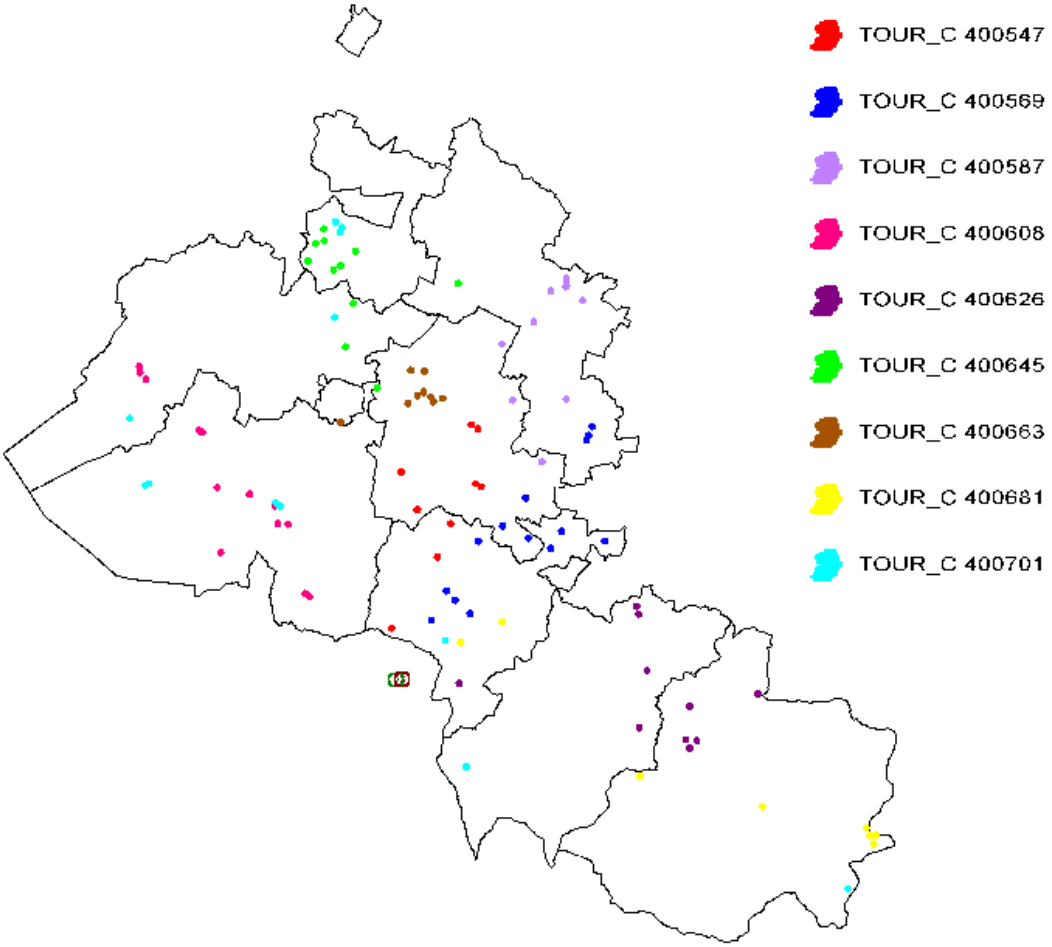



FIG. 8.3 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3, 4, 5, 6\}$. Visualisation des routes démarrant à la date $s = 3$

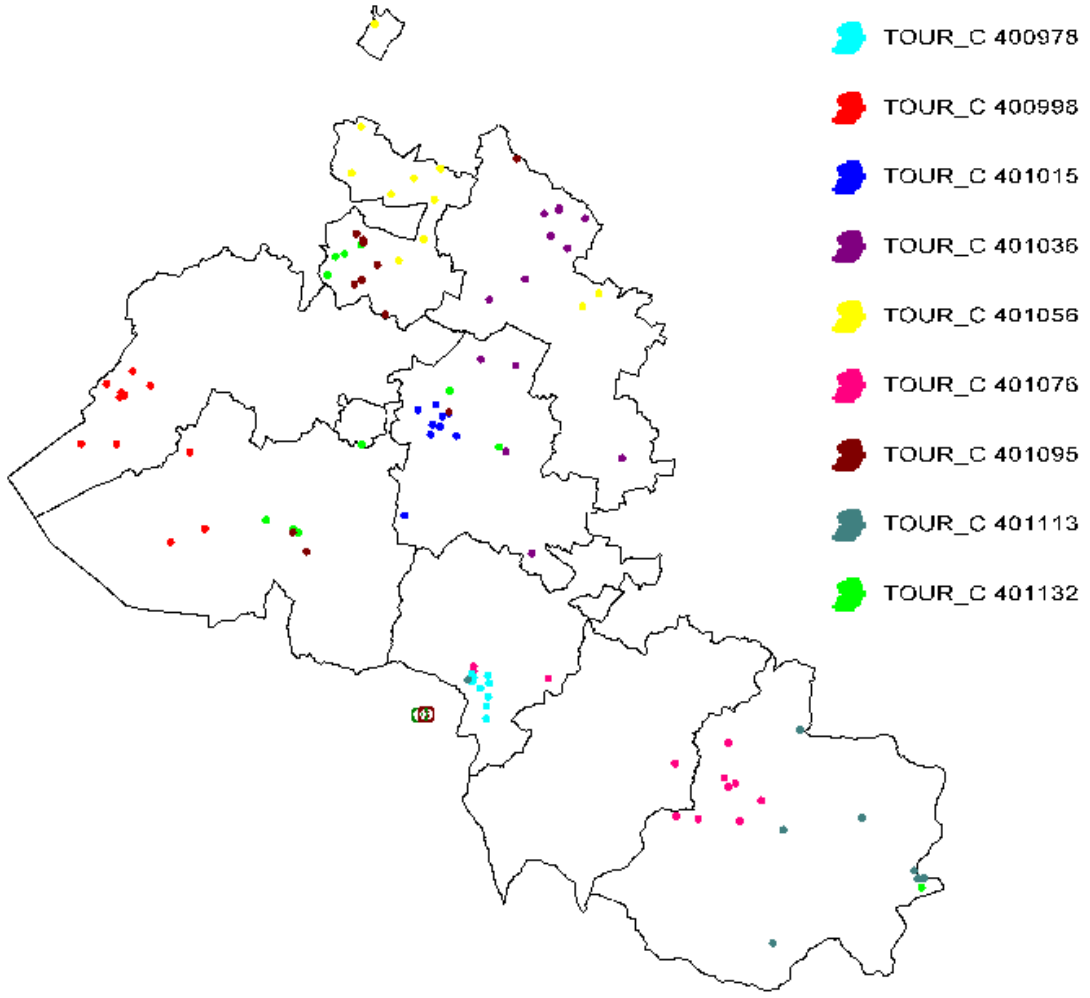



FIG. 8.4 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3, 4, 5, 6\}$. Visualisation des routes démarrant à la date $s = 4$

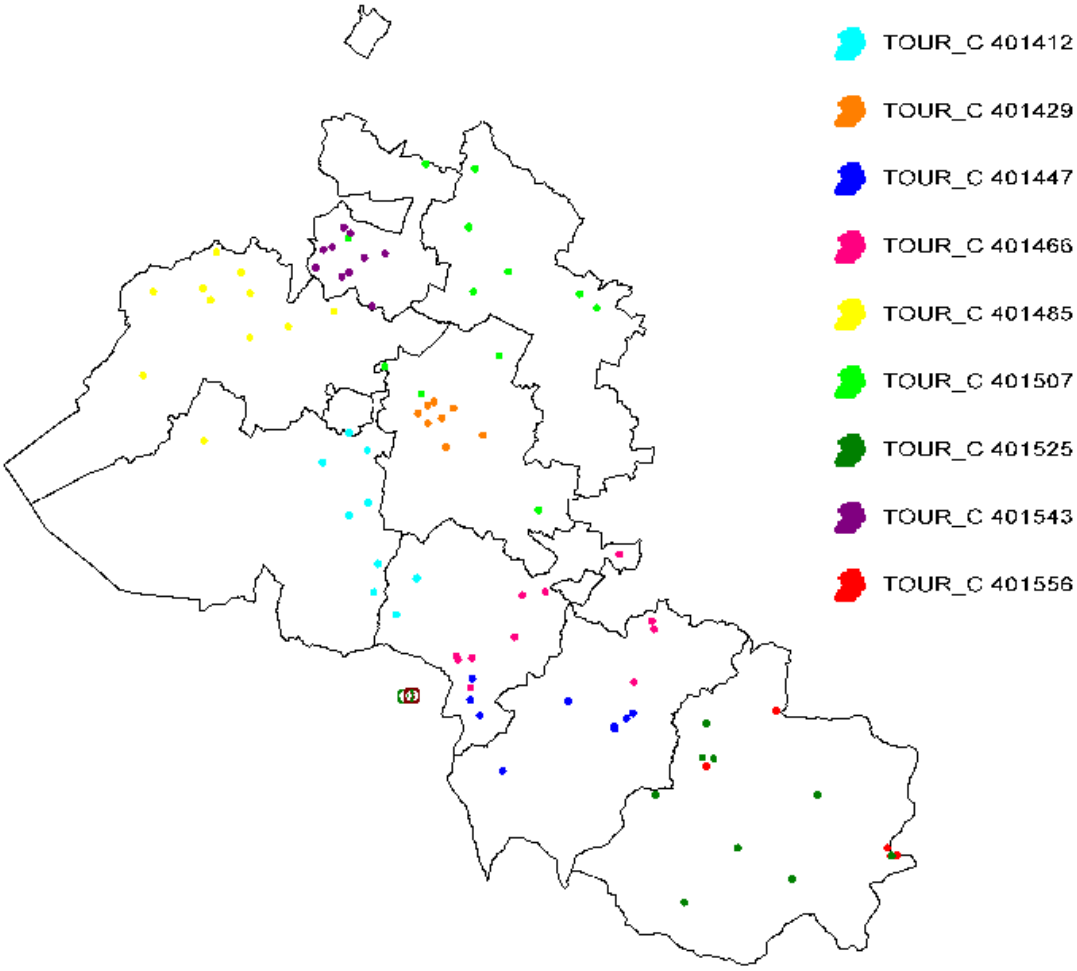



FIG. 8.5 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3, 4, 5, 6\}$. Visualisation des routes démarrant à la date $s = 5$

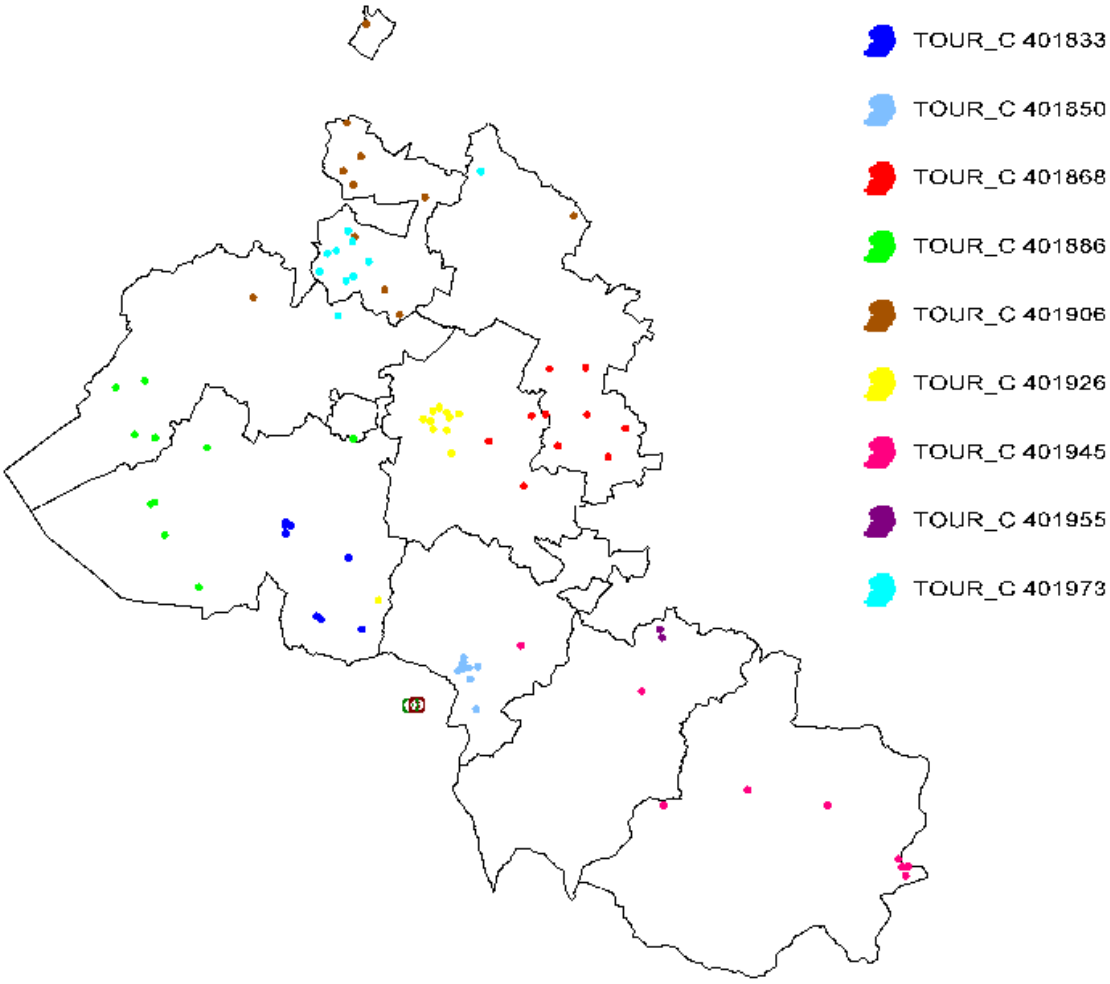



FIG. 8.6 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3, 4, 5, 6\}$. Visualisation des routes démarrant à la date $s = 6$

Lorsque l'ensemble des périodicités est $P = \{1, 2, 3\}$, nous obtenons une solution primale de coût 907.965 et dans laquelle 9 véhicules sont utilisés. La déviation à l'optimum est de 8.33%. Cette solution a été obtenue à la racine, les branchements ne l'ont pas améliorée. Cette solution a été obtenue au bout de 1h49m (le temps est réparti équitablement entre le nœud racine et le nœud 2). 38m sont passées dans l'heuristique d'arrondi. Cette solution est améliorée avec l'heuristique de recherche locale. L'amélioration est de 0.13%, son coût est de 906.773 et la déviation à l'optimum est de 8.18%. Toujours 9 véhicules sont utilisés. Il a fallu 2h53m pour obtenir cette solution. 2h05m sont consacrées à l'heuristique.

Les détails des routes de la solution sont dans le tableau 8.3. Toutes les routes ont une périodicité égale à 3, la solution est donc 3−périodique et tous les plannings individuels sont 3−périodiques. On peut visualiser ces routes sur les figures de 8.7 à 8.9 (une figure par date de départ). Chaque route est utilisée une seule fois. Les clusters sont plus homogènes que dans la solution où $P = \{1, 2, 3, 4, 5, 6\}$. De plus, on remarque que les routes sont mieux équilibrées (par rapport aux routes de la solution où $P = \{1, 2, 3, 4, 5, 6\}$, le temps maximum de parcours est plus petit et la charge minimum d'une route est plus grande). En moyenne, dans la solution où $P = \{1, 2, 3\}$, plus de borne sont collectées chaque semaine que dans la solution où $P = \{1, 2, 3, 4, 5, 6\}$, mais la distance et les temps de parcours sont moins importants.

| p | Caract | $s = 1$ | $s = 2$ | $s = 3$ |
|---|---|---|---|---|
| 3 | Tour | C396338 | C396775 | C397218 |
| | $b$ | 15 | 19 | 17 |
| | $W$ | 29841 | 29951 | 29931 |
| | $D$ | 82.82 | 62.81 | 108.33 |
| | $C$ | 4.15.37 | 3.55.36 | 5.16.38 |
| 3 | Tour | C396356 | C396800 | C397237 |
| | $b$ | 10 | 17 | 18 |
| | $W$ | 29942 | 29974 | 29928 |
| | $D$ | 64.00 | 87.73 | 24.43 |
| | $C$ | 3.12.59 | 4.35.26 | 1.58.50 |
| 3 | Tour | C396382 | C396823 | C397265 |
| | $b$ | 18 | 15 | 20 |
| | $W$ | 30000 | 29892 | 29923 |
| | $D$ | 78.01 | 127.45 | 97.60 |
| | $C$ | 4.21.00 | 5.44.52 | 5.10.10 |
| 3 | Tour | C396400 | C396844 | C397285 |
| | $b$ | 10 | 13 | 12 |
| | $W$ | 29927 | 29960 | 29927 |
| | $D$ | 47.43 | 65.02 | 65.35 |
| | $C$ | 2.39.51 | 3.30.00 | 3.25.40 |
| 3 | Tour | C396419 | C396864 | C397303 |
| | $b$ | 11 | 12 | 11 |
| | $W$ | 29795 | 29929 | 29933 |
| | $D$ | 89.48 | 95.31 | 54.17 |
| | $C$ | 4.08.57 | 4.25.36 | 2.58.19 |
| 3 | Tour | C396438 | C396880 | C397323 |
| | $b$ | 11 | 8 | 11 |
| | $W$ | 29404 | 25202 | 29024 |
| | $D$ | 97.82 | 82.86 | 102.00 |
| | $C$ | 4.25.37 | 3.40.41 | 4.33.59 |
| 3 | Tour | C396457 | C396895 | C397347 |
| | $b$ | 11 | 7 | 16 |
| | $W$ | 29750 | 22750 | 29773 |
| | $D$ | 58.00 | 95.95 | 117.19 |
| | $C$ | 3.05.59 | 3.41.50 | 5.29.21 |
| 3 | Tour | C396475 | C396917 | C397363 |
| | $b$ | 10 | 14 | 8 |
| | $W$ | 29038 | 28087 | 29168 |
| | $D$ | 87.73 | 68.11 | 84.86 |
| | $C$ | 4.00.26 | 3.41.13 | 3.44.48 |
| 3 | Tour | C396488 | C396933 | C397382 |
| | $b$ | 5 | 8 | 11 |
| | $W$ | 17501 | 27126 | 28555 |
| | $D$ | 94.54 | 57.36 | 86.19 |
| | $C$ | 3.49.04 | 2.49.48 | 4.02.22 |
| | $b$ | 101 | 113 | 117 |
| | $W$ | 255198 | 252871 | 266162 |
| | $D$ | 699.83 | 732.58 | 740.11 |
| | $C$ | 1.09.59.29 | 1.12.04.58 | 1.12.40.02 |

TAB. 8.3 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3\}$. Caractéristiques des routes utilisées.

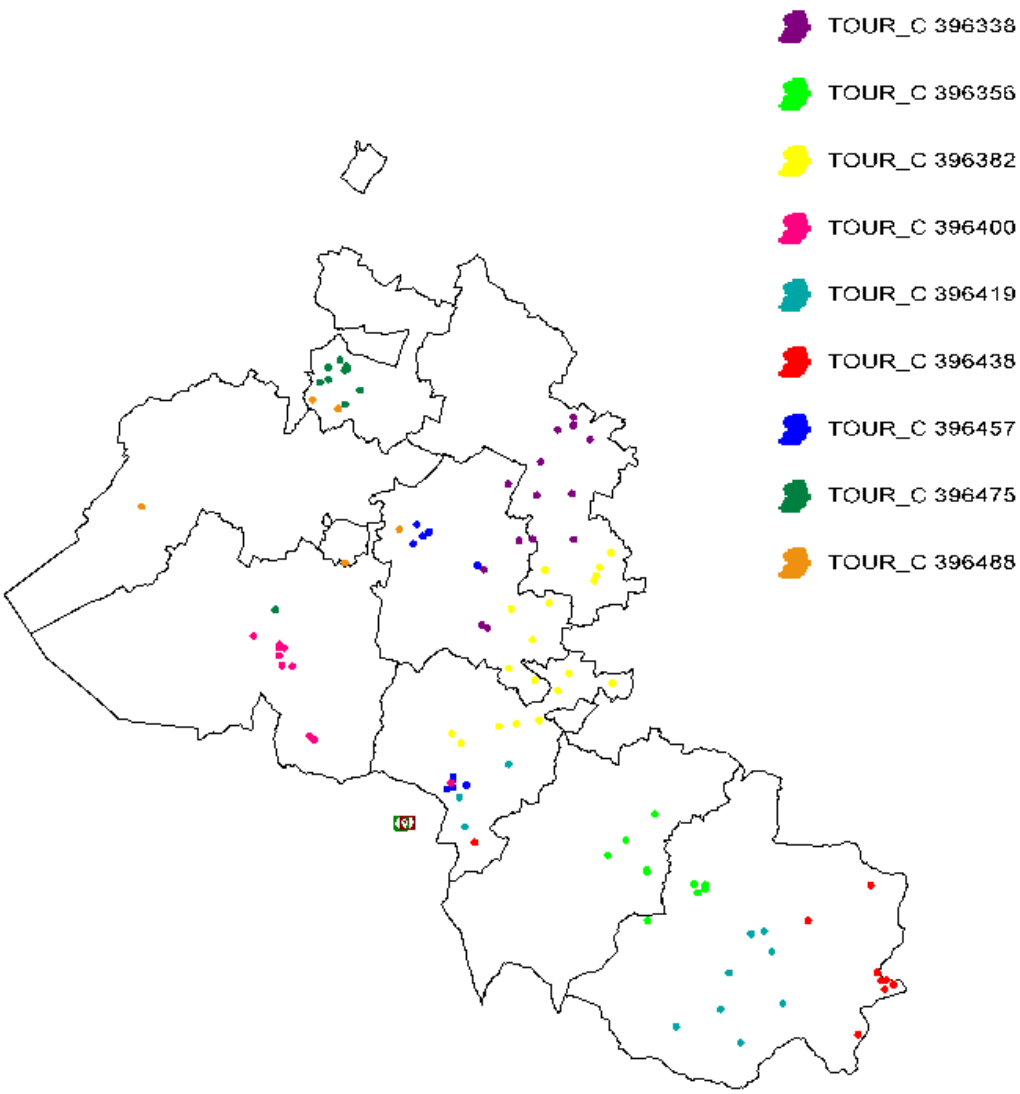



FIG. 8.7 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3\}$. Visualisation des routes démarrant à la date $s = 1$

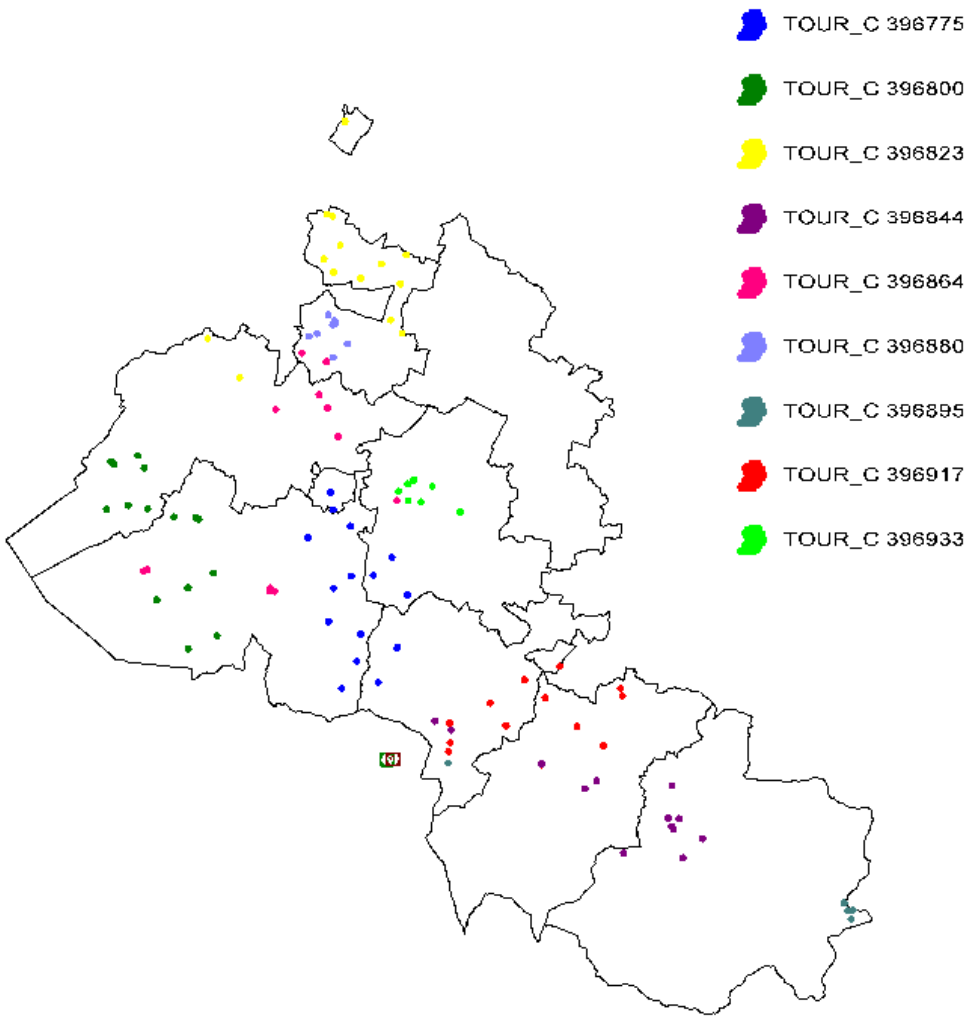



FIG. 8.8 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3\}$. Visualisation des routes démarrant à la date $s = 2$

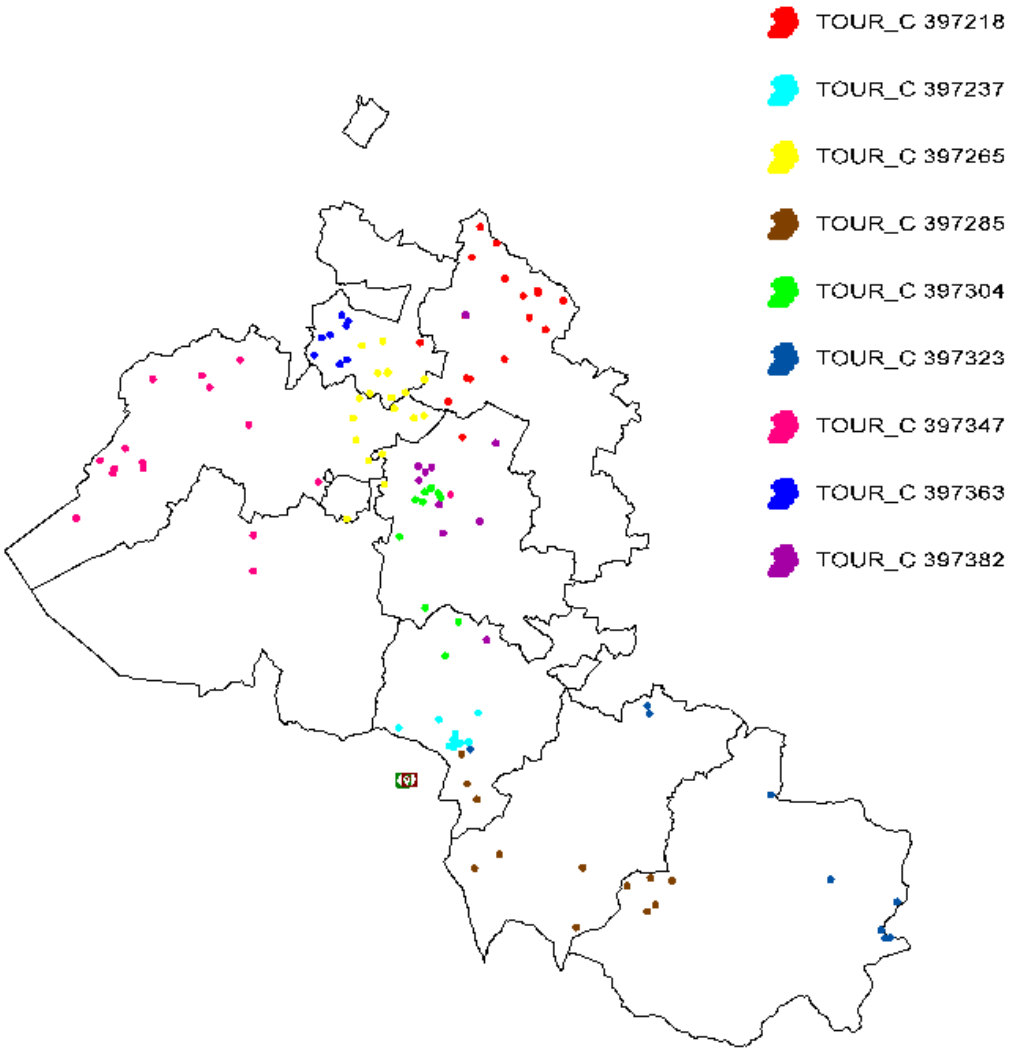



FIG. 8.9 – Solution obtenue avec l'heuristique de recherche locale sur l'instance industrielle lorsque $P = \{1, 2, 3\}$. Visualisation des routes démarrant à la date $s = 3$

Avec la politique de partition fixe PF, nous obtenons une solution primale de coût 1039.83 et dans laquelle 11 véhicules sont utilisés. La déviation à l'optimum est de 24.059%. Cette solution a été obtenue à la racine, les branchements ne l'ont pas améliorée. Il a fallu 2h22m pour obtenir cette solution (dont 1h passée au nœud racine, 53m au nœud 1 irréalisable et 28m au nœud 2). 57m sont passées dans l'heuristique d'arrondi.

Les détails des routes de la solution sont dans le tableau 8.4. Lorsqu'il y a un trait dans les cases associées aux charges, cela signifie qu'à cette date il y a répétition de la route "Tour". Nous n'avons pas donné la charge des six premières semaines, car les charges des 60 semaines sont différentes. Toutes les périodicités apparaissent, la solution est donc 60−périodique. Comme les clients appartiennent à une seule route, chaque planning individuel est $p-$périodique, $p \in P$. On peut visualiser ces routes sur les figures de 8.10 à 8.15 (une figure par périodicité).

Cette politique est souvent utilisée dans la littérature lorsque l'horizon est infini (voir la section 1.3). Les solutions obtenues par cette politique sont très pratique pour l'utilisateur car les clients sont partitionnés en clusters, et chaque cluster est servi indépendamment. De plus, lorsqu'un client du cluster est visité, tous le sont. Cependant, lorsque l'hypothèse d'"order up to level" est faite (i.e. à chaque visite, le borne est entièrement vidée), cette politique est restrictive. Dans la solution obtenue, les clusters sont assez homogènes, mais leur surface est plus importante que pour les clusters des solutions précédentes. Les routes sont aussi assez équilibrées et la charge des semaines aussi. La partition des clients s'est faite en fonction de leur position géographique et de leur compatibilité de fréquence de collecte. Le principal inconvénient de cette solution est le nombre de véhicules nécessaires.

| p | Caract | $s=1$ | $s=2$ | $s=3$ | $s=4$ | $s=5$ | $s=6$ |
|---|---|---|---|---|---|---|---|
| 1 | Tour | C406093 | C406093 | C406093 | C406093 | C406093 | C406093 |
| | $b$ | 10 | - | - | - | - | - |
| | $W$ | 29750 | - | - | - | - | - |
| | $D$ | 66.99 | - | - | - | - | - |
| | $C$ | 3.18.58 | - | - | - | - | - |
| 1 | Tour | C406110 | C406110 | C406110 | C406110 | C406110 | C406110 |
| | $b$ | 9 | - | - | - | - | - |
| | $W$ | 29868 | - | - | - | - | - |
| | $D$ | 84.53 | - | - | - | - | - |
| | $C$ | 3.49.03 | - | - | - | - | - |
| 1 | Tour | C406125 | C406125 | C406125 | C406125 | C406125 | C406125 |
| | $b$ | 7 | - | - | - | - | - |
| | $W$ | 24500 | - | - | - | - | - |
| | $D$ | 88.09 | - | - | - | - | - |
| | $C$ | 3.46.10 | - | - | - | - | - |
| 1 | Tour | C406137 | C406137 | C406137 | C406137 | C406137 | C406137 |
| | $b$ | 4 | - | - | - | - | - |
| | $W$ | 12250 | - | - | - | - | - |
| | $D$ | 95.71 | - | - | - | - | - |
| | $C$ | 3.46.24 | - | - | - | - | - |
| 2 | Tour | C406075 | C406487 | C406075 | C406487 | C406075 | C406487 |
| | $b$ | 9 | 8 | - | - | - | - |
| | $W$ | 24500 | 28000 | - | - | - | - |
| | $D$ | 127.35 | 53.85 | - | - | - | - |
| | $C$ | 5.14.40 | 2.42.41 | - | - | - | - |
| 3 | Tour | C405999 | C406500 | C406850 | C405999 | C406500 | C406850 |
| | $b$ | 9 | 8 | 9 | - | - | - |
| | $W$ | 29757 | 28008 | 27483 | - | - | - |
| | $D$ | 60.34 | 117.37 | 93.37 | - | - | - |
| | $C$ | 3.00.40 | 4.49.43 | 4.06.44 | - | - | - |
| 3 | Tour | C406040 | | C406836 | C406040 | | C406836 |
| | $b$ | 12 | | 10 | - | | - |
| | $W$ | 29577 | | 29319 | - | | - |
| | $D$ | 70.12 | | 93.09 | - | | - |
| | $C$ | 3.35.13 | | 4.11.10 | - | | - |
| 4 | Tour | C406058 | C406434 | C406818 | C407168 | C406058 | C406434 |
| | $b$ | 10 | 9 | 10 | 10 | - | - |
| | $W$ | 29092 | 29636 | 29556 | 29868 | - | - |
| | $D$ | 118.57 | 60.38 | 107.06 | 76.80 | - | - |
| | $C$ | 5.02.08 | 3.00.45 | 4.39.06 | 3.38.34 | - | - |
| 5 | Tour | | C406453 | | C407150 | | |
| | $b$ | | 11 | | 9 | | |
| | $W$ | | 29855 | | 29945 | | |
| | $D$ | | 96.44 | | 95.64 | | |
| | $C$ | | 4.22.52 | | 4.11.16 | | |
| 6 | Tour | C406020 | C406417 | C406781 | C407133 | C407449 | C407750 |
| | $b$ | 13 | 12 | 10 | 12 | 13 | 14 |
| | $W$ | 29946 | 29928 | 29784 | 29712 | 29892 | 29832 |
| | $D$ | 51.36 | 52.05 | 55.53 | 91.52 | 80.39 | 93.91 |
| | $C$ | 3.02.45 | 2.59.05 | 2.56.02 | 4.18.01 | 4.00.46 | 4.32.48 |
| 6 | Tour | | C406471 | C406800 | | C407468 | |
| | $b$ | | 10 | 11 | | 11 | |
| | $W$ | | 29544 | 29964 | | 29868 | |
| | $D$ | | 97.59 | 99.41 | | 106.50 | |
| | $C$ | | 4.20.11 | 4.28.48 | | 4.42.59 | |

TAB. 8.4 – Solution obtenue avec l'heuristique d'arrondi sur l'instance industrielle avec PF. Caractéristiques des routes utilisées.

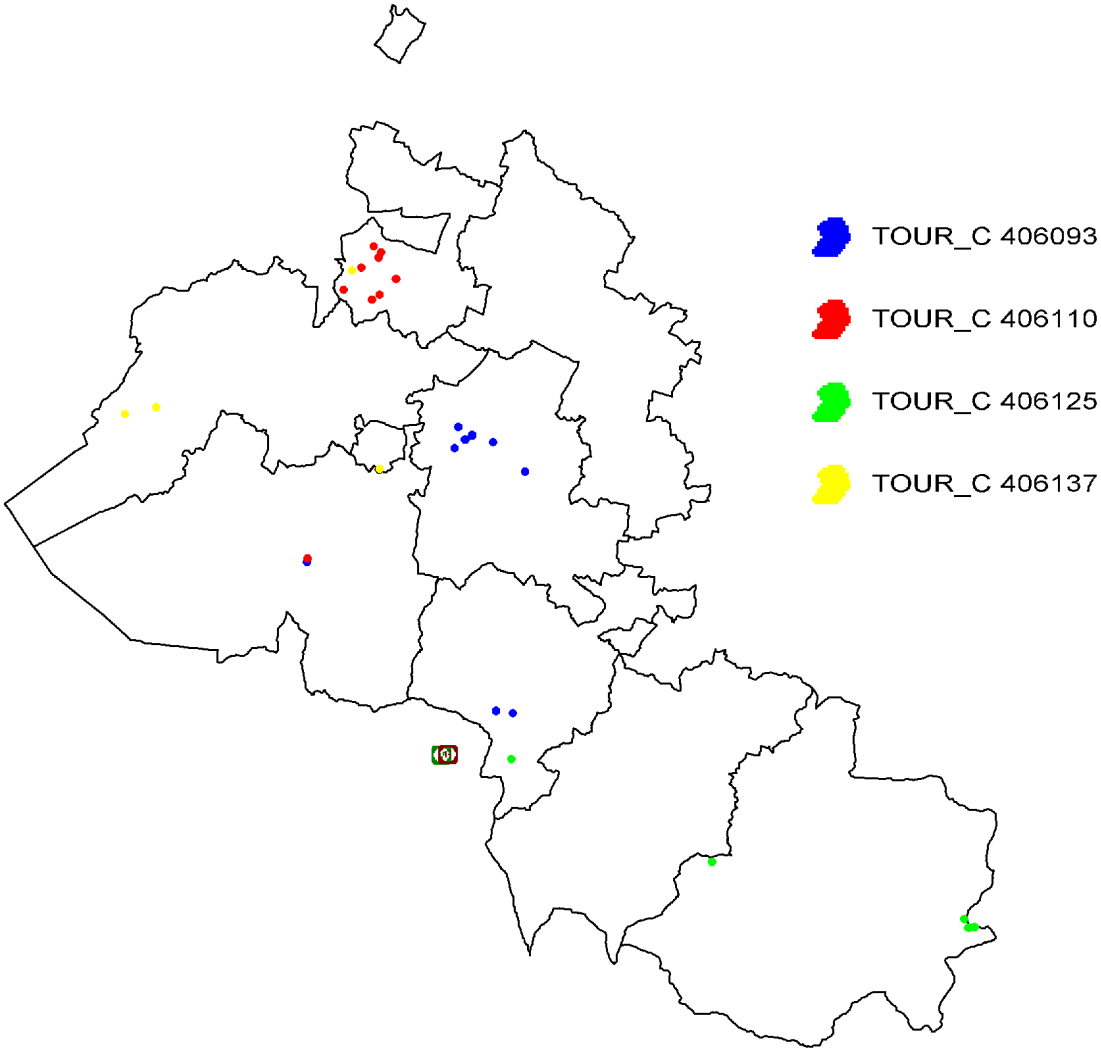



FIG. 8.10 – Solution obtenue avec l'heuristique d'arrondi sur l'instance industrielle avec PF. Visualisation des routes de périodicité $p = 1$

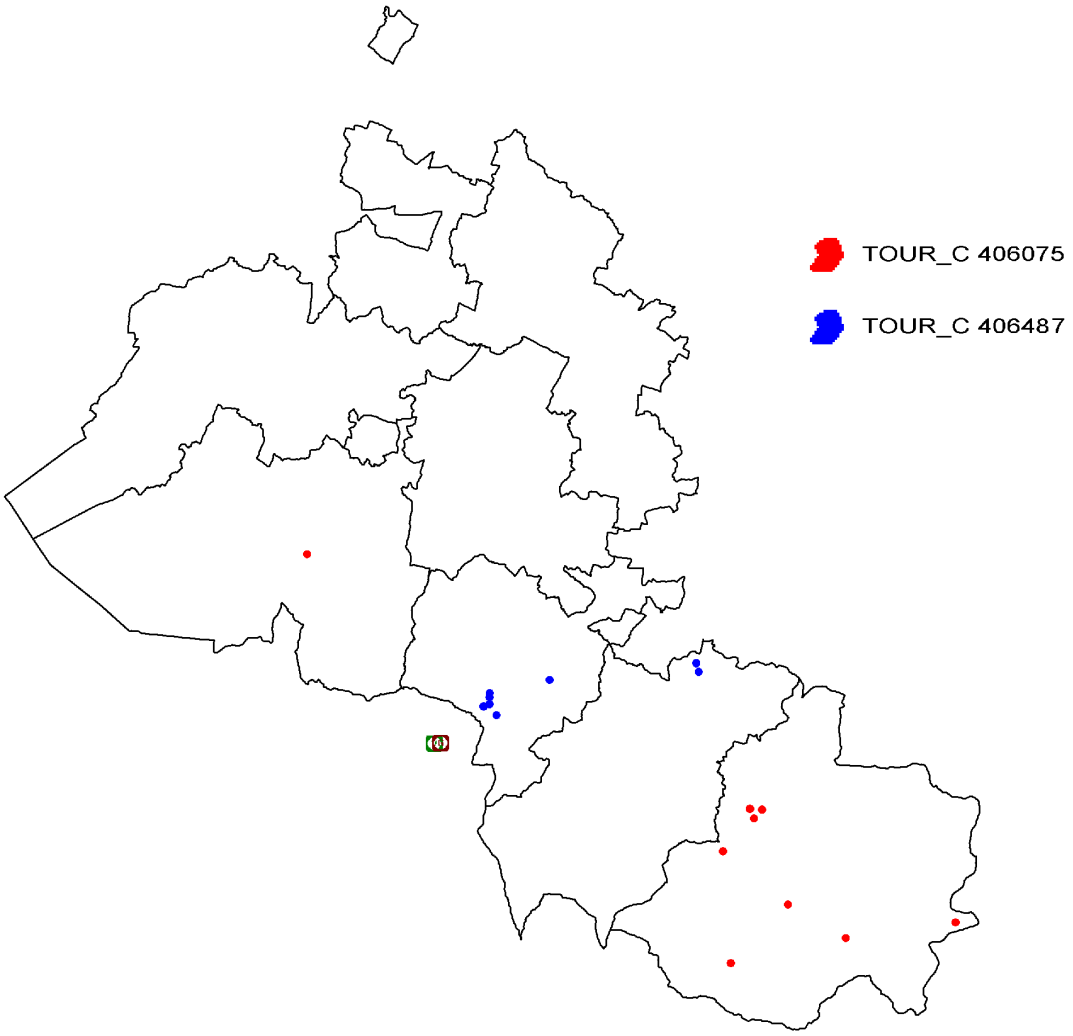



FIG. 8.11 – Solution obtenue avec l'heuristique d'arrondi sur l'instance industrielle avec PF. Visualisation des routes de périodicité $p = 2$

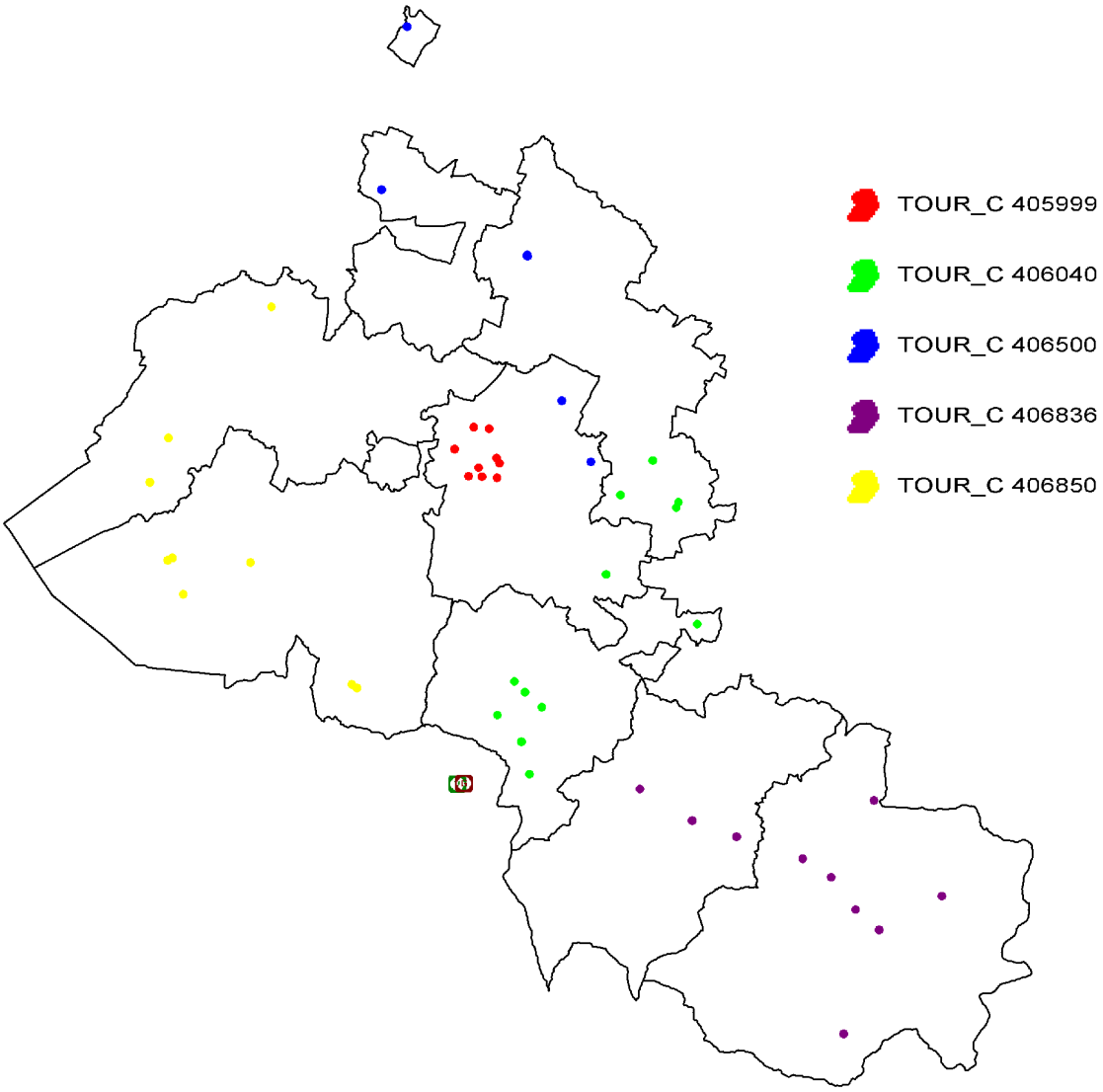



FIG. 8.12 – Solution obtenue avec l'heuristique d'arrondi sur l'instance industrielle avec PF. Visualisation des routes de périodicité $p = 3$

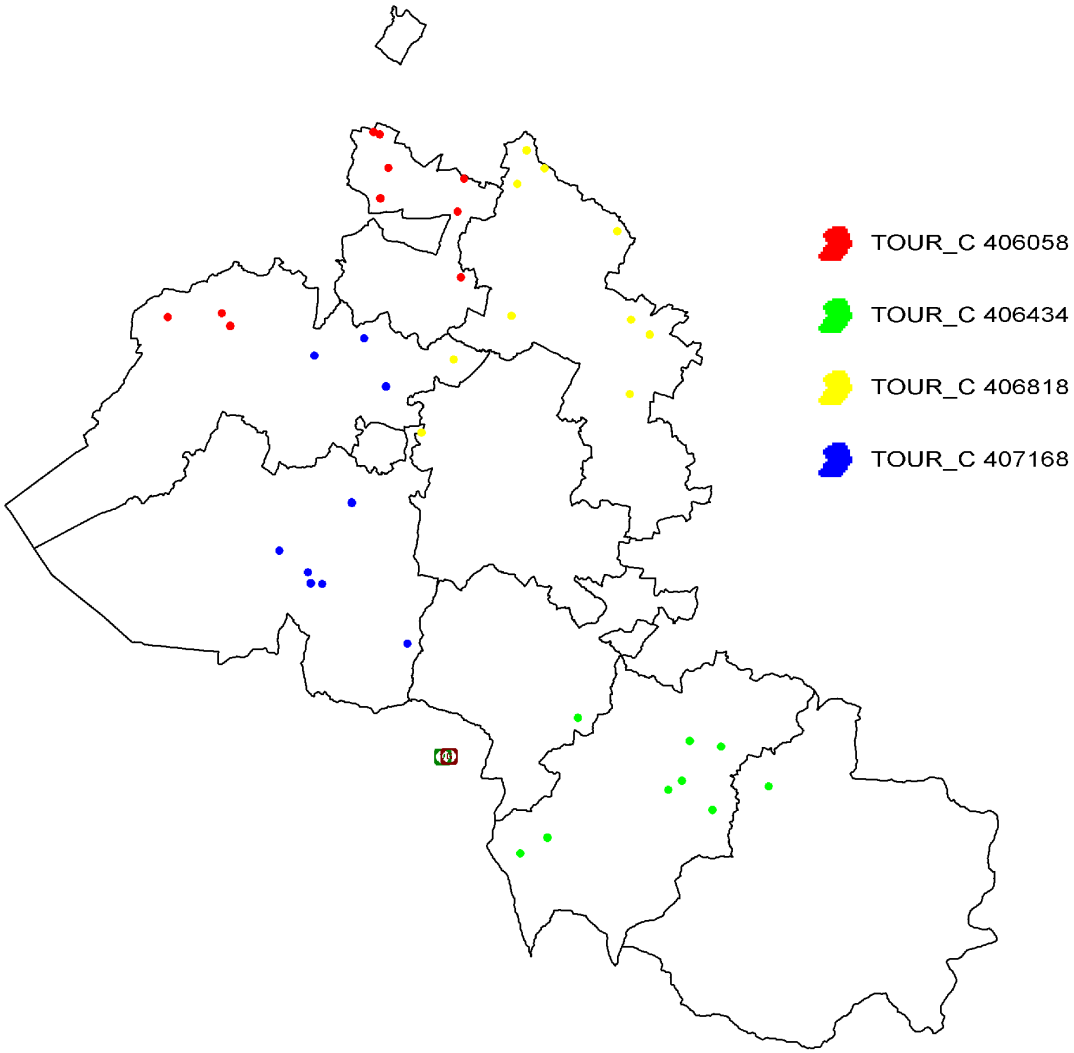



FIG. 8.13 – Solution obtenue avec l'heuristique d'arrondi sur l'instance industrielle avec PF. Visualisation des routes de périodicité $p = 4$

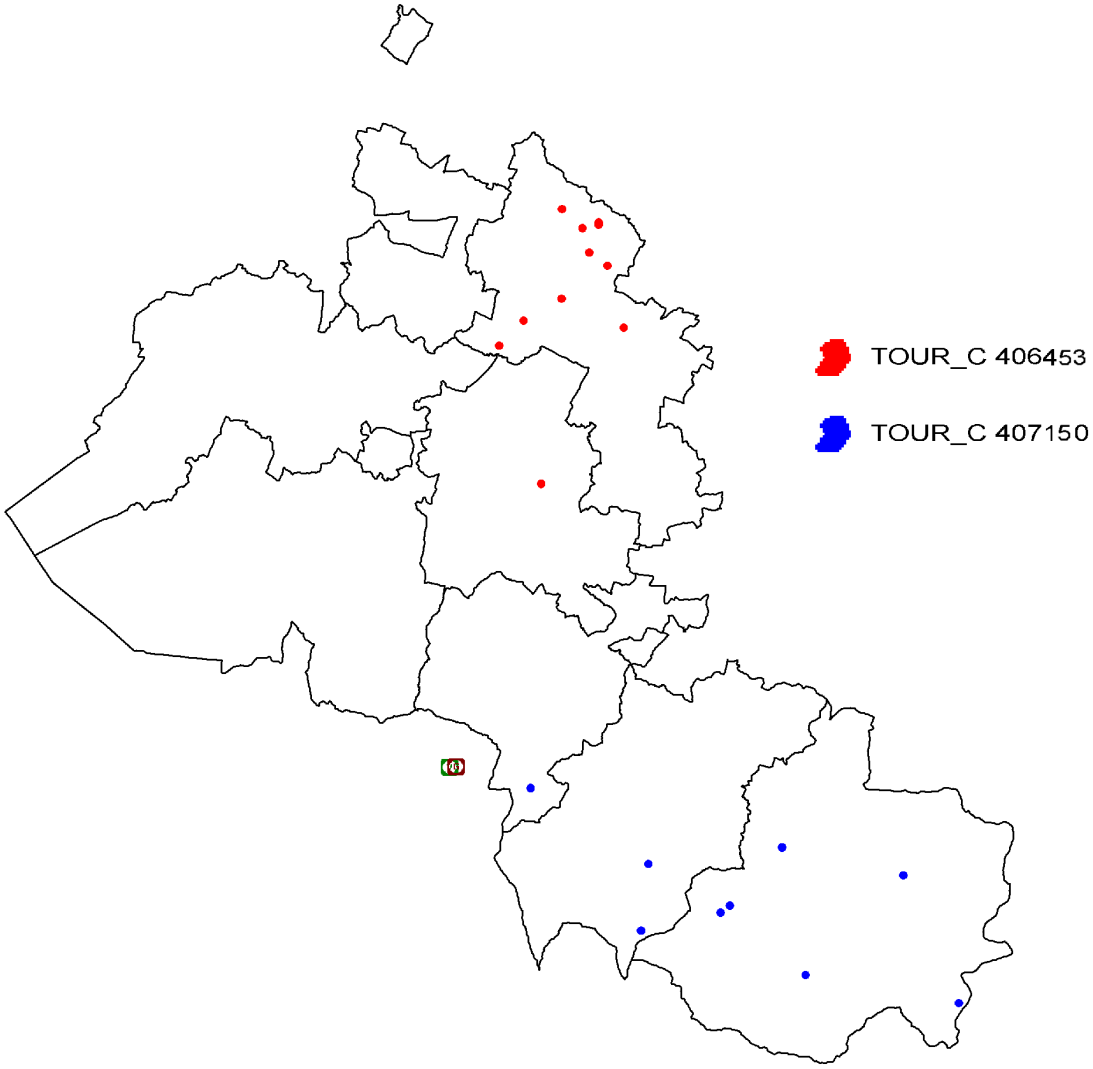



FIG. 8.14 – Solution obtenue avec l'heuristique d'arrondi sur l'instance industrielle avec PF. Visualisation des routes de périodicité $p = 5$

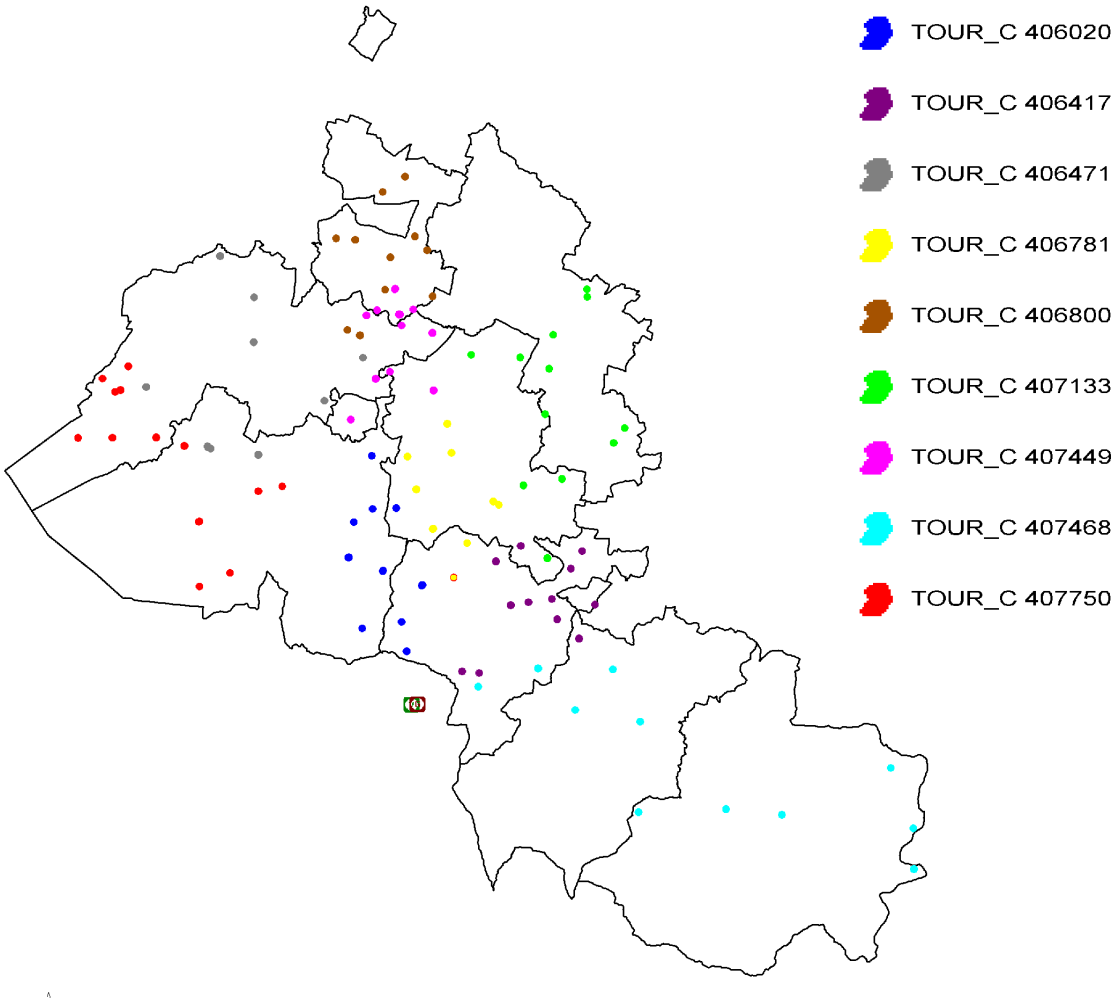



FIG. 8.15 – Solution obtenue avec l'heuristique d'arrondi sur l'instance industrielle avec PF. Visualisation des routes de périodicité $p = 6$

## 8.4 Conclusion

Les heuristiques développées ont permis de résoudre le problème industriel en un temps raisonnable (pour le niveau tactique). Les solutions obtenues répondent bien à l'objectif tactique qui consiste à déterminer d'une part le nombre de véhicules nécessaires, et d'autre part les regroupements de clients proches géographiquement et/ou ayant une périodicité de collecte similaire. La restriction de l'espace à l'ensemble $P = \{1, 2, 3\}$ permet de mieux répondre à l'objectif de regroupement des clients. On constate en effet que la structure de la solution est déterminée par les bornes fréquemment collectées.

Notre meilleure déviation à l'optimum est égale à 8.18%. Cette déviation est due aux coûts des routes. Remarquons que des gains pourraient être fait en transformant certaines routes de périodicité 3 par deux routes de périodicité 6 qui collecteraient chacune moins de bornes (on passerait moins de temps à vider les bornes). Par exemple, supposons une route de périodicité 3 commençant à la date 1 qui visite deux bornes telles que $tmax_i = 6$ (i.e. ces deux bornes sont collectées de l'équivalent de 3 périodes de production aux dates 1,4,7,10,13,16,...). On peut visiter ces bornes toutes les 6 semaines, en visitant l'une aux dates 1,7,13,... et l'autre aux dates 4,10,16.... (i.e. à chaque collecte de ces bornes, l'équivalent de 6 périodes de production sont ramassées). Cependant, nous pensons aussi qu'une partie de la déviation à l'optimum vient de la borne duale (obtenue par une relaxation du problème).

La comparaison avec la solution actuellement mise en place par le client est difficile car cette solution est opérationnelle et doit faire face aux aléas. Cependant, nous avons comparé le comportement moyen sur une semaine de la solution utilisée en 2005 par rapport à notre meilleure solution tactique. Actuellement, en moyenne, 10 routes collectant le verre sont effectuées chaque semaine. Ces routes collectent 59 bornes mais en réalité elles en visitent plus, ne vidant que celles qui sont suffisamment pleines ; 782.63 km sont parcourus. Dans notre solution, les 9 routes collectent 110 bornes en moyenne et 724.173 km sont parcourus. Par semaine, une route est économisée, et moins de km sont parcourus. En revanche beaucoup plus de bornes sont collectées. Cela n'est pas surprenant vu que dans notre solution, des bornes ayant une périodicité supérieure à 6 sont quand même collectées toutes les 3 semaines alors qu'elles sont à moitié vide. Cette stratégie a un avantage collatéral : elle permet une meilleure connaissance du terrain et on risque moins de trouver des bornes

dont la capacité est dépassée.

Nous pensons que notre solution tactique peut aider le modèle opérationnel à produire de bonnes solutions pratiques. Le modèle opérationnel se baserait sur les fréquences de visite et les regroupements des bornes dictés par la solution tactique. Par la suite, il serait intéressant de comparer la solution opérationnelle obtenue en s'aidant de cette solution tactique à la solution opérationnelle actuelle.

# Conclusion

Cette thèse est une étude d'un problème d'optimisation des tournées lié à la gestion de stock, connu sous le nom d'"Inventory Routing Problem" (IRP). L'application sous-jacente est la collecte des bornes de déchets recyclables. La problématique nous a été présentée par notre partenaire industriel Synoptis. Notre travail a consisté à trouver une modélisation et à développer une méthode de résolution basée sur une approche exacte.

Nous avons proposé une classification des problèmes de type IRP avec un aperçu des méthodes utilisées dans la littérature. Nous avons montré qu'il existe de nombreuses variantes à ce problème, dépendant principalement de l'application sous-jacente et du niveau de décision auquel on se place (tactique ou opérationnel). La variante du problème qui fait l'objet de notre étude n'a pas encore été traitée dans la littérature. Il est donc difficile de comparer les méthodes que nous proposons à celles de la littérature, d'autant plus qu'il n'existe pas de jeux de données de type "benchmark". Cependant, on constate que la plupart des études précédentes utilisent une méthode heuristique basée sur une approche en deux phases : la sélection des clients à livrer et l'élaboration des routes (i.e. une optimisation hiérarchisée).

Nous obtenons des solutions approchées à notre problème (dont la déviation à l'optimum est donnée) en appliquant le principe de décomposition de Dantzig-Wolfe et en développant des approches heuristiques basées sur la programmation mathématique. Nous avons revu, dans le chapitre 2, les heuristiques basées sur une approche de décomposition utilisées dans la littérature. Les schémas génériques des heuristiques de type glouton, d'arrondi et de recherche locale donnés ont permis de mieux faire apparaître les variantes existantes.

Dans les chapitres 3 à 5, nous analysons différentes modélisations pour la

construction des routes (opérationnelles ou tactiques), pour l'élaboration du planning de collecte, et pour le problème global (choix de décomposition). Nous avons montré par une étude comparative que la modélisation la mieux adaptée à notre problème est de considérer un horizon de temps infini et des routes définies comme des ensembles homogènes de clients. Elle permet d'avoir une solution continue plus rapidement et les solutions entières sont plus pratiques pour l'utilisateur.

La formulation exacte de ce modèle (en temps discret) est très coûteuse en temps de calcul. Cela est principalement dû à la symétrie dans le temps de cette formulation. Aussi nous avons proposé une formulation agrégée qui est obtenue en utilisant la technique de "state space relaxation". Cette formulation agrégée (qui modélise un comportement moyen) évite la symétrie en temps. Nous avons montré que les deux formulations, discrète et agrégée, fournissent la même solution $PL$, mais la solution $PLE$ est différente. La formulation agrégée est une relaxation du problème. Notre algorithme se base donc sur ces deux formulations pour calculer une borne primale et une borne duale respectivement. Des instances de taille réelle peuvent ainsi être résolues.

Nos résultats du chapitre 6 montre que des bornes duales de qualité peuvent être obtenues par une méthode de Branch-and-Price-and-Cut tronquée appliquée à la formulation agrégée. L'ajout de coupes basées sur des fonctions superadditives permet une amélioration peu significative de la borne duale. Une amélioration plus franche est obtenue en branchant sur le nombre de véhicules utilisés. La meilleure borne duale obtenue en combinant l'ajout des coupes et les contraintes de branchement est enregistrée pour évaluer nos solutions entières.

Au chapitre 7, nous avons montré comment construire de bonnes solutions primales à l'aide d'heuristiques basées sur l'approche de décomposition qui font une utilisation croisée des formulations discrète et agrégée : ces heuristiques utilisent la solution continue obtenue par la formulation agrégée mais construisent une solution entière pour la formulation discrète. Nous avons testé plusieurs heuristiques. Les meilleures solutions sont obtenues par l'heuristique d'arrondi. La variante la plus performante est celle qui appelle cette heuristique sur la solution $PL$ optimale (sans les coupes) de la racine et après le branchement sur le nombre de véhicules utilisés. Nous avons développé une heuristique de recherche locale pour améliorer les solutions obtenues. La recherche locale s'appuie d'ailleurs sur l'heuristique d'arrondi pour compléter la

solution partielle obtenue après la suppression de quelques colonnes. Pour que les temps de calcul restent raisonnables, l'heuristique de recherche locale est appelée seulement sur les solutions obtenues à la racine de l'arbre de Branch-and-Price.

Dans le dernier chapitre, nous appliquons notre meilleur algorithme aux données industrielles. La solution obtenue répond bien à l'objectif tactique fixé : minimiser le nombre de véhicules nécessaires et regrouper les clients proches géographiquement et/ou ayant une périodicité de collecte similaire. La restriction de l'espace à l'ensemble $P = \{1, 2, 3\}$ permet de mieux répondre à cet objectif de regroupement. On constate en effet que ce sont les bornes fréquemment collectées qui vont déterminer la structure de la solution. Nous pensons que la solution tactique obtenue peut permettre d'améliorer la solution opérationnelle existante dans laquelle les bornes à collecter sont sélectionnées principalement sur le critère de la fréquence de collecte. Des comparaisons directes sont difficiles, cependant notre solution tactique utilise une route de moins par semaine et les distances parcourues sont réduites.

Il existe plusieurs perspectives à nos travaux de recherche. Une extension possible concerne la méthode exacte de Branch-and-Price-and-Cut. De nouvelles inégalités valides pourraient être cherchées pour couper plus de solutions fractionnaires. De nouvelles contraintes de branchement pourraient permettre de développer partiellement l'arbre d'énumération (par exemple, on pourrait brancher sur le nombre de véhicules utilisés pour une périodicité $\hat{p}$, $\sum_{r,p^r=\hat{p}} \lambda_r$). Ensuite, une étude plus poussée du traitement de la symétrie dans la formulation discrète pourrait peut être permettre de ne pas passer par la formulation agrégée. Enfin, utiliser une autre méthode que le simplex pour résoudre le maître (comme la méthode de Bundle inexacte) pourrait peut être faire gagner du temps (en ayant une génération de colonnes plus stable).

On peut aussi envisager une étude plus poussée au niveau des heuristiques basées sur l'approche de décomposition. Par exemple, l'hybridation de l'heuristique glouton (qui itérativement génère une colonne puis la fixe) avec les autres heuristiques (la résolution du maître restreint, l'heuristique d'arrondi et la recherche locale) pourraient mener à un gain en temps de calcul, mais peut être au détriment de la qualité des solutions. Une méta-heuristique utilisant comme base l'heuristique de recherche locale pourrait être développée. D'autre part, il serait intéressant de tester l'heuristique de recherche locale sur d'autres

problèmes.

Enfin, un autre axe de recherche concerne la problématique. Nous avons traité le niveau tactique, il serait intéressant de développer le niveau opérationnel qui utiliserait les informations de la solution tactique. Le modèle tactique apporte une solution d'équilibre qui indique au modèle opérationnel les visites normalement prévues sur la semaine et les regroupements de bornes. Le modèle opérationnel, qui consiste à gérer le transitoire et les aléas tout en considérant une situation initiale, est ainsi moins "myope". Les visites pourraient être planifiées sur un horizon roulant d'une semaine découpé en périodes d'un jour ; seules les routes du premier jour seraient mises en œuvre ; puis on réitérerait la résolution décalée d'un jour. Les visites à effectuer une semaine intégreraient alors les visites prévues par la solution tactique plus celles qui se sont ajoutées comme demandes ponctuelles. Le but est d'intégrer à la solution d'équilibre du niveau tactique les demandes exceptionnelles tout en préservant autant que possible les regroupements des bornes dictés par la solution tactique tout en prenant en compte les remplissages observés sur le terrain.

# Bibliographie


[1] T. Achterberg et T. Berthold. Improving the feasibility pump. Technical report, ZIB-Report, 2005.

[2] Y. Agarwal, K. Mathur, et H. Salkin. A set partitioning based exact algorithms for the vehicle routing problem. *Networks*, 19 :731–749, 1989.

[3] EH. Aghezzaf, B. Raa, et H. Van Landeghem. Modeling inventory routing problems in supply chains of high consumption products. *European Journal of Operational Research*, 169 :1048–1063, 2006.

[4] RK. Ahuja, TL. Magnanti, et JB. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[5] S. Anily et A. Federgruen. One warehouse multiple retailer systems with vehicle routing costs. *Management Science*, 36 :792–114, 1990.

[6] P. Avella, M. Boccia, et A. Sforza. Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research*, 152 :170–179, 2004.

[7] M0. Ball, TL. Magnanti, CL. Monna, et GL. Nemhauser, eds. *Network Routing*, vol 8. Handbooks in OR and Management Science, Amsterdam, North Holland, 1995.

[8] C. Barnhart, EL. Johnson, GL. Nemhauser, MWP. Savelsbergh, et PH. Vance. Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, 46 :316–329, 1998.

[9] W. Bell, L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, et P. Prutzman. Improving the distribution of industrial gases with on-line computerized routing and scheduling optimizer. *Interfaces*, 13 :4–23, 1983.

[10] G. Belov et G. Scheithauer. A cutting plane algorithm for the one dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, 141 :274–294, 2002.

[11] G. Belvaux. *Modelling and solving Lot Sizing Problems by Mixed Integer Programming.* Thèse, Faculté des sciences appliquées, Université catholique de Louvain, Belgique, 1999.

[12] L. Bertazzi, G. Paletta, et MG. Speranza. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36 :119–132, 2002.

[13] R. Borndorfer, M. Grotschel, et A. Lobel. Scheduling duties by adaptive column generation. Technical report, ZIB-Report, 2001.

[14] J. Bramel et D. Simchi-Levi. A location based heuristic for general routing problems. *Operations Research*, 43 :649–660, 1995.

[15] AM. Campbell, LW Clarke, et MWP. Savelsbergh. *Inventory routing in practice*, P. Toth et D. Vigo, eds, *The Vehicle Routing Problem*, chapitre 12, p 309 – 330. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[16] AM. Campbell et MWP. Savelsbergh. A decomposition approach for the inventory routing problem. *Transportation Science*, 38 :488–502, 2004.

[17] A. Chabrier. Heuristic branch-and-price-and-cut to solve a network design problem. *Proceedings CPAIOR*, Montréal, Canada, mai 2003.

[18] A. Chabrier, E. Danna, et C. Le Pape. Coopération entre génération de colonnes et recherche locale appliquées au problème de routage de véhicules. *Huitièmes Journées Nationales sur la résolution de Problèmes NP-Complets (JNPC)*, p 83–97, Nice, France, mai 2002.

[19] LMA. Chan, A. Federgruen, et D. Simchi-Levi. Probabilistic analyses and pratical algorithms for inventory-routing models. *Operations Research*, 46 :96–106, 1998.

[20] TW. Chien, A. Balakrishnan, et RT. Wong. An integrated inventory allocation and vehicle routing problem. *Transportation Science*, 23 :67–76, 1989.

[21] M. Christiansen. Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, 33 :3–13, 1999.

[22] M. Christiansen et B. Nygreen. A method for solving ship routing problems with inventory constraints. *Annals of Operations Research*, 81 :357–378, 1998.

[23] N. Christofides et JE. Beasley. The period routing problem. *Networks*, 14 :237–256, 1983.

[24] N. Christofides et S. Eilon. Expected distances in distribution problems. *Operational Research Quaterly*, 20 :437–443, 1969.

[25] FL. Cimelière. *Optimisation du traitement de l'ordre de fabrication dans l'industrie textile*. Thèse, Université Bordeaux 1, France, 2004.

[26] G. Clarke et J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12 :568–581, 1964.

[27] K. Cousineau-Ouimet. A tabu search heuristic for the inventory routing problem. *Proceedings of the 37th Annual ORSNZ conference*, Auckland, New Zealand, novembre 2002.

[28] AL. Custódio et RC. Oliveira. A system of logistic management integrating inventory management and routing. *Optimization 2001*, Aveiro, Portugal, juillet 2001.

[29] E. Danna. *Intégration des techniques de recherche locale à la programmation linéaire en nombres entiers*. Thèse, équipe Recherche Opérationnelle et Optimisation du Laboratoire d'informatique, Université d'Avignon, France, 2004.

[30] Z. Degraeve et R. Jans. A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot sizing problem with set-up times. *ERIM Report Series in Management*, ERS-2003-010-LIS, 2003.

[31] RS. Dembo et PL. Hammer. A reduction algorithm for knapsack problems. *Methods of Operations Research*, 36 :49–60, 1980.

[32] G. Desaulniers, J. Desrosiers, A. Erdmann, MM. Solomon, et F. Soumis. *VRP with pickup and delivery*, P. Toth et D. Vigo, eds, *The Vehicle Routing Problem*, chapitre 9, p 225 – 242. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[33] G. Desaulniers, J. Desrosiers, et MM. Solomon. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. Technical Report G-99-36, Les cahiers du GERAD, 1999.

[34] M. Desrochers, J. Desrosiers, et M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40 :342–354, 1992.

[35] J. Desrosiers, Y. Dumas, M. Solomon, et F. Soumis. *Time Constrained Routing and Scheduling*, M0. Ball, TL. Magnanti, CL. Monna, et GL. Nemhauser, eds, *Network Routing*, vol 8, p 35–139. Handbooks in OR and Management Science, Amsterdam, North Holland, 1995.

[36] M. Dror, ed. *Arc Routing : Theory, Solution and Applications*. Kluwer Academic Publishers, 2000.

[37] M. Dror et M. Ball. Inventory/routing : reduction from an annual to a short-term problem. *Naval Research Logistics*, 34 :891–905, 1995.

[38] ME. Dyer, N. Kayal, et J. Walker. A branch-and-bound algorithm for solving the multiple choice knapsack problem. *Journal of Computational and Applied Mathematics*, 11 :231–249, 1984.

[39] A. Federgruen et D. Simchi-Levi. *Analysis of Vehicle Routing and Inventory Management Problems*, *Network Routing*, vol 8 of *Handbooks in OR and Management Science*, p 297–371. Amsterdam, Elsevier edition, 1995.

[40] D. Feillet, P. Dejax, et M. Gendreau. The profitable arc tour problem : Solution with a branch-and-price algorithm. *Transportation Science*, 39 :439–552, 2005.

[41] D. Feillet, P. Dejax, et M. Gendreau. Traveling salesman problem with profit. *Transportation Science*, 39 :188–205, 2005.

[42] D. Feillet, P. Dejax, M. Gendreau, et C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints : application to some vehicle routing problems. *Networks*, 44 :216–229, 2004.

[43] ML. Fisher. *Vehicle Routing*, M0. Ball, TL. Magnanti, CL. Monna, et GL. Nemhauser, eds, *Network Routing*, vol 8, p 1–33. Handbooks in OR and Management Science, Amsterdam, North Holland, 1995.

[44] ML. Fisher et R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11 :109–124, 1981.

[45] M. Fishetti, F. Glover, et A. Lodi. The feasibility pump. *Mathematical Programming*, 104 :91–104, 2005.

[46] M. Fishetti et A. Lodi. Local branching. *Mathematical Programming*, 98 :23–47, 2003.

[47] R. Fukasawa, M. Poggi de Aragão, O. Porto, et E. Uchoa. Robust branch-and-cut-and-price for the capacitated minimum spanning tree problem. *Proceedings of the International Network Optmization Conference*, p 231–236, Evry, France, 2003.

[48] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, et RF. Werneck. Robust branch-and-cut-and-price for the capacited vehicle routing problem. *Proceedings of the tenth Integer Programming and Combinatorial Optimization conference IPCO'04*, vol 3064, p 1–15, New York, 2004.

[49] M. Gamache, F. Soumis, G. Marquis, et J. Desrosiers. A column generation approach for large scale aircrew rostering problem. *Operations Research*, 47 :247–263, 1999.

[50] V. Gaur et ML. Fisher. A periodic inventory routing problem at a supermarket chain. *Operations Research*, 52 :813–822, 2004.

[51] AM. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Studies*, 2 :82–114, 1974.

[52] O. Günlük, T. Kimbel, L. Ladangi, B. Schieber, et G. Sorkin. Vehicle routing and staffing for sedan service. Technical Report RC23208, IBM Research Report, 2005.

[53] B. Golden, A. Assad, et R. Dahl. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale System 7*, p 181–190, 1984.

[54] M. Göthe Lundgren, T .Larsson, et A. Westerlund. A stabilized column generation and cutting plane scheme for the traveling salesman subtour problem. Technical report, Linkoping University. Department of Mathematics, 2005.

[55] G. Hadley et TM. Whitin. *Etude et pratique des modèles de stocks*. Dunod, 1966.

[56] P. Hansen et B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79 :191–215, 1997.

[57] S. Irnich et D. Villeneuve. The shortest path problem with ressource constraints and k-cycle elimination for k$\geq$ 3. Technical Report G-2003-55, Les cahiers du GERAD, 2003.

[58] P. Jaillet, J. Bard, L. Huang, et M. Dror. Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science*, 36 :292–300, 2002.

[59] V. Kaibel et ME. Pfetsch. Orbitopes. *10th Workshop on Combinatorial Optimization*, Aussois, France, janvier 2006.

[60] K. Kiwiel. An inexact bundle approach to cutting stock problems. Technical report, 2005.

[61] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, et D.B. Shmoys, eds. *The Traveling Salesman Problem, a guided tour of Combinatorial Optimization*. John Wilsey and Sons, 1985.

[62] AN. Letchford et JJ. Salazar-González. Projection results for vehicle routing. *Mathematical Programming*, 105 :251–274, 2006.

[63] Q. Liu, HC. Lau, D. Seah, et S. Chong. An efficient near exact algortihm for large scale vehicle routing with time windows. *Proceedings of the 5th World Congress on ITS*, Korea, 1998.

[64] V. Malépart, F. Boctor, J. Renaud, et S. Labillois. Nouvelles approches pour l'approvisionnement des stations d'essence. *Revue Française de Gestion Industrielle*, 22 :15–31, 2002.

[65] F. Margot. Symmetry in integer programming. *IMA Special Workshop : Mixed-Integer Programming*, Minneapolis, USA, juillet 2005.

[66] F. Menezes, L. Moreno, M. Poggi, et E. Uchoa. Planning flight schedules to offshore platforms. *19ème International Symposium on Mathematical Programming*, Rio de Janeiro, Brésil, août 2006.

[67] M. Mourgaya. *The periodic vehicle routing problem : planning before routing.* Thèse, Université Bordeaux 1, France, 2004.

[68] GL. Nemhauser et LA. Wolsey. *Integer and Combinatorial Optimization*, chapitre II.1.4 : *The theory of valid inequalities : superadditive functions and valid inequalities*, p 229–237. John Wiley & Sons, 1988.

[69] N. Perrot. *Integer Programming Column Generation Strategies for the Cutting Stock Problem and its Variants.* Thèse, Université Bordeaux 1, France, 2005.

[70] D. Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83 :394–410, 1995.

[71] D. Pisinger. Optimization codes for knapsack problems written in C. 1995. http ://www.diku.dk/ pisinger/codes.html.

[72] Y. Pochet. *Mathematical programming models and formulations for deterministic production planning problems*, M. Jünger et D. Naddef, eds, *Computational Combinatorial Optimization*, p 57–111. Springer Verlag LNCS 2241, 2001.

[73] Y. Pochet et LA. Wolsey. Algorithms and reformulations for lot sizing problems. *DIMACS Series in Discrete Mathematics and theorical Computer Science*, 20 :245–293, 1995.

[74] M. Poggi de Aragão et E. Uchoa. Integer program reformulation for robust branch-and-cut-and-price algorithms. *http ://citeseer.ist.psu.edu/poggidearagao03integer.html*, 2003.

[75] projet ROCOCO. http ://www.rnrt.org/rnrt/projets/pres_d107_ap99.htm.

[76] W. Qu, J. Bookbinder, et P. Iyogun. An integrated inventory transportation system with modified periodic policy for multiple products. *European Journal of Operational Research*, 115 :254–269, 1999.

[77] CC. Ribeiro et F. Soumis. A column generation approach to the multiple depot vehicle scheduling. *Operations Research*, 42 :41–52, 1994.

[78] E. Taillard. A heuristic column generation method for the heterogeneous vrp. *RAIRO Operations Research*, 33 :1–14, 1999.

[79] D. Taqa Allah, J. Renaud, et F. Boctor. Le problème d'approvisionnement des stations d'essences, the gas stations supply problem. *Journal européen des systèmes automatisés*, 34 :11–33, 2000.

[80] P. Toth et D. Vigo, eds. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[81] P. Vance, A. Atamturk, C. Barnhart, E. Gelman, A. Krishna E. Johnson, D. Mahidhara, G. Nemhauser, et R. Rebello. A heuristic branch-and-price approach for the airline crew pairing problem. *citeseer.ist.psu.edu/vance97heuristic.html*, 1997.

[82] F. Vanderbeck. Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Operations Research*, 48 :915–926, 2000.

[83] F. Vanderbeck. *Implementing Mixed Integer Column Generation*, G. Desaulniers, J. Desrosiers, et MM. Solomon, eds, *Column Generation*. Kluwer, 2005.

[84] B. Verweij et LA. Wolsey. Uncapacitated lot-sizing with buying, sales and backlogging. *Optimization Methods and Software*, 19 :427–435, 2004.

[85] I. Webb et R. Larson. Period and phase of customer replenishment : A new approach to the strategic inventory/routing problem. *European Journal of Operational Research*, 85 :132–148, 1995.

[86] M. Witucki, P. Dejax, et M. Haouari. Un modèle et un algorithme de résolution exacte pour le problème de tournées de véhicules multipériodique : une application à la distribution des gaz industriels. *Deuxième congrès international franco-québecois de génie industriel*, Albi, France, 1997.

[87] Xpress. *User guide and Reference Manual, Dash Optimization*. http ://www.dashoptimization.com.

# OPTIMISATION DES TOURNEES DE VEHICULES COMBINEES A LA GESTION DE STOCK

**Résumé :** Dans le problème de la collecte des conteneurs de déchets recyclables, une flotte de véhicules est affectée à collecter un seul produit sur différents sites. Chaque site a son propre taux d'accumulation et sa capacité de stockage. A chaque visite, le stock est vidé. Dans la phase de planification tactique, nous cherchons une solution périodique qui est répétée dans le temps. L'objectif est de minimiser la taille de la flotte et les coûts de transport tout en donnant un découpage régionale de l'espace par une partition des sites entre les véhicules. Au terme d'une analyse comparative nous proposons une modélisation adéquate. La structure du problème est exploitée pour développer une approche de décomposition de Dantzig-Wolfe. Des plannings périodiques sont générés pour les véhicules en résolvant un problème de sac-à-dos à choix multiple. L'élaboration des plannings de collecte pour les sites est traitée dans le programme maître. Ce dernier est reformulé en terme de variables agrégées pour éviter la symétrie en temps. Les bornes duales sont calculées par une méthode tronquée de Branch-and-Price-and-Cut. Les bornes primales sont obtenues par des heuristiques d'arrondi et de recherche locale développées dans le contexte de la génération de colonnes. Des instances réelles sont résolues avec une déviation à l'optimalité raisonnable.

**Mots-clés :** *tournées de véhicules, planification, génération de colonnes, heuristiques primales*

# INVENTORY ROUTING PROBLEM

**Abstract:** We consider an application of the inventory routing problem. A fleet of vehicles is devoted to collecting a single product from geographically dispersed sites. Each site has its own accumulation rate and stock capacity. On each visit, the vehicle empties the stock. In the tactical planning phase, we search for a periodic solution to be repeated over time. The objectives are to minimise the vehicle fleet size as well as the transportation cost, while achieving some form of regional clusterisation in partitioning the sites between the vehicles. We propose an adequate modelisation after a comparative analysis. The structure of the problem is exploited to develop a Dantzig-Wolfe decomposition approach. Periodic plannings are generated for vehicles by solving a multiple choice knapsack problem. The issues related to the construction of the customer plannings are dealt with in a master program. The latter program is reformulated in terms of aggregated variables to avoid the symmetry in time. Dual bounds are obtained by truncated Branch-and-Price-and-Cut. Primal bounds are derived through rounding and local search procedures specially developed for use in the context of a column generation approach. Real-life instances of the problem are solved with reasonable optimality gaps.

**Keywords:** *vehicle routing, planification column generation, primal heuristics*