Problèmes de production avec transport des composants

Carlos Heitor Pereira Liberalino

▶ **To cite this version:**

Carlos Heitor Pereira Liberalino. Problèmes de production avec transport des composants. Autre [cs.OH]. Université Blaise Pascal - Clermont-Ferrand II, 2012. Français. NNT : 2012CLF22230 . tel-00719567

HAL Id: tel-00719567

https://theses.hal.science/tel-00719567

Submitted on 20 Jul 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : 2230
EDSPIC : 555

Université Blaise Pascal - Clermont-Ferrand II

Ecole Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

THÈSE

présenté par

Heitor LIBERALINO

pour obtenir le grade de

Docteur d'Université

Spécialité : Informatique

Problèmes de Production avec Transport des Composants

Soutenue publiquement le 22 mars 2012
devant le jury composé de :

Président :

Safia KEDAD-SIDHOUM	Maître de Conférences, Université Pierre et Marie Curie, Paris, France

Rapporteurs :

Mauricio C. DE SOUZA	Maître de Conférences, Universidade Federal de Minas Gerais, Belo Horizonte, Brésil
Alexandre DOLGUI	Professeur, Ecole Nationale Supérieure des Mines, Saint-Etienne, France
Aziz MOUKRIM	Professeur, Université de Technologie de Compiègne, Compiègne, France

Directeurs de thèse :

Alain QUILLIOT	Professeur, Université Blaise Pascal, Clermont-Ferrand, France
Christophe DUHAMEL	Maître de Conférences, Université Blaise Pascal Clermont-Ferrand, France

Problèmes de Production avec Transport des Composants

Thèse préparée au sein du laboratoire LIMOS (Laboratoire d'Informatique de Modélisation et d'Optimisation des Systèmes) Unité Mixte de Recherche 6158 du CNRS .

LIMOS
Complexe scientifique des Cézeaux,
63173 AUBIERE cedex, FRANCE.

Remerciements

Je tiens à remercier,

Tout d'abord Celui qui a plusieurs noms, dans plusieurs langues, mais que j'appelle affectueusement mon Père.

Ma famille brésilienne, spécialement ma mère Fátima et ma sœur Camila, qui m'a encouragé et m'a soutenu tout ce temps et qui, même de loin, a été présente dans ma vie.

Ma famille française Bernard, Monique et Julien qui m'ont adopté de tout leur cœur.

Ma chérie Émilie pour avoir partagé tous les moments vécus ici en faisant preuve de compréhension et patience.

Mes responsables de thèse Alain Quilliot et Christophe Duhamel qui m'ont bien guidé dans la voie de la recherche opérationnelle.

Safia Kedad-Sidhoum et Phillipe Chrétienne pour la collaboration dans le projet ANR.

Mon eternel maître Dario Aloise qui a toujours veillé sur ses élèves comme s'ils étaient ses enfants.

Mes amis du bureau Andréa, Libo, Raksmey, Jeremy, Jonathan, Hélène et Philippe, pour toute l'amitié et pour être toujours disponibles pour m'écouter et me donner un coup de main.

A mes amis Renata, Datta, Erol, Alexandre, Nathalie, Amandine et Laurie qui m'ont aidé (directement ou indirectement) dans mes travaux de recherche ainsi que dans ma vie privée. Chacun de vous m'a donné un bon exemple de conduite et de principes.

A tous ceux qui ont cru en moi et qui ont rendu ma vie plus agréable même avec un simple et sincère bonjour.

Enfin, mon fils Lucca pour me donner une raison de devenir de plus en plus un bon exemple de conduite morale. C'est à lui je dédie cette thèse.

D'ici qu'à l'infini je vous redirez merci.

Viver,
E não ter a vergonha
de ser feliz
Cantar e cantar e cantar
a beleza de ser
um eterno aprendiz...

O que é o que é?
Gonzaguinha

Résumé

Dans ce travail nous considérons des problèmes de planification de production sur plusieurs sites avec transport de produits entre ces sites. L'objectif est de synchroniser les deux problème (planification et transport) et de construire une solution globale. Le système de production sur chaque site est modélisé comme un problème de *Capacitated Lot-Sizing* où nous travaillons avec stock et ressources. Le transport de produits entre les sites se ramène à une version simplifiée du *Vehicle Routing Problem* où le temps est discrétisé. D'abord nous proposons un modèle linéaire en nombres entiers que nous appelons le « *Lot-Sizing and Vehicle Routing Problem* » (LSVRP). Puis nous présentons deux cas particuliers : le *Single-item* LSVRP (S-LSVRP) et le *Single-level* LSVRP (1-LSVRP). Les problèmes sont traités ici par six heuristiques que nous avons développé. Quatre de ces méthodes sont des heuristiques qui utilisent la programmation en nombres entiers et prennent en compte la relaxation linéaire de quelques variables du problème. Elles s'appuient sur l'exploration partielle de l'arbre de décision et la fixation de variables. Les deux autres sont spécifiques pour les cas particuliers. La première, qui traite le S-LSVRP, est basée sur la propagation des ordres de production sur chaque site. Puis à chaque itération elle calcule le plan de transport compatible et essaie d'améliorer la solution en modifiant la production sur les sites. L'autre méthode consiste en une relaxation lagrangienne qui travaille sur une modélisation du 1-LSVRP en un problème de flot. Des résultats numériques et des analyses sont présentés pour évaluer l'efficacité de ces heuristiques.

Mots-clés : Production, planification, transport, heuristique, relaxation lagrangienne, multiflot.

Abstract

In this work we consider some problems of scheduling both a production distributed on several sites and the transportation of items between those sites. By doing so, the objective is to synchronize the two components and to build a better overall solution. The production system on each site is modeled as a Capacitated Lot-Sizing Problem where stock both on resources and produced items is available. The inter-site items transportation is a simplified version of the Vehicle Routing Problem where time is discretized. We first propose a mixed integer linear programming formulation that we call "The Lot-Sizing and Vehicle Routing Problem" (LSVRP). Then we present two particular cases: The Single-item LSVRP (S-LSVRP) and The Single-level LSVRP (1-LSVRP). All those cases are treated here by the six heuristics we develloped. Four of those methods are MIP based heuristics and take in account the the linear relaxation of some variables of the problem. They rely on partial decision tree exploration along with variable fixing. The other two are specifics for the two particular cases. The one who treats the S-LSVRP is based on production order propagation over the sites. Then, at each iteration, it computes a compatible transportation schedule and it tries to improve the solution by modifying the production on the sites. The other method consists in a lagrangian relaxation that works with an adaptation of the 1-LSVRP into a flow problem. Computational results and analysis are presented to evaluate the efficiency of those heuristics.

Keywords: Production, scheduling, transportation, heuristics, matheuristics, lagrangian relaxation, multi-flow.

Table des matières

Résumé **9**

Abstract **11**

Introduction **17**

1 État de l'art **21**
1.1 Introduction 21
1.2 Motivation 23
1.3 Modèles de type *Lot-Sizing* 26
1.3.1 Définitions et caractéristiques 26
1.3.2 Modèles de *lot-sizing* 32
1.4 Problèmes de tournée de véhicules 39
1.4.1 Définitions et caractéristiques 39
1.4.2 Modèles de problèmes de tournées 40
1.5 Méthodes de résolution 45
1.5.1 Méthodes exactes 45
1.5.2 Méthodes approchées 52

2 Modèle intégré de production et transport **57**
2.1 Introduction 57
2.2 Modélisation mathématique 62
2.3 Méthodes de résolution pour le cas général 72
2.3.1 Heuristique *Relax-and-Fix* 73
2.3.2 Heuristique *Insert-and-Fix* 75
2.3.3 Heuristique *Fractional Relax-and-Fix* 77
2.3.4 Comparaison entre les heuristiques *Relax-and-Fix*, *Insert-and-Fix* et *Fractional Relax-and-Fix* 79
2.3.5 Heuristique *Diving* 80
2.4 Cas particulier de production dédiée 81
2.4.1 Modèle mathématique 81
2.4.2 Une heuristique simple pour le cas particulier S-LSVRP 84

3 Analyse et résultats **95**
3.1 Génération des instances . . . 95
3.2 Analyse des formulations . . . 97
3.2.1 Comparaison entre formulations . . . 98
3.3 Résultats des heuristiques . . . 99
3.3.1 Calibrage des heuristiques . . . 99
3.3.2 Résultats pour le problème LSVRP . . . 101
3.3.3 Résultats pour le cas particulier (S-LSVRP) . . . 108
3.4 Difficultés rencontrées dans l'implémentation . . . 116

4 Schéma lagrangien pour le LSVRP mono-niveau avec transfert **119**
4.1 Objectifs . . . 119
4.2 Présentation du problème . . . 120
4.2.1 Données et variables du problème . . . 120
4.2.2 Coûts . . . 121
4.2.3 Capacités . . . 122
4.3 Reformulation du problème sous une forme de flot/graphe . . . 122
4.3.1 Graphe de base . . . 122
4.3.2 Graphe réduit . . . 123
4.3.3 Reformulation multiflot du problème de *lot-sizing* . . . 123
4.3.4 Le cas sans capacités de stockage/transfert et le réseau réduit H . . . 125
4.4 Le problème sans capacité : problème de localisation . . . 125
4.4.1 Problème de localisation mono-produit . . . 125
4.4.2 Un algorithme simple pour le problème de localisation mono-produit . . . 126
4.4.3 Le problème sans capacité vu comme un problème de localisation 128
4.5 Traitement des contraintes de capacités de production . . . 128
4.5.1 Maximisation du lagrangien . . . 129
4.5.2 Lissage . . . 131
4.5.3 Schéma algorithmique général . . . 132
4.6 Traitement des contraintes de capacités en deux étapes . . . 133
4.6.1 Maximisation du lagrangien . . . 134
4.6.2 Lissage . . . 135
4.6.3 Schéma algorithmique général . . . 136
4.7 Traitement en simultané des contraintes de capacités . . . 137
4.7.1 Maximisation du lagrangien . . . 137
4.7.2 Lissage . . . 139
4.8 Compléments . . . 140
4.8.1 Perfectionnement du calcul de $\lambda_{(max)}$ . . . 141
4.8.2 Perfectionnement des procédures de lissage . . . 142
4.8.3 Perfectionnement des procédures de lissage : extension . . . 142

4.8.4 Gestion des échecs : réparation des routages 143
4.9 Résultats 143

Conclusion **147**

Index **149**

Introduction

Le scénario post révolution industrielle représente la montée de ce que nous appelons la « société de consommation ». Ce qui avant était fabriqué de manière artisanale à coûts élevés et en une infime quantité, a commencé à être conçu par des machines à vapeur, en grande quantité, et plus accessibles à la population. Cette facilité d'acquisition a générée des habitudes auprès de la société, et ces habitudes ont changé la manière de travailler dans les ateliers (aujourd'hui plus connues comme des usines où des sites de fabrication). La façon « moderne » de vivre a amené les fabricants à organiser leur production en imposant une systématisation nécessaire pour respecter les demandes de la société.

Parallèlement à cette nécessité de répondre aux besoins du marché, la normalisation des processus de fabrication se traduit par la maîtrise de chaque élément du système (machines, stocks, fournisseurs) de manière à mettre en place un plan de production le plus efficace possible. Un plan de production comprend l'acquisition des ressources et des matières premières, et la planification des activités de production qui sont nécessaires pour transformer la matière première en produits finaux de façon la plus économique possible [82]. Cette économie peut être réalisée par le groupement des produits par lots. Dans ce contexte une question importante qui se pose est « quelle est la taille d'un lot (ou des lots) à fabriquer pour satisfaire les demandes ? ». Ce problème s'appelle *Lot-Sizing* (dimensionnement de lots) et joue un rôle assez important dans le processus de production. La décision de la taille d'un lot à fabriquer comprend plusieurs questions liées au respect de la demande du client et à la minimisation des coûts impliqués : quelle quantité fabriquer ? à quelle période ? quelle quantité stocker ? est-ce qu'un retard de production est plus rentable qu'une augmentation de la capacité de fabrication ? etc.

A cause de l'expansion et de la mondialisation du marché, les industries et les entreprises ont fait beaucoup d'efforts afin de réduire la complexité du processus de production et de maîtriser les temps et coûts de démarrage (*setup*), de manière à fournir des planifications de production adaptées aux exigences des clients. Actuellement, le problème de dimensionnement de lots n'est qu'une partie d'un système moderne de production. La croissance du marché a exacerbé la concurrence entre les entreprises de taille nationale et internationale. De ce fait, pour rester compétitifs les entreprises cherchent à faire des économies sur différents plans. Cela se traduit classiquement par les phénomènes de délocalisation pour rechercher des lieux où les

coûts de fabrication sont les plus faibles possibles. Mais en délocalisant, une entreprise prend le risque d'éparpiller ses sites de production et elle fragilise sa chaîne logistique. De plus, elle réduit sa capacité d'optimiser les coûts logistiques. Ainsi, bien que la production reste la préoccupation centrale, prendre en compte la logistique inter-sites offre la possibilité d'améliorer les plans de production sur l'ensemble des sites, que la logistique soit confiée à un prestataire de service (externalisation) ou non (service interne à l'entreprise). Alors, dans ce contexte, un nouvel élément intervient dans les systèmes de production : le transport.

Selon [73], la délocalisation des centres de production dans les pays où la main d'œuvre est moins chère a engendré des coûts de transport extrêmement onéreux. Ainsi, depuis les années 1970 l'intégration du transport dans la production a été prise en compte d'une manière plus sérieuse, et plusieurs modèles et approches de résolution ont été développés pour traiter ce couple de problèmes.

Le but de ce travail est de traiter ce couplage entre problème de production et le problème de transport dans une chaîne logistique. Notre étude se déroule sur trois types de systèmes :

- Un système multi-niveaux et multi-produit de fabrication pour lequel plusieurs composants sont considérés et pour lequel les sites de fabrication sont géographiquement distants, de manière à ce que la durée de transport soit supérieure à la durée de production. Nous appelons ce problème le « *Lot-Sizing with Vehicle Routing Problem* » (LSVRP) car il vient du problème *Lot-Sizing* couplé avec le problème de tournée de véhicules (*Vehicle Routing Problem*) ;
- Le deuxième type de problème est un cas particulier du LSVRP dont la production est dédiée, c'est-à-dire que chaque produit est fabriqué sur un seul site de production et chaque site fabrique un seul type de produit. Ce cas particulier est appelé « *Single-item* LSVRP » (ou S-LSVRP) ;
- Le dernier type de problème considère un système multi-produits mais mono-niveau, c'est-à-dire qu'il n'y a pas de consommation de composants. De cette façon la chaîne de production complète est centralisée dans une seule région. Alors, le transport de produits entre sites et clients (ou sites qui ont besoin de produits pour couvrir une rupture de stock) peut être considéré comme un transfert instantané qui se passe à la période de production. Ce problème est appelé « *Single-level* LSVRP » (ou 1-LSVRP).

L'étude de ces problèmes a produit des bons résultats. Nous avons pu présenté les premiers résultats pour le S-LSVRP dans le 11$^{\text{ème}}$ congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2010). De même, les premiers avancements des heuristiques ont été présentés à ROADEF 2011. De plus, dans l'année 2011 nous avons participé au *IEEE Symposium Series on Computational Intelligence* (SSCI 2011). A la même année nous avons participé au XLIII *Simpósio Brasileiro de Pesquisa Operacional* (Symposium Brésilien de

Recherche Opérationnelle - SBPO 2011), à Ubatuba, état de São Paulo, Brésil. Dans ce symposium, nous avons publié un article complet sur nos méthodes de résolution.

Cette thèse est organisée comme suit. Initialement, dans le chapitre 1, nous présentons une étude bibliographique sur différents modèles de *lot-sizing* qui ont influencé ce travail. De même nous donnons une vision générale des problèmes de tournée de véhicules. Ensuite, dans le même chapitre, nous montrons quelques méthodes de résolution connues dans la littérature pour résoudre chacun de ces problèmes. Quelques méthodes présentées ont servi comme base pour les heuristiques que nous avons développées.

Le chapitre 2 présente la formulation mathématique pour le problème LSVRP, reposant sur la programmation linéaire mixte, et aussi pour le cas particulier S-LSVRP. Pour chaque problème nous présentons aussi les heuristiques que nous avons développées.

Dans le chapitre 3, nous faisons une analyse des résultats fournis par les heuristiques proposées. De même, nous analysons le LSVRP et son cas particulier, tout en montrant les différences et quelles sont les caractéristiques qui jouent sur la complexité de chacun.

Le chapitre 4 présente le problème 1-LSVRP. Il propose une modélisation et une résolution par multiflot pour ce problème. La méthode choisie pour le résoudre est basée sur la relaxation lagrangienne. Les résultats pour cette méthode et leur analyse concluent ce chapitre.

Pour finir, nous présentons les conclusions sur les travaux réalisés et nous discutons des directions que le projet pourrait prendre. Plusieurs possibilités d'extension peuvent être envisagées.

Chapitre 1

État de l'art

1.1 Introduction

La volonté de rationaliser les processus de production est un principe fondateur de la révolution industrielle. Parallèlement à la nécessité de répondre aux besoins du marché (demandes) l'accent est mis sur la maitrise de chaque élément du système (machines, stock, fournisseurs) de manière à mettre en place un plan de production le plus efficace possible. Une planification de production englobe l'acquisition des ressources et des matières premières jusqu'à la planification la plus économique possible des activités de production pour la transformation de celles-ci en produits finaux. [82].

Un des premiers travaux publiés dans ce domaine date de 1913, dans lequel Harris [60] suggère que la fabrication en lots de plusieurs produits peut réduire les coûts globaux de production. Les premières années du XX^{ème} siècle correspondent à l'expansion de l'industrialisation dans le monde, où la main-d'œuvre est progressivement remplacée par les machines qui produisent en plus grande quantité. Une des conséquences de ce changement est que les habitudes de consommation de la population urbaine des pays industrialisés ont changé aussi. Ce changement demande aux industries de s'adapter et d'améliorer leur efficacité.

Dans son travail, Harris introduit des termes tels que « *setup costs* » et « *economical size of lots* », que nous retrouvons encore dans les modèles actuels. Il affirme que les intérêts sur le capital immobilisé dans les salaires, le matériel et les frais généraux fixent une limite maximum à la quantité de pièces qui peuvent être fabriquées profitablement en une seule fois, ce qui définit le problème comme le rapport entre la minimisation des coûts et la maximisation de la production.

De nos jours la mondialisation des marchés et les besoins étant de plus en plus exigeants, cela oblige les industries à réorganiser la production de façon optimale afin d'obtenir une meilleure rentabilité. Les industries et les entreprises doivent fournir des planifications de production adaptées aux exigences des clients de manière à atteindre leurs objectifs. Ces dernières années, beaucoup d'efforts ont été faits dans l'industrie afin de réduire la complexité du processus de production et de maîtriser

les temps et les coûts de démarrage.

Le niveau décisionnel responsable de la gestion de ce processus s'appelle la « planification et contrôle de production » (PCP). Il coordonne toutes les activités, de l'acquisition des matières premières à la livraison des produits finaux. La structure hiérarchique d'un système PCP est divisée en trois niveaux de planification distincts : stratégique, tactique et opérationnel. Le premier est associé au plus haut niveau de prise de décisions, où sont définis les buts globaux d'une entreprise et les politiques appropriées pour les atteindre, en déterminant les objectifs à long terme. Le deuxième est responsable de l'implantation des stratégies définies dans le premier niveau, de manière à utiliser efficacement les ressources disponibles, en prenant les décisions à moyen terme. Enfin, la planification opérationnelle, traite les décisions de court terme, prenant en compte l'objectif à exécuter selon les plans définis précédemment [31].

Dans ce contexte le problème de dimensionnement de lots (*Lot-Sizing Problem* - LSP) joue un rôle essentiel. Selon [89], le terme « *lot-sizing* » vient d'un environnement de production discret, considéré ici comme l'organisation des demandes d'un même produit à différentes périodes pour un même lot de production. Ainsi les demandes de clients possédant des dates de livraisons différentes par produit sont regroupées pour former une seule demande de production. Dans la littérature cette opération concerne les problèmes de prise de décision liés à la planification tactique [25]. La planification de production à ce niveau consiste à déterminer un plan qui prévoit combien produire et/ou acheter pour les prochaines périodes de temps appelé ici « horizon de planification ». Le problème prend aussi en compte la détermination des niveaux de stock, des ressources nécessaires, le temps de fabrication nécessaires pour chaque lot produit, la machine et le site de fabrication du produit, la séquence des lots, ainsi que d'autres caractéristiques nécessaires pour exécuter le plan.

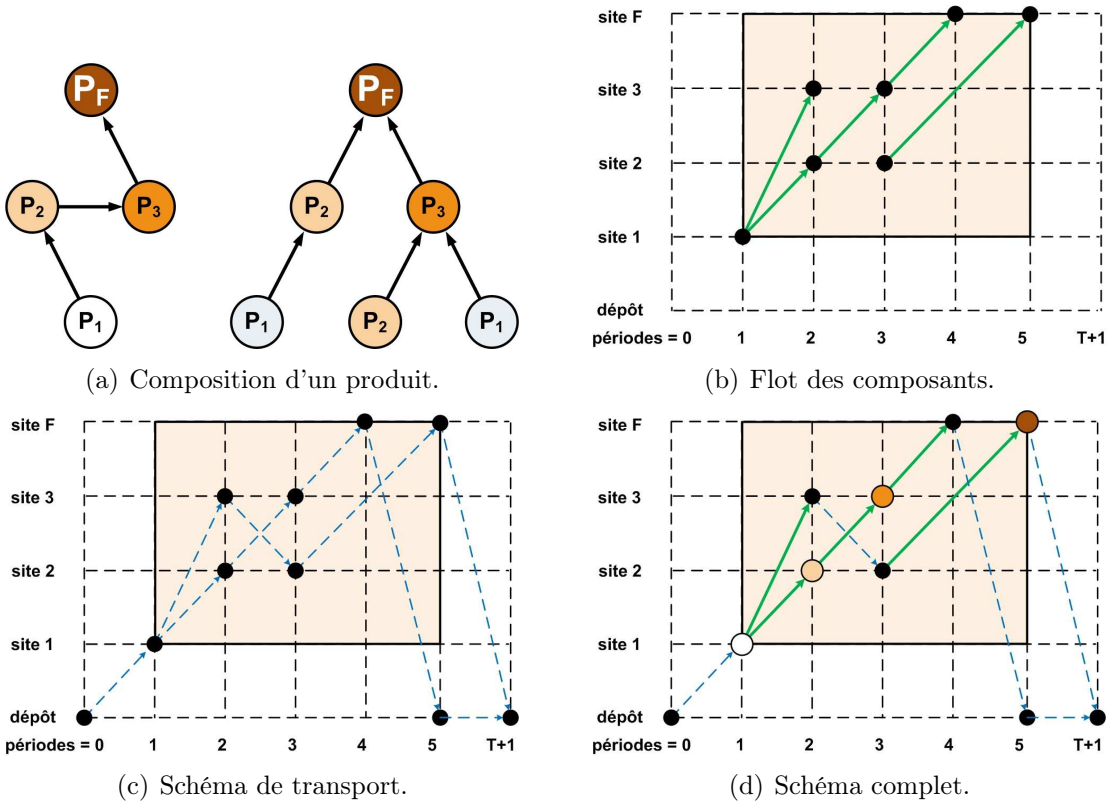
L'objectif de ce chapitre est de présenter un aperçu général du LSP qui permettra la compréhension des variantes du problème, en particulier le couplage avec les problèmes de routage, qui est le sujet de cette thèse. Aussi, ce chapitre se compose de quatre parties : la première donne la motivation de cette thèse concernant l'étude d'un modèle de production et de transport et son rôle dans une chaîne logistique ; la deuxième décrit les caractéristiques du *lot-sizing*, en détaillant chaque composant, et donne un état des lieux de plusieurs modèles depuis Harris [60] jusqu'aux modèles actuels, tout en montrant leurs évolutions et leurs ramifications qui conduisent au modèle présenté dans ce travail ; la troisième décrit le problème de tournées de véhicules et plusieurs modèles ; la dernière partie présente les méthodes les plus utilisées pour résoudre les problèmes de production avec ou sans transport.

1.2 Motivation

Le problème que nous étudions ici, et que nous appelons de « *Lot-Sizing with Vehicle Routing Problem* » (LSVRP), concerne la fabrication de produits et leur transport entre différents sites de fabrication. Il s'agit d'un couplage entre les problèmes de production de lots (*Lot-Sizing Problem* - LSP) et de tournées de véhicules (*Vehicule Routing Problem* - VRP). Nous décrivons le premier comme : minimiser les coûts de planification de fabrication de produits, sur plusieurs périodes, en utilisant une machine de capacité M et en respectant les contraintes de production, de stockage et de satisfaction de la demande. Ici, nous considérons un système multi-niveau de production pour lequel plusieurs composants sont considérés. Ces composants sont fabriqués sur différents sites et ils sont nécessaires pour la fabrication d'un produit final. Le problème intègre également la livraison des composants en plus de la planification de production. Dans ce contexte, nous considérons le second problème comme un VRP. Il consiste à définir une tournée pour chaque véhicule de manière à minimiser les coûts de transport, en respectant les contraintes de capacité des véhicules, de durée de transport et de flux d'entrée/sortie des produits dans chaque site.

La figure (1.1) montre la structure du problème. Pour illustrer le LSVRP, nous travaillons dans un graphe temporel. L'axe x représente un horizon de planification de $T + 1$ périodes et commençant par 0. L'axe y quant à lui représente les sites de fabrication ainsi que le dépôt. Les nomenclatures utilisées, c'est-à-dire la structure d'un produit, sont de type série et assemblage (détaillées plus loin dans ce chapitre). Elles sont représentées dans la figure (1.1(a)) (série à gauche et assemblage à droite) où le produit final, P_F, est le résultat de la composition des produits P_1, P_2 et P_3. Le flot d'entrée et sortie des produits composants dans les sites sont exprimés dans la figure (1.1(b)), où les flèches en traits pleins représentent les composants qui sortent d'un site i, à la période t, vers un site j, à la période t' (les cercles noirs). Donc, pour respecter ce flot, une flotte de véhicules doit sortir du dépôt au début de la planification, accomplir les tâches de livraison, et retourner au dépôt à la fin de l'horizon. L'illustration (1.1(c)) exprime ce processus. Ici les flèches pointillées représentent les déplacements des véhicules. La figure (1.1(d)) montre un schéma complet de production (cercles plus grands) et de transport pour un problème avec 5 périodes de temps et 4 sites de production. Ici nous supposons que le délai de transport est supérieur à celui de la production.

La raison qui nous a motivé à travailler avec ce couplage de problèmes est que dans une chaîne logistique, les coûts de transport représentent des montants significatifs dans le coût global. Selon [73], la délocalisation des centres de production dans les pays où la main d'œuvre est moins chère a engendré des coûts de transport extrêmement onéreux. Ainsi, depuis les années 1970 l'intégration du transport dans la production a été prise en compte d'une manière plus sérieuse, et plusieurs modèles et approches de résolution ont été développés pour traiter ce couple de problèmes.



(a) Composition d'un produit.

(b) Flot des composants.

(c) Schéma de transport.

(d) Schéma complet.

FIGURE 1.1 – Processus de production, envoi et consommation des composants.

Par la suite, nous allons décrire quelques travaux qui ont été réalisés.

Un des premiers travaux sur ce sujet se trouve dans l'article de Glover *et al.* [51], où les auteurs ont développé un système qui intègre production, distribution et stockage pour une entreprise de produits chimiques. Le système prend en compte plusieurs questions sur les aspects du problème qui ont été nécessaires pour une modélisation plus réaliste. Le but a été de concevoir un système global basé sur un réseau bien adapté aux nécessités de l'entreprise et capable de réduire considérablement les coûts de transport.

Une question courante dans les modèles intégrés de transport et production est la mesure de temps correspondant à chaque problème. Hahm et Yano [59] insèrent dans la littérature l'*Economic Lot and Delivery Scheduling Problem* (ELDSP). Il reprend le modèle du problème *Economic Lot Scheduling* et ajoute le problème de livraison. Ils montrent que les périodes de production (T_p) peuvent être un multiple de l'intervalle de livraison (T_l), c'est à dire $T_p = M \times T_l$. Dans cet article, les auteurs proposent une manière de déterminer la fréquence de production d'un composant et la fréquence de livraison de ce composant vers un client qui le consomme à taux constant. Les modèles proposés permettent de calculer la taille des périodes et la quantité de production, ainsi que la réduction des coûts de démarrage et du temps de reconfiguration de machine (*setup costs* et *setup times*), pour atteindre une meilleure synchronisation entre la fabrication et la livraison.

Chandra et Fisher [23] considèrent un scénario particulier où un site fabrique un certain nombre de produits, et stocke les produits finis pour ensuite les livrer en utilisant une flotte de véhicules. Les demandes pour chaque produit sont connues à chaque période. Pour résoudre ce problème, les auteurs supposent que les opérations de fabrication utilisent le modèle de *lot-sizing* avec une capacité limitée de ressources. Les opérations de tournées reposent sur des modèles de livraisons locales avec périodes multiples. Ils utilisent deux approches, une découplée, où chaque problème est résolu séparément par des heuristiques classiques, et une autre couplée, utilisant une heuristique d'amélioration locale. Les résultats servent alors de support pour déterminer sous quelles conditions des changements opérationnels doivent être réalisés.

Lei *et al.* [69] considèrent un seul produit et un ensemble de sites, chacun avec ses caractéristiques, une flotte de véhicules hétérogènes associée à chaque site et un ensemble de centres de demandes de clients distants. Le problème consiste à coordonner les trois aspects du problème en satisfaisant la demande de manière à minimiser les coûts. Les auteurs ont développé une méthode en deux étapes pour résoudre le modèle couplé. La première étape consiste à résoudre une version du problème qui considère des contraintes d'expédition directe entre le producteur et le consommateur. Le résultat optimal du sous-problème donne une borne supérieure au problème complet et fournit la quantité à produire, stocker et transporter. Il indique aussi le nombre de trajets possibles par période. La deuxième étape consiste à résoudre le problème de transport, sans contraintes d'expédition directe, en utilisant une heuristique. La méthode supprime les trajets moins intéressants et affecte les meilleurs aux transporteurs. Cette approche essaye d'optimiser le problème de production et de transport simultanément.

Plusieurs méthodes ont été développées ces dernières années sur des couplages production/transport. Bard et Nananukul [8] utilisent une approche similaire à Lei *et al.* [69]. Ils résolvent le problème dans le scénario où un site de fabrication doit satisfaire les demandes d'un ensemble de clients. Pour faire la livraison, le modèle dispose d'une flotte de véhicules. La méthode de résolution consiste en une recherche tabou réactive pour résoudre le problème complet, suivie d'un algorithme de recomposition de chemins (*path relinking*) pour améliorer les résultats.

Les travaux récents de Boudia, Louly et Prins, que nous allons voir en suite, proposent de traiter des méthodes heuristiques et métaheuristiques pour le problème couplé et découplé de production/transport où le but est de minimiser les coûts totaux. Dans [15], ils considèrent qu'un site de production doit satisfaire la demande de plusieurs clients. Une flotte homogène de véhicules de capacité limitée est utilisée pour faire la livraison. Il est supposé aussi que le site de production ainsi que les clients peuvent stocker les produits. La méthode de résolution consiste en une heuristique gloutonne où une extension de l'algorithme de Wagner-Whitin [97] est utilisée pour traiter la composante production. L'algorithme de Clarke-Wright [24] est utilisé pour la partie transport. Cette heuristique suit trois étapes : détermi-

ner la quantité à livrer à chaque période pour chaque client ; établir les tournées de livraison ; déterminer les dates de production des quantités à livrer. Les auteurs ont également développé une méthode découplée, c'est-à-dire en résolvant les problèmes séparément, utilisant les mêmes heuristiques pour comparer les résultats. La conclusion est que l'optimisation d'une façon globale amène des meilleurs résultats.

Nous présentons dans la suite quelques modèles de *lot-sizing* en détaillant les caractéristiques principales de chacun d'eux.

1.3 Modèles de type *Lot-Sizing*

1.3.1 Définitions et caractéristiques

Le problème original de *Lot-Sizing* (LPS) a dérivé plusieurs variantes, celles-ci sont de plus en plus complexes à cause de diverses contraintes qui sont venues les enrichir. Ces contraintes visent à rendre le modèle plus réaliste et par conséquent cela limite l'espace des solutions originelles. Pour faciliter la compréhension du problème, nous présentons quelques définitions et caractéristiques. Fatemi *et al.* [43], Suerie [89] et Nascimento *et al.* [78] présentent des synthèses avec les concepts de base du LSP. Nous avons choisi ici une subdivision en quatre groupes de caractéristiques : trois basées sur celle proposée par [89], qui sont les « ressources », les « produits » et le « temps », et nous avons ajouté une quatrième les « objectifs ».

1.3.1.1 Modélisation des ressources

Le premier type d'attributs concerne les ressources utilisées dans le processus de production. Celles-ci peuvent être renouvelables, non renouvelables ou partiellement renouvelables.

Capacité : cet attribut limite la production par période de temps. Le fonctionnement des machines, la disponibilité de la main-d'œuvre, les heures de travail, etc., sont des exemples de capacité. La capacité de production peut être finie ou infinie dans une période, ce qui induit des problèmes à capacité limitée (*capacitated*) ou non limitée (*uncapacitated*). Elle influe fortement sur la complexité du problème, ceux avec capacité étant de complexité NP-Difficile selon l'étude fait par [14].

Attribution des produits et ressources : les ressources et produits sont liés par la consommation de plusieurs façons. Par exemple, dans une production dite dédiée, chaque produit a l'exclusivité d'une ressource (tâche unique). Dans le cas où un site fabrique plusieurs produits, ces ressources sont normalement partagées (et utilisées à la fois par chaque produit) et il faut prendre en considération des coûts additionnels pour les opérations de changement de type de production.

Nombre de ressources : concerne l'utilisation des ressources disponibles (machines, main d'œuvre, etc.) pour le processus de fabrication. Si une opération utilise plus d'une ressource en même temps, nous disons que ce sont des ressources parallèles. De même, si l'opération est faite sur des ressources déjà parallèles, nous caractérisons le problème comme multi-ressources.

Structure de production et d'opération : elle désigne le flot de matériaux dans le système de production. Il peut être cyclique si le processus revient à une deuxième fabrication d'un type de produit, ou au contraire il est considéré acyclique.

Taux minimal d'utilisation : il est nécessaire pour éviter le gaspillage des ressources dans une production surtout lorsque nous considérons une fabrication à échelle réduite. Dans certains cas il est plus raisonnable de reporter la production à une autre période pour économiser l'utilisation de machines, énergie, matériaux, etc., ce qui concerne le principe même d'un système de production.

Opérations de *setup* : ces opérations sont essentiellement présentes dans les problèmes sous la forme d'une durée et/ou d'un coût qui concerne le processus de démarrage et d'ajustement de la production. Les *setups* sont fortement liés aux ressources, dans le sens où ces opérations changent leur l'état. Du point de vue de la complexité, les opérations d'ajustement (ou de reconfiguration) augmentent la difficulté du problème car elles réduisent en partie la capacité de production. Elles doivent donc être réduites au minimum possible [91]. Nous définissons deux types de structure de *setup* : simple et complexe. La première est indépendante de la séquence et des décisions prises aux périodes précédentes. Au contraire, la seconde dépend de l'historique. Nous subdivisons les *setups* complexes en quatre catégories : les *setups* de transition (*setup carry-over* [58]) permettent de continuer la production de la période précédente sans coûts de *setup* additionnels ; les *setups* de classe (*major setups*) sont induits par des similitudes dans le processus et le design d'un ensemble de produits alors que les *setup* de produit (*minor setups*) ont lieu quand il y a changement de production entre produits d'une même famille [83] ; enfin, le *setup* dépendant de la séquence (*sequence dependent setups* [84]) arrive lorsque les coûts de *setup* et des reconfigurations dépendantes des séquences de production [43].

Opérations de *start-up* : une opération de *start-up* se produit au début d'une production lorsqu'une machine commence à fonctionner. C'est le premier démarrage selon une configuration de machine spécifique. Elle se répète dès qu'arrive une commutation pour la fabrication d'un autre produit. Selon Constantino [26] la différence entre *start-up* et *setup* réside dans le fait que le premier est une opération qui aura lieu chaque fois que la machine produit, alors que les *setup* sont des ajustements initiaux à chaque période.

1.3.1.2 Modélisation des produits

Nous présentons ensuite des attributs correspondants au processus de fabrication des produits.

Demande : c'est une quantité à respecter dans une production. Elle peut être dynamique ou statique, selon que la demande varie (ou pas) dans la période de production. Si la valeur est connue à l'avance, la demande est déterministe. Si elle est inconnue et doit être fondée sur des probabilités, la demande est stochastique [77].

Nombre de produits : c'est un attribut important qui permet de dire si un problème est à production unitaire (ou mono-produit, quand un seul produit est fabriqué dans le système de production) ou multiple quand plusieurs produits sont considérés. En termes de complexité, la multi-production induit une complexité beaucoup plus élevée.

Nombre de niveaux : un problème de type *lot-sizing* (LSP) prend en compte un ou plusieurs niveaux de planification en fonction de l'interdépendance entre les produits. Une production est mono-niveau (*single-level*) quand elle concerne un produit qui ne dépend pas des autres produits, c'est-à-dire quand la matière première est transformée directement en produit final sans sous-assemblage intermédiaire. C'est une production « indépendante ». Une production multi-niveaux (*multi-level*) considère que les produits finis sont fabriqués à partir d'opérations sur composants. Ainsi la production en sortie devient l'entrée d'une nouvelle étape de la production. Dans ce cas, c'est une production « dépendante ». Les systèmes à multi-niveaux sont plus difficiles à traiter et peuvent être classifiés en fonction de leur structure de production [43], comme nous allons le voir en détails.

Nomenclature (*Bill of materials* - BOM) : elle concerne la manière dont les matières premières et les composants sont composés (ou décomposés) pour générer les produits finaux [66]. Elle se rapporte à la structure de la production. Le BOM peut se décrire simplement comme la liste qui indique le nombre de composants (et sous-composants) et leurs quantités nécessaires pour fabriquer un produit [90]. Un BOM comprend différents types de structures concernant la composition d'un produit. Ces dernières sont : structure convergente (assemblage), structure divergente, structure de série et structure générale.
La figure (1.2) est tirée de [89]. Elle montre ce type de relations dans un exemple avec 4 niveaux d'assemblage. Dans la structure d'assemblage, à partir du deuxième niveau, chaque produit est issu d'un assemblage de produits de niveaux antérieurs. Par contre chaque produit a seulement un successeur dans la chaîne de production. La structure divergente exprime qu'un produit d'entrée peut donner plusieurs produits de sortie [66]. Cependant il est issu uniquement d'un seul produit prédécesseur. La structure en série montre un cas spécifique d'héritage entre une structure d'assemblage et divergente, où un

produit a un seul successeur et prédécesseur (par exemple le traitement par étapes d'un produit jusqu'à sa finalisation). La dernière structure est une généralisation des autres structures, en prenant en compte leurs caractéristiques.

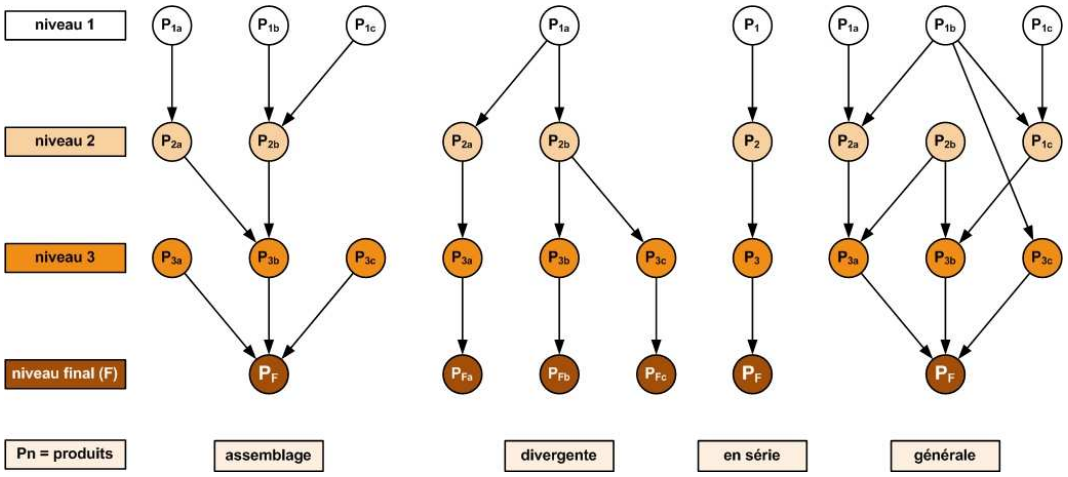


FIGURE 1.2 – Types de nomenclature

La bonne conception d'un BOM est essentielle pour la production parce qu'une seule erreur compromet le plan de production.

Processus d'approvisionnement : c'est la supposition qu'à chaque période de temps un produit devient disponible pour une nouvelle étape après avoir été traité dans une étape précédente. Nous considérons que chaque produit simple sera soit disponible tout de suite pour une prochaine étape de traitement, soit seulement si la totalité du lot est terminée.

Temps de traitement (*lead time*) : la quantité de temps requis pour réaliser une tâche (de production de lot, par exemple) et est défini comme endogène ou exogène. C'est-à-dire qu'il dépend de facteurs internes ou externes à la production. En ce qui concerne le premier type, il est inclus dans le processus de production (conséquence de la modélisation), alors que le deuxième est donné par les circonstances extérieures au processus de production, comme par exemple le temps de livraison d'un composant sur un site lors d'une fabrication de type multi-niveaux avec la structure d'assemblage.

Contraintes de stockage : quelques contraintes de limite maximum et minimum sont ajoutées pour gérer l'inventaire de fabrication. Elles représentent, par exemple, une capacité de stockage dans un entrepôt ou la réserve de sécurité nécessaire pour démarrer une production.

Politiques de service : elles concernent le respect d'une demande qui est connue à priori. Il y a des modèles où une demande peut être perdue (perte de ventes - *lost sales*) ou retardée (*backlogging*), et d'autres où il y a des coûts de pénalité pour les demandes non satisfaites (*shortage costs*). La non satisfaction

de la demande (retards, par exemple) est justifiée par des raisons économiques comme des coûts de *setup* plus élevés que les coûts de pénurie [25], ou des situations où la capacité de production n'est pas suffisante pour respecter ces demandes. L'enjeu ici est de décider dans quelles mesures les demandes doivent être satisfaites.

Règles supplémentaires : nous pouvons appliquer des règles supplémentaires aux modèles de *lot-sizing*, comme, par exemple, des contraintes de limites de ressources, de stockage, de durée de production, de reconfiguration de machine, etc.

1.3.1.3 Modélisation du temps

Quand nous traitons un problème de planification de production, nous nous intéressons à la durée d'exécution de cette planification. Les différents problèmes de *lot-sizing* traitent cet aspect de différentes manières selon la situation, ce qui nous conduit à séparer en deux subdivisions basiques : plans à court terme et à moyen terme. Les plans de production à moyen terme sont basés sur une échelle de temps discrète composée de semaines ou mois avec peu de détails de micro-gestion. Dans ceux à court terme, les plans opérationnels ont besoin de plus de détails et ont une échelle de temps beaucoup plus fine, voire se déroulent d'une manière continue, c'est-à-dire, sans discrétisation en périodes de temps [89].

La structure de temps utilisée par les modèles est définie par certains éléments :

Horizon de planification : l'horizon de planification est le temps durant lequel la production doit être réalisée. Ce temps peut être fini ou infini, avec des grandes ou petites périodes de temps, nommées « *big-buckets periods* » (BBP) ou « *small-buckets periods* » (SBP).

Essentiellement, une BBP permet la production de plusieurs produits dans une longue période (comme par exemple, une semaine avec un horizon de six mois). Il est approprié pour des problèmes où il existe seulement des coûts de *setup* et de reconfiguration de machine [13]. Il supprime aussi le problème de déclaration de séquence de demandes s'il y a fabrication de plusieurs produits dans une même période. Pour le SBP, le temps de production est court (par exemple, heures ou demi-journées) et mène à la fabrication d'un ou deux produits par période [12]. Les SBP permettent d'ajuster les décisions selon la demande connue et sont appropriées pour les situations où il est nécessaire de contrôler précisément la capacité consommée par les *setup*. Les BBP sont utilisés pour les modèles de *lot-sizing* où, normalement, se prennent des décisions fondées sur l'ensemble [4, 89, 78]. Dans les BBP il y a aussi la possibilité d'éviter les *setups* si un même produit est fabriqué à la fin d'une période et au début d'une autre [13].

Dans la pratique le nombre de périodes dans les SBP est souvent très grand, ce qui pose des problèmes pour les méthodes mathématiques, mais qui peuvent être raisonnablement traités par des heuristiques [37]. Belvaux et Wolsey [12] ont proposé une classification des problèmes de production de mono-niveau en prenant les BBP et les SBP comme deux grands groupes de base. Cette classification est présentée dans la figure (1.3). Les BBP sont séparés en problèmes avec reconfiguration de machine *setup times* et de séquence de production dépendante des coûts de changement. Les période de type *small-buckets* (SBP) sont séparés en problèmes avec la fabrication de 1 ou 2 produits par période. Une sous-division des SBP concerne la capacité des machines.

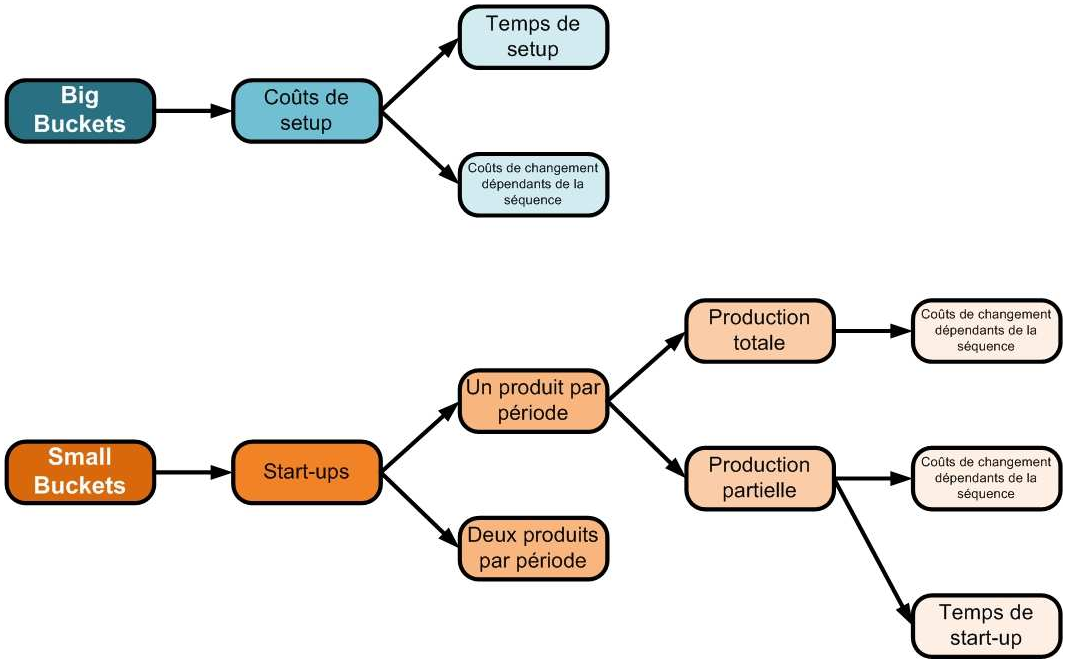


FIGURE 1.3 – Classification des problèmes par rapport à la taille des périodes [12].

Échelle de temps : elle est définie comme continue ou discrète. Pour une échelle de temps de type discrète, le temps est divisé en périodes qui peuvent être grandes ou petites, uniformes ou non-uniformes. La plupart des modèles de problèmes *lot-sizing* utilisent les échelles de temps discrètes avec périodes uniformes, c'est-à-dire de même taille. Parmi les modèles utilisant une échelle de temps continue l'*Economic Lot Scheduling Problem* (ELSP) sera plus détaillé dans la section suivante.

Développement temporel des paramètres et données : les paramètres comme la capacité, les coûts, etc., peuvent varier avec le temps. Un modèle de LSP peut être décrit comme dynamique ou statique selon la variabilité temporelle des demandes.

Disponibilité et connaissance des paramètres et données : nous classons les modèles comme déterministes ou stochastiques par rapport à la disponibilité et

la connaissance des données. Dans le premier type, les paramètres et les données sont connus à l'avance par le système. Dans le cas stochastique, le modèle essaie de prendre en compte l'incertitude dans le type de planification.

1.3.1.4 Modélisation des objectifs

Le dernier groupe de caractéristiques concerne les objectifs du problème. Ceux-ci varient selon le modèle et sont normalement attribués à la minimisation des coûts de production, stock, transport, etc., et des facteurs liés à la demande.

Fonction objectif : bien que dans la plupart des cas l'objectif est de minimiser les coûts de production, la fonction objectif peut aussi intégrer d'autres critères tels que les bénéfices ou les ventes. Le problème peut aussi prendre en compte le niveau des ressources disponibles (objectif de la performance physique de la production) ou la minimisation du retard maximum ou le temps de conclusion du processus (objectif temporel) par exemple.

Coûts composants : par défaut, le problème vise à minimiser les coûts dépendant de plusieurs facteurs. Les plus fréquents sont les coûts de production unitaires, de stockage et de *setup*. Les deux premiers sont habituellement des fonctions linéaires de la quantité de produits qui sont fabriqués ou stockés à chaque période de temps. Les coûts de *setup* sont dérivés du processus de production et correspondent aux démarrages de la machine. Plusieurs modèles associent le *setup* à un temps de reconfiguration de machine (*setup times*), et le coût de cette opération est la consommation de la capacité de production. D'autres coûts sont associés selon la modélisation, comme par exemple les coûts de retard pour la demande (*backlogging*), et pour les ventes perdues (*lost sales*), expliqué précédemment, coûts de préservation de l'état de *setup* même s'il n'y a plus de production (*reservation costs*), coûts de démarrage de production (*start-up costs*), coûts d'ajustement de machine à la fin du processus (*swich-off costs*), etc. Une analyse concernant les coûts de *setup* et d'autres coûts associés est présentée en détail dans [89].

1.3.2 Modèles de *lot-sizing*

Depuis la publication de Harris [60], où il présentait le modèle *Economic order quantity* (EOQ), de nombreux travaux avec différents modèles et approches de résolution pour les problèmes de *lot-sizing* (LSP) ont été faits. Ces modèles sont classifiés en fonction de différents types d'attributs et leur large diversité est expliquée par la complexité de modéliser un système physique : certains objectifs de planification ont besoin d'une modélisation détaillée alors que d'autres ont besoin uniquement d'informations basiques [25]. Nous détaillons dans la suite quelques modèles.

1.3.2.1 *Economic order quantity* (EOQ)

Le travail de Harris [60] a été conçu dans la période d'expansion de la révolution industrielle, où les machines ont été utilisées pour accélérer le processus de fabrication en augmentant le volume de biens de consommation. Le modèle EOQ propose de déterminer la taille la plus économique d'un lot de fabrication en suivant des facteurs impliqués : les coûts de production unitaire (P), d'ajustement de machine (*setup*, Q), et de stockage (H). L'objectif du problème est de déterminer la taille (x) du lot et la quantité de circulation de stock (s) dans chaque intervalle (t) d'un mois, de manière à minimiser le coût total (Z).

La méthode de résolution tient compte de l'expression suivante pour le coût total :

$$Z = \frac{1}{240s}(Px + Q) + \frac{Q}{x} + P \tag{1.1}$$

où $\frac{1}{240s}$ représente l'intérêt annuel et le coût de dépréciation pour $12s$ unités utilisées par année.

L'analyse de la solution est faite à travers des graphes qui représentent l'évolution des coûts par rapport à la taille de l'ordre de fabrication. L'étude ne comprend pas certains aspects réels du problème. Ce travail est devenu un paradigme dominant pour les analyses d'ordres de quantité pendant les 40 années suivantes. Un historique de ce travail, avec sa trajectoire et ses dérivées directes est présenté dans l'article de Erlenkotter [39].

1.3.2.2 *Economic lot and scheduling problem* (ELSP)

L'ELSP est aussi considéré comme un des premiers travaux dans le domaine de la planification de production. D'après [38] ce problème était présent depuis 1915, mais son étude a débuté formellement dans les années 50. Dans [85], le problème est présenté sous une forme généralisée intégrant la multi-production et les contraintes de capacité. Les ressources sont limitées et partagées entre produits indépendants (production mono-niveau), les demandes sont statiques (ne changent pas au fil du temps) et l'horizon de planification est infini et continu (il n'y a pas de division d'horizon en périodes). Le modèle comprend les coûts de *setup*, de reconfiguration, et les coûts de stockage.

Pour illustrer le problème, supposons le cas où deux (ou plusieurs) produits ont besoin des ressources du site de production. La question qui se pose est de savoir comment procéder pour minimiser le coût de la planification résultante si nous souhaitons maintenir un schéma cyclique de production basée sur la quantité économique de fabrication (*economic manufacturing quantity* - EMQ) [38]. Ce problème a été bien étudié parce qu'il capture plusieurs caractéristiques importantes qui sont fréquemment retrouvées dans les problèmes réels [101]. Il est NP-Difficile.

Plusieurs approches, modèles et méthodes de résolution ont été étudiés, dont nous citons [36, 101, 37]. Des modèles étendus basés sur ELSP ont aussi été développés,

comme par exemple dans [59] (présenté de manière plus détaillé à la section 1.2) qui reprend les idées principales du ELSP et fait un couplage avec le problème de livraison.

1.3.2.3 *Economic lot-sizing model* (ELS)

Ce modèle est une autre approche pour l'*economic order quantity* (EOQ). Les travaux de Wagner-Whitin [97] et Manne [72], dans les années 50, introduisent le problème d'une façon discrète où nous supposons un plan de production fini sous-divisé en plusieurs périodes. Les demandes sont données pour chaque période et varient au travers du temps (demandes dynamiques), ce qui ajoute la caractéristique de taille dynamique de lot (*dynamic lot-sizing*) au problème. Ici la capacité de production n'est pas considérée, et le problème est mono-niveau avec un seul produit (mono-produit) [37]. L'objectif de ce problème est de déterminer combien et à quelle période devrions-nous produire de manière à minimiser les coûts de fabrication, les *setups* et les stockages tout en respectant les demandes. L'algorithme de Wagner-Whitin est polynomial et se résout en $O(T^2)$. D'autres approches ont été proposées, comme par exemple les travaux de Wagelmans-Hoesel-Kolen [96] et Federgruen-Tzur [44] où sont présentés des algorithmes de complexité en $O(Tlog(T))$ ou même de $O(T)$ pour certaines situations.

1.3.2.4 *Uncapacitated lot-sizing problem* (ULS)

C'est un des modèles les plus basiques pour le problème *lot-sizing* (LSP) et en dépit de ça, plusieurs articles, livres et périodiques ont été publiés sur le sujet, ainsi que des extensions intéressantes sur ce modèle.

Dans les travaux de Wolsey et Pochet-Wolsey [98, 99, 82] est présentée une récapitulation détaillée de ce problème avec un et plusieurs produits. Nous trouverons aussi une extension du modèle où les coûts de production et de stockage sont agrégés en une seule variable. Donc, le problème est reformulé en soulevant la question « quelle quantité produire dans la période t pour satisfaire une demande à la période t' ? ». Les modèles donnent la même valeur pour la fonction objectif, par contre la distance entre la solution du problème entier et celle du problème relaxé est inférieure dans le deuxième modèle, ce qui montre que son espace de solution est plus délimité. D'autres modifications et extensions pertinentes du modèle et des méthodes de résolution significatives sont aussi détaillées.

L'article de Brahimi *et al.* [20] fait aussi une récapitulation et une description des problèmes de *lot-sizing* mono-produit. Deux remarques importantes extraites de cet article sont :

1) il existe des industries où il est possible d'agréger des produits pour obtenir un seul produit et où la capacité n'est pas une grande préoccupation, comme par exemple le cas où la différence entre produits est la couleur ;

2) plusieurs approches de résolution des problèmes complexes de *lot-sizing* amènent à des sous-problèmes impliquant ULS, comme l'application de relaxation lagrangienne pour le LSP avec capacité (CLSP) et la génération des colonnes pour le LSP discret (DLSP) (la section suivante présente ces méthodes de résolution).

Smith [87], définit le modèle ULS en considérant qu'une demande probable doit être trouvée en un certain pourcentage à n'importe quelle période. L'auteur appelle ce modèle « ULS avec contraintes probabilistes » (*uncapacitated lot-sizing problem with probabilistic constraints* - ULSP), et utilise un logiciel d'optimisation pour le résoudre. Le modèle permet aussi l'utilisation de *backlogging*. Pour réduire le temps de processus, le modèle est divisé en utilisant une décomposition de Benders et aussi une recherche des possibles inégalités valides pour définir une région de faisabilité.

Un autre travail intéressant traite de la planification de production d'un seul produit dans plusieurs centres de production (*lot-sizing problems with several production centers*) [92]. Le modèle ne prend pas en compte les contraintes de capacité et permet l'utilisation de *backlogging*. L'article présente une modélisation du problème avec un ou plusieurs centres de production. L'auteur propose aussi des algorithmes pour trouver la solution et les analyses de performance.

Par la suite, nous présentons le modèle ULS et son extension avec coûts de production et stockage en une seule variable.

Étant donnée une demande (D_t) pour le produit dans une période t de temps. La fonction objectif (1.2) vise à minimiser les coûts de fabrication (P_t) d'un produit dans un horizon de T périodes ($t = 1, \ldots, T$) de même taille, en prenant en compte le *setup* (Q_t) et le stock (H_t). Les variables du modèle sont : x_t, pour la quantité fabriquée, y_t, pour le *setup*, et s_t pour le stock.

$$\min \sum_{t=1}^{T} (P_t x_t + H_t s_t + Q_t y_t) \tag{1.2}$$

Les contraintes du problème sont :

$$x_t + s_{t-1} = D_t + s_t \quad \forall t = 1, \ldots, T \tag{1.3}$$

$$x_t \leq M y_t \quad \forall t = 1, \ldots, T \tag{1.4}$$

$$x_t, s_t \geq 0, y_t \in \{0, 1\} \quad \forall t = 1, \ldots, T \tag{1.5}$$

Les contraintes (1.3) expriment la conservation de flot de stock entre les périodes. En conséquence de la minimisation, le stock au début et à la fin doit être égal à zéro. Les contraintes (1.4) font la liaison entre la production et les opérations de *setup*. Dans les modèles avec capacité, elles servent aussi à limiter la quantité fabriquée dans une période mais ici M représente une capacité « infinie ». (1.5) c'est le domaine des variables.

La formulation suivante exprime le modèle étendu du problème ULS. Dans ce modèle nous avons les coûts de production et de stockage combinés en $C_{tt'}$, qui exprime le coût pour fabriquer le produit à la période t pour une utilisation à la période t'. Cela implique que le produit est stocké jusqu'à t'. L'un des avantages de ce modèle est que son espace de solution est contenu à l'intérieur de l'espace du modèle antérieur, ce qui fournit une recherche moins profonde dans l'arbre de décision. L'une des autres avantages est que le gap d'intégralité est plus petit et peut fournir des bonnes bornes pour le problème primal. Cependant le temps de calcul augmente en raison du nombre de variables de production $w_{tt'}$ (la production en t destinée à la période t') qui passent d'$O(NT)$ à $O(N^2T)$, comme nous pouvons constater dans l'égalité (1.6) suivante où est établie une relation entre les variables de production des deux modèles :

$$x_t = \sum_{t'=t}^{T} w_{tt'} \quad \forall t = 1, \ldots, T \tag{1.6}$$

Fonction objectif :

$$\min \sum_{t=1}^{T} (\sum_{t'=t}^{T} C_{tt'}\, w_{tt'} + Q_t\, y_t) \tag{1.7}$$

Les contraintes du problème sont :

$$\sum_{t=1}^{t'} w_{tt'} = D_{t'} \quad \forall t = 1, \ldots, T \tag{1.8}$$

$$w_{tt'} \leq D_{t'}\, y_t \quad \forall t, t' = 1, \ldots, T,\ t \leq t' \tag{1.9}$$

$$w_{tt'} \geq 0, y_t \in \{0, 1\} \quad \forall t = 1, \ldots, T \tag{1.10}$$

Les coûts de production et de stockage $C_{tt'}$ sont calculés de la manière suivante :

$$\begin{cases} C_{tt} = P_t & \forall t = 1, \ldots, T \\ C_{tt'} = P_t + \sum_{t''=t}^{t'-1} H_{t''} & \forall t, t' = 1, \ldots, T, t \leq t' \\ C_{0t'} = \sum_{t'=1}^{T} H_{t'-1} & \forall t = 1, \ldots, T \end{cases}$$

La première égalité donne le coût de fabrication du produit à la période t pour satisfaire la demande à la même période (pas de stock). La deuxième combine les coûts de production à la période t aux coûts de stockage jusqu'à la période t'. La troisième donne le coût associé à un produit disponible initialement et stocké jusqu'à la période t'.

Les contraintes (1.8) expriment la satisfaction de la demande. Les contraintes

(1.9) assurent qu'il y aura une production pour chaque opération de *setup*. Elles fournissent aussi une borne supérieure de fabrication. Les contraintes (1.10) expriment le domaine des variables.

Les problèmes de type *lot-sizing* sont en grande part classifiés comme NP-Difficile. Les variations de complexité sont obtenues avec la suppression ou l'adjonction de certaines contraintes ou variables. Par exemple, un problème sans opérations de *setup* et avec des contraintes linéaires est de faisabilité P [31]. Cependant, l'ajout de ces opérations augmente la complexité, comme par exemple dans le problème avec un seule produit, ressources limitées et coûts de *setup* [47] qui est considéré comme NP-Difficile. Un autre exemple serait les problèmes NP-Difficiles qui ajoutent le temps de reconfiguration de machine (*setup times*), devenant ainsi NP-complets [71].

1.3.2.5 *Capacitated lot-sizing problem* (CLSP)

Le problème de *lot-sizing* avec capacité est un des plus connus dans la littérature. C'est une extension du problème de Wagner et Within [97] en ajoutant une contrainte de capacité. Dans le CLSP, le temps et la taille des lots sont planifiés pour la fabrication de plusieurs produits qui partagent une contrainte simple de capacité de ressources. C'est un problème d'une seule étape où aucun produit ne peut être prédécesseur d'un autre. Les coûts et demandes peuvent varier dans un horizon fini de périodes de temps discret. Le CLSP n'ajoute pas de séquence de travail et de planification dans une période, de manière à ressembler au système MRP (*materials requirements planning* - planification des besoins en composants) qui laisse les décisions des détails de la planification au système de contrôle du site [91].

Les caractéristiques principales du modèle sont les mêmes que celles de l'*uncapacitated lot-sizing* (ULS), sauf que maintenant il y a une capacité M_t disponible par période. Nous incluons le temps de reconfiguration (Z^k) et aussi la consommation de capacité pour la fabrication d'un produit k (B^k).

Le modèle mathématique pour le CLSP est défini par Trigeiro *et al.* [91] comme :

$$\min \sum_{k \in U} \sum_{t=1}^{T} (P_t^k x_t^k + H_t^k s_t^k + Q_t^k y_t^k) \tag{1.11}$$

Contraintes :

$$s_{t-1}^k + x_t^k = D_t^k + s_t^k \quad \forall k \in U, \forall t = 1, \ldots, T \tag{1.12}$$

$$\sum_{k \in U} (B^k \, x_t^k + Z^k \, y_t^k) \leq M_t \quad \forall t = 1, \ldots, T \tag{1.13}$$

$$x_t^k \leq M y_t^k \quad \forall k \in U, \forall t = 1, \ldots, T \tag{1.14}$$

$$x_t^k, s_t^k \geq 0, y_t^k \in \{0, 1\} \quad \forall k \in U, \forall t = 1, \ldots, T \tag{1.15}$$

La fonction objectif (1.11) cherche à réduire au minimum les coûts de *setup*, de production et de stockage. Les contraintes (1.12) montrent l'évolution du stock au cours du temps et aussi la conservation du flot de la production et du stock. Comme dans le modèle ULS, le stock, au début et à la fin, doit être égal à zéro. Les contraintes (1.13) expriment la consommation de capacité. Les contraintes (1.14) renforcent la liaison entre production et *setup*. Ici, M représente un nombre suffisamment grand. Enfin, les dernières contraintes (1.15) imposent que les variables soient positives et/ou binaires.

Le CLSP est NP-Difficile [37], devenant NP-Complet avec les temps de *setup* incorporés.

Ce problème a une vaste littérature. Parmi quelques travaux basés sur le CLSP, nous citons la variante avec plusieurs sites de fabrication (*multi-plant capacitated lot-sizing problem* - MPCLSP [78]), qui concerne la production avec transfert des coûts entre sites avec demandes individuelles par période. Ces coûts sont inclus parce qu'il est permis à un site de produire pour un autre site. De la même manière, un site peut stocker les produits d'un autre site. Nous sous-classifions le problème comme le MPCLSP dépendant, où les sites ont besoin d'autres sites pour produire, et le MPCLSP indépendant, dans le cas contraire. Ce problème est plutôt rencontré dans les secteurs de fabrication, tels que l'acier inoxydable, les boissons, les matelas, etc., où les sites sont à des distances géographiques considérables.

Sambasivan et Yahya [86] traitent de la planification de production avec plusieurs sites sur plusieurs périodes dans un environnement avec capacité et transfert entre sites (*multi-plant, multi-item, multi-period capacitated lot-sizing problems with inter-plant transfers*). L'étude se base sur une situation réelle avec quatre sites de production localisés en différentes parties des États-Unis. Les produits fabriqués peuvent être envoyés directement au client à partir du site où il a été produit, par contre le client devra payer uniquement pour le transport qui vient du site le plus proche où il a fait la demande. Le stock est permis ainsi que l'utilisation de demande entre les sites de production.

De nombreux travaux qui ajoutent des caractéristiques supplémentaires aux problèmes de *lot-sizing* ont été publiés. Nous citons les articles de Lee *et al.* [42], Hwang et Jaruphonga [62] et Wolsey [100] qui traitent du problème en ajoutant des fenêtres de temps (*time windows*). Ce problème considère qu'un produit doit être livré chez le client dans une période déterminée. Pendant une fenêtre de temps aucun coût de stockage ou retard n'est inclus au processus de fabrication. Wolsey [100] traite aussi le problème de fenêtre de temps de production, où au lieu d'avoir des limites de temps pour faire une livraison, le problème utilise ce concept pour la fabrication d'un produit. Dans son article, il montre deux variations du problème : avec commandes spécifiques, où chaque produit prend en compte une période spécifique pour être fabriqué ; et non spécifiques, où la fenêtre de temps exprime les entrées et les sorties des produits pendant une période (organisation de temps). Le travail de Heuvel et Wagelmans [61] montre le même problème dans un comparatif entre quatre modèles,

où les trois autres sont : le *lot-sizing* avec limites de stock (*lot-sizing with bounded inventory*), qui considère une borne inférieure et supérieure pour le stock à chaque période ; le *lot-sizing* avec l'option de remise à neuf (*lot-sizing with remanufacturing option*), qui prend en compte le stock de retour et le stock de production (neuf ou remisé) ; et le *lot-sizing* avec capacité cumulative (*lot-sizing with cumulative capacities*), où la capacité non utilisée d'une période peut être mise à profit pour la période suivante (exemples : l'argent, les matières premières. Contre-exemple : le temps).

Ensuite nous présentons une synthèse sur les problèmes de tournée de véhicules, en mettant l'accent sur les variations avec *pick-up and delievery* (collecte et livraison) et avec les fenêtres de temps, concepts sur lesquels nous nous sommes basés pour concevoir nos modèles.

1.4 Problèmes de tournée de véhicules

Les problèmes de routage sont au cœur de l'activité de logistique externe, en particulier le transport. L'objectif de base est d'attribuer un parcours aux agents transporteurs de manière à respecter des activités de collecte et/ou livraison tout en minimisant les coûts impliqués. Les agents normalement sont des entités mobiles et les activités peuvent être exprimées comme des livraisons ou retraits d'articles, personnes, données, etc.

Parmi les nombreuses applications basées sur les problèmes de routage, nous mettons l'accent sur le problème de tournée de véhicules (*Vehicle Routing Problem* - VRP) qui a été introduit à la fin des années 1950 par Dantzig et Ramser [30]. Il a comme objectif de définir une tournée pour chaque véhicule qui se déplace entre le dépôt et les points de livraison [32]. C'est un des problèmes d'optimisation combinatoire le plus étudié dans la littérature. Sa version la plus connue est le problème de tournées avec capacités. Son importance est due à sa large applicabilité, allant de la collecte des déchets à la prestation du service de courrier et de soins d'urgence [55]. L'objectif basique du problème est de minimiser le nombre total de véhicules, ainsi que le temps et la distance du parcours, en prenant en compte les contraintes spécifiques pour chaque problème.

1.4.1 Définitions et caractéristiques

Le cadre des problème de type VRP peut être définit d'un point de vue mathématique comme suit :

Soit $G = (V, E)$ un graphe orienté et complet, où $V = \{0, \ldots, N\}$ est l'ensemble des N sommets et E l'ensemble d'arcs. Étant donné un ensemble de consommateurs chacun avec une demande de collecte P_i (ou de livraison D_i), un dépôt 0 où les produits sont stockés et un nombre K de véhicules avec capacité W. En prenant en compte que chaque arc (i, j) a un coût associé non négatif I_{ij} qui représente les distances, les temps ou les coûts. L'objectif du problème est de définir un parcours

pour les véhicules de manière à ce qu'ils partent du dépôt, satisfassent les consommateurs, tout en optimisant les temps de transport, et retournent au dépôt. Pour que cette tâche soit faite, le problème doit respecter les contraintes suivantes :

- Les demandes ont une valeur non négative, avec $P_0 = D_0 = 0$;
- Chaque route commence et termine au dépôt ;
- Tout client est visité une fois et par un seul véhicule en tout point de la route ;
- La charge de chaque véhicule ne doit pas dépasser la capacité W ;
- Toutes les demandes doivent être satisfaites.

D'autres éléments contraignants sont ajoutés pour des variantes du problème, comme par exemple :

- La demande en livraison, en collecte, demande point à point ;
- L'intervalle de temps pour satisfaire une demande ;
- La distance maximale d'un déplacement ;
- La limite de sommets par route ;
- Le respect d'ordre de visite de sommets (le véhicule doit passer d'abord par i pour aller vers j) ;
- Le type de flotte de véhicules (homogène ou hétérogène) ;
- La limite de véhicules ;
- etc.

La sous-section suivante propose un survol de quelques modèles mathématiques pour le problème de tournées de véhicules et quelques variantes.

1.4.2 Modèles de problèmes de tournées

Les problèmes de type *Vehicle Routing Problem* (VRP) sont une extension du problème du voyageur de commerce (*Traveling Salesman Problem* - TSP). Le TSP est un des premiers problèmes sur les graphes à avoir été considéré formellement. Dans ce problème, un ensemble de villes doivent être visitées en totalité par un voyageur qui veut vendre ses marchandises. Les villes sont représentées par des sommets et les chemins entre les villes sont les arêtes du graphe de transport sous-jacent. Sous hypothèse de connexité, ce graphe peut être rendu complet en définissant les arcs comme le plus court chemin entre les deux sommets. Le but ici est de minimiser le coût de parcours en visitant toutes les villes au moins une fois. Il s'agit donc d'identifier un cycle hamiltonien de coût minimum.

1.4.2.1 Problème du voyageur de commerce

Le TSP est fondamental en théorie de la complexité, il est aussi l'un des 21 problèmes NP-Complet identifiés par Karp [64]. Nous présentons une formulation

pour le cas asymétrique. Nous considérons $G = (V, E)$ le graphe support où V est l'ensemble des $N = |N|$ sommets et E l'ensemble des $M = N(N-1)$ arcs. Nous avons des variables de décision $g_{ij} \in \{0, 1\}$ pour chaque arc $(i, j) \in E$, auxquelles nous associons un coût I_{ij}. Nous notons $E(S)$ l'ensemble des arcs du sous-graphe induit par les sommets $S \subseteq V$. En considérant l'inégalité du triangle, nous avons le modèle suivant :

$$\min \sum_{(i,j)\in E} I_{ij} g_{ij} \tag{1.16}$$

$$\sum_{i\in V} g_{ij} = 1 \quad \forall j \in V \tag{1.17}$$

$$\sum_{j\in V} g_{ij} = 1 \quad \forall i \in V \tag{1.18}$$

$$\sum_{(i,j)\in E(S)} g_{ij} \leq |S| - 1 \quad \forall\, S \subset V,\ 2 \leq |S| \leq \left\lfloor \frac{V}{2} \right\rfloor \tag{1.19}$$

$$g_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \tag{1.20}$$

La fonction objectif (1.16) minimise les coûts. Les contraintes (1.17) et (1.18) imposent un arc en entrée et en sortie de chaque sommet. Les contraintes (1.19) sont les contraintes d'élimination de cycles. Il existe d'autres moyens d'interdire les cycles, comme les contraintes proposées par Miller, Tucker et Zemlin (MTZ) [75] et Gavish et Graves [48]. Les contraintes (1.20) définissent le type de variables.

Le m-TSP est une variation du problème du voyageurs de commerce dans laquelle K agents de transport sont utilisés. Ils doivent sortir d'un dépôt, normalement le sommet 0, visiter les N villes et retourner au dépôt. Chaque ville est visitée par un seul agent. L'objectif (1.21) est de minimiser les coûts des routes attribuées aux K transporteurs. Le modèle est exprimé comme [5] :

$$\min \sum_{(i,j)\in E} I_{ij} g_{ij} \tag{1.21}$$

$$\sum_{j\in V} g_{0j} = K \tag{1.22}$$

$$\sum_{i\in V} g_{i0} = K \tag{1.23}$$

$$\sum_{i\in V} g_{ij} = 1 \quad \forall j \in V\backslash\{0\} \tag{1.24}$$

$$\sum_{j\in V} g_{ij} = 1 \quad \forall i \in V\backslash\{0\} \tag{1.25}$$

$$\sum_{(i,j)\in E(S)} g_{ij} \leq |S| - 1 \quad \forall\, S \subset V,\ 2 \leq |S| \leq \left\lfloor \frac{V}{2} \right\rfloor \tag{1.26}$$

$$g_{ij} \in \{0,1\} \quad \forall (i,j) \in E \tag{1.27}$$

Les contraintes (1.22) et (1.23) imposent la sortie et la rentrée des K agents au dépôt. Les contraintes (1.24) et (1.25) spécifient qu'un sommet est visité exactement par un agent (en dehors du dépôt). Les contraintes (1.26) interdisent les sous-cycles et les contraintes (1.27) définissent les variables.

Il existe beaucoup d'autres variantes du TSP, avec des contraintes qui le rapprochent des problèmes réels. Parmi les nombreuses études sur le TSP, nous citons le travail présenté par Laporte [67] qui montre une synthèse des principaux algorithmes utilisés pour la résolution de ce type de problème. Parmi eux nous pouvons citer les méthode exactes de Dantzig *et al.* [29] et Miller *et al.* [75] ; les algorithmes basés sur le *Branch-and-bound* comme Carpaneto and Toth [22], Balas and Christofides [7] et Miller and Pekny [76], et aussi les métaheuristiques comme les algorithmes génétiques [56], le *Simulated Annealing* [49], la recherche tabou [53], le GRASP [46], etc.

1.4.2.2 Problème de tournées de véhicules avec capacité

Le problème de tournées de véhicules avec capacité (*Capacitated VRP* - CVRP) concerne la planification des tournées de collecte ou de livraison sur les clients de manière à minimiser les coûts. La différence par rapport au m-TSP est la capacité des véhicules et la demande des clients. Les demandes associées à chaque véhicule doivent donc respecter sa capacité. Le graphe du problème est : $G = (V, E)$, où V correspond à l'ensemble des N clients $V' = \{1, \ldots, N\}$. De plus, nous avons un sommet de départ et un sommet d'arrivé pour les véhicules ayant nommés comme 0 et $N+1$. E est l'ensemble des arcs $(i,j) \in V$ de manière à ce que $i \neq j$, $i \neq N+1$ et $j \neq 0$, c'est-à-dire, respectivement, qu'il n'existe pas de boucle, qu'il n'y a pas d'arcs qui commencent en $N+1$ ou qui terminent en 0. A chaque client $i \in V'$ nous associons une demande D_i. La capacité des véhicules est W. Nous notons $r(S)$ le nombre minimum de véhicules pour servir un sous-ensemble S de clients, donc $r(S) = [(\sum_{i \in S} D_i)/W]$.

Une des formulations mathématique pour ce problème est celle trouvée dans [93] :

$$\min \sum_{(i,j) \in E} I_{ij}\, g_{ij} \tag{1.28}$$

$$\sum_{i=1}^{N} g_{ij} = 1 \quad \forall j = 1, \ldots, N, \forall j \neq i \tag{1.29}$$

$$\sum_{j=1}^{N} g_{ij} = 1 \quad \forall i = 1, \ldots, N, \forall i \neq j \tag{1.30}$$

$$\sum_{i \in V} g_{i0} = K \tag{1.31}$$

$$\sum_{j \in V} g_{0j} = K \tag{1.32}$$

$$\sum_{(i,j) \in E(S)} g_{ij} \leq |S| - r(S) \quad \forall S \subseteq V \backslash \{0\}, S \neq 0 \tag{1.33}$$

$$g_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \tag{1.34}$$

La fonction objectif (1.28) minimise les coûts de routage. Les contraintes (1.29) et (1.30) imposent qu'il existe un seul arc qui rentre et un seul arc qui sort de chaque sommet. Les contraintes (1.31) et (1.32) représentent la flotte de véhicules qui sortent et rentrent au dépôt. Les contraintes (1.33) éliminent les sous-tours et les contraintes (1.34) définissent le type des variables.

Il existe de nombreuses études et applications pour les problèmes de tournées de véhicules. Dans la littérature, nous pouvons trouver des bonnes synthèses comme [93, 11, 80, 81, 21] qui détaillent des modèles, des méthodes exactes et des heuristiques pour la résolution du problème.

Parmi la grande diversité des problèmes de type VRP, nous présentons de manière brève quelques variantes les plus connues dans la littérature et qui ont une liaison avec nos modèles :

VRP avec contraintes de distance (DVRP) : le problème *Distance-Constrained VRP* est une variante qui concerne le remplacement des contraintes de capacité par une limite de distance (autonomie des véhicules). Dans le cas où nous imposons une capacité, nous avons un problème DCVRP.

VRP avec fenêtres de temps (VRPTW) : dans cette version du CVRP, appelé *VRP with Time Windows*, nous imposons un intervalle de temps $[a_i, b_i]$ pour la visite des clients. Chaque client i doit être livré avant la date de fermeture b_i. Un véhicule peut arriver chez le client i avant la date d'ouverture a_i, mais doit alors attendre jusqu'à b_i.

VRP avec *Backhauls* (VRPB) : l'ensemble des clients est réparti en deux sous-ensembles : l'un de clients avec requêtes de livraison (*linehauls*), et l'autre de clients avec requêtes de collecte (*backhauls*). Pour chaque tournée le véhicule sort du dépôt chargé des requêtes de livraison, puis effectue les livraisons avant de réaliser les collectes et de retourner au dépôt. La variante du problème qui ajoute les fenêtres de temps s'appelle le VRPBTW.

VRP avec collecte et livraison (VRPPD) : dans le problème *VRP with Pick-up and Delievery* nous associons deux requêtes à chaque client : une de collecte (*pick-up*, P_i) et une autre de livraison (*delivery*, D_i). Nous imposons que toutes les livraisons soient faites avant les collectes. L'objectif est de trouver des tournées de livraisons et de collectes de coût minimum. Dans le cas où les origines et les destinations ne sont pas spécifiées, le problème est considéré comme de collecte et livraison simultanées (*VRP with Simultaneous Pick-up and Delievery* - VRPSPD). Si nous ajoutons une fenêtre de temps, nous traitons le problème

VRPPDTW.

Dial-a-Ride Problem (DARP) : ce problème est semblable au VRPPD mais dirigé au transport de personnes, raison pour laquelle il ajoute la qualité de service dans la fonction objectif [94]. Le but de ce problème est de satisfaire les demandes faites pour les passagers tout en minimisant les coûts de transport et en respectant la qualité de service. Pour cela le modèle intègre plusieurs agents mobiles et utilise des fenêtres de temps dans la caractérisation des demandes ainsi que des contraintes de durée de transport et de temps d'attente. Deux exemples d'utilisation de ce problème sont les transports scolaires et le transport pour personnes à mobilité réduite.

Ces variantes du problème de tournées de véhicules sont décrites dans la figure (1.4) [93] où nous ajoutons aussi des schémas graphiques qui représentent chaque problème. Nous faisons deux remarques par rapport aux schémas :

1) Un VRP « pur » concerne soit des collectes soit des livraisons, selon la situation qu'il modélise ;
2) A titre de simplification, nous considérons un seul véhicule dans l'illustration.

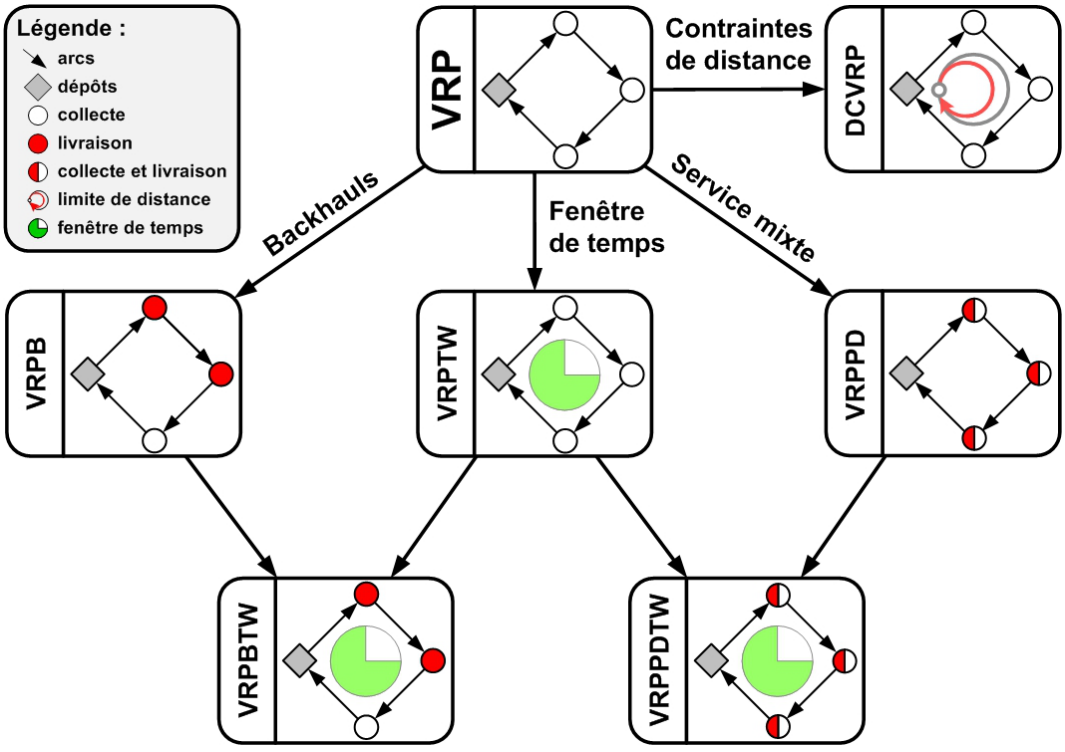


FIGURE 1.4 – Classification des problèmes par VRP avec capacité [93].

En dehors de ces variantes il en existe des nombreuses autres qui ont, d'une certaine manière, une liaison indirecte avec l'étude de cette thèse, comme par exemple l'*Heterogeneous Fleet VRP* (HVRP), le *Vehicle Fleet Mix Problem* (VFMP), le *Split*

Delivery VRP (SDVRP), le *Site-Dependant VRP* (SDVRP), le *Multi-Compartment Vehicle Routing Problem* (MC-VRP), etc. [94].

Nous présentons maintenant les méthodes de résolution les plus connues dans la littérature pour les problèmes de *lot-sizing* ainsi que ceux pour le problème de tournées de véhicules.

1.5 Méthodes de résolution

Les problèmes de *lot-sizing* (LSP) et de tournées de véhicules (VRP) ont une vaste littérature, avec plusieurs modèles et méthodes de résolution. Par la suite, nous allons décrire les plus connus et les plus utilisés pour résoudre les problèmes abordés.

La section est divisée en deux sous-sections qui traitent des méthodes exactes et des méthodes approchées.

1.5.1 Méthodes exactes

Nous commençons par les méthodes exactes de résolution qui sont plus recommandées pour traiter les problèmes qui sont résolus en temps polynomial.

1.5.1.1 Programmation dynamique

La programmation dynamique est une des techniques utilisées pour résoudre des problèmes d'optimisation. Elle a été développée par Bellman dans les années 1950 et est très utilisée dans le domaine de la planification de production en raison de l'algorithme de Wagner et Whitin [97] pour le problème *lot-sizing* sans capacité (ULS).

Cette technique peut être appliquée à des problèmes généraux d'optimisation linéaire ou non-linéaire, continue ou discrète, avec fonctions dérivées ou non-dérivées, etc. Son fonctionnement consiste à décomposer le problème original en une séquence de sous-problèmes plus petits et plus simples à résoudre [5]. Il peut être aussi vu comme une reformulation du problème en un problème de plus court chemin et sa formalisation repose sur quelques concepts basiques :

Étapes : une étape est un point où les décisions sont prises.

États : chaque étape comprend un nombre d'états du système, qui fournit l'ensemble des informations nécessaires pour la prise de décision.

Décisions : une décision amène le système à un nouvel état, et pour cette opération il y a un coût associé.

Principe d'optimalité : dans un ensemble de décisions optimales il existe la propriété suivante : quelle que soit la première décision et l'état initial, les décisions subséquentes doivent être optimales par rapport à l'état résultant de cette

première décision. Plus simplement (deuxième forme du principe), toute sous-politique d'une politique optimale est optimale [40].

Récursivité : la politique optimale du problème initial est obtenue à partir des politiques optimales des ses sous-problèmes. Cette affirmation reflète la réductibilité de plusieurs approches pour les problèmes de *lot-sizing* complexes à sous-problèmes ULS [20].

Pour illustrer, utilisons l'algorithme de Wagner-Whitin en exprimant le problème comme la recherche d'un chemin plus court dans un graphe $G = (V, E)$. Ici V représente les sommets qui indiquent chaque période, et E l'ensemble des arcs $(e_{tt'})$ qui expriment des productions avec leurs coûts de *setup* (Q_t), de production (P_t) et de stockage (H_t) de la période t à t'. D_t représente la demande connue par période. La méthode consiste à trouver le chemin le plus court dans ce réseau étant donné les coûts des arcs, qui sont calculés selon la formule (1.35) suivante [77] :

$$e_{tt'} = Q_{t+1} + P_{t+1} \sum_{l=t+1}^{t'} D_l + \sum_{l=t+2}^{t'-1} H_l \sum_{l'=l+1}^{t'} D_{l'}$$

$$\forall t, t' = 1, ..., T, t < t' \quad (1.35)$$

$$s_{t-1} x_t = 0 \quad \forall t = 1, ..., T \quad (1.36)$$

La figure (1.5) montre un exemple avec 5 périodes de temps et une solution possible (flèches en trait plein). La solution respecte la propriété (1.36), démontrée par les auteurs, qui exprime que nous avons soit une production à la période t, soit un stock qui viendra de la période $t-1$ (jamais les deux en même temps) pour satisfaire les demandes. Cette propriété est évidente, car il n'existe pas de limite de capacité de production et de stockage dans une période.

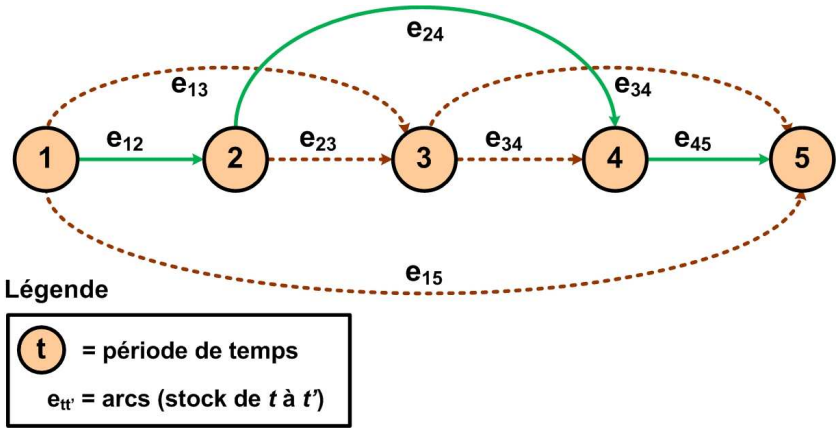


FIGURE 1.5 – Exemple de réseau pour la méthode de Wagner-Whitin.

A l'origine cet algorithme a une complexité en $O(T^2)$ pour traiter le problème de *lot-sizing*. Au début des années 90, de nombreux chercheurs (comme par exemple

Federgruen et Tzur [44] et Wagelmans *et al.* [96]) ont réussi à améliorer la borne de complexité pour obtenir $O(TlogT)$.

1.5.1.2 Programmation linéaire

Un programme linéaire (PL) est exprimé comme :

$$Z(X) = \min_{(x)}\{Cx : Ax \geq B, \quad x \in \mathbb{R}^n_+\} \tag{1.37}$$

où

- X est considéré comme l'ensemble de solutions faisables et il est composé par $x \geq 0$;
- $Z(X)$ est la valeur optimale de la fonction objectif quand l'optimisation est faite sur l'ensemble X des solutions faisables ;
- x est le vecteur colonne des variables non-négatives ;
- $C \in \mathbb{R}^n$ est le vecteur ligne des coefficients ;
- A est la matrice des contraintes avec des coefficients réels. Elle est de dimension $(m \times n)$;
- $B \in \mathbb{R}^m$ est le vecteur colonne des termes de droite des contraintes.

Les problèmes de programmation linéaire utilisent des variables fractionnaires et donnent des solutions dans le même espace. Cependant, si un problème utilise des variables discrètes, nous passerons à la programmation linéaire en nombre entier (*integer linear programming* - ILP, ou *integer programming* - IP).

Un autre type de programmation linéaire est appelé « entier mixte » (*mixed integer (linear) program* - MIP). Il s'agit d'un PL qui comprend des variables continues et des variables entières [82]. Un MIP peut être écrit comme :

$$Z(X) = \min_{(x,y)}\{Cx + Fy : \quad Ax + Dy \geq B \quad x \in \mathbb{R}^n_+ \quad y \in \mathbb{N}^p\} \tag{1.38}$$

- X est considéré comme l'ensemble de solutions faisables et il est composé par $(x, y) \in \mathbb{R}^n_+ \times \mathbb{N}^p$;
- $Z(X)$ est la valeur optimale de la fonction objectif quand l'optimisation est faite sur l'ensemble X des solutions faisables ;
- x et y sont respectivement les vecteurs colonnes des variables continues non-négatives et des variables entières non-négatives ;
- $C \in \mathbb{R}^n$ et $F \in \mathbb{R}^p$ sont les vecteurs lignes des coefficients ;
- A et D sont les matrices de contraintes avec des coefficients réels et des dimensions $(m \times n)$ et $(m \times p)$;

– $B \in \mathbb{R}^m$ est le vecteur colonne des termes de droite des contraintes.

De même, un programme linéaire binaire mixte (*mixed binary linear program* - MBP), ou programme mixte 0-1, est un MIP dans lequel les variables entières $y \in X$ sont booléennes ($y \in \{0, 1\}$).

Un dernier concept est la relaxation linéaire (RL). Il s'agit d'enlever les restrictions d'intégralité des variables $y \in \mathbb{N}^p$ d'un MIP, ce qui a pour conséquence d'augmenter l'espace de solutions faisables pour le problème.

Parmi les méthodes connues pour résoudre les problèmes MIP, nous en citons trois qui sont très célèbres dans la littérature :

Branch-and-bound : appelée « séparation et évaluation progressive » (SEP) en français, c'est une méthode fondamentale pour résoudre les problèmes IP et MIP. Elle se base sur le principe de séparer et évaluer des ensembles de solutions pour un problème.

Considérons S comme l'ensemble de N solutions $\{S^i : i = 1, ..., N\}$ d'un problème P. Chaque solution correspond à un sous-problème P^i et sa valeur est limité par des bornes supérieures (Z^i_{sup}) et inférieures (Z^i_{inf}). Les sous-ensembles et leurs problèmes associés forment une hiérarchie arborescente, appelée « arbre de décision ». La partie du domaine de réalisabilité du nœud courant en plusieurs sous-ensembles disjoints est l'étape de séparation de la méthode. Elle est nécessaire lorsque chercher des solutions en réalisant des partitions est plus facile et plus efficace que de chercher globalement.

L'objectif de la méthode est de parcourir l'arbre de décision en trouvant à chaque P^i des successions de bornes qui convergent en une solution où Z_{sup} est égal à Z_{inf} du problème original. Dans l'étape d'évaluation la méthode décidera pour chaque nœud trouvé, de confirmer ou abandonner la recherche en cours dans cette branche de l'arbre.

Plusieurs auteurs ont utilisé le *branch-and-bound* comme méthode (ou sous-méthode) de résolution de divers problèmes. Parmi les problèmes de *lot-sizing* (LSP), Nascimento [77] utilise cette méthode comme approche optimale pour traiter le LSP avec capacité et plusieurs sites. L'auteur utilise Cplex pour fournir des paramètres pour le *branch-and-bound*. Les stratégies considèrent une sélection des nœuds et des branches dans l'arbre de décision. Les expérimentations ont été réalisées sur des instances fournies par [91] et d'autres générées selon [6]. Les résultats montrent qu'une bonne configuration de la méthode de recherche conduit à de bonnes solutions pour le problème.

D'autres méthodes utilisent le *branch-and-bound* comme une partie du processus d'optimisation. Par exemple Salah Merzouk [73] traite d'une chaîne logistique linéaire composée d'une succession d'entreprises en cascade. Il utilise deux modèles : un plus simple où il s'agit de résoudre une structure fournisseur-client assistée par un transporteur, et un autre pour la chaîne linéaire complète. Les modèles étudiés considèrent des données déjà connues pour le problème et

aussi le temps de production, de chargement et de déchargement des produits dans un véhicule, et de voyage entre deux sites. Les méthodes de résolution consistent en un *branch-and-bound* et un algorithme génétique qui donne des résultats équilibrés entre solution/temps d'exécution.

Branch-and-cut : l'algorithme *branch-and-cut* est une extension du *branch-and-bound* dans laquelle les plans de coupe sont générés à chaque nœud de l'arbre de décision. La méthode ne se limite pas à réduire le nombre de nœuds, elle peut essayer aussi d'autres approches comme un pré-traitement ou une heuristique primale à chaque nœud [99].

Belvaux et Wolsey [12] ont développé le système « *bc-prod* », qui est basé sur *branch-and-cut* pour résoudre plusieurs types de problèmes *lot-sizing*. Le système exige comme entrée des fichiers écrits dans le langage de programmation spécifique pour le logiciel d'optimisation XPRESS-MP. Une fois que le problème est chargé dans une matrice il peut subir des remodelages spécifiques pour quelques types de problèmes. Ensuite, le *bc-prod* utilise une variété de routines de séparation, y compris les routines faites par [26] pour le LSP avec un seul produit. La méthode compte sur des heuristiques de relaxation et fixation de variables (*relax-and-fix*) pour certains problèmes. Le système est capable de résoudre une grande variété de problèmes, avec certaines exceptions, de manière satisfaisante.

Plusieurs travaux ont été faits en utilisant le *branch-and-cut*, comme par exemple ceux de Constantino [26, 27]. Les deux travaux traitent du problème *lot-sizing* avec multi-production. Le premier article montre une formulation renforcée par des plans de coupe de manière à trouver des bornes pour le problème programmation linéaire. Dans le second, il dérive des familles d'inégalités valides et développe des algorithmes polynomiaux pour résoudre le problème de séparation correspondant à chaque famille. Les deux méthodes ont été testées en plusieurs instances, y compris des problèmes réelles, et donnent des résultats intéressants.

Génération de colonnes : cette méthode représente une composante de la décomposition de Dantzig et Wolfe, et repose sur la division de l'ensemble des contraintes du problème en sous-ensembles. Normalement les meilleures formulations sont celles où le nombre de variables est exponentiel, par contre ce principe va contre le temps d'exécution. Donc, l'objectif de la génération de colonnes est de réduire l'utilisation d'un grand nombre de variables (les colonnes) dans la résolution du problème. La méthode comprend deux structures qui sont le problème dit « maître » (*Restrictive Master Problem* - RMP) et le sous-problème, qui comporte une fonction objectif modifié de l'originale. Alors, la méthode consiste à résoudre le sous-problème, et celui-ci donne comme solution les colonnes qui vont entrer dans la base du problème maître. Nous pouvons retrouver des descriptions plus approfondies chez Cordeau *et al.* [28], Lübbecke et Desrosiers [34] et Feillet [45], avec des applications en problèmes

réels comme les tournées de véhicules.

Branch-and-price : cette méthode a été développée par Barnhart *et al* [9] et son principe est le même que celui du *branch-and-cut*, mais en utilisant la génération de colonnes dans l'évaluation de chaque sommet de l'arbre de décision. Les colonnes restent en dehors de la relaxation du programme linéaire, cela à cause de la difficulté de les gérer car elles sont nombreuses. De ce fait, pour évaluer la qualité d'une solution la méthode consiste à résoudre un sous-problème pour identifier les colonnes qui vont entrer dans la base. Ce sous-problème est appelé « *pricing problem* » et sa solution amène soit à la re-optimisation du programme linéaire, soit au branchement de l'arbre, c'est le cas où elle ne respecte pas les conditions d'intégralité.

Programmation par contraintes : c'est une technique exacte pour résoudre les contraintes difficiles [74]. Elle peut aussi être considérée comme l'étude des systèmes de calcul basée sur des contraintes [79]. Elle concerne l'utilisation de contraintes dans la construction de relations entre variables. Elle définit aussi des critères pour arriver à une solution au lieu de dire quelles sont les séquences d'étapes qui doivent être accomplies. C'est la raison pour laquelle elle est classée comme un type de « programmation déclarative ». Dans les dernières années cette méthodologie a été bien étudiée par des experts de plusieurs domaines à cause de son habilité pour résoudre des problèmes réels difficiles, comme par exemple les problèmes d'ordonnancement. De bonnes synthèses ont été faites par Bartak [10] qui décrit la programmation par contraintes depuis ses origines, avec les concepts d'intelligence artificielle et de programmation logique par contraintes [63], Pesant et Rousseau [79], qui donnent aussi des exemples pour les diverses applications de cette méthode, et Gendreau et Pesant [50], qui proposent un *framework* qui intègre la méthodologie de programmation par contraintes avec les méthodes de recherche locale.

Relaxation lagrangienne : c'est un type de relaxation utilisé pour fournir des limites aux algorithmes *branch-and-bound* et résoudre plus facilement les problèmes de programmation linéaire (PL). L'idée principale est de dualiser certaines contraintes, c'est-à-dire ajouter ces contraintes dans la fonction objectif au travers d'un vecteur de multiplicateurs de Lagrange, et ensuite les éliminer de l'ensemble de contraintes. Le temps de calcul du problème lagrangien est plus petit que ce des méthodes exactes et la solution fournit sert comme une borne pour le problème initial.

Pour expliquer la méthode, considérons un problème P où nous avons des variables de décision (x), des vecteurs de coûts (C), des matrices de coefficients (A et D), et des vecteurs de termes indépendantes (B et E) [41]. Ainsi, nous

avons :

$$\begin{aligned}(P)\min\{Cx :\\ Ax \leq B\\ Dx \leq E\\ x \in \mathbb{N}\}\end{aligned} \tag{1.39}$$

Une relaxation lagrangienne d'un problème P (soit P_R) peut être exprimée par la multiplication des contraintes $Ax \leq B$ par un vecteur multiplicateur R. Ainsi, nous avons :

$$\begin{aligned}(P_R)\min\{Cx + R(Ax - B) :\\ Dx \leq E\\ x \in \mathbb{R}\}\end{aligned} \tag{1.40}$$

La qualité des bornes générées par P_R dépend de la valeur de R. Cette valeur doit être positive ou nulle pour assurer une borne inférieure à P. Si les contraintes sont du type $Ax \geq B$ les multiplicateurs R doivent être inférieurs ou égaux à zéro, et si elles sont du type $Ax = B$, le signe de R n'a pas de restriction.

Une solution optimale obtenue pour la relaxation doit valider les conditions $(Ax^* - B) \leq 0$, où x^* est la solution optimale pour $R = R^*$, et les conditions $R^*(Ax^* - B) = 0$.

Parmi les problèmes de *lot-sizing* qui sont résolus par la relaxation lagrangienne, nous citons le CLSP avec multi-produits, stock de sécurité et coût de demande non satisfaite (*Multi-item Capacitated Lot-sizing Problem with Safety stocks and demand Shortage costs* - MCLSS), traité par Absi et Kedad-Sidhoum en [1]. Dans ce problème, la demande est totalement (ou partiellement) perdue parce que le retard (*backlogging*) n'est pas permis. La fabrication d'un produit k dans une certaine période de temps t implique une consommation de ressource et de temps de préparation. Ici, l'objectif est de trouver un stock de sécurité afin d'assurer une production. L'article propose une relaxation lagrangienne des contraintes de capacité et de ressource, et développe un algorithme de programmation dynamique pour résoudre les sous-problèmes.

Toledo et Armentano [6] traitent du CLSP en une seule étape avec la fabrication de plusieurs produits en machines parallèles sans aucune relation entre eux (*Capacitated Lot-Sizing Problem in Parallel machines* - CLSPP). L'article décrit une heuristique basée sur une relaxation lagrangienne des contraintes de capacité et propose une optimisation sous-gradient où, à chaque étape, une procédure de faisabilité et d'autres améliorations sont appliquées à la solution du problème lagrangien.

1.5.2 Méthodes approchées

Dans la littérature nous pouvons trouver plusieurs méthodes approchées de résolution pour les problèmes de *lot-sizing*, comme des heuristiques, métaheuristiques et algorithmes d'approximation. Dans cette section nous allons présenter quelques travaux connus dans la littérature et aussi donner une description de certaines méthodes qui sont utilisées pour résoudre notre problème, en particulier l'heuristique *Relax-and-Fix*, qui a servi comme base pour certains des méthodes développés dans ce travail.

1.5.2.1 Heuristiques

Une heuristique, en une définition courte, est une méthode de résolution de problèmes d'optimisation combinatoire qui donne des solutions non nécessairement optimales, mais en un temps relativement court. Il existe plusieurs méthodes heuristiques, chacun avec une spécificité, comme par exemple : les méthodes d'heuristiques de construction, qui fournissent des solutions réalisables à partir d'une combinaison des éléments selon un critère (ex.: heuristique gloutonne) ; les heuristiques de voisinage, qui essayent d'améliorer une solution initiale à partir des échanges, suppressions et/ou insertions des éléments (ex.: recherche locale), les heuristiques systématiques, où les espaces de solutions sont parcourus selon des critères de branchement et coupes ; les heuristiques hybrides, qui combinent deux ou plusieurs heuristiques ; les métaheuristiques, où une procédure gère des heuristiques pour explorer l'instance du problème et son espace de solution ; etc.

D'autres méthodes sont plus spécifiques pour des problèmes de programmation linéaire en nombres entières, comme les relaxations utilisées pour résoudre problèmes *lot-sizing* (LSP). Le travail de Trigeiro *et al.* [91] à la fin des années 80 utilise une heuristique lagrangienne pour résoudre le problème avec capacité (CLSP) et temps de reconfiguration. L'algorithme fait des itérations entre les procédures primal et dual. Il relaxe les contraintes de capacité, ce qui divise le problème en sous-problèmes sans capacité et susceptibles d'être résolus par l'algorithme de Wagner et Whitin [97]. Une procédure d'heuristique d'atténuation est utilisée pour générer des solutions réalisables. La conclusion selon les auteurs est que les problèmes avec les reconfigurations sont encore plus difficiles quand la capacité est courte, les coûts de *setup* sont grands, les machines sont moins partagées et le nombre de périodes est petit. Autrement dit, leur algorithme est bien adapté pour les problèmes de grandes tailles.

Robinson et Lawrence [68] utilisent aussi une heuristique lagrangienne pour résoudre trois types de LSP avec demande dynamique : CLSP avec un seul produit, CLSP avec multi-produit et ULS. L'article montre des formulations en programmation entière mixte (MIP) et des procédures basées sur programmation linéaire et relaxation lagrangienne. Cette dernière s'est montrée dominante et a été capable de trouver des bonnes bornes inférieures, surtout pour les grandes instances. Les

méthodes ont été testées en problèmes industrielles et ont fournies une réduction considérable dans les coûts.

Une heuristique bien connue dans le domaine des problèmes de production s'appelle *Relax-and-Fix* (RF). Sa parution eut lieu dans les années 90 depuis l'article séminal de Dillenbeger et Escudero [35]. D'après [2] les problèmes de *lot-sizing* sont bien adaptés à cette méthode parce que les décisions faites au début de l'horizon de production sont plus importantes que celles de la fin. Sa caractéristique principale est la possibilité de travailler avec des sous-problèmes plus petits et potentiellement plus faciles que le problème initial.

Dans ce type d'heuristique, l'ensemble des variables entières binaires d'un MIP sont partagées en R sous-ensembles disjoints $(Q^1, ..., Q^R)$. A chaque itération deux groupes de variables sont définis comme entières et les autres comme relaxées de manière à créer un nouveau sous-problème MIP. Une fois que nous obtenons la réponse, nous répétons le processus de manière à ce que le premier sous-ensemble qui était entier devienne fixe, que les deux prochains soient considérés comme entiers et que les autres continuent à être relaxés. Le processus se répète jusqu'à la résolution du problème avec toutes les véhicules considérés comme entières.

Le travail de Stadtler [88] utilise des fenêtres de temps prédéfinies qui se chevauchent pour essayer de trouver des bons résultats. Dans une fenêtre une partie est relaxée et les autres sont considérées comme entières. La fixation des variables dans une fenêtre se produit quand elles ne se chevauchent pas avec la fenêtre suivante. Dans cette approche chaque sous-problème est traité séparément.

Un autre travail intéressant qui utilise RF est le système *bc-prod* (décrit précédemment) développé pour Belvaux et Wolsey [12]. L'outil est spécialisé en *branch-and-cut* pour résoudre les problèmes de *lot-sizing* et dispose aussi des algorithmes RF et des fenêtres de temps prédéfinies. Les variables à relaxer sont celles hors de la fenêtre. Une fois que le problème est résolu, les variables dans la fenêtre doivent être fixées. Le principal avantage du système est que ses formulations fortes fournissent de bonnes bornes inférieures pour le problème [2], par contre il n'utilise pas de fenêtres superposées, ce qui ne permet pas de considérer les effets de futures décisions de *setup*.

1.5.2.2 Métaheuristiques

Selon Fred Glover, les métaheuristiques, dans leur définition originale, sont des méthodes de résolution qui orchestrent une interaction entre procédures d'amélioration locale et stratégies de haut niveau pour créer un processus capable d'échapper à un optimum local et réaliser une recherche robuste d'un espace de solution [52].

En d'autres termes, une métaheuristique vise à résoudre des problèmes difficiles d'optimisation combinatoire pour lesquels nous ne connaissons pas de méthodes classiques efficaces (comme les problèmes NP-Complets, par exemple).

Une métaheuristique suit des itérations qui progressent vers l'optimum global, et sont basées sur une combinaison entre choix aléatoires et connaissance des résultats

antérieurs pour suivre les recherches dans les voisinages. En raison de ce niveau d'abstraction, ces méthodes sont bien adaptées à plusieurs types de problèmes.

Plusieurs métaheuristiques sont basées sur des systèmes naturels dans la physique, biologie, éthologie, etc. Parmi eux, nous pouvons citer les algorithmes génétiques, les colonies de fourmis, les réseaux de neurones, la recomposition de chemin, etc. Dans le cadre des problèmes de production, la plupart des métaheuristiques ont été faites pour résoudre les modèles plus complexes (que le *lot-sizing* sans capacité, par exemple). Nous mettons l'accent sur deux métaheuristiques particulièrement connues dans la littérature :

GRASP : la *Greedy Randomized Adaptative Search Procédure* (en français, « procédure de recherche gloutonne aléatoire adaptative ») est une métaheuristique qui travaille en deux phases : celle de la construction, où elle produit une solution réalisable, et celle de la recherche locale. Dans l'étape de construction, plusieurs candidats sont ensembles dans une liste qui est triée de manière « gloutonne » par rapport aux meilleures solutions. Ensuite un de ces candidats est tiré aléatoirement. Dans la deuxième étape, la recherche locale est appliquée sur la solution trouvée précédemment.

Path relinking (PR) : la « recomposition de chemin » est une stratégie évolutive qui permet d'exploiter les trajectoires dans l'espace de solution en connectant les solutions de bonne qualité [78]. Le principe est de construire un chemin entre deux bonnes solutions de manière à évaluer la valeur des solutions intermédiaires, qui peuvent être meilleures que les deux extrêmes du chemin. Cette méthode possède des variantes [54] : la variation et *tunneling*, la recomposition extrapolée, le générateur multiple, la voisinage constructive et la construction de vocabulaire (*vocabulary building*). Chaque variante possède des caractéristiques particulières par rapport aux points intermédiaires entre les extrémités du chemin.

Un bon aperçu pour les métaheuristiques est présentée par Glover et Kochenberger [52], dans un ensemble de travaux de plusieurs auteurs qui s'appelle « *Handbook of Metaheuristics* ». Pour une révision des métaheuristiques pour des problèmes *lot-sizing*, en particulier le DLSP, nous recommandons l'article de Jans et Degraeve [33] qui montre non seulement un historique des métaheuristiques, mais aussi des modèles plus connus et des méthodes généralement utilisées pour les résoudre. Pour une lecture plus détaillée sur le *path relinking*, l'article de Glover et Laguna [54] montre le fonctionnement de ses variantes ainsi que de la méthode *scatter search*, qui a une liaison directe avec PR, tout ça avec des exemples et illustrations.

En [16, 17, 18], Boudia *et al.* montrent des approches basées sur GRASP et *path relinking*. Leur première heuristique [16] assume le modèle du travail précédent et résout le problème complet. La méthode de résolution est faite en deux étapes : une phase de construction qui utilise GRASP et une phase d'amélioration qui utilise la

recherche locale. Les résultats obtenus sont comparés avec les méthodes précédentes et montrent la pertinence de l'utilisation de la méthode GRASP.

D'autres travaux de Boudia *et al.* ont été faits avec GRASP. Dans l'article [17], les auteurs traitent le problème en utilisant quatre approches : GRASP simple (vue dans [16]), GRASP réactive, GRASP avec *path relinking* à la fin du processus et GRASP avec *path relinking* dans les étapes intermédiaires. Ils ont fait la comparaison de ces approches avec les deux autres heuristiques vues en [15]. Plusieurs instances générées aléatoirement ont été testées. Les solutions ont montré que la quatrième méthode donne de meilleurs résultats.

Nascimento *et al.* [78] présentent une heuristique GRASP avec une procédure d'intensification de recomposition de chemins (*path relinking intensification*) pour trouver des solutions de coût effectif pour le MPCLSP. L'heuristique est utilisée aussi pour résoudre les instances du CLSP avec des machines parallèles.

Une dernière citation concerne le travail de Boudia et Prins [19] qui propose une métaheuristique appelée « algorithme mémétique avec gestion de population » (*Memetic Algorithme with Population Management* - MA|PM). Cette algorithme résout le problème en une seule phase et les auteurs comparent les résultats du MA|PM avec ceux obtenus par les méthodes précédentes. L'approche MA|PM se montre plus efficace que les autres, même si pour quelques instances le temps d'exécution est plus grand, ce qui est qualifié par les auteurs comme acceptable pour les problèmes concernant un niveau de décision tactique.

1.5.2.3 Algorithmes d'approximation

L'objectif de ces algorithmes est de déterminer des solutions heuristiques polynomiales dont la raison d'approximation vers l'optimum soit la plus petite possible. Cette borne de qualité est le pire cas, assurée par des propriétés mathématiques, où la méthode doit retourner une réponse. Une notation communément utilisée explique la distance de la solution d'un algorithme par rapport à la solution optimale. Par exemple, un algorithme 4-approximatif trouve une solution au pire à 4 fois plus de la solution optimale.

Ces algorithmes peuvent être classifiés comme déterministes ou aléatoires. Les premières produisent toujours une même sortie avec un temps d'exécution identique. Les deuxièmes produisent différentes solutions à chaque exécution.

Khumawala *et al.* [65] présentent une comparaison entre les méthodes exactes et les méthodes d'approximation pour résoudre le problème de localisation des sites, ceux-ci sans contraintes de capacité. Ils font une analyse de coût-avantage entre ces méthodes, ce qui montre un avantage des méthodes exactes pour les problèmes de petite et moyenne taille.

Levi *et al.* [70] présentent des algorithmes d'approximation pour *lot-sizing* avec capacité et multi-production. Ils montrent la difficulté du problème et proposent une méthode qui résout une relaxation linéaire basée sur des inégalités de couverture d'arcs puis arrondit sa solution optimale, ce qui donne une valeur maximum

deux fois plus élevées que l'optimum. Cette procédure fournit un algorithme à 2-approximation.

Hoesel et Wagelmans [95] montrent un schéma composé par deux algorithmes d'approximation polynomiale pour le problème *economic lot-sizing* avec capacité et mono-produit. Le premier forme la base du schéma d'optimisation et est basé sur un algorithme de programmation dynamique qui donne une solution réalisable dont l'écart absolu par rapport à l'optimum est borné et dépendent de T. Le deuxième est plus simple et donne une solution réalisable d'un écart relatif de $2T$ d'optimum, qui est la borne supérieure. La méthode s'est montrée applicable à plusieurs problèmes de ce type.

Chapitre 2

Modèle intégré de production et transport

2.1 Introduction

Dans ce chapitre nous introduisons le problème *Lot-Sizing with Vehicle Routing Problem* (LSVRP) dans lequel nous traitons des problèmes de dimensionnement de lot (*Lot-Sizing Problem* - LSP) sur un ensemble de sites de fabrication. A ce problème de planification de la production nous ajoutons un problème de tournées de véhicules (*Vehicle Routing Problem* - VRP) pour effectuer les livraisons des produits fabriqués entre les sites. En effet la production d'un site peut servir à alimenter un autre site, il faut alors transporter cette production d'un site vers l'autre.

Tout d'abord nous présentons quelques hypothèses que décrivent le domaine du LSVRP, en simplifiant certains aspects du problème. Ensuite, en (2.2) nous présentons une formulation mathématique du LSVRP avant de proposer plusieurs heuristiques (2.3) et de présenter un cas particulier de production dédiée (2.4) où chaque site fabrique un seul produit.

Production : la partie production du LSVRP repose sur le problème classique de *lot-sizing* multi-produit, où nous le traitons de manière séparée sur chaque site producteur d'un ensemble de sites. Même si le traitement de la fabrication est indépendante sur chaque site, il existe quand même une dépendance au niveau d'assemblage des produits (production multi-niveaux). C'est-à-dire que la plupart des sites fabriquent uniquement s'ils ont assez de composants disponibles. La consommation des composants est faite par rapport à une gamme d'assemblage où chaque produit k consomme des produits k' à une quantité fixé.

Transport : dès qu'un produit k doit être transporté d'un site i vers un site j, nous faisons affecter un véhicule pour réaliser ce transport. Alors nous avons dans le LSVRP une flotte homogène d'agents transporteurs qui doivent être coordonnés avec les activités de production de manière à respecter le flot de

composants entre les sites. Les véhicules sont stockés dans un dépôt, qui est considéré aussi comme le point de départ des parcours. A la fin de l'horizon de production, tous les véhicules doivent retourner à ce dépôt. Le nombre de véhicules dans le système de transport est illimité. Cependant afin de ne pas utiliser un trop grand nombre de véhicules nous affectons un coût assez élevé à l'utilisation d'un véhicule (ce coût peut être vu comme une pénalité). Le coût de transport d'un produit entre deux sites comprend une partie variable (qui dépend de la quantité) et une partie fixe (qui dépend du chemin emprunté). Les coûts de déplacements entre sites et dépôt ne sont pas pris en compte (nous supposons que le service de livraison est externalisé).

Produits : nous avons trois types de produits dans le système : les produits basiques, qui ne nécessitent pas des composants, mais qui sont consommés par la fabrication d'autres types de produits ; les produits intermédiaires qui consomment des composants et sont aussi consommés par la fabrication d'autres produits ; et les produits finaux qui sont consommateurs des deux autres types de produits.

Les produits finaux ne sont pas pris en compte dans le système de transport car nous considérons qu'ils sont directement livrés au client. La logistique aval est ainsi découplée de la logistique interne. Dans une situation avec plusieurs sites d'assemblage final, nous pourrions considérer une collecte des produits finaux pour les stocker en un lieu unique.

Demandes : nous considérons comme donnée du problème uniquement la demande en produit final. Elle est connue à l'avance, et c'est la demande que toute la ligne de production prend comme base. A partir d'elle tous les autres sites vont ajuster leur production de composants. La production globale doit donc respecter les quantités et limites de temps imposées par les demandes des clients en produit final.

Remarque : nous ne prenons pas en compte les dimensions et volume des produits, ils sont considérés comme unités. Cette simplification évite, par exemple, de traiter des sous-problème spécifiques pour optimiser le stockage et le transport.

Stocks : bien que le modèle LSVRP a été conçu pour accepter un stock de démarrage de la production, nous allons considérer que les stocks de produits fabriqués ou de composants sur un site sont vides en début de planning, sans perte de généralité. De plus, les stocks seront aussi vides en fin de planning.

Temps de reconfiguration (*setup times*) : dans un site, la fabrication de plusieurs produits normalement implique en une préparation spécifique de machines et de ressources pour démarrer la fabrication de chaque type de produit. Le modèle LSVRP considère des temps de reconfiguration de machine qui se traduisent en une consommation supplémentaire de capacité au début de chaque type de production, comme en [91] et [82]. La figure (2.1) montre une fabrica-

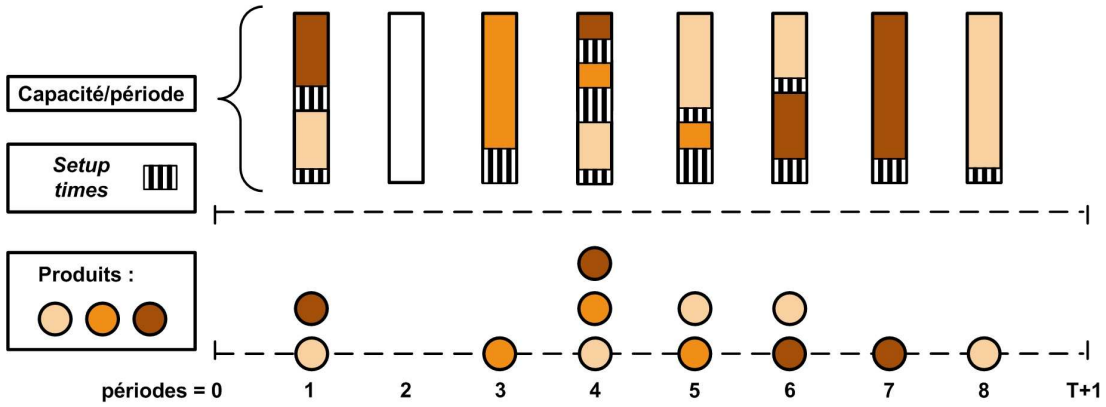


FIGURE 2.1 – Exemple de production avec temps de reconfiguration

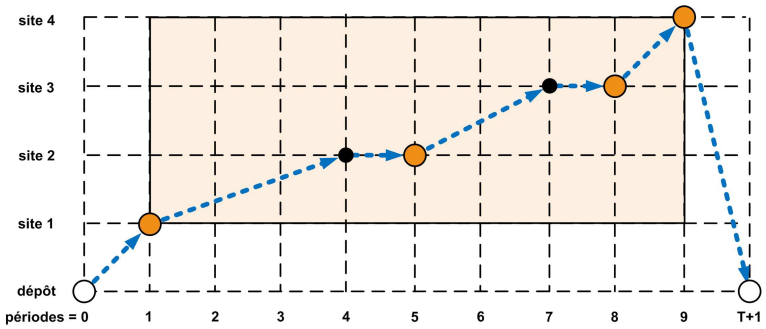
tion de trois produits dans un site. Les opérations de changement de produit dans le système sont exprimées par les parties hachurées dans la barre de capacité.

Nomenclature (*Bill of Materials*) : nous considérons une nomenclature de type assemblage (convergente) ou en série (ligne de production). Les cas divergente et généraux (expliqués dans le chapitre 1) ne sont donc pas traités. Associé aux nomenclatures, chaque produit non basique a un taux fixe de consommation en composants nécessaires à leur fabrication. Ce taux de consommation définit les besoins en produits et la demande non connue du système.

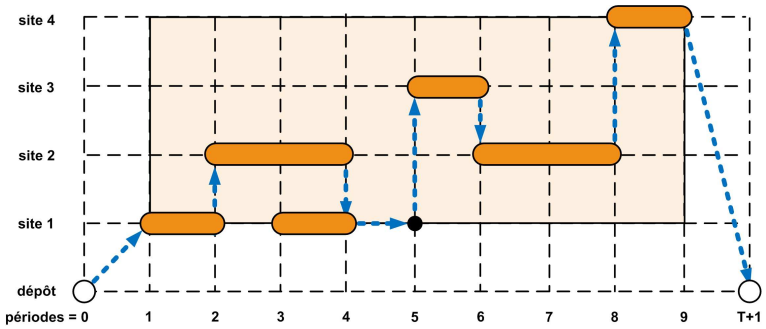
Capacités : les capacités sont présentes dans le modèle pour les deux problèmes (production et transport). Pour la production elle est associée à chaque site et est renouvelable à chaque période. Pour le transport, elle est associée à chaque véhicule. Cependant, nous avons décidé de ne pas imposer une capacité pour le flot des produits car il est déjà limité implicitement pour la capacité de production. De plus avoir une borne pour le transport de composants implique uniquement une augmentation du nombre de véhicules. De même, pour le stockage nous n'avons pas de capacité.

Durée des opérations : la production et le transport ont la même échelle de temps, ce qui implique une discrétisation du temps pour les tournées de véhicules. Alors nous avons décidé de traiter le problème de transport d'un manière plus simple dans un réseau temporel. De ce fait, l'adaptation que nous avons faite pour le traitement du temps est telle que dans l'article [59] : la durée de livraison et la durée de production sont mesurées en période de temps (dans l'horizon de planification) de manière à ce que l'ordre de grandeur (secondes, minutes, heures, jours,...) de l'échelle de temps de l'une soit identique à l'ordre de grandeur de l'échelle de temps de l'autre.

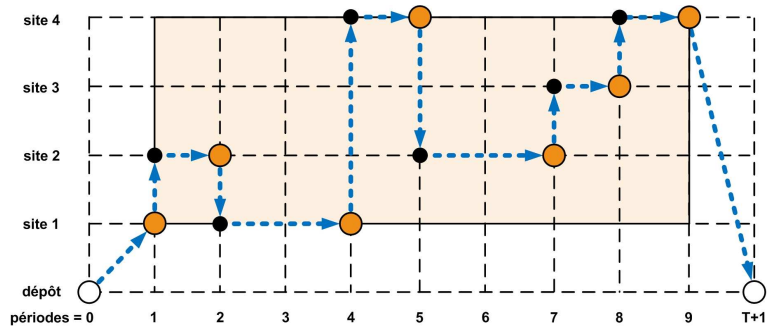
Prenons un exemple avec 4 sites de fabrication et un horizon de 9 périodes de temps. Les figures dans (2.2) montrent les trois possibilités : a) temps de production soumis au temps de transport ($T_P < T_T$) ; b) temps de transport



(a) Durée de transport > durée de production.



(b) Durée de transport < durée de production.



(c) Durée de transport = durée de production.

FIGURE 2.2 – Durées des opérations.

soumis au temps de production ($T_P > T_T$) ; et temps de production et de transport égaux ($T_T = T_P$). Nous remarquons dans la figure (2.2(a)) que la production est « instantanée » dans le système. De même, la figure (2.2(b)) montre un système où il existe un délai de production (*lead time*) et le transport est plutôt considéré comme un transfert instantané de produits entre sites. La figure (2.2(c)) montre la situation où les durées de transport et de production sont égales. Dans ces figures, les arcs représentent les durées de transport, les cercles en blanc représentent le dépôt, les cercles en noir sont les sommets des arcs (les sites), et les autres cercles et rectangles arrondis sont les durées de production.

Le problème traité dans ce chapitre correspond à la situation où **la fabrication d'un lot commence et se termine dans la même période** (mono-

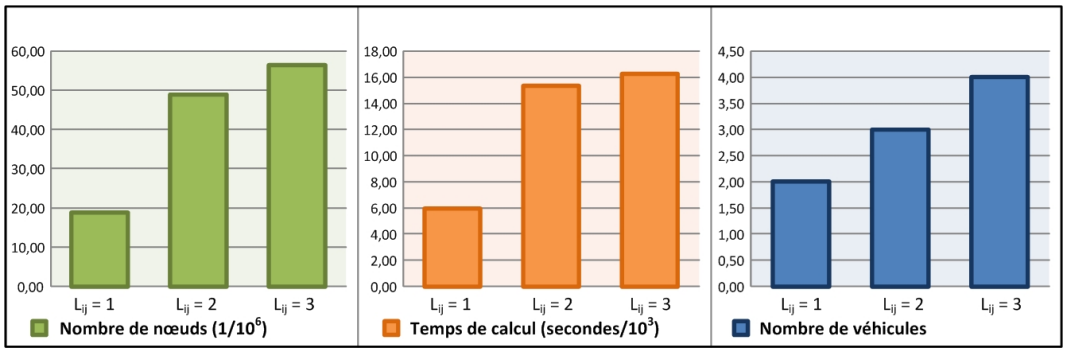


FIGURE 2.3 – Impact des durées de transport.

périodicité) et la durée de transport devient dominante du point de vue des contraintes de temps. Cette situation représente une problématique plus riche en termes combinatoires. Elle correspond aussi à des situations courantes dans le cadre industriel actuel où les entreprises importent les composants de leurs produits de sites géographiquement distants, car cela est financièrement plus rentable. Pour montrer l'impact des durées de transport dans le système de production, nous avons fait des tests en prenant quelques instances et en restreignant la durée de transport entre les sites i et j à 1, 2 ou 3 périodes de temps. La figure (2.3) exprime la moyenne des nombres de nœuds parcourus pour résoudre 10 problèmes de taille moyenne (4 sites de fabrication et 36 périodes de temps, c'est-à-dire 4×36 sommets dans le graphe). La figure montre aussi les valeurs pour le temps de calcul et le nombre de véhicules utilisés.

Dans la figure (2.3) c'est évident que le nombre de nœuds, le temps de calcul et le nombre de véhicules grandit avec l'augmentation des durées de transport. La raison à cela est la diminution du nombre de possibilités de trajet (nombre d'arcs) valides en raison des déplacements de longues durée. Par exemple, dans la figure (2.2(a)) il serait impossible de réaliser un déplacement du site 1 vers le site 2 (dont $L_{12} = 3$ est la distance en périodes) à la période 8 car le véhicule arriverait au site 2 après la fin de l'horizon de production.

Le problème dont les durées de production sont supérieures à celles de livraison ne sera pas étudié dans ce travail, et la situation où les durées de production et de transport sont instantanées sera traité dans le chapitre 4 sur une forme de « multiflot » dans un graphe.

Pour finaliser cette section, nous présentons un exemple illustratif du problème. Prenons une instance avec 4 sites de fabrication, 4 produits dans le système dont 1 est final. Nous avons un système de production où deux des quatre sites (sites 2 et 3) produisent deux types différents de composants. Nous considérons le produit du site 1 comme un composant basique du système et le produit du site 4 comme le produit final. Les produits sont représentés par des cercles en différentes couleurs dans la figure (2.4), qui montre une possible solution pour la satisfaction d'une demande à

la période 9. Le transport peut être réalisé par deux véhicules qui transitent entre les sites pour livrer les composants. Les arcs en traits pointillés représentent les véhicules vides, les autres représentent le transport de produits.

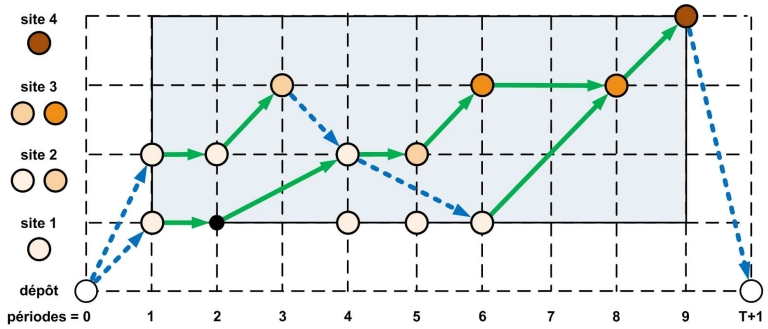


FIGURE 2.4 – Exemple d'un problème LSVRP

2.2 Modélisation mathématique

Pour la partie production sur chaque site, nous nous appuyons sur le modèle *Capacitated Lot-sizing Problem* (CLSP). Le but est de minimiser les coûts de fabrication et de stockage dans l'horizon de planification d'un site, tout en respectant les contraintes de consommation de capacité de production. Comme il y a plusieurs sites interdépendants, nous introduisons une liaison entre la production de composants et leur transport entre les différents sites.

La structure générale de notre modèle peut être divisée en trois blocs : un bloc de production, un bloc de transport et un bloc de couplage, qui concerne l'entrée et la sortie des composants sur chaque site. La figure (2.5) illustre cette structure en blocs. Nous supposons que x est un vecteur qui contient les variables et B contient les termes constants (ressources du système). Notons que la relaxation des blocs de couplage induit deux problèmes séparés de production et de transport.

Dans le bloc de transport nous avons une version simplifié du *Vehicle Routing Problem* (VRP). Cela est dû au fait de que le couplage avec le problème de production impose des contraintes qui n'appartiennent pas au VRP de base. Alors des adaptations ont été nécessaires, comme nous avons vu dans les hypothèses en (2.1).

La figure (2.6) montre le graphe de transport (2.6(a)) et sa discrétisation en périodes (2.6(b)). La distance entre chaque pair de sites est définie en périodes de temps (arc $(i,\ j)= t$ périodes). Exemples : arc $(1, 2) = 3$, arc $(2, 3) = 2$, etc. Cette distance constitue une limite minimale d'arrivée pour les véhicules. Les transporteurs peuvent aussi arriver et rester chargés au lieu d'effectuer la livraison, mais cette opération implique un coût d'attente pour le véhicule. Dans la figure, nous remarquons des arcs diagonaux de différentes tailles. Ce sont les représentations des durées du trajet entre deux sites. Les arcs horizontaux représentent l'attente d'un véhicule dans un site.

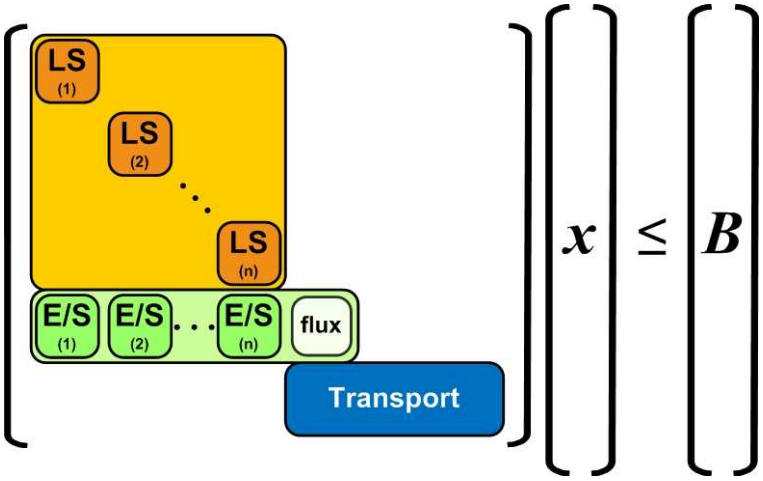
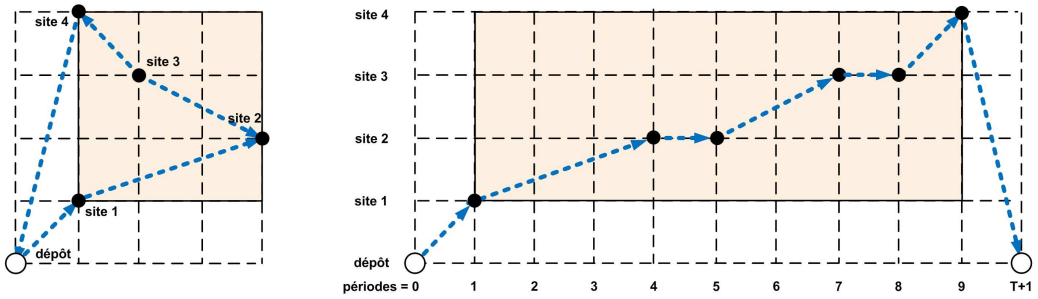


FIGURE 2.5 – Structure en bloc du problème.



(a) Graphe de transport.

(b) Réseau discrétisée en périodes.

FIGURE 2.6 – Réseau de transport.

Données

Comme vu dans le chapitre précédent, nous avons un ensemble V de sites i producteurs qui doivent fabriquer des produits k pour satisfaire les demandes $(D^k_{i,t})$ dans un horizon de T périodes de temps $(t = 1, \ldots, T)$. $V = \{V_0\} \cup V'$, où V_0 est le dépôt des véhicules et $V' = \{V_1, \ldots, V_N\}$ est l'ensemble de sites. Nous avons aussi un ensemble de produits U, et chaque site $i \in V'$ fabrique des produits k qui appartiennent à une liste $U_i \subset U$ de produits fabriqués sur le site i. De même, chaque produit k a une liste $V'_k \subset V'$ de sites qui le fabriquent. Pour cette opération, chaque k en i consomme des composants k' à un taux de consommation $A^{k'}_k$. Il faut aussi prendre en compte les coûts de production $(P^k_{i,t})$, de *setup* $(Q^k_{i,t})$ et de stockage $(H^k_{i,t})$ à chaque période t. Sur chaque site nous imposons une limite individuelle de production par produit $(M^k_{i,t})$ et une capacité de production $M_{i,t}$ renouvelable à chaque période. Cette capacité est consommée par la production de produit k au site i à un taux B^k_i, et également par les reconfigurations de machine (*setup times*, Z^k_i) dans le cas où plusieurs produits sont fabriqués sur le site i à la même période.

Dans la composante transport du problème, chaque arc $(i, j) \in E$ représente le

plus court chemin de i vers j, avec une durée L_{ij} (en périodes de temps). Ces arcs ont un coût fixe (I_{ij}) d'utilisation. Nous ne prenons pas en compte les coûts des déplacements entre sites et dépôt (externalisation du service de livraison). Le transport unitaire pour chaque produit k coûte J_{ij}^k, et la quantité transportée ne doit pas dépasser W unités. Cependant, nous considérons cette capacité comme infinie, comme expliqué dans les hypothèses en (2.1).

Variables

Soit k un produit, i et j des sites de fabrication et t la période de temps actuelle. Nous définissons des variables qui concernent la quantité à produire ($x_{i,t}^k$), la décision de *setup* ($y_{i,t}^k$), la quantité à stocker en produits fabriqués sur place ($s_{i,t}^k$) et la quantité à stocker en composants (ressources) venus d'autres sites ($r_{i,t}^k$). Nous pourrions utiliser une seule variables pour les deux types de stock, cependant nous avons choisi de les séparer pour bien différencier leur flot dans le système de production.

Pour le transport nous définissons des variables g qui agrègent les véhicules sur les arcs, dont $g_{ij,t}$ représente la quantité de véhicules quittant i à l'instant t pour se rendre en j à l'instant $t + L_{ij}$. Les parcours des véhicules eux-même sont obtenus en décomposant ce flot en cycles. Le nombre de véhicules utilisés dans le système de transport est exprimé par la variable v qui n'est pas explicitement limitée, mais qui est pénalisée par un coût assez important pour éviter une forte augmentation de sa valeur.

Attaché au flot des véhicules, nous avons aussi le flot des produits envoyés d'un site à l'autre ($f_{ij,t}^k$). Là nous avons une simplification : si la capacité d'un véhicule n'est pas assez grande pour respecter le flot f, alors nous pouvons répartir cette quantité entre les véhicules. Le transport compte aussi avec des variables d'entrée et de sortie de produits dans un site (respectivement, $\sigma_{i,t}^k$ et $\tau_{i,t}^k$).

Fonction objectif

Nous cherchons à minimiser les coûts globaux de transport et de production tout en respectant les contraintes présentes dans les blocs de production, de transport et de couplage.

$$\min \sum_{\substack{(i,j)\in E \\ i,j\in V'}} \sum_{t=1}^{T} \left[I_{ij}\, g_{ij,t} + \sum_{k\in U} J_{ij}^k\, f_{ij,t}^k \right] + X\, v$$

$$\sum_{k\in U} \sum_{i\in V'} \sum_{t=1}^{T} \left[Q_{i,t}^k\, y_{i,t}^k + P_{i,t}^k\, x_{i,t}^k + H_{i,t}^k (s_{i,t}^k + r_{i,t}^k) \right] \quad (2.1)$$

Contraintes

Les contraintes suivantes (2.2 - 2.8) concernent les blocs de production. Les deux premières (2.2 et 2.3) sont responsables de la propagation et de l'utilisation du stock des produits fabriqués sur place ainsi que du stock des composants transportés d'un site vers un autre entre les périodes $t-1$ et t. Les contraintes (2.4)-(2.6) concernent le *setup* et les délais liés aux *setup*. Elles représentent, respectivement, la limite de production pour chaque produit, la consommation de machine pour la fabrication de chaque produit et l'obligation de fabriquer s'il y a une opération de *setup*. Les contraintes (2.7 et 2.8) représentent le respect de la demande en produit final et la limite supérieure de production pour cette demande.

$$s^k_{i,t-1} + x^k_{i,t} = s^k_{i,t} + \tau^k_{i,t} \quad \forall k \in U, \forall i \in V'_k, \forall t = 1, \ldots, T \tag{2.2}$$

$$r^{k'}_{i,t-1} + \sigma^{k'}_{i,t} = r^{k'}_{i,t} + \sum_{k \in U_i} A^{k'}_k \, x^k_{i,t} \quad \forall k' \in U, \forall i \in V', \forall t = 1, \ldots, T \tag{2.3}$$

$$x^k_{i,t} \leq M^k_{i,t} \, y^k_{i,t} \quad \forall k \in U, \forall i \in V'_k, \forall t = 1, \ldots, T \tag{2.4}$$

$$\sum_{k \in U_i} (B^k_i \, x^k_{i,t} + Z^k_i \, y^k_{i,t}) \leq M_{i,t} \quad \forall i \in V', \forall t = 1, \ldots, T \tag{2.5}$$

$$x^k_{i,t} \geq y^k_{i,t} \quad \forall k \in U, \forall i \in V'_k, \forall t = 1, \ldots, T \tag{2.6}$$

$$\sum_{i \in V'_k} \tau^k_{i,t} = D^k_{j,t} \quad \forall k \in F, \forall j \in V'_k, \forall t = 1, \ldots, T \tag{2.7}$$

$$x^k_{i,t} \leq D^k_{i,t} \, y^k_{i,t} \quad \forall k \in F, \forall i \in V'_k, \forall t = 1, \ldots, T \tag{2.8}$$

Les contraintes suivantes concernent le flot de composants entre les sites de fabrication. Elles lient la production avec le système de transport. Les contraintes (2.9) expriment la conservation de flot d'un produit sur un site. Les contraintes (2.10) et (2.11) lient le flot interne de produits et leur entrée ou sortie d'un site. Elles constituent l'interface de chaque site.

$$\sum_{i \in V'} f^k_{ij,t-L_{ij}} - \sigma^k_{j,t} = \sum_{i \in V'} f^k_{ji,t} - \tau^k_{j,t} \quad \forall k \in U, \forall j \in V', \forall t = 1, \ldots, T \tag{2.9}$$

$$\sigma^k_{j,t} = \sum_{i \in V'} f^k_{ij,t-L_{ij}} \quad \forall k \in U, \forall j \in V', \forall t = 1, \ldots, T \tag{2.10}$$

$$\tau^k_{j,t} = \sum_{i \in V'} f^k_{ji,t} \quad \forall k \in U, \forall j \in V', \forall t = 1, \ldots, T \tag{2.11}$$

Le dernier bloc de contraintes concerne le problème de transport entre les sites. Comme dit précédemment, les contraintes classiques de VRP ont été adaptées pour traiter les tournées de véhicules sur un graphe temporel. Les distances entre sites (L_{ij}) sont exprimées en périodes de temps de production et les variables $g_{ij,t}$ in-

diquent le nombre de véhicules qui prennent cet arc.

Les contraintes (2.12) imposent le respect de la capacité de flot pour chaque arc.

$$\sum_{k\in U} f^k_{ij,t} \leq W\, g_{ij,t} \quad \forall (i,j) \in E, i,j \in V', \forall t = 1,\ldots,T \tag{2.12}$$

Les contraintes (2.13) limitent le nombre de véhicules (v) qui sortent du dépôt à la période t et les contraintes (2.14) expriment la conservation de flot de véhicule sur chaque site.

$$\sum_{j\in V'} \sum_{t=1}^{T} g_{0j,t} \leq v \tag{2.13}$$

$$\sum_{\substack{j\in V' \\ t-L_{ji}\geq 0}} g_{ji,t-L_{ji}} = \sum_{\substack{j\in V' \\ t+L_{ij}\leq T}} g_{ij,t} \quad \forall i \in V', \forall t = 1,\ldots,T \tag{2.14}$$

Nous avons aussi les contraintes (2.15) qui limitent le nombre de véhicules qui transitent dans une période de temps t. Cette contrainte peut être considérée comme une redondance car elle est une conséquence de (2.13) et (2.14).

$$\sum_{\substack{(i,j)\in E \\ i\in V'}} \sum_{\substack{t'\geq t \\ t'<t+L_{ij}}} g_{ij,t'} \leq v \quad \forall t = 1,\ldots,T \tag{2.15}$$

Les dernières contraintes (2.16) et (2.17) définissent le domaine des variables.

$$x^k_{i,t},\ s^k_{i,t},\ r^k_{i,t},\ \sigma^k_{i,t},\ \tau^k_{i,t},\ f^k_{ij,t},\ g_{ij,t} \in \mathbb{N} \qquad \forall k \in U, \forall i,j \in V', \forall t = 1,\ldots,T \tag{2.16}$$

$$y^k_{i,t} \in \{0,1\} \quad \forall k \in U, \forall i \in V', \forall t = 1,\ldots,T \tag{2.17}$$

Les variables entières concernant le flot de production dans les système peuvent être traitées aussi comme réelles positifs, c'est-à-dire $x^k_{i,t}$, $s^k_{i,t}$, $r^k_{i,t}$, $f^k_{ij,t}$, $\sigma^k_{i,t}$, $\tau^k_{i,t} \in \mathbb{R}_+$. Dans ce cas, au lieu de résoudre un problème de lots physiques, nous résolvons un problème de valeurs statistiques de production. Les résultats sont les mêmes, du fait de la propriété d'unimodularité, mais le temps d'exécution et le nombre de nœuds évalués par l'algorithme sont plus petits. De même, la variable de décision de *setup* ($y^k_{i,t}$) peut adopter une valeur réelle. Dans ce cas, elle est interprétée comme une fraction de la demande de k en i à la période t. En utilisant la formulation étendue du modèle (qui nous allons voir ensuite), cela représenterait la portion de la demande cumulative fabriquée à la période t et destinée satisfaire la demande à la période t' [3].

Ensuite nous avons le modèle complet du LSVRP :

Minimiser

$$\sum_{\substack{(i,j)\in E\\ i,j\in V'}} \sum_{t=1}^{T} \left[I_{ij}\, g_{ij,t} + \sum_{k\in U} J_{ij}^{k}\, f_{ij,t}^{k} \right] + X\, v$$

$$\sum_{k\in U} \sum_{i\in V'} \sum_{t=1}^{T} \left[Q_{i,t}^{k}\, y_{i,t}^{k} + P_{i,t}^{k}\, x_{i,t}^{k} + H_{i,t}^{k}(s_{i,t}^{k} + r_{i,t}^{k}) \right] \quad (2.1)$$

Sous les contraintes

$$s_{i,t-1}^{k} + x_{i,t}^{k} = s_{i,t}^{k} + \tau_{i,t}^{k} \quad \forall k \in U, \forall i \in V_k', \forall t = 1, \ldots, T \quad (2.2)$$

$$r_{i,t-1}^{k'} + \sigma_{i,t}^{k'} = r_{i,t}^{k'} + \sum_{k\in U_i} A_k^{k'} x_{i,t}^{k} \quad \forall k' \in U, \forall i \in V', \forall t = 1, \ldots, T \quad (2.3)$$

$$x_{i,t}^{k} \leq M_{i,t}^{k}\, y_{i,t}^{k} \quad \forall k \in U, \forall i \in V_k', \forall t = 1, \ldots, T \quad (2.4)$$

$$\sum_{k\in U_i} (B_i^{k}\, x_{i,t}^{k} + Z_i^{k}\, y_{i,t}^{k}) \leq M_{i,t} \quad \forall i \in V', \forall t = 1, \ldots, T \quad (2.5)$$

$$x_{i,t}^{k} \geq y_{i,t}^{k} \quad \forall k \in U, \forall i \in V_k', \forall t = 1, \ldots, T \quad (2.6)$$

$$\sum_{i\in V_k'} \tau_{i,t}^{k} = D_{j,t}^{k} \quad \forall k \in F, \forall j \in V_k', \forall t = 1, \ldots, T \quad (2.7)$$

$$x_{i,t}^{k} \leq D_{i,t}^{k}\, y_{i,t}^{k} \quad \forall k \in F, \forall i \in V_k', \forall t = 1, \ldots, T \quad (2.8)$$

$$\sum_{i\in V'} f_{ij,t-L_{ij}}^{k} - \sigma_{j,t}^{k} = \sum_{i\in V'} f_{ji,t}^{k} - \tau_{j,t}^{k} \quad \forall k \in U, \forall j \in V', \forall t = 1, \ldots, T \quad (2.9)$$

$$\sigma_{j,t}^{k} = \sum_{i\in V'} f_{ij,t-L_{ij}}^{k} \quad \forall k \in U, \forall j \in V', \forall t = 1, \ldots, T \quad (2.10)$$

$$\tau_{j,t}^{k} = \sum_{i\in V'} f_{ji,t}^{k} \quad \forall k \in U, \forall j \in V', \forall t = 1, \ldots, T \quad (2.11)$$

$$\sum_{k\in U} f_{ij,t}^{k} \leq W\, g_{ij,t} \quad \forall (i,j) \in E, i, j \in V', \forall t = 1, \ldots, T \quad (2.12)$$

$$\sum_{j\in V'} \sum_{t=1}^{T} g_{0j,t} \leq v \quad (2.13)$$

$$\sum_{\substack{j\in V'\\ t-L_{ji}\geq 0}} g_{ji,t-L_{ji}} = \sum_{\substack{j\in V'\\ t+L_{ij}\leq T}} g_{ij,t} \quad \forall i \in V', \forall t = 1, \ldots, T \quad (2.14)$$

$$\sum_{\substack{(i,j)\in E\\ i\in V'}} \sum_{\substack{t'\geq t\\ t'<t+L_{ij}}} g_{ij,t'} \leq v \quad \forall t = 1, \ldots, T \quad (2.15)$$

$$x^k_{i,t},\ s^k_{i,t},\ r^k_{i,t},\ \sigma^k_{i,t},\ \tau^k_{i,t},\ f^k_{ij,t},\ g_{ij,t} \in \mathbb{N} \quad \forall k \in U, \forall i,j \in V', \forall t = 1,\dots,T \tag{2.16}$$

$$y^k_{i,t} \in \{0,1\} \quad \forall k \in U, \forall i \in V', \forall t = 1,\dots,T \tag{2.17}$$

Reformulation du LSVRP

Nous présentons maintenant une reformulation étendue du modèle de production du LSVRP. Au lieu de considérer les opérations de production et de stockage séparément, nous définissons une variable $w^k_{i,tt'}$ qui indique la quantité du produit k fabriquée sur le site i à la période t pour être utilisée à la période t'. Elle intègre donc le stockage et le modèle est modifié en conséquence. Associé à ces variables, nous avons aussi des coûts variables ($C^k_{i,tt'}$) qui correspondent aux opérations de production du produit k sur le site i à la période t et de stockage jusqu'à la période t'.

Le modèle peut être formalisé autour de la question suivante : « combien devrions-nous produire dans la période t pour satisfaire une demande dans la période t' ? ». Elle conduira à une réformulation des blocs de production comme vu dans le modèle étendu du ULS (chapitre 1). Selon Wolsey [99], cette façon de travailler avec une seule référence de production/stockage à la place de deux est meilleure dans le sens où sa relaxation est plus proche de la solution entière. Par contre le nombre de variables dans le modèle augmente considérablement à cause de l'ajout d'une autre dimension. Néanmoins, nous utiliserons cette formulation comme la principale dans la suite de ce travail.

La fonction objectif (2.1) devient donc :

$$\begin{aligned}\min \sum_{\substack{(i,j)\in E\\ i,j\in V'}} \sum_{t=1}^{T} \left[I_{ij}\, g_{ij,t} + \sum_{k\in U} J^k_{ij}\, f^k_{ij,t} \right] + X\, v \\ + \sum_{k\in U} \sum_{i\in V'} \sum_{t=1}^{T} \left[Q^k_{i,t}\, y^k_{i,t} + H^k_{i,t}\, r^k_{i,t} + \sum_{t'=1}^{t} C^k_{i,t't}\, w^k_{i,t't} \right] \\ + \sum_{k\in U} \sum_{i\in V'} \sum_{t=1}^{T+1} C^k_{i,0t}\, w^k_{i,0t}\end{aligned} \tag{2.18}$$

L'intégration des coûts en (2.18) repose sur les éléments suivants :

$$\begin{cases} C^k_{i,tt} = P^k_{i,t} & \forall k \in U, \forall i \in V'_k, \forall t = 1, \ldots, T \\ C^k_{i,tt'} = P^k_{i,t} + \sum_{l=t}^{t'-1} H^k_{i,l} & \forall k \in U, \forall i \in V'_k, \forall t, t' = 1, \ldots, T \\ C^k_{i,0t'} = \sum_{t=1}^{t'} H^k_{i,t} & \forall k \in U, \forall i \in V'_k, \forall t' = 1, \ldots, T+1 \end{cases}$$

La première égalité donne le coût de fabrication du produit k à la période t pour satisfaire la demande à la même période (pas de stock). La deuxième combine les coûts de production à la période t aux coûts de stockage jusqu'à la période t'. La troisième donne le coût associé à un produit disponible initialement et stocké jusqu'à la période t'.

Les premières contraintes du bloc de production (2.19) concernent la consommation du stock initial disponible au début de la production sur chaque site ($S^k_{i,0}$). Normalement nous considérons $S^k_{i,0} = 0$, mais le modèle permet d'avoir un stock pour démarrer la production.

$$\sum_{t=1}^{T+1} w^k_{i,0t} = S^k_{i,0} \quad \forall k \in U, \forall i \in V'_k \tag{2.19}$$

Les autres contraintes sont similaires à celles du modèle précédent, avec une adaptation pour considérer les coûts intégrés. Les contraintes suivantes (2.20)-(2.27) expriment, dans cette ordre, la consommation et le flux des composants sur l'horizon de production, la limite individuelle de production, la consommation de machine, l'obligation de production s'il y a une opération *setup*, si les produits qui sortent proviennent de la fabrication sur place où d'un stock déjà existent au début de l'horizon (où des deux), la satisfaction de la demande, la limite de production de produits finaux par période et le domaine des variables.

$$r^{k'}_{i,t-1} + \sigma^{k'}_{i,t} = r^{k'}_{i,t} + \sum_{k \in U_i} (A^{k'}_k \sum_{t'=t}^{T} w^k_{i,tt'}) \quad \forall k' \in U, \forall i \in V', \forall t = 1, \ldots, T \tag{2.20}$$

$$\sum_{t'=t}^{T} w^k_{i,tt'} \leq M^k_{i,t}\, y^k_{i,t} \quad \forall k \in U, \forall i \in V'_k, \forall t = 1, \ldots, T \tag{2.21}$$

$$\sum_{k \in U_i} (B^k_i \sum_{t'=t}^{T} w^k_{i,tt'} + Z^k_i\, y^k_{i,t}) \leq M_{i,t} \quad \forall i \in V', \forall t = 1, \ldots, T \tag{2.22}$$

$$\sum_{t'=t}^{T} w^k_{i,tt'} \geq y^k_{i,t} \quad \forall k \in U, \forall i \in V'_k, \forall t = 1, \ldots, T \tag{2.23}$$

$$w^k_{i,0t'} + \sum_{t=1}^{t'} w^k_{i,tt'} = \tau^k_{i,t'} \quad \forall k \in U, \forall i \in V'_k, \forall t' = 1, \ldots, T \tag{2.24}$$

$$\sum_{i \in V'_k} \tau^k_{i,t} = D^k_{j,t} \quad \forall k \in F, \forall j \in V'_k, \forall t = 1, \ldots, T \tag{2.25}$$

$$w^k_{i,tt'} \leq D^k_{i,t'}\, y^k_{i,t} \quad \forall k \in U, \forall i \in V'_F, \forall t, t' = 1, \ldots, T, \forall t' \geq t \tag{2.26}$$

$$w^k_{i,tt'},\ r^k_{i,t},\ \sigma^k_{i,t},\ \tau^k_{i,t},\ f^k_{ij,t},\ g_{ij,t} \in \mathbb{N} \quad \forall k \in U, \forall i, j \in V', \forall t, t' = 1, \ldots, T,\ t \leq t' \tag{2.27}$$

$$y^k_{i,t} \in \{0, 1\} \quad \forall k \in U, \forall i \in V', \forall t = 1, \ldots, T \tag{2.17}$$

Les contraintes des blocs de couplage et de transport, tel que le domaine des variables de *setup*, restent identiques au modèle initial.

La différence majeur entre ces deux modèles est la manière dont les coûts de production et de stockage son traités. Le résultat de la résolution entière est la même, par contre le gap d'intégrité entre la résolution entière et celle relaxée est plus petit dans le deuxième modèle.

Le modèle complet du LSVRP reformulé est :

Minimiser

$$\sum_{\substack{(i,j) \in E \\ i,j \in V'}} \sum_{t=1}^{T} \left[I_{ij}\, g_{ij,t} + \sum_{k \in U} J^k_{ij}\, f^k_{ij,t} \right] + X\, v + \sum_{k \in U} \sum_{i \in V'} \sum_{t=1}^{T} \left[Q^k_{i,t}\, y^k_{i,t} + H^k_{i,t}\, r^k_{i,t} + \sum_{t'=1}^{t} C^k_{i,t't}\, w^k_{i,t't} \right] + \sum_{k \in U} \sum_{i \in V'} \sum_{t=1}^{T+1} C^k_{i,0t}\, w^k_{i,0t} \tag{2.18}$$

Sous les contraintes

$$\sum_{t=1}^{T+1} w^k_{i,0t} = S^k_{i,0} \quad \forall k \in U, \forall i \in V'_k \tag{2.19}$$

$$r^{k'}_{i,t-1} + \sigma^{k'}_{i,t} = r^{k'}_{i,t} + \sum_{k \in U_i} (A^{k'}_k \sum_{t'=t}^{T} w^k_{i,tt'}) \quad \forall k' \in U, \forall i \in V', \forall t = 1, \ldots, T \tag{2.20}$$

$$\sum_{t'=t}^{T} w^k_{i,tt'} \leq M^k_{i,t}\, y^k_{i,t} \quad \forall k \in U, \forall i \in V'_k, \forall t = 1, \ldots, T \tag{2.21}$$

$$\sum_{k\in U_i}(B_i^k \sum_{t'=t}^{T} w_{i,tt'}^k + Z_i^k\, y_{i,t}^k) \leq M_{i,t} \quad \forall i \in V', \forall t = 1,\ldots,T \tag{2.22}$$

$$\sum_{t'=t}^{T} w_{i,tt'}^k \geq y_{i,t}^k \quad \forall k \in U, \forall i \in V_k', \forall t = 1,\ldots,T \tag{2.23}$$

$$w_{i,0t'}^k + \sum_{t=1}^{t'} w_{i,tt'}^k = \tau_{i,t'}^k \quad \forall k \in U, \forall i \in V_k', \forall t' = 1,\ldots,T \tag{2.24}$$

$$\sum_{i\in V_k'} \tau_{i,t}^k = D_{j,t}^k \quad \forall k \in F, \forall j \in V_k', \forall t = 1,\ldots,T \tag{2.25}$$

$$w_{i,tt'}^k \leq D_{i,t'}^k\, y_{i,t} \quad \forall k \in U, \forall i \in V_F', \forall t, t' = 1,\ldots,T, \forall t' \geq t \tag{2.26}$$

$$\sum_{i\in V'} f_{ij,t-L_{ij}}^k - \sigma_{j,t}^k = \sum_{i\in V'} f_{ji,t}^k - \tau_{j,t}^k \quad \forall k \in U, \forall j \in V', \forall t = 1,\ldots,T \tag{2.9}$$

$$\sigma_{j,t}^k = \sum_{i\in V'} f_{ij,t-L_{ij}}^k \quad \forall k \in U, \forall j \in V', \forall t = 1,\ldots,T \tag{2.10}$$

$$\tau_{j,t}^k = \sum_{i\in V'} f_{ji,t}^k \quad \forall k \in U, \forall j \in V', \forall t = 1,\ldots,T \tag{2.11}$$

$$\sum_{k\in U} f_{ij,t}^k \leq W\, g_{ij,t} \quad \forall (i,j) \in E, i,j \in V', \forall t = 1,\ldots,T \tag{2.12}$$

$$\sum_{j\in V'} \sum_{t=1}^{T} g_{0j,t} \leq v \tag{2.13}$$

$$\sum_{\substack{j\in V' \\ t-L_{ji}\geq 0}} g_{ji,t-L_{ji}} = \sum_{\substack{j\in V' \\ t+L_{ij}\leq T}} g_{ij,t} \quad \forall i \in V', \forall t = 1,\ldots,T \tag{2.14}$$

$$\sum_{\substack{(i,j)\in E \\ i\in V'}} \sum_{\substack{t'\geq t \\ t'<t+L_{ij}}} g_{ij,t'} \leq v \quad \forall t = 1,\ldots,T \tag{2.15}$$

$w_{i,tt'}^k,\ r_{i,t}^k,\ \sigma_{i,t}^k,\ \tau_{i,t}^k,\ f_{ij,t}^k,\ g_{ij,t} \in \mathbb{N}$

$$\forall k \in U, \forall i,j \in V', \forall t,t' = 1,\ldots,T,\ t \leq t' \tag{2.27}$$

$$y_{i,t}^k \in \{0,1\} \quad \forall k \in U, \forall i \in V', \forall t = 1,\ldots,T \tag{2.17}$$

Selon Trigeiro [91], la complexité du problème *lot-sizing* devient NP-Complet une fois que nous considérons les temps de reconfiguration de machine (*setup times*). Cependant, nous pouvons également travailler le problème sans ces reconfigurations. Pour cela nous remplaçons les contraintes (2.22) du modèle par des contraintes plus simples de consommation de capacité. Ces contraintes supposent que les taux de consommation de machine par produit (B_i^k) valent 1 et la consommation de capacité

par la reconfiguration de machine (Z_i^k) valent 0. Alors, (2.22) deviennent :

$$\sum_{k \in U_i} \sum_{t'=t}^{T} w_{i,tt'}^k \leq M_{i,t} \quad \forall i \in V', \forall t = 1, \ldots, T \tag{2.28}$$

qui représente la somme de toute la production d'un site en une période de temps.

Selon [91] tester l'existence d'une solution faisable devient plus facile quand nous ignorons les reconfigurations de machine, une fois qu'il suffit de comparer la demande cumulative avec la capacité totale disponible. Néanmoins, le problème reste encore NP-Difficile.

2.3 Méthodes de résolution pour le cas général

Les problèmes nécessitant un modèle entier mixte sont souvent couteux à résoudre (en temps de calcul) et sont classées dans la catégorie des problèmes NP-difficiles. Par conséquent, l'utilisation des heuristiques est une alternative nécessaire dès que la taille du problème devient importante.

Parmi les nombreuses heuristiques existantes, il y a celles qui utilisent la structure des programmes entiers mixtes (*Mixed Integer Programs* - MIP) de manière à trouver de bonnes solutions en un temps de calcul réduit. Une stratégie bien connue est d'utiliser la relaxation linéaire d'un MIP pour gagner en performance de temps de résolution et/ou trouver des limites pour la valeur de la fonction objectif.

Les prochaines sous-sections présentent quelques heuristiques basées sur la formulation MIP pour résoudre le problème LSVRP. La plupart d'entre elles sont bien appropriées pour les problèmes de planification de production en particulier lorsque la relaxation des variables fournit une bonne prévision des opérations de *setup*. L'idée principale de ces heuristiques est de travailler sur des sous-ensembles de variables de décision du problème initial à chaque itération, cela par des relaxations linéaires et fixation des variables.

Comme le problème LSVRP est un hybride de deux problèmes de caractéristiques distinctes, nous avons réalisé plusieurs tests pour trouver les points compatibles entre ces problèmes et les paramétrer. Ces points sont référents à la taille du problème et des sous-ensembles, au type de variables et au temps d'exécution.

Alors, les heuristiques MIP que nous proposons fonctionnent avec les paramètres suivants :

Taille des sous-ensembles (TS) : à chaque itération, nous partitionnons les variables en sous-ensembles qui correspondent à des intervalles de périodes. L'un d'entre eux sera fixé dans l'itération suivante, et le reste des sous-ensembles sera relaxé. Notre méthode nous permet de fixer la taille de ces sous-ensembles, ce qui peut influencer le temps d'exécution et la qualité de la solution fournit par l'heuristique.

Type de variables (TV) : dans le modèle LSVRP, nous avons choisi les variables de décision de *setup* ($y^k_{i,t}$) et de flot de véhicule ($g_{ij,t}$) à fixer. Ainsi, nous travaillons avec les trois combinaisons possibles en termes de variables : y, g et $y|g$.

Temps d'exécution (TE) : pour la résolution de nos heuristiques nous utilisons le logiciel d'optimisation Cplex pour lequel nous pouvons fixer le temps maximal d'exécution. A l'issue de ce temps, nous récupérons la meilleure solution entière calculée.

2.3.1 Heuristique *Relax-and-Fix*

Les heuristiques de relaxation et de fixation de variables (*Relax-and-Fix*-RF) sont bien utilisées dans le domaine des problèmes de production depuis l'article de Dillenbeger et Escudero [35] dans les années 90. D'après [2] les problèmes de *lot-sizing* sont bien adaptés à cette méthode parce que les décisions faites au début de l'horizon de production sont plus importantes (aux sens qu'elles ont plus d'impact) que celles prises à la fin.

La caractéristique principale de cette heuristique est la possibilité de travailler en sous-problèmes MIP plus petits et potentiellement plus faciles à résoudre que le problème initial. Chaque MIP est composé de sous-ensembles de variables positives pouvant être considérées comme entières ou réelles. A chaque itération, l'heuristique fixe un sous-ensemble de variables entières et crée un nouveau MIP.

Soit un problème entier mixte :

$$\begin{aligned} \min_{(x,y)}\{C\,x + F\,y : \quad & A\,x + D\,y \geq B \\ & x \in \mathbb{R}^n_+ \\ & y \in \mathbb{N}^p\} \end{aligned} \tag{2.29}$$

où,

- L'ensemble de solutions faisables est composé par $(x, y) \in \mathbb{R}^n_+ \times \mathbb{N}^p$;
- x et y sont respectivement les vecteurs colonnes des variables continues non-négatives et des variables entières non-négatives ;
- $C \in \mathbb{R}^n$ et $F \in \mathbb{R}^p$ sont les vecteurs lignes des coefficients ;
- A et D sont les matrices de contraintes avec des coefficients réels et de dimensions $(m \times n)$ et $(m \times p)$;
- $B \in \mathbb{R}^m$ est le vecteur colonne des termes de droite des contraintes.

Au début de la méthode nous considérons $y \in \mathbb{R}^p_+$ et nous faisons une partition des ces variables en sous-ensembles Q^r, $r = 1, ..., R$. Nous considérons aussi U^r, $r = 1, ..., R-1$, des sous-ensembles auxiliaires, tel que $U^r \subseteq Q^{r+1}$ pour $r = 1, ..., R-1$.

Nous commençons par résoudre un problème (MIP$^{(1)}$) dans lequel deux sous-

ensembles (soient Q^1 et U^1) seront traités comme entiers et les autres comme réels. La solution fournira des valeurs entières de $y \in Q^1 \cup U^1$ auxquels seuls les $y \in Q^1$ seront fixées pour être utilisées dans la prochaine itération.

Ensuite, la méthode résoudra le deuxième problème (MIP$^{(2)}$) qui sera composé par les variables fixées de MIP$^{(1)}$ plus les sous-ensembles $Q^2 \cup U^2$ entiers et les autres relaxés.

L'heuristique suit le schéma de relaxation et fixation, tout en modifiant des sous-ensembles à chaque MIP, jusqu'à trouver une solution où toutes les variables sont des nombres entiers ou échouer.

La formalisation pour les r MIPs de la méthode peut être exprimé comme [82] :

$$\begin{aligned} \min_{(x,y)}\{Cx + Fy : \quad & Ax + Dy \geq B \\ & x \in \mathbb{R}^n_+ \\ & y_j = y_j^{r-1} \in \{0,1\} \quad \forall j \in Q^1 \cup ... \cup Q^{r-1} \\ & y_j \in \{0,1\} \quad \forall j \in Q^r \cup U^r \\ & y_j \in [0,1] \quad \forall j \in Q \backslash (Q^1 \cup ... \cup Q^r \cup U^r)\} \end{aligned} \tag{2.30}$$

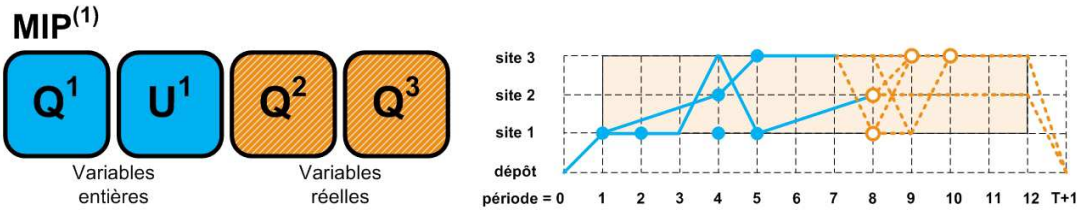
Par exemple, prenons une instance à 3 sites de fabrication et 12 périodes. Pour ce problème, nous paramétrons la taille des sous-ensembles à 25%, c'est-à-dire que nous aurons quatre sous-ensembles de variables $(Q^1, ..., Q^4)$, chacun associé à trois périodes de temps, et dont $U^1, ..., U^3$ sont les sous-ensembles auxiliaires. Nous allons travailler avec les variables de *setup* et flot au même temps $(y|g)$ et le temps d'exécution à la limite de 10 secondes pour chaque itération.

Le premier MIP contiendra le sous-ensemble $Q^1 \cup U^1$ qui comporte les variables associées aux périodes de 1 à 6 et qui seront traitées comme entières. Les autres variables seront relaxées. La figure (2.7) montre la division des sous-ensembles suivi d'une représentation en graphe temporel de la solution du problème (2.7(a)).

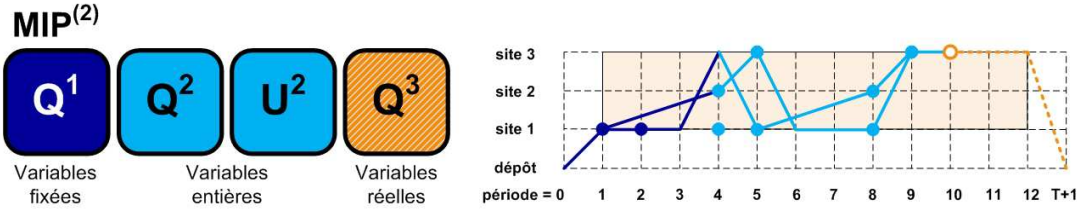
Le deuxième MIP fixe les valeurs des variables de Q^1 fournies par la solution précédente. Maintenant, le sous-ensemble $Q^2 \cup U^2$ est considéré comme entier et comporte les variables liées aux périodes de 3 à 9. Les variables associées aux périodes de 10 à 12 $(Q\backslash(Q^1 \cup Q^2 \cup U^2))$ seront relaxées (2.7(b)).

Pour la dernière itération nous aurons un MIP avec les variables de Q^1 et Q^2 fixes et le sous-ensemble $Q^3 \cup U^3$ considéré comme entier. Sa résolution fournira la solution de l'heuristique *relax-and-fix* pour cette instance. Nous n'avons pas besoin de résoudre un quatrième problème étant donnée que la dernière itération correspond au problème complet entier (2.7(c)).

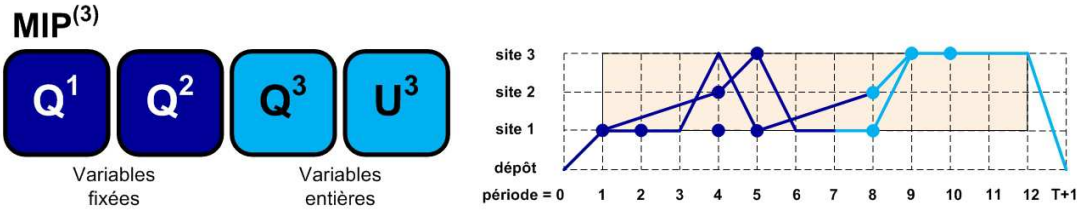
Notre version du *relax-and-fix* ajoute un traitement d'erreur pour les situations où les sous-MIPs ne donnent pas une solution réalisable. Ce traitement permet de revenir à l'étape précédente et ensuite d'ajouter d'autres blocs de variables. Ce traitement se répète jusqu'à ce que le sous-problème ait une solution pour l'ensemble entier et l'ensemble relaxé. S'il ne trouve pas de réponse, alors l'heuristique RF



(a) Exemple RF : première itération.



(b) Exemple RF : deuxième itération.



(c) Exemple RF : troisième itération.

FIGURE 2.7 – Évolution de l'heuristique *relax-and-fix* (RF).

configurée avec ces paramètres n'est pas applicable à cette instance.

2.3.2 Heuristique *Insert-and-Fix*

L'étude de l'heuristique *relax-and-fix* nous a amené à la conception de deux autres méthodes. La première, que nous appelons « heuristique d'insertion et de fixation de variables » (*Insert-and-Fix* - IF), utilise la résolution du problème de façon différente de l'heuristique RF. Au lieu de considérer toutes les variables du problème, cette heuristique résout des sous-MIPs de tailles inférieures ou égales au problème initial. C'est-à-dire que MIP$^{(1)}$ sera plus petit ou égal à MIP$^{(2)}$, qui sera plus petit ou égal à MIP$^{(3)}$, etc.

Une autre différence par rapport à l'heuristique *Relax-and-Fix* est qu'il n'y a pas de relaxation dans les sous-ensembles, mais uniquement des variables entières à chaque itération et qui seront fixées à l'itération suivante. Le regroupement des variables est le même que pour la RF, mais au lieu de relaxer les sous-ensembles l'heuristique *Insert-and-Fix* ne les considère pas dans le MIP.

Soit le problème entier mixte (2.29). Nous considérons une partition des variables y en Q^r sous-ensembles $r = 1, \ldots, R$, où U^r, $r = 1, \ldots, R-1$ sont les sous-ensembles auxiliaires ($U^r \subseteq Q^{r+1}$).

Nous commençons par résoudre MIP$^{(1)}$ où les sous-ensembles $Q^1 \cup U^1$ sont entiers,

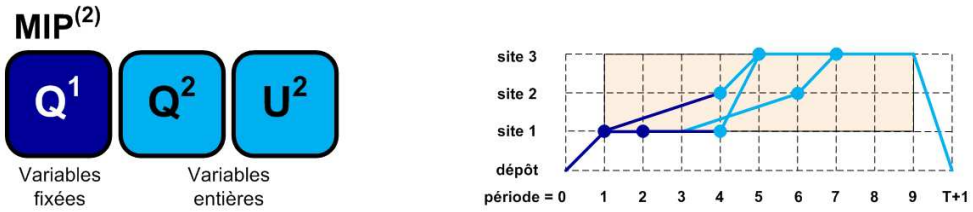
et nous ne considérons pas $Q \backslash (Q^1 \cup U^1)$ dans ce problème. Une fois que nous avons une solution, nous passons à MIP$^{(2)}$ où nous fixons les variables $y \in Q^1$ aux valeurs fournies par le problème précédent et nous ajoutons un nouveau sous-ensemble au problème, soit $Q^2 \cup U^2$.

De cette manière, nous formalisons le problème pour chacun des r MIPs comme :

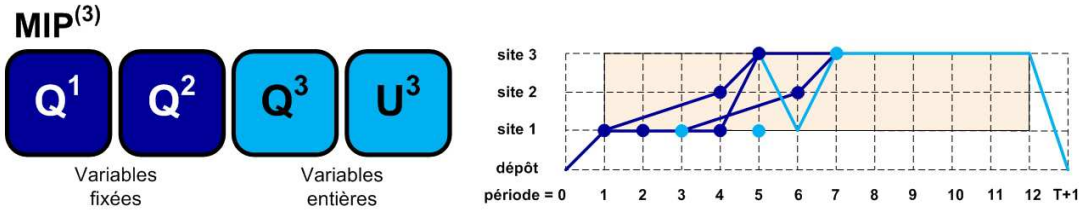
$$\begin{aligned} \min_{(x,y)} \{ C\,x + F\,y : \quad & A\,x + D\,y \geq B \\ & x \in \mathbb{R}^n_+ \\ & y_j = y_j^{r-1} \in \{0,1\} \quad \forall j \in Q^1 \cup \ldots \cup Q^{r-1} \\ & y_j \in \{0,1\} \quad \forall j \in Q^r \cup U^r \} \end{aligned} \tag{2.31}$$

Par exemple, nous prenons la même instance utilisée pour expliquer l'heuristique *relax-and-fix*. L'évolution de la méthode est illustrée par la figure (2.8). Ici nous fixons la taille des sous-ensembles à 25%, ce qui nous permettra de traiter le problème en trois MIPs : un avec la moitié des variables (2.8(a)), un avec les trois quarts (2.8(b)) et un autre avec toutes les variables (2.8(c)). Le type de variables de travail et le temps d'exécution sont respectivement $y|g$ et 10 secondes.

(a) Exemple IF : première itération.



(b) Exemple IF : deuxième itération.



(c) Exemple IF : troisième itération.

FIGURE 2.8 – Évolution de l'heuristique *insert-and-fix* (IF).

MIP$^{(1)}$ comportera les sous-ensembles $Q^1 \cup U^1$ qui contiennent les variables associées aux périodes de 1 à 6 et qui seront traitées comme entières. Les autres variables

ne sont pas considérées dans le problème, de sorte que $Q \backslash (Q^1 \cup U^1)$ n'existe pas en MIP$^{(1)}$.

Lorsque l'heuristique trouve la réponse, elle commence la composition du MIP$^{(2)}$ avec les variables fixées du sous-ensemble Q^1 et les variables de $Q^2 \cup U^2$ entières. De même que dans le problème précédent, $Q \backslash (Q^1 \cup Q^2 \cup U^2)$ n'existe pas en MIP$^{(2)}$.

Le dernier problème (MIP$^{(3)}$) prend en compte les variables de tous les sous-ensembles, avec $Q^1 \cup Q^2$ fixes et $Q^3 \cup U^3$ considérées comme entiers. La solution fournie par MIP$^{(3)}$ est la solution finale.

Cette heuristique a l'avantage de résoudre des sous-problèmes plus légers, ce qui améliore le temps d'exécution. Cependant, elle perd l'effet de « prévision » fournit par la relaxation des variables en *relax-and-fix*, ce qui est défavorable à la résolution de certains problèmes.

2.3.3 Heuristique *Fractional Relax-and-Fix*

Nous appelons cette variante du *relax-and-fix* l'« heuristique de relaxation et fixation fractionnaire » (*Fractional Relax-and-Fix* - FRF), elle travaille avec une fraction du problème initial (comme dans *insert-and-fix*) et aussi avec des relaxation des variables (comme dans *relax-and-fix*).

En considérant le problème entier mixte (2.29), nous faisons une partition des variables y (maintenant appartenant à $\mathbb{R}^p_+$) en Q^r sous-ensembles $r = 1, \ldots, R$, où U^r, $r = 1, \ldots, R-1$ sont les sous-ensembles auxiliaires ($U^r \subseteq Q^{r+1}$).

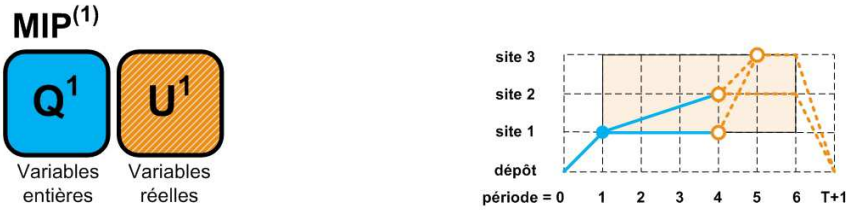
Le fonctionnement de *fractional relax-and-fix* est identique à celui de *relax-and-fix* et *insert-and-fix* : dans la première itération nous cherchons à résoudre un problème (MIP$^{(1)}$) où une partie des variables du problème initial est entière (Q^1), l'autre est relaxée (U^1) et l'autre n'est pas considérée $Q \backslash Q^1 \cup U^1$.

Une fois que nous avons une solution, nous passons à MIP$^{(2)}$ où nous fixons les variables $y \in Q^1$, et nous ajoutons un nouveau sous-ensemble au problème : $Q^2 \cup U^2$, dont une partie sera considérée comme entière (Q^2) et une autre comme réelle (U^2).

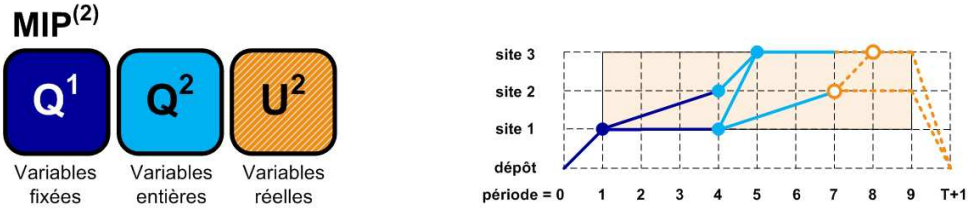
La formalisation pour les r MIPs peut être exprimée comme :

$$\begin{aligned} \min_{(x,y)} \{ Cx + Fy : \quad & Ax + Dy \geq B \\ & x \in \mathbb{R}^n_+ \\ & y_j = y_j^{r-1} \in \{0,1\} \quad \forall j \in Q^1 \cup \ldots \cup Q^{r-1} \\ & y_j \in \{0,1\} \quad \forall j \in Q^r \\ & y_j \in [0,1] \quad \forall j \in U^r \} \end{aligned} \tag{2.32}$$

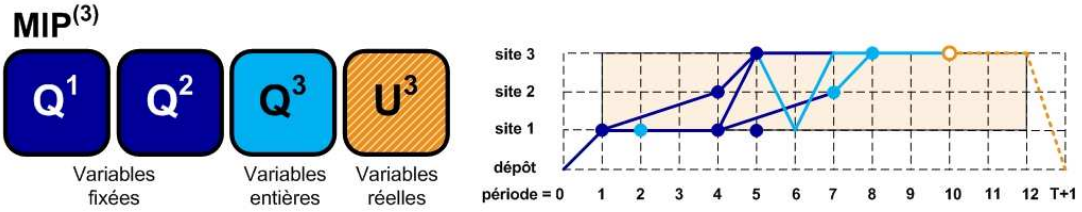
Pour illustrer la méthode, prenons le même exemple que celui utilisé pour expliquer la RF et l'IF : 3 sites de fabrication sur un horizon de 12 périodes de temps, paramétré de la même manière (taille des sous-ensembles = 25%, variables de travail = $y|g$ et temps d'exécution = 10 secondes). La figure (2.9) illustre la méthode.



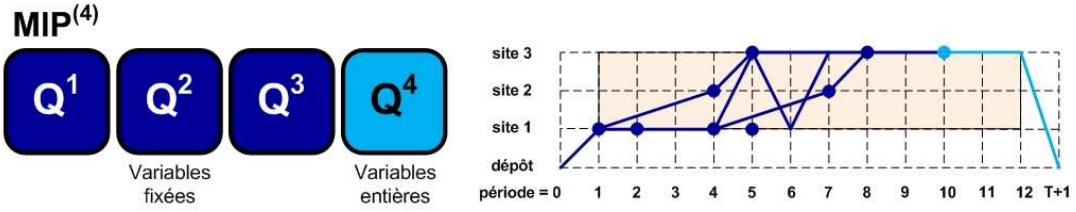
(a) Exemple FRF : première itération.



(b) Exemple FRF : deuxième itération.



(c) Exemple FRF : troisième itération.



(d) Exemple FRF : quatrième itération.

FIGURE 2.9 – Évolution de l'heuristique *fractional relax-and-fix*.

Le premier sous-problème MIP$^{(1)}$ comportera $Q^1 \cup U^1$, avec Q^1 considéré comme entier et U^1 comme relaxé. Les variables du sous-ensemble $Q \backslash Q^1 \cup U^1$ ne seront pas considérées dans ce sous-problème (2.9(a)).

Une fois que nous avons la résolution de MIP$^{(1)}$, nous fixons le sous-ensemble Q^1 entier et ajoutons $Q^2 \cup U^2$ pour former MIP$^{(2)}$. Ici Q^2 sera considéré comme entier et U^2 comme relaxé. Le sous-ensemble $(Q \backslash Q^1 \cup Q^2 \cup U^2)$ n'existe pas en MIP$^{(2)}$ (2.9(b)).

Ensuite, MIP$^{(3)}$ envisagera tout l'ensemble Q avec $Q^1 \cup Q^2$ fixé, Q^3 considéré comme entier et U^3 comme relaxé (2.9(c)).

A la différence des autres approches, cette heuristique ne s'arrête pas dans l'itération $R-1$ étant donné qu'il reste une partie à traiter comme entier. Dans la dernière itération, MIP$^{(4)}$ reprendra le problème complet en nombre entier (2.9(d)) et sa solution sera la solution finale de l'heuristique.

2.3.4 Comparaison entre les heuristiques *Relax-and-Fix*, *Insert-and-Fix* et *Fractional Relax-and-Fix*

Les trois heuristiques sont basées sur l'idée de résoudre des sous-problèmes MIP à chaque itération. Ce type de méthode est bien efficace, surtout pour les instances de grande taille, et donne des bonnes solutions en temps raisonnable de calcul. Comme nous l'avons vu, chacune de ces heuristiques traite les sous-MIPs d'une manière particulière, ce qui favorise soit le temps de calcul, soit le nombre de variables traitées, soit la qualité de la solution.

Ensuite nous montrerons un comparatif entre les trois heuristiques. Les analyses sont faites sur les paramètres de taille des sous-ensembles des variables (TS), de type de variable utilisée (TV), et de temps d'exécution par itération (TE).

L'impact causé par l'augmentation de la taille des sous-ensembles de variables se reflète directement dans le nombre d'itérations des trois méthodes et, par conséquent, dans le temps global de calcul. Plus de variables dans un sous-ensemble signifie moins d'itérations de la méthode. Cependant, cette augmentation de taille amène des effets négatifs comme la diminution de performance de l'heuristique. Comme la combinatoire augmente (en raison du nombre de variables) et que le temps disponible de calcul par itération reste le même, il est naturel que la solution du problème soit moins raffinée. Le résultat donne des gaps plus grands pour les instances de taille plus importantes.

De la même manière, une diminution excessive de la taille des sous-ensembles peut amener à l'échec des méthodes *insert-and-fix* et *fractional relax-and-fix*. Cela s'explique par le manque d'information du problème initial. Pour la *relax-and-fix*, diminuer la taille des sous-ensembles signifie augmenter le temps global de calcul, ce qui n'est pas une bonne stratégie quand il s'agit de petites instances.

En ce qui concerne le type de variables de travail (TV), nous remarquons que traiter uniquement celles de flot de véhicule (g) donne des meilleures résultats que

pour traiter celles de *setup* (y). Cependant, traiter les deux types de variables ($y|g$) a des avantages sur la manipulation individuelle de chacune.

Pour les instances de taille moins importantes, travailler uniquement avec les variables y donne de meilleurs résultats. L'explication pour cela est que la dimension la plus grande des problèmes de notre jeu d'instances est le nombre de périodes. Avoir plus de périodes dans l'horizon de planification signifie avoir plus de sommets dans le graphe du problème et, par conséquent, plus de possibilités pour construire un réseau de transport. Alors, dès que nous incluons les variables de transport dans le traitement d'un problème de petite taille, nous aurons le risque de commencer un mauvais plan de routage qui pourra difficilement être réparé par la suite de la méthode.

Au niveau des temps d'exécution (TE), l'heuristique *insert-and-fix* est la seule qui se termine, dans la plupart des cas, avant la limite imposée par chaque itération. Mais c'est évident que ce paramètre dépend des deux autres dont nous avons discuté précédemment (TS et TV).

Pour terminer cette analyse comparative des méthodes, la performance des trois méthodes au niveau des résultats est bien équilibrée. RF fournit les meilleures résultats pour tout le jeu d'instances, alors que IF et FRF fournissent des bons résultats pour les instances de grande taille et moins bons pour celles de taille moins importantes.

2.3.5 Heuristique *Diving*

L'heuristique *Diving*, ou *Dive-and-Fix* (DF) selon [99] (qui signifie « plonger et fixer »), est une stratégie qui réalise une recherche en profondeur dans l'arbre de décision, tout en utilisant des conditions de limitation et de fixation dans les variables 0-1 du problème. Ce processus permet de simplifier le problème à chaque itération une fois que les limites imposées amènent à une solution faisable.

La méthode utilise la relaxation linéaire à chaque itération. A partir du résultat obtenu, elle applique des critères pour limiter et fixer en entier les valeurs fractionnaires des variables.

Pour expliquer l'heuristique de base, considérons le problème original relaxé décrit comme suit :

$$\begin{aligned} \min_{(x,y)}\{Cx + Fy : \quad & Ax + Dy \geq B \\ & x \in \mathbb{R}^n_+ \\ & y \in [0,1]\} \end{aligned} \tag{2.33}$$

A chaque itération, l'heuristique utilise la relaxation des variables y pour trouver une solution $(\widehat{x}, \widehat{y})$. Selon un critère, l'heuristique doit choisir des variables pour les fixer à l'entier le plus proche.

Les critères utilisés par notre version du *diving* sont :

Tolérance décroissante (TD) : les variables sont fixées selon une certaine tolérance. A chaque itération, la méthode parcourt le vecteur de variables fractionnaires en les arrondissant à 1 lorsqu'elles sont suffisamment proches de cette valeur. S'il n'y a pas eu d'arrondi dans une itération, cette tolérance sera augmentée et le vecteur sera ré-analysé.

Fixation par intervalle (FI) : cette méthode définit les variables qui sont dans un intervalle $[a, b]$ en les arrondissant à la valeur 1 et en les fixant. Cet intervalle est modifié à chaque itération selon un critère donnée (par exemple, les deux plus grandes valeurs trouvées dans l'ensemble des variables fractionnaires).

Alors, considérons une exécution du *diving* avec le critère de fixation par intervalle. Soit R l'ensemble des variables 0-1 dont la valeur est fractionnaire. Tant que $R \neq \emptyset$, l'heuristique cherche les variables y à fixer dans l'intervalle $[a_r, b_r]$ (où r correspond à l'itération) jusqu'à arriver à l'une des trois conditions :

1) Il n'existe plus de variables fractionnaires ($R = \emptyset$) ;
2) Trouver une solution où la valeur de y est dans l'intervalle et l'arrondi à 1 ;
3) La solution n'est pas réalisable, alors l'heuristique à échoué.

Le processus est illustré dans la figure (2.10), où nous observons l'évolution de la solution selon le critère de fixation par intervalle. Chaque étape correspond à la solution partielle qui est associée à un sous-MIP. La figure (2.10(a)) montre une solution où les valeurs des variables sont fractionnaires (cercles en blanc). Après avoir défini l'intervalle de fixation comme $[0,83\,;\ 0,92]$ pour cette itération, l'heuristique résout un nouveau MIP et trouve le résultat de la figure (2.10(b)). La méthode se poursuit avec l'intervalle $[0,66\,;\ 0,83]$, qui donne la solution illustrée en (2.10(c)), et $[0,46\,;\ 0,66]$ qui donne la solution finale (2.10(d)).

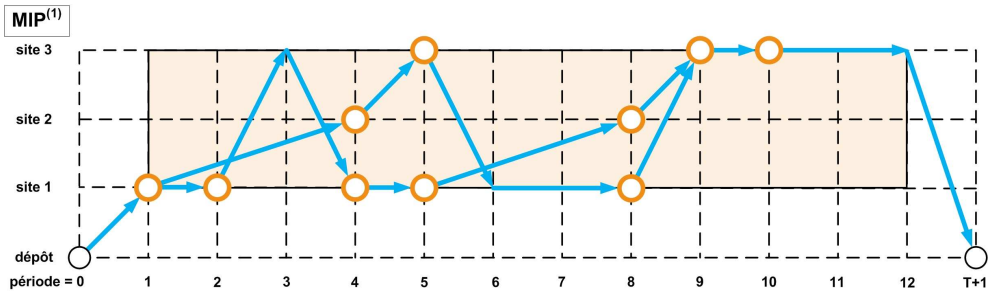
Quelques remarques sur notre version du *diving* :

1) Elle utilise uniquement les variables 0-1 de *setup* (y). Malgré les tentatives pour appliquer la méthode aux variables de flot de véhicules (g), ces variables appartiennent aux nombres entiers et la méthode a été conçue pour travailler avec des variables binaires, alors cette approche est moins efficace ;
2) Comme la décision de production est dominante dans le problème, sa fixation à 0 peut compromettre la solution. Alors la fixation a été faite avec soin en favorisant l'arrondi à la limite supérieure.

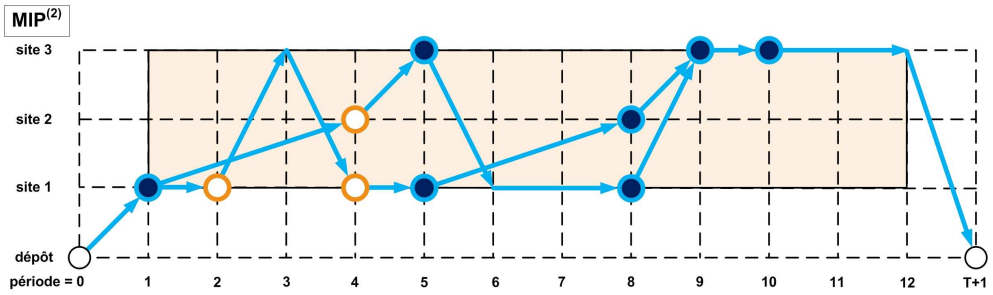
2.4 Cas particulier de production dédiée

2.4.1 Modèle mathématique

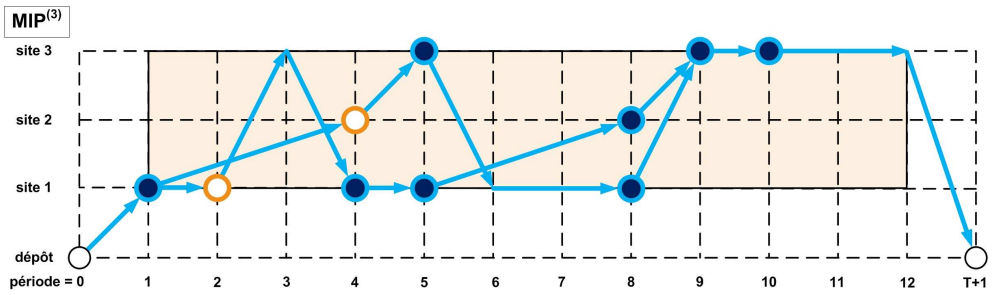
Dans ce chapitre nous traitons une variante simplifiée du problème LSVRP que nous appelons *Single-item Lot-Sizing with Vehicle Routing Problem* - (S-LSVRP).



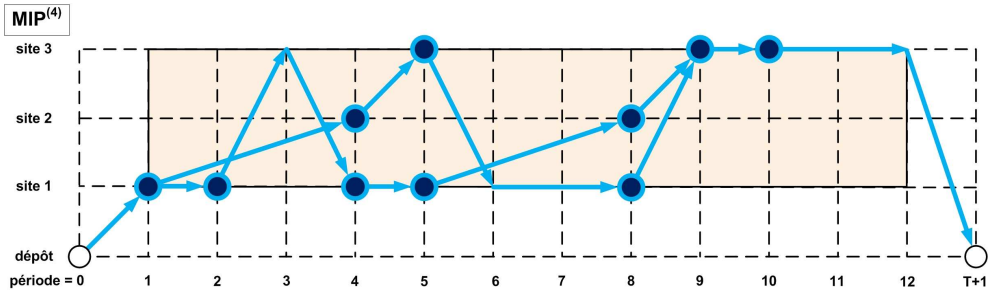
(a) Exemple DV : première itération.



(b) Exemple DV : deuxième itération.



(c) Exemple DV : troisième itération.



(d) Exemple DV : quatrième itération.

FIGURE 2.10 – Évolution d'une solution par *Diving* (DV).

Elle considère essentiellement l'hypothèse que chaque site est dédié uniquement à une tâche à accomplir (chaque site fabrique un seul type de produit, et un même produit n'est produit que sur un site). D'un point de vue économique, c'est généralement plus rentable parce que nous ne prenons pas en compte les temps et coûts associés aux reconfigurations de machines (*setup times*) sur une période. De plus, en cas de problèmes (défauts), il est plus facile de localiser la source.

Basiquement, pour ce modèle l'indice sur les produits (k) devient inutile, il disparaît donc des coûts de *setup* ($Q_{i,t}$), de stockage ($H_{i,t}$) et des coûts agrégés de production et stockage ($C_{i,tt'}$), ainsi que des variables référents à la décision de fabriquer ($y_{i,t}$) et de quantité à fabriquer ($w_{i,tt'}$).

Alors, le modèle suivant exprime le S-LSVRP :

Fonction objectif

$$\min \sum_{\substack{(i,j)\in E \\ i,j\in V'}} \sum_{t=1}^{T} \left[I_{ij}\, g_{ij,t} + \sum_{k\in U} J_{ij}^{k}\, f_{ij,t}^{k} \right] + X\, v + \sum_{i\in V'} \sum_{t=1}^{T} \left[Q_{i,t}\, y_{i,t} + H_{i,t} \sum_{k\in U} r_{i,t}^{k} + \sum_{t'=1}^{t} C_{i,t't}\, w_{i,t't} \right] + \sum_{i\in V'} \sum_{t=1}^{T+1} C_{i,0t}\, w_{i,0t} \quad (2.34)$$

Sous les contraintes

$$\sum_{t=1}^{T+1} w_{i,0t} = S_{i,0} \quad \forall i \in V' \quad (2.35)$$

$$r_{i,t-1}^{k'} + \sigma_{i,t}^{k'} = r_{i,t}^{k'} + A_{k}^{k'} \sum_{t'=t}^{T} w_{i,tt'} \quad \forall k', k \in U, \forall i \in V_k', \forall t = 1, \ldots, T \quad (2.36)$$

$$\sum_{t'=t}^{T} w_{i,tt'} \leq M_{i,t}\, y_{i,t} \quad \forall i \in V', \forall t = 1, \ldots, T \quad (2.37)$$

$$\sum_{t'=t}^{T} w_{i,tt'} \geq y_{i,t} \quad \forall i \in V', \forall t = 1, \ldots, T \quad (2.38)$$

$$w_{i,tt'} \leq D_{i,t'}\, y_{i,t} \quad \forall i \in V_F', \forall t, t' = 1, \ldots, T, \forall t' \geq t \quad (2.39)$$

$$w_{i,0t'} + \sum_{t=1}^{t'} + w_{i,tt'} = \tau^k_{i,t'} \quad \forall k \in U, \forall i \in V'_k, \forall t' = 1, \ldots, T \tag{2.40}$$

$$\sum_{i \in V'_k} \tau^k_{i,t} = D_{j,t} \quad \forall k \in F, \forall j \in V', \forall t = 1, \ldots, T \tag{2.41}$$

$$w_{i,tt'},\ r^k_{i,t},\ \sigma^k_{i,t},\ \tau^k_{i,t},\ f^k_{ij,t},\ g_{ij,t} \in \mathbb{N} \quad \forall k \in U, \forall i \in V'_k, \forall t, t' = 1, \ldots, T, t \leq t' \tag{2.42}$$

$$y_{i,t} \in \{0, 1\} \quad \forall i \in V', \forall t = 1, \ldots, T \tag{2.43}$$

Les premières contraintes du bloc de production (2.35) concernent la consommation du stock initial disponible au début de la production sur chaque site ($S_{i,0}$). Les contraintes (2.36)-(2.41) expriment, dans cet ordre, la consommation et le flux des composants sur l'horizon de production, le respect de la capacité, l'obligation de production à l'existence d'un *setup*, la limite de cette production en produits finaux, si les produits qui sortent proviennent d'une fabrication où d'un stock (où des deux), et la satisfaction de la demande. Les deux dernières contraintes (2.42)-(2.43) correspondent aux demandes des variables.

2.4.2 Une heuristique simple pour le cas particulier S-LSVRP

Dans cette sous-section nous présentons l'« heuristique de Fusion de Séquences de Production » (*Production Sequences Fusion* - PSF). Bien que nous sommes sur un système de production multi-sites et multi-produits, avec des nomenclatures en série et d'assemblage, cette heuristique repose sur la particularité que chaque produit k est affecté à un seul site i et vice-versa.

La PSF a pour principe d'améliorer un plan de production « juste-à-temps » (*just-in-time* - JIT), c'est-à-dire fait au plus tard de manière à réduire les coûts de stockage, en fusionnant des séquences de production. Nous désignons par « arbre de production » ces séquences, parce qu'elles représentent la composition d'un produit dans une structure d'arbre (les assemblages nécessaires pour obtenir un composant). De même cette structure représente la dépendance entre les sites qui vont fabriquer ces produits.

Dans un premier temps, nous effectuons l'hypothèse que seules les expéditions directes de produits entre sites sont autorisées, c'est-à-dire le trajet d'un véhicule doit être fait du site où il vient de collecter les produits vers le site de livraison, sans passer par des sites intermédiaires. Ceci oblige le véhicule à livrer puis à retourner vide soit au dépôt, soit sur un autre site pour recharger.

L'heuristique utilise comme structure de données auxiliaires deux listes : l'une composée de requêtes pour des produits (r) et l'autre d'ordres de fabrication (o). Les requêtes sont faites par un site demandeur et chacune consiste en un quintuplet (r_{ini}, i, j, t, q) qui contient respectivement la requête initiale r_{ini} (les premières sont les demandes connues du problème), le site producteur i, le site demandeur j, le temps t auquel le produit doit être livré et la quantité q demandée. De cette manière,

chaque ordre correspond à une requête et est affecté à un site producteur. Un ordre consiste en un triplet (r, t', q'), contenant respectivement la requête r associée, la période de temps t' à laquelle cet ordre doit être exécuté ($t' \leq t$) et la quantité q' à produire (limité par la capacité du site à la période t').

Un ordre de production peut être propagé en remontant en arrière sur les périodes de temps. La raison pour cela c'est la limitation de la capacité par des contraintes sur chaque site. Ainsi, une requête peut générer plusieurs ordres. La figure (2.11) montre un schéma où un site j, à la période 10, a une demande de 25 unités. Cette requête a été prise en compte par le site i et a été transformée en un ordre de production qui doit être exécuté à la période 8 (en supposant que la durée $L_{ij} = 2$). Etant donné que la capacité de production du site i est limitée à 10 unités par période, la requête de j devrait commencer à être traitée par i dans les périodes précédentes, plus précisément à partir de la période 5.

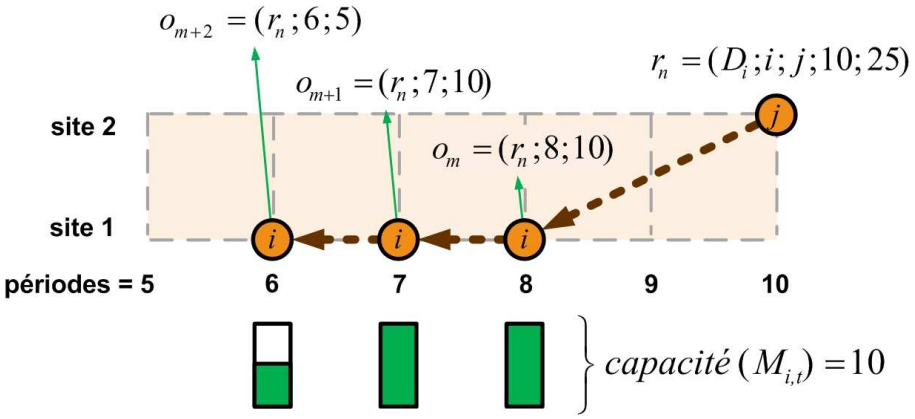


FIGURE 2.11 – Propagation d'un ordre de fabrication.

L'heuristique PSF est divisée en trois étapes, en *pipeline*, dont deux se répètent en boucle tant qu'il existe un gain dans la solution du problème. La première consiste en la construction d'un plan de production réalisable et son amélioration par fusion d'arbres. La deuxième étape correspond à la construction d'un schéma de transport qui respecte le plan de production trouvé par l'étape précédente. Dans la troisième étape l'heuristique refait un plan de production mieux adapté au plan de transport. Ces deux dernières étapes se répètent de manière à ce que la production et le transport convergent en une solution équilibrée entre ces deux aspects.

Première étape : construction et amélioration d'un plan de production initial

La première procédure reprend l'idée de la production juste-à-temps qui vise à organiser la production sans espace de temps supplémentaire pour permettre une autre décision (de production, de stockage, de transport, etc.). Bien qu'une telle construction tend à réduire les quantités stockées, elle conduit en pratique à l'aug-

mentation des opérations de *setup* et de transport. Une amélioration du plan sera donc nécessaire.

Pendant la génération des ordres de fabrication, deux situations peuvent se produire :

1) Si les ordres générés dépassent la capacité, l'algorithme génère une (ou plusieurs) nouvelle requête (sur le même site) pour la période précédente avec l'excédent de demande ;
2) Si les ordres ne dépassent pas la capacité, l'algorithme cherche si pour ces ordres il est nécessaire de consommer des composants. Dans ce cas, une nouvelle requête sur les composants est adressée au site correspondant.

Pour essayer d'améliorer le plan de production initial, PSF utilise deux procédures qui fusionnent des arbres de production en réduisant les effets causés par la production au plus tard. Dans le graphe correspondant au problème cela se traduit par une réduction des arcs de transport. Les procédures sont :

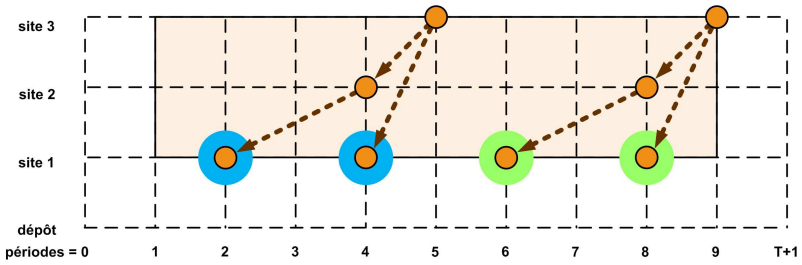
Direct Shift **:** c'est le mouvement le plus simple de cette heuristique. Il consiste à parcourir la liste des requêtes de production r (r_{ini}, i_r, j_r, t_r, q_r) du plan généré par la construction en cherchant des éléments susceptibles d'être fusionnés. Ce mouvement travaille sur les requêtes (et ses ordres et des requêtes dérivés) liées directement au site de composants basiques (1^{er} niveau de la production où il n'y a pas de consommation d'autres produits). Nous parcourons la liste de la fin vers le début de la planification. Chaque fois qu'un mouvement est validé, la requête au plus tard et ses dérivées sont déplacées. Les périodes sont ajustées et tous les coûts impliqués sont recalculés.

La figure (2.12) montre le processus de fusion des arcs directs. Dans la figure (2.12(a)) deux arbres de production impliquent trois sites (k, j, et i), dont le site i, en évidence dans la figure, est le producteur de composants basiques). La figure (2.12(b)) représente le résultat après l'application de *direct shift*. Le nombre de *setup* est réduit (cercles) en i et les sites j et k utilisent maintenant un stockage (exprimé par des arcs horizontaux).

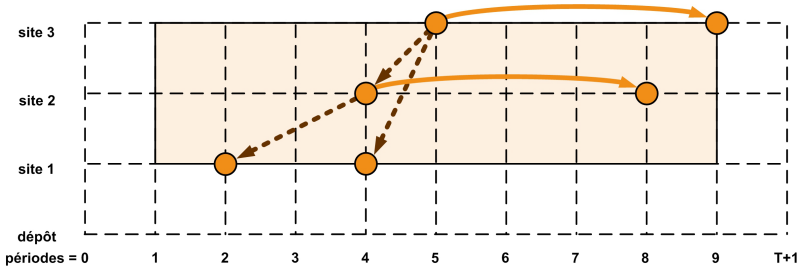
Pendant l'exécution ce mouvement utilise deux autres procédures simples de vérification : une qui actualise la capacité disponible dans tous les sites, et une autre qui énumère le nombre d'arcs existant dans le graphe du problème après la fusion. Cette dernière procédure est utilisée dans le critère de validation du mouvement de fusion.

Subtree Shift **:** ce mouvement généralise le *direct shift*. Il permet également de réduire le nombre d'arcs du graphe associé en respectant la capacité et les limites de temps de production. La différence est qu'il cherche non seulement les liaisons entre les sites du 1^{er} niveau, mais aussi sur tous les arbres et sous arbres de niveaux ultérieurs.

Le mouvement utilise des requêtes r (r_{ini}, i_r, j_r, t_r, q_r) et nous cherchons à réaliser la fusion des arbres (et sous-arbres) de production en partant du



(a) Arbres de production.



(b) Résultat de la fusion des arbres en (a).

FIGURE 2.12 – Fusion d'arbres de production par *Direct Shift*.

dernier niveau de la production, où la demande concerne les produits finaux, vers le premier, où nous avons les produits basiques. Pour chaque niveau, nous parcourons de la fin de l'horizon de planification vers le début.

La figure (2.13) montre le processus avant (2.13(a)) et après (2.13(b)) une fusion de deux arbres. Ensuite, l'arbre résultant est fusionné avec d'autres sous-arbres du problème (2.13(c)). Dans (2.13(d)) nous avons le résultat final.

Ce mouvement utilise cinq autres opérations : deux qui ont déjà été présentées avant (vérification de capacité et énumération des arcs) et trois nouvelles qui sont :

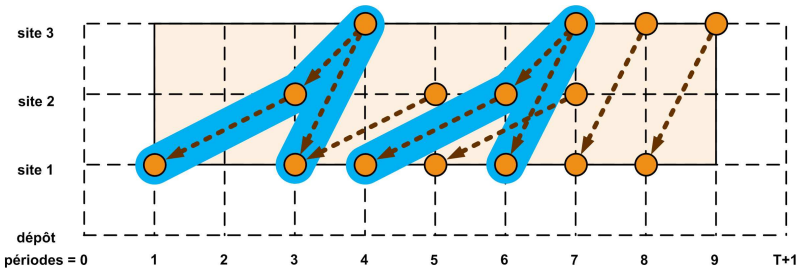
Analyser un arbre : cette procédure parcourt tous les sommets depuis le site d'assemblage (racine) en vérifiant qu'il y ait une capacité disponible pour agréger la production d'un autre arbre.

Fusionner des arbres : une fois que deux arbres sont analysés, cette procédure parcourt tous les sommets en déplaçant la production de l'arbre le plus proche de la fin de l'horizon de production vers l'autre.

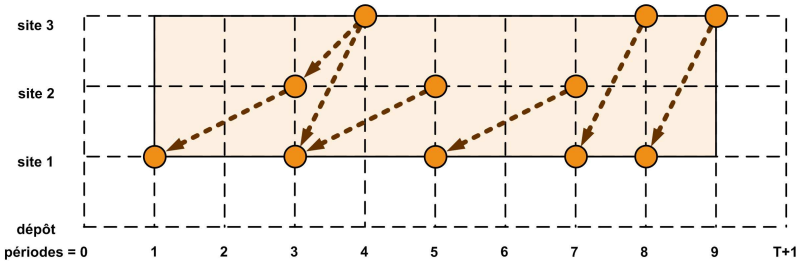
Séparer des arbres : après la fusion, si la nouvelle solution n'amène pas une amélioration, cette procédure parcourt l'arbre crée par la fusion en séparant chaque sommet.

Deuxième étape : construction d'un schéma de transport

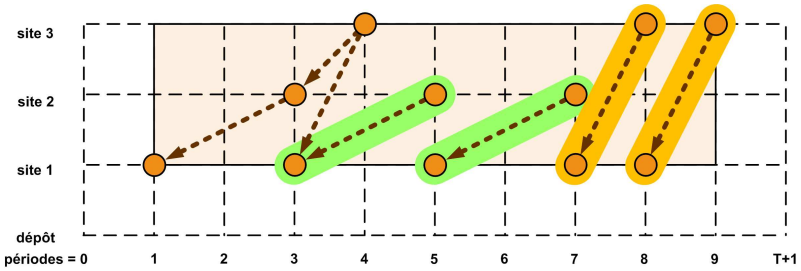
L'étape suivante de l'heuristique est de construire un schéma de routage de manière à minimiser l'utilisation des véhicules, c'est à dire les coûts fixes et les coûts



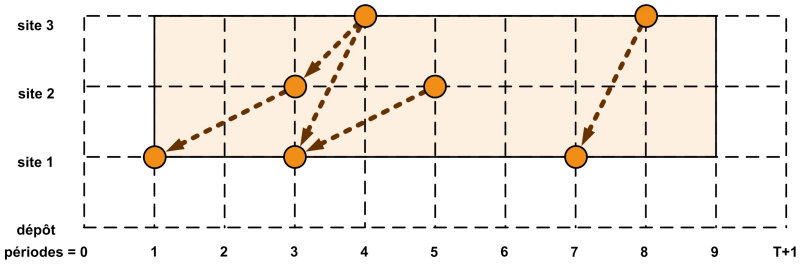
(a) Détection d'arbres à fusionner.



(b) Résultat de la première fusion.



(c) Détection de sous-arbres à fusionner.



(d) Résultat de la deuxième fusion.

FIGURE 2.13 – Fusion d'arbres de production par *Subtree Shift*.

unitaires de transport. Cette étape comprend deux sous-étapes qui sont exprimées comme des problèmes de flot minimum. Les sous-étapes 1 et 2 consistent, respectivement à :

1) Déterminer le nombre minimal de véhicules nécessaires ;
1) Étant donné ce nombre minimal de véhicules, identifier les tournées dont le coût de transport est aussi minimal.

A l'issue de la première étape, nous disposons d'une planification de la production sur chaque site représentant la capacité ainsi que les délais minimaux de transport. La planification ayant été faite au plus tard, aucun retard dans le transport n'est permis. Chaque composant doit être transporté au moment de sa mise en disponibilité. Un véhicule doit donc le charger pour le livrer immédiatement au destinataire.

Dans le graphe temporel, certaines connexions (arcs) entre sites sont obligatoires pour la livraison d'un composant. Nous devons organiser un ensemble de transport de coût minimal couvrant ces arcs. Le graphe étant temporel, il est donc cyclique.

De ce fait, le problème sous-jacent de transport, qui est à la base un problème de type VRP, se simplifie en un problème de flot de coût minimal couvrant certains arcs. Le coût de transport fait intervenir deux éléments : un coût fixe par véhicule et un coût fixe par arc utilisé.

La figure (2.14) illustre la résolution de la partie transport. (2.14(a)) est un exemple de graphe de planification lequel doit être couvert par un minimum de transporteurs (2.14(b)). Dès que le nombre de véhicules est fixé, une couverture à coût minimal doit être trouvée (2.14(c)).

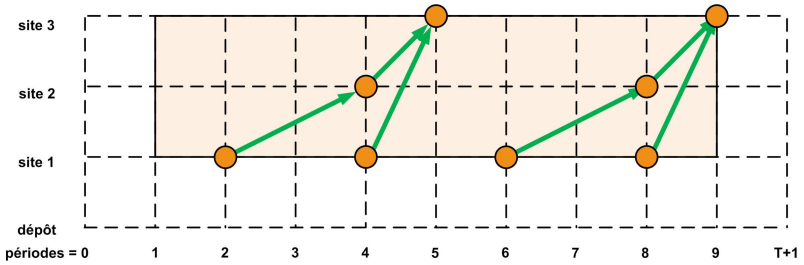
Troisième étape : construction d'un nouveau plan de production

Après la résolution du sous-problème de transport, nous obtenons une couverture minimale d'arcs qui respecte le flot de production imposée par la première étape. Nous avons donc une solution complète, à la fois sur l'aspect production et sur l'aspect transport. Cependant, comme la construction a été centrée sur la production, la composante transport peut être très dégradée. Pour essayer d'améliorer cette solution, nous fournissons cette couverture au logiciel d'optimisation Cplex pour qu'il puisse identifier un nouveau plan de production qui respecte le transport imposé.

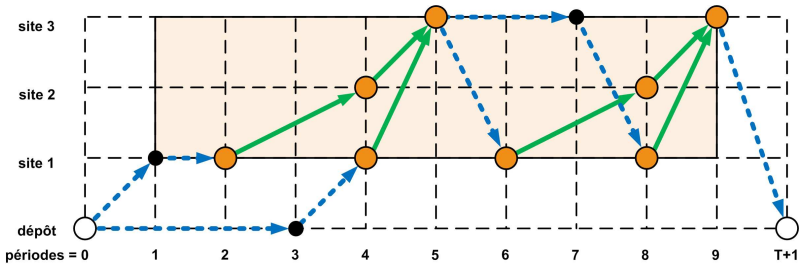
Nous fixons une partie des variables de flot de véhicules ($g_{ij,t}$) présentes dans les contraintes de capacité de flot. De cette manière, le problème ne prend pas en compte les autres contraintes concernant le transport puisqu'elles sont implicitement satisfaites et il n'a plus qu'à identifier un meilleur plan de production.

Une fois que le nouveau plan de production est obtenu, l'heuristique revient à la deuxième étape. Ce processus se répète tant qu'il y a une amélioration de solution. En l'absence d'amélioration, l'heuristique se termine.

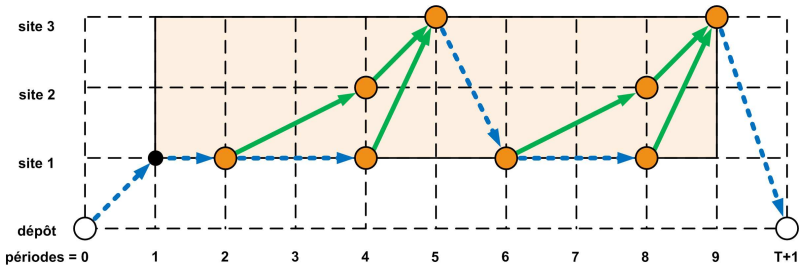
La figure (2.15) montre le processus de construction d'un nouveau plan de pro-



(a) Exemple de planification.

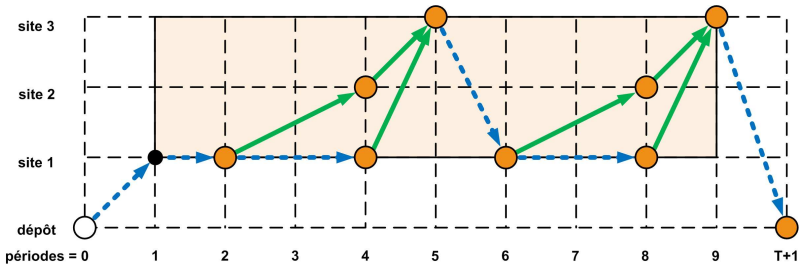


(b) Réseau initial de transport.

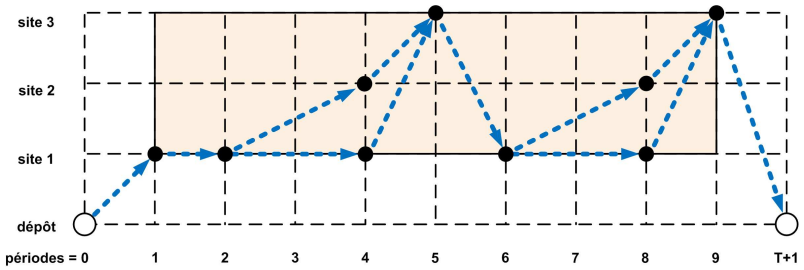


(c) Réseau amélioré à coût minimal.

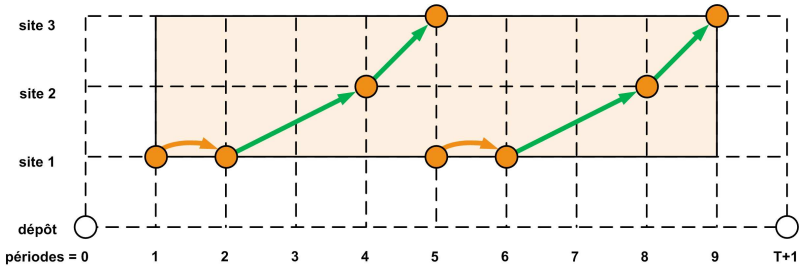
FIGURE 2.14 – Construction du réseau de transport.



(a) Solution initiale.



(b) Réseau initial de transport.



(c) Plan de production amélioré.

FIGURE 2.15 – Amélioration du plan de production.

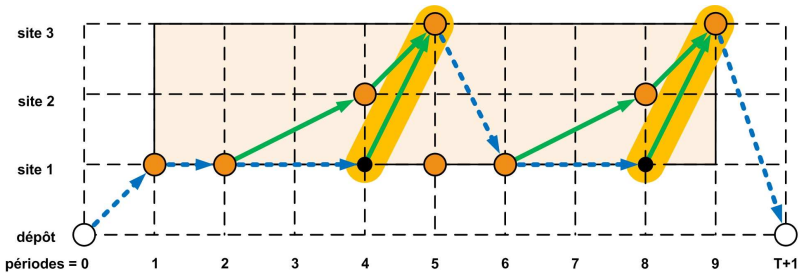
duction. En commençant par une solution où la production et le transport sont déjà définis (2.15(a)), le système utilise le réseau de transport (2.15(b)) pour construire un nouveau plan de production (2.15(c)). Dans les figures, les arcs pleins représentent les véhicules chargés et les arcs en pointillés représentent les véhicules vides (à l'exception des arcs courbes en (2.15(c)) qui représentent les stocks).

Après un dernier ajustement de production, il est fort possible qu'il y ait un transport résiduel vide dans la solution, c'est-à-dire des arcs de flot nul. Afin de supprimer ces arcs, PSF fixe leurs variables de transport associées à zéro (variables $f^k_{ij,t}$ du modèle mathématique). Alors, les coûts sont recalculés et le transport est ajusté, ce qui nous donne la solution finale pour une exécution de cette heuristique.

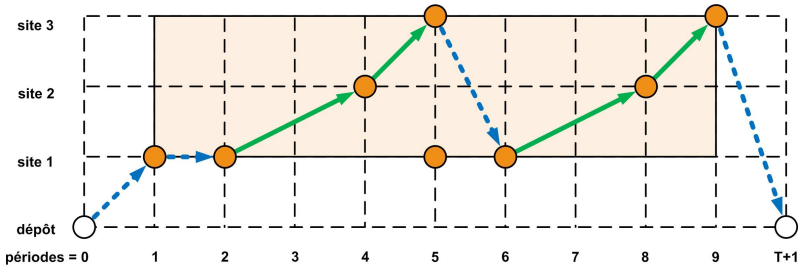
Les figures (2.16(a)) et (2.16(b)) suivantes illustrent, respectivement, le graphe du problème avant et après la suppression des arcs vides (en évidence en 2.16(a)).

2.4.2.1 Schéma algorithmique

Le fonctionnement de l'heuristique PSF peut être exprimé de manière simplifié selon le schéma algorithmique suivant :



(a) Solution final avec des arcs vides.



(b) Solution final après suppression des arcs vides.

FIGURE 2.16 – Suppression des arcs vides.

PRODUCTION-SEQUENCES-FUSION (Demandes connues, coûts impliqués)

1. A partir des requêtes et des ordres de production dérivés des demandes, exécuter les étapes suivantes :
2. **Étape 1 : construire un plan de production**
3. Tant qu'il existe une réduction des coûts de production, faire :
4. Fusionner des arbres de production par *Subtree shift* et *Direct shift* ;
5. Calculer les coûts de production et vérifier s'ils ont diminué ;
6. **Étape 2 : construire un plan de transport**
7. Résoudre un problème de **flux minimum** pour trouver le nombre v de véhicules qui respecte le plan de production ;
8. Résoudre un problème de **flux de coût minimum**, avec v véhicules, pour trouver un plan de transport qui respecte le plan de production trouvé en « Étape 1 » ;
9. **Étape 3 : construction d'un nouveau plan de production**
10. Refaire le plan de production de manière à ce qu'il soit mieux ajusté au plan de transport trouvé en « Etape 2 » ;
11. Vérifier s'il existe des arcs résiduels de flot nul et les supprimer ;
12. Si la solution a été améliorée
13. Retourner à « Étape 2 » ;

14. Sinon
15. Finaliser PSF et retourner la solution ;

Ensuite, dans le prochain chapitre, nous allons présenter les résultats obtenus par les heuristiques présentées dans ce chapitre. Plusieurs analyses sur les méthodes et les problèmes seront discutées.

Chapitre 3

Analyse et résultats

Dans ce chapitre nous allons montrer les résultats obtenus avec les heuristiques présentées dans le chapitre précédent. Notre jeu d'instances comprend plusieurs problèmes de type *lot-sizing with vehicle routing* (LSVRP), aussi bien que des problèmes de type *single-item* LSVRP (S-LSVRP). Chaque instance a tout d'abord été résolue à l'aide de Cplex afin de connaître la solution optimale. Nous limitons le temps d'exécution à 3 heures. Si la solution optimale n'est pas trouvée en 3 heures nous gardons la meilleure solution.

Initialement nous allons montrer comment les instances ont été générées et comment elles sont classées selon les caractéristiques du problème (3.1). Dans la section suivante (3.2), nous allons faire une analyse comparative entre le modèle LSVRP et sa reformulation par rapport aux références de production et de stock. La section (3.3) compare les valeurs des résultats obtenus par les heuristiques proposées avec les solutions fournies par Cplex. Pour finir ce chapitre, nous présentons en (3.4) les difficultés rencontrées pendant l'implémentation des méthodes.

3.1 Génération des instances

La création des instances a été faite de manière aléatoire au niveau des coûts. Les paramètres d'entrée sont : le type de nomenclature, le nombre total de sites, le nombre de produits par site, le nombre de périodes et la capacité des machines. La génération est faite de manière à garantir une production réalisable. Pour les autres données du problème, nous avons utilisé la gamme de valeurs de Armentano *et al.* [6] dans l'intervalle $[a,\, b]$ avec une distribution uniforme $U[a,\, b]$:

- Coût unitaire de production $(P_{i,t}^k) : U[1,5;\ 2,5]$;
- Coût fixe de production $(Q_{i,t}^k) : U[5,0;\ 95,0]$;
- Coût unitaire de stockage $(H_{i,t}^k) : U[0,2;\ 0,4]$;
- Taux de consommation de composant $(A_k^{k'}) : U[1;\ 3]$ unités ;
- Consommation de machine par produit $(B_i^k) : U[1;\ 3]$ unités ;
- Consommation de machine par *setup times* $(Z_i^k) : U[1;\ 3]$ unités ;

– Durée de transport (L_{ij}) : $U[1;\ 3]$ périodes de temps ;
– Coût unitaire de transport (J_{ij}^k) : $U[12,0;\ 45,0]$;
– Coût d'arc (I_{ij}) : $U[5,0;\ 95,0]$.

La capacité de transport (W) par véhicule est considérée infinie (pour les raisons citées dans les hypothèses) et pour chaque véhicule utilisé nous avons une pénalité (X) de coût 1000.

Pour garantir qu'une instance soit réalisable nous avons conçu un algorithme de génération de demande. Cet algorithme calcule toutes les demandes (représentés par les variables $d_{i,t}^k$) de produits k pour un site i et une période t, et retourne la demande maximale pour les sites d'assemblage de produits finaux ($k \in U_F$), tout en respectant les limites de capacité ($M_{i,t}$) et en prenant en compte les taux de consommation ($A_k^{k'}$) de composants et de machine (B_i^k) et le temps nécessaire pour rendre disponible un produit (L_{ij}, la distance minimale en périodes d'un site à l'autre). Toutefois, en raison du taux de consommation accumulé, la demande pour les produits non-finaux ($k \notin U_F$) sera élevée. Soit $t_{DP(i)}$ la période de début de la production en i, calculé par l'algorithme du plus long chemin pour trouver le chemin critique (c'est-à-dire, le chemin lequel un retard dans une activité implique un retard global [5]). Alors, pour le calcul de la demande en produit k pour chaque site en chaque période est fait de la manière suivante :

Maximiser

$$\sum_{k \in U_i} \sum_{i \in V_F} \sum_{t=1}^{T} d_{i,t}^k \tag{3.1}$$

Sous les contraintes

$$d_{i,t}^k = \sum_{k' \in U_j} \sum_{\substack{j \in V' \\ t \leq T - L_{ij}}} A_k^{k'}\ d_{j,t+L_{ij}}^{k'} \quad \forall k \in U_i, \forall i \in V', \forall t = t_{PD(i)}, \ldots, T \tag{3.2}$$

$$B_i^k\ d_{i,t}^k \leq M_{i,t} \quad \forall k \in U_i, \forall i \in V', \forall t = \ldots T \tag{3.3}$$

$$\sum_{k \in U_i} \sum_{t=1}^{T} d_{i,t}^k > 1 \quad \forall i \in V' \tag{3.4}$$

$$d_{i,t}^k \geq 0 \quad \forall k \in U, \forall i \in V', \forall t = 1, \ldots, T \tag{3.5}$$

La fonction objectif (3.1) vise à trouver la demande maximale possible pour les produits finaux à chaque période. Les contraintes (3.2) définissent les demandes dépendantes pour les produits non finaux. Les contraintes (3.3) imposent le respect de la capacité disponible en chaque site à chaque période. Les contraintes (3.4) imposent que au moins 1 produit final doit être demandé en chaque site d'assemblage,

et les contraintes (3.5) définissent le domaine des variables.

Après avoir trouvé les demandes maximales, nous avons le choix de générer un problème avec ces demandes (donc le système utilisera la capacité maximale de production permise), d'utiliser une pourcentage de ces demandes dans les instances, de redéfinir les demandes de manière aléatoire entre une quantité minimale et la quantité trouvé par l'algorithme, d'utiliser d'autres types de fonctions pour générer les demandes mais qui respectent les demandes maximales connues, etc.

Nous avons généré plusieurs types d'instances, chacune avec des caractéristiques différentes par rapport au nombre de sites, de produits, etc. Alors, pour décrire ces instances nous avons adopté des sigles qui les définissent selon le :

- Type de nomenclature : la nomenclature utilisée par le problème est du type en série (N_S) ou d'assemblage (N_A) ;
- Nombre de produits par site : si le problème comprend des sites multi-produits, le nombre de produits (k) par site est précisé (k_P) ;
- Nombre de sites d'assemblage final : si les produits finaux sont assemblés en plusieurs sites i (i_A) ;
- Présence de *setup times* (ST) ;
- Présence de contraintes d'expédition directe de produits entre le site fabricant et le site consommateur (ED).

Pour obliger les véhicules à faire des livraisons directes, nous avons ajouté les contraintes

$$\sum_{\substack{k \in U_i \\ k \notin U_j}} f^k_{ij,t} = 0 \quad \forall (i,j) \in E, \forall t = 1, \ldots, T \tag{3.6}$$

qui expriment qu'un composant k fabriqué dans le site i ($k \in U_i$) ne peut être envoyé vers le site j que si ce produit est impliqué dans la production d'un produit $k' \in U_j$.

3.2 Analyse des formulations

Nous avons utilisé le langage de programmation C++ avec la bibliothèque *Ilog Cplex* pour la résolution des programmes linéaires. Nous avons aussi utilisé la bibliothèque graphique *OpenGL* pour générer une visualisation des résultats. Les expérimentations ont été réalisées sur une machine munie d'un processeur Intel Core2 Quad 2GHz, 4Go de mémoire vive et système Windows XP. Nous avons également ajusté certains paramètres de Cplex, comme le temps d'exécution à 10800 secondes (3 heures) et la tolérance de convergence à 1e-006.

Les instances générées sont de type LSVRP et S-LSVRP, et comportent de 3 à 6 sites de fabrication. Chaque site fabrique deux types de produits, qui consomment des composants qui viennent d'autres sites. Les horizons de planification sont de 12, 24, 36, 48 et 60 périodes de temps. Pour chaque combinaison (site × période) nous

avons généré 10 problèmes et pour chacun nous avons traité 4 situations différentes :

- Problème avec temps de reconfiguration et expédition directe des produits (ST_ED) ;
- Problème avec temps de reconfiguration et sans l'expédition directe des produits (ST) ;
- Problème sans temps de reconfiguration et avec l'expédition directe des produits (ED) ;
- Problème sans temps de reconfiguration ni expédition directe des produits.

Les problèmes de type S-LSVRP ont été traités uniquement dans les deux dernières situations.

3.2.1 Comparaison entre formulations

Dans cette sous-section nous allons comparer les résultats des deux formulations du problème LSVRP présentées dans le chapitre 2. Pour cela nous avons choisis la cas particulier de production dédiée (S-LSVRP), ce qui simplifie les analyses des résultats pour cette comparaison. Alors le problème choisi est de type nomenclature d'assemblage, avec un seul produit par site, un seul site d'assemblage du produit final et sans la contrainte d'expédition directe ($N_A_1_P_1_A$).

Le problème correspond à une chaine de production dans laquelle chaque produit (à part ceux du premier niveau) dépendent d'un ou plusieurs composants. Nous avons le tableau (3.1) qui montre les résultats pour les instances à 3 et 4 sites de fabrication (instances de petite et moyenne tailles). Les deux premières colonnes du tableau concernent la formulation naturelle (fn) du problème, où nous avons les coûts de production et stockage séparés, et montrent les moyennes des gaps entre la solution trouvée par Cplex en 3h d'exécution (S) et la solution du problème relaxé (RL) et les temps d'exécution. Les deux colonnes suivantes montrent les mêmes types de résultats mais pour la formulation étendue (fe), où les coûts de production et de stockage sont groupés en un seul coût. Enfin, les dernières colonnes comparent les résultats entre les deux formulations et montrent la différence entre les gaps et les temps d'exécution.

En analysant le tableau (3.1), nous voyons que les gaps entre la solution entière et la relaxation sont un peu plus serrés dans ce modèle étendu. Ce « gap d'intégrité » montre que l'espace de solution du deuxième modèle est plus restreint que celui de la formulation naturelle. En ce qui concerne le temps d'exécution, celui du modèle étendu devient supérieur car cette formulation nécessite beaucoup plus de variables.

Malgré son temps de calcul pour les instances de plus grande taille, nous avons choisi la deuxième formulation comme base pour nos méthodes. Notre choix est dû au fait de que la relaxation linéaire est meilleure dans cette formulation, ce qui devient un avantage pour nos heuristiques.

TABLE 3.1 – Comparaison entre formulations($N_A_1_P_1_A$).

	fn		fe		**différences entre fn et fe**	
Inst	**gap**$_{(\%)}$	**tmp**$_{(s)}$	**gap**$_{(\%)}$	**tmp**$_{(s)}$	**gap**$_{(fn-fe)}$	**tmp**$_{(\frac{fe}{fn})}$
3×12	19,43	0,50	18,85	0,68	0,59	1,86
3×24	10,87	75,59	10,81	188,51	0,06	1,74
3×36	7,84	7556,87	7,62	3320,50	0,20	0,66
3×48	5,43	3253,80	5,16	2958,38	0,24	1,07
3×60	5,07	3443,02	4,81	3215,43	0,28	0,99
4×12	32,32	2,35	32,32	3,39	0,41	1,58
4×24	13,45	2506,81	13,33	4089,78	0,12	2,09
4×36	10,23	5463,10	9,98	5178,68	0,27	0,96
4×48	6,97	3084,67	6,60	4639,32	0,21	1,44
4×60	6,46	1754,10	6,25	4622,42	0,19	2,42

3.3 Résultats des heuristiques

Ensuite nous présentons les résultats obtenus par les heuristiques proposées. Nous avons un total de 2000 instances qui ont été regroupées selon les critères présentés précédemment. Alors nous allons traiter 10 ensembles de 200 instances chacun, identifiés par :

- $N_A_2_P_1_A$
- $N_A_2_P_1_A_ED$
- $N_A_2_P_1_A_ST$
- $N_A_2_P_1_A_ST_ED$

pour le LSVRP, et par :

- $N_S_1_P_1_A$
- $N_S_1_P_1_A_ED$
- $N_A_1_P_1_A$
- $N_A_1_P_1_A_ED$
- $N_A_1_P_2_A$
- $N_A_1_P_2_A_ED$

pour le S-LSVRP.

3.3.1 Calibrage des heuristiques

Avant de tester nos heuristiques sur la totalité des instances, nous avons fait un calibrage qui consiste à évaluer toutes les combinaisons de paramètres, de toutes les méthodes, pour trouver quels sont les meilleurs réglages. Le jeu d'instances choisi pour le test correspond à des problèmes de petite et moyenne tailles (3 et 4 sites) de type $N_A_1_P_1_A$, en production dédiée. Ce problème sert de base pour la plupart

TABLE 3.2 – Tableau de calibrage de *Relax-and-Fix* (RF).

	RF (y)			**RF (g)**			**RF (y|g)**		
	30"	**60"**	**90"**	**30"**	**60"**	**90"**	**30"**	**60"**	**90"**
20%	0,02	0,02	0,02	0,10	0,10	0,10	0,21	0,21	0,21
	54,11	6,32	3,95	8,44	3,12	1,30	6,39	1,40	**0,84**
25%	0,02	0,02	0,02	0,07	0,07	0,07	0,10	0,10	0,10
	-	10,21	-	5,85	4,39	1,15	5,90	1,99	1,00
34%	0,01	0,01	0,01	0,00	0,00	0,00	0,01	0,01	0,01
	51,60	9,51	6,09	12,56	5,00	1,91	12,15	7,39	1,59

des autres sur lesquels nous avons travaillé. Les résultats heuristiques que nous montrons dans cette section ont été trouvés en utilisant les meilleurs paramètres. Cependant, l'heuristique de fusion de séquences de production (PSF) n'a pas eu besoin de calibrage et son exécution est telle que décrite dans le chapitre 2.

Ensuite nous avons les tableaux de calibrage (3.2), (3.3) et (3.4) qui illustrent l'influence des paramètres pour les heuristiques *Relax-and-Fix* (RF), *Insert-and-Fix* (IF) et *Fractional Relax-and-Fix* (FRF). Ils montrent l'impact des variations sur le temps d'exécution (TE) à 30, 60 et 90 secondes, la taille des sous-ensembles (TS) à 20%, 25% et 34% de l'horizon de planification, et du type de variable de décision à fixer (TV), qui sont celles de *setup* (y), de flot de véhicules (g), et les deux ($y|g$). Les lignes du tableau correspondent à differents tailles de sous-ensembles et les colonnes à differrents temps d'exécution imposés à Cplex à chaque itération. Nous présentons les gaps (en pourcentage entre la valeur de la solution trouvée par Cplex en 3 heures d'exécution et la valeur obtenue par l'heuristique) sur l'instance la plus petite (en haut, 3 sites par 12 périodes) et sur l'instance la plus grande (en bas, 4 sites par 60 périodes) en nombre de variables parmis 100 problèmes de type $N_A_1_P_1_A$. Dans ces tableaux nous pouvons constater que le meilleur réglage pour les 3 méthodes est de travailler avec les deux variables et d'utiliser 20% de taille de sous-ensemble avec 90 secondes de temps d'exécution à chaque itération. Cette paramètrage favorise plutôt les problèmes de grande taille.

D'autres réglages donnent ont été faites avec les instances à 5 et 6 sites de fabrication. Pour celles-ci nous avons décidé d'utiliser 25% de taille de sous-ensembles avec 180 secondes de temps d"'exécution. Les variables traitées par les méthodes sont y et g ensembles ($y|g$).

Le tableau (3.5) montre les solutions fournies par l'heuristique *Diving* (DV) pour le jeu d'essai. Le tableau compare les gaps et les temps d'exécution pour les deux critères de l'heuristique : fixation par intervalle (FI) et tolérance décroissante (TD). Les colonnes correspondent aux limites de temps de 30, 60 et 90 secondes pour chaque itération de la méthode. Les lignes expriment le gap par rapport à la solution du Cplex, pour l'instance la plus petite (en haut) et la plus grande (en bas) en nombre de variables.

TABLE 3.3 – Tableau de calibrage de *Insert-and-Fix* (IF).

	IF (y)			**IF (g)**			**IF (y|g)**		
	30"	**60"**	**90"**	**30"**	**60"**	**90"**	**30"**	**60"**	**90"**
20%	0,10	0,10	0,10	1,11	1,11	1,11	1,17	1,17	1,17
	22,95	37,15	3,99	5,44	1,43	0,45	2,80	1,18	**0,22**
25%	0,05	0,05	0,05	0,71	0,71	0,71	0,71	0,71	0,71
	29,43	10,56	3,34	2,94	2,03	0,72	2,27	1,95	0,54
34%	0,03	0,03	0,03	0,91	0,91	0,91	0,91	0,91	0,91
	58,37	9,26	3,76	16,44	4,06	1,12	10,03	1,75	1,04

TABLE 3.4 – Tableau de calibrage de *Fractional Relax-and-Fix* (FRF).

	FRF (y)			**FRF (g)**			**FRF (y|g)**		
	30"	**60"**	**90"**	**30"**	**60"**	**90"**	**30"**	**60"**	**90"**
20%	0,15	0,15	0,15	1,16	1,16	1,16	1,39	1,39	1,39
	-	7,45	3,45	4,09	0,92	1,28	2,37	0,61	**0,36**
25%	0,07	0,07	0,07	1,09	1,09	1,09	1,13	1,13	1,13
	-	43,50	4,64	10,71	1,86	0,38	3,10	0,95	0,39
34%	0,01	0,01	0,01	1,14	1,14	1,14	0,62	0,62	0,62
	41,86	29,86	4,72	4,73	1,70	0,74	2,73	1,17	0,56

TABLE 3.5 – Comparaison entre les paramètres de *Diving* pour $N_A_1_P_1_A$.

	FI			**TD**		
	30"	**60"**	**90"**	**30"**	**60"**	**90"**
$\mathbf{gap}_{(\%)}$	0,59	0,59	0,59	0,76	0,76	0,76
	-	4,35	3,46	-	-	4,36

Nous voyons que le meilleur critère pour cette heuristique est le critère de fixation par intervalle. Les gaps sont semblables à ceux des autres heuristiques en utilisant uniquement les variables de *setup*.

3.3.2 Résultats pour le problème LSVRP

Dans cette section nous allons montrer les résultats fournis par les heuristiques *relax-and-fix*, *insert-and-fix*, *fractional relax-and-fix* et *diving* pour 4 variantes du problème LSVRP. Nous avons divisé les résultats en quatre tableaux concernant les problèmes sans et avec temps de reconfiguration (3.6 et 3.7, respectivement) et les variantes respectives avec contraintes d'expédition directe (3.8 et 3.9). Dans chaque tableau nous avons des colonnes pour chacune de ces heuristiques et des colonnes pour les moyennes des gaps ($gap(\%)$) et des temps ($tps(s)$) d'exécution de chaque méthode. Ces premiers sont obtenus selon la formule $gap = (ValH - ValCpx) * 100/ValCpx$, où $ValH$ est la valeur heuristique et $ValCpx$ est la valeur

fournie par Cplex à une limite de 3 heures d'exécution. Le temps exprimé par $tps = (TpsH/TpsCpx)$, où $TpsH$ est le temps de calcul de la méthode et $TpsCpx$ est le temps de calcul utilisé par Cplex, indique la vitesse de la méthode par rapport au solveur.

En analysant le tableau de calibrage pour l'heuristique RF (3.2), pour chaque type de variable à fixer nous observons une amélioration des résultats sur les grandes instances lorsque le temps d'exécution augmente et lorsque la taille des sous-ensembles diminue. Pour les petites instances, temps d'exécution n'a aucune influence car la solution optimale est toujours obtenue au but de 30 secondes sur les sous-problèmes. En revanche taille des sous-ensembles joue un rôle positif : en fait sur les instances de petit taille, la méthode trouve la plupart des solution fournies par Cplex, même lorsque nous augmentons la taille du sous-problème. De ce fait, il est préférable de lui donner une valeur élevée. Sur des instances de grande taille, la méthode n'obtient pas toujours la solution optimale, même lorsque la taille des sous-ensembles est faible. Par conséquent, augmenter la taille des sous-ensembles augmente les risques d'obtenir des solution éloignées de l'optimum.

Dans le tableau (3.3) qui présente l'analyse des résultats pour l'heuristique *insert-and-fix*, nous voyons que le comportement général est globalement le même que pour l'heuristique *relax-and-fix*. Cette heuristique est plus efficace que RF pour les problèmes de grande taille. La raison est que le nombre global de variables dans chaque sous-problème de IF est inférieur à celui de *relax-and-fix*, ce qui améliore la capacité de résolution du solveur dans le temps imposé à chaque itération. Cependant, l'absence des variables relaxées dans *insert-and-fix* invalide l'effet de « prévision » sur les périodes suivantes. Cela constitue un désavantage sur les petites instances car le traitement des variables de décision dans un petit problème est plus délicat et l'absence de l'une d'entre elles peut compromettre la solution globale.

Comme pour l'heuristique *relax-and-fix*, la meilleur configuration de paramètres pour l'heuristique *insert-and-fix* est de temps d'exécution à 90 secondes, taille des sous-ensembles à 20%, et variables de type *setup* et flot de véhicules traitées en même temps. Ce réglage est également le meilleur pour la *fractional relax-and-fix*, comme nous le constatons dans le tableau (3.4). Le fonctionnement de cette heuristique garde des caracteristiques des méthodes *relax-and-fix* et *Insert-and-Fix*.

Les tests de cette section pour l'heuristique *diving* montrent que cette heuristique est inapte à résoudre des grosses instances car elle travaille sur un nombre infime de variables par itération, au contraire des autres méthodes. Le critère de fixation choisi est la fixation par intervalle. Dans la plupart des instances nous observons que les gaps ont un ordre de croissance logique, avec des petites variations, qui suivent la taille et la difficulté du problème.

Dans le tableau (3.6) nous avons les résultats pour le problème $N_A_2_P_1_A$. Pour ce premier jeu d'instances, l'heuristique *insert-and-fix* montre des avantages par rapport aux autres. Elle est plus efficace dans la résolution des instances de tailles moyennes et grandes que dans la résolution des petites instances, celles-ci

TABLE 3.6 – Solutions de RF, IF, FRF et DV pour ($N_A_2_P_1_A$).

	Relax-and-Fix		***Insert-and-Fix***		***Fractional RF***		***Diving***	
Inst	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**
3×12	2,09	2,80	4,46	1,77	2,45	2,41	0,24	2,00
3×24	0,31	0,47	1,87	0,22	1,64	0,41	0,38	0,61
3×36	0,06	0,12	0,28	0,03	0,46	0,08	0,34	0,16
3×48	0,07	0,11	0,09	0,03	0,26	0,07	0,52	0,10
3×60	0,52	0,11	0,36	0,07	0,18	0,09	2,72	0,14
4×12	7,01	1,18	1,45	0,89	0,76	1,05	0,67	0,91
4×24	0,91	0,38	3,03	0,51	3,84	0,66	0,37	0,73
4×36	0,49	0,12	0,73	0,06	1,37	0,07	4,31	0,32
4×48	1,17	0,07	1,10	0,05	1,97	0,07	4,01	0,23
4×60	1,87	0,08	0,73	0,08	1,81	0,09	-	0,29
5×12	6,91	0,83	0,40	0,72	0,61	0,53	2,59	0,30
5×24	2,42	0,09	5,55	0,04	5,08	0,05	20,22	0,16
5×36	1,56	0,09	2,43	0,05	2,91	0,07	-	0,21
5×48	1,70	0,09	2,91	0,07	7,07	0,09	-	0,07
5×60	4,16	0,08	4,96	0,06	6,92	0,07	-	0,04
6×12	6,70	0,05	4,84	0,03	7,42	0,03	8,49	0,05
6×24	7,99	0,05	9,01	0,02	10,01	0,03	-	0,07
6×36	6,49	0,06	6,23	0,04	8,99	0,06	-	0,05
6×48	7,88	0,08	6,76	0,07	13,96	0,08	-	0,02
6×60	9,35	0,09	6,15	0,06	31,75	0,07	-	0,02

résolues de manière plus compétant par *diving*. Cependant, *diving* ne résout pas les problèmes de taille plus importantes.

Le tableau (3.7) présente les solutions pour le problème $N_A_2_P_1_A_ST$. En comparant avec les résultats du problème précédent, nous voyons que *insert-and-fix* est toujours la plus efficiente et *diving* la moins. Toutefois l'ajout de temps de reconfiguration induit une amélioration de performance de la part des heuristiques *relax-and-fix* et *fractional relax-and-fix*. Cette augmentation d'efficacité vient du fait que ces deux heuristiques travaillent également la relaxation sur une partie des ses variables. La relaxation linéaire, déjà commentée dans le chapitre 2, donne un effet de prévision de la production pour les périodes suivantes. Si nous ajoutons le fait que les reconfigurations augmentent la combinatoire du problème, alors *insert-and-fix* tend à dégrader les solutions une fois qu'elle n'utilise pas la relaxation linéaire des variables.

En utilisant les heuristiques RF et IF comme exemples nous pouvons exprimer l'impact des reconfigurations dans le problème, au niveau du gap et du temps d'exécution, par les figures (3.1) et (3.2). Ces figures montrent, respectivement, la relation entre la taille du problème et le gap, et entre la taille du problème et le temps d'exécution.

TABLE 3.7 – Solutions de RF, IF, FRF et DV pour (N_A_2_P_1_A_ST).

	Relax-and-Fix		*Insert-and-Fix*		*Fractional RF*		*Diving*	
Inst	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**
3×12	2,05	2,53	4,47	1,68	2,39	2,36	0,23	1,72
3×24	0,28	0,56	1,92	0,21	1,68	0,38	0,43	0,58
3×36	0,09	0,12	0,33	0,04	0,47	0,09	0,41	0,18
3×48	0,10	0,07	0,12	0,02	0,27	0,05	0,63	0,06
3×60	0,12	0,06	0,38	0,03	0,26	0,05	3,29	0,10
4×12	7,01	1,36	1,45	1,24	0,76	1,23	0,67	1,17
4×24	0,86	0,28	3,06	0,20	3,93	0,42	0,28	0,39
4×36	0,82	0,10	0,58	0,03	1,79	0,10	5,42	0,27
4×48	1,35	0,04	1,27	0,03	2,21	0,04	19,14	0,18
4×60	1,95	0,04	0,62	0,04	1,18	0,04	-	0,11
5×12	7,09	0,93	0,17	0,60	0,48	0,82	2,94	0,32
5×24	3,50	0,08	5,01	0,04	3,96	0,05	16,83	0,13
5×36	1,53	0,06	1,66	0,03	2,55	0,04	-	0,14
5×48	2,96	0,06	3,27	0,05	6,80	0,05	-	0,08
5×60	4,52	0,05	5,10	0,03	11,56	0,04	-	0,01
6×12	6,65	0,03	7,76	0,02	5,55	0,02	5,72	0,03
6×24	5,20	0,04	15,59	0,01	10,25	0,02	33,49	0,07
6×36	4,08	0,05	5,48	0,03	6,79	0,04	-	0,03
6×48	9,32	0,04	9,44	0,04	10,24	0,05	-	0,02
6×60	13,11	0,05	8,67	0,03	14,70	0,04	-	0,01

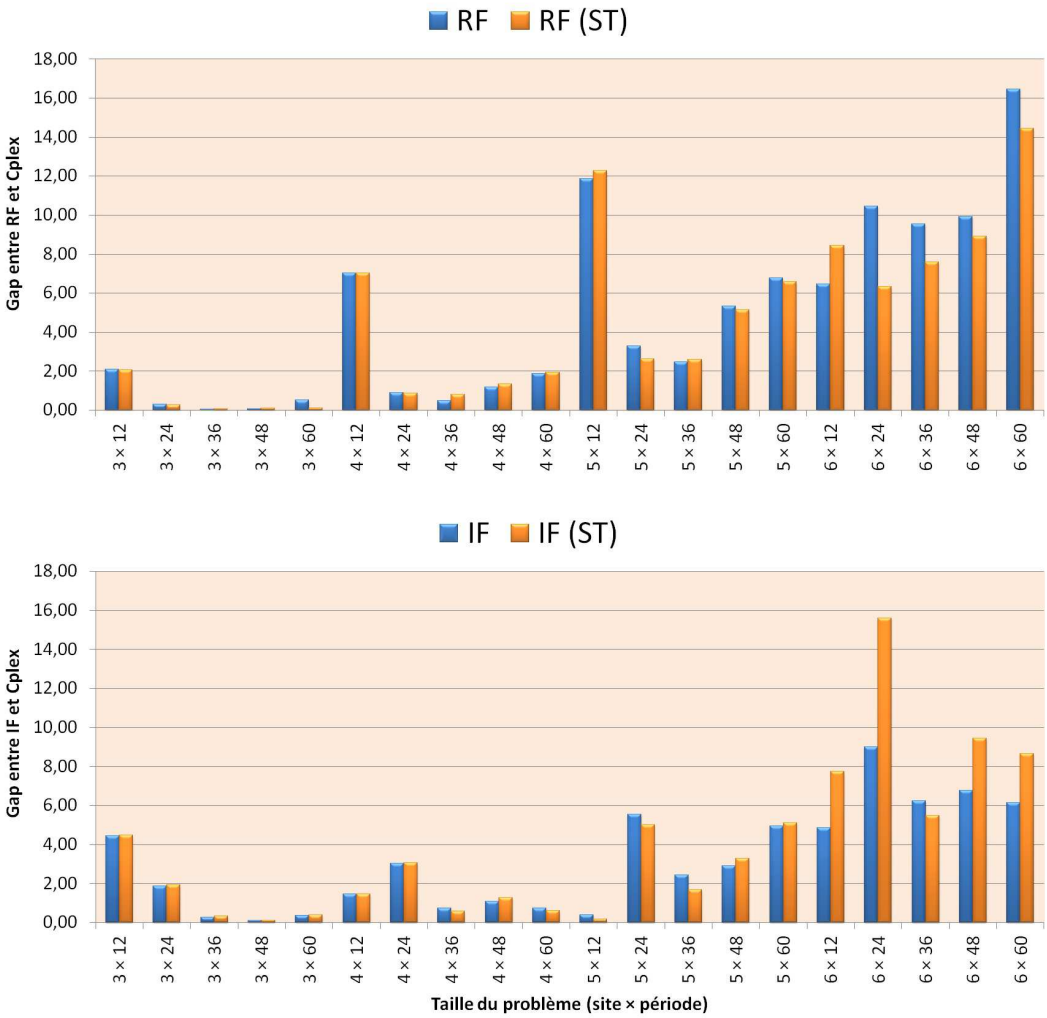


FIGURE 3.1 – Evolution du gap en fonction de la taille du problème.

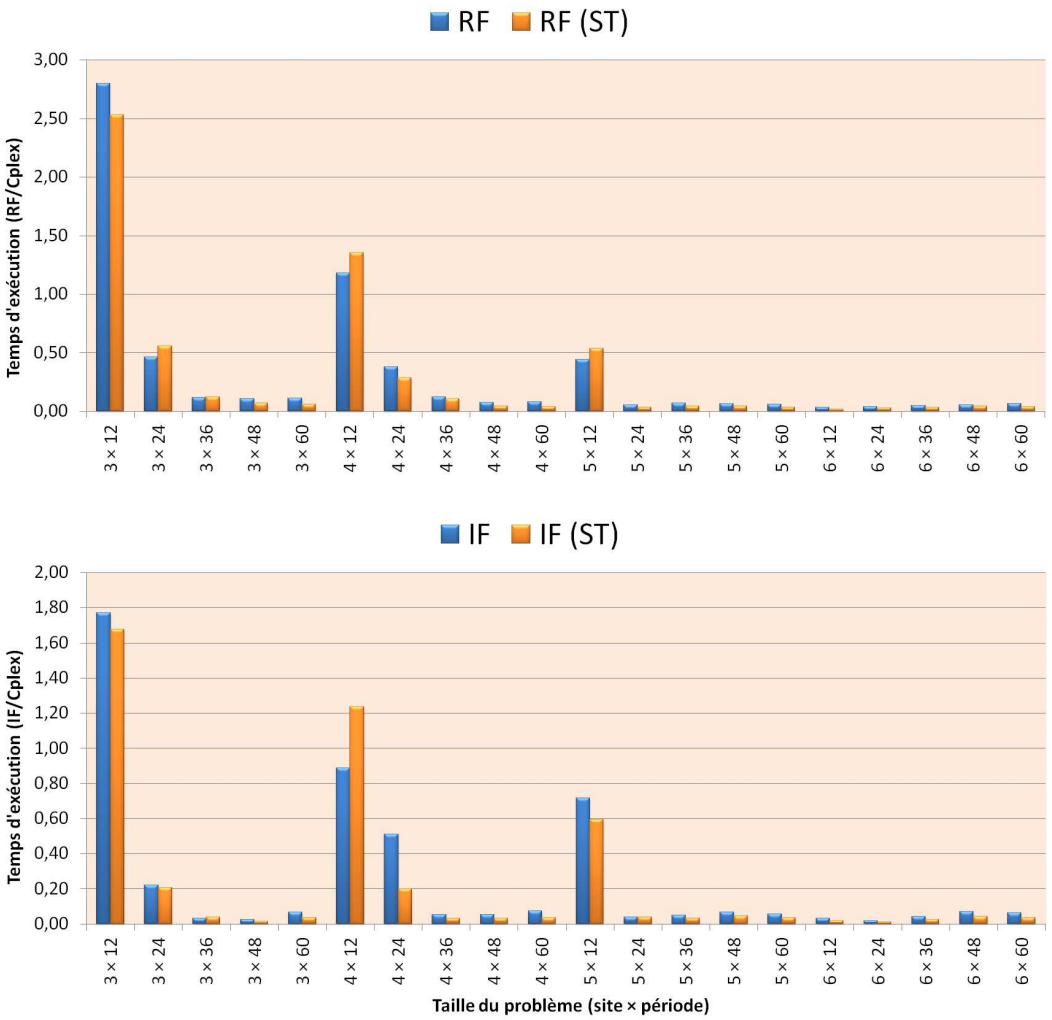


FIGURE 3.2 – Evolution du temps d'exécution en fonction de la taille du problème.

Ensuite nous allons présenter les résultats pour les deux problèmes avec des contraintes d'envoi direct et avec et sans temps de reconfiguration (respectivement les problèmes $N_A_2_P_1_A_ST_ED$ et $N_A_2_P_1_A_ED$). Alors le tableau (3.8) correspond au premier type de problème.

Nous remarquons que l'heuristique *insert-and-fix* est globalement dominante et que *diving* continue à donner de bons résultats pour les petites instances. Cependant, nous voyons que les gaps entre *relax-and-fix* et Cplex s'approchent des gaps entre *insert-and-fix* et Cplex ce qui est dû aux contraintes d'envoi direct. Ce fait peut être vu de manière plus claire dans les figures (3.3) et (3.4), qui montrent la relation entre la taille des instances, le gap et le temps d'exécution fournis par *relax-and-fix* et *insert-and-fix*.

TABLE 3.8 – Solutions de RF, IF, FRF et DV pour ($N_A_2_P_1_A_ED$).

	Relax-and-Fix		*Insert-and-Fix*		*Fractional RF*		*Diving*	
Inst	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**
3×12	2,09	2,72	4,48	1,60	2,45	2,32	0,24	1,77
3×24	0,31	0,41	1,87	0,20	1,64	0,41	0,38	0,60
3×36	0,06	0,10	0,27	0,03	0,45	0,08	0,38	0,16
3×48	0,07	0,09	0,10	0,03	0,25	0,06	0,51	0,08
3×60	0,53	0,10	0,30	0,06	0,36	0,08	2,83	0,16
4×12	6,70	1,15	1,51	1,01	0,76	1,14	0,66	1,20
4×24	0,84	0,32	2,85	0,23	3,47	0,56	0,45	0,59
4×36	0,40	0,08	0,79	0,04	1,47	0,05	6,41	0,19
4×48	0,62	0,08	0,88	0,07	1,67	0,08	19,00	0,36
4×60	1,26	0,08	0,56	0,07	1,34	0,08	-	0,31
5×12	7,16	0,73	0,51	0,61	0,57	0,50	3,27	0,27
5×24	2,69	0,08	5,21	0,04	3,48	0,05	16,20	0,11
5×36	1,12	0,09	1,81	0,05	3,48	0,07	-	0,26
5×48	2,66	0,09	3,29	0,06	8,71	0,09	-	0,18
5×60	3,00	0,08	4,88	0,06	8,37	0,07	-	0,04
6×12	6,93	0,05	5,99	0,03	5,64	0,03	7,21	0,06
6×24	7,32	0,05	8,21	0,02	12,75	0,03	7,61	0,08
6×36	3,80	0,06	5,15	0,04	11,31	0,06	-	0,03
6×48	3,64	0,08	6,83	0,07	14,67	0,08	-	0,02
6×60	11,80	0,07	11,22	0,06	17,81	0,07	-	0,02

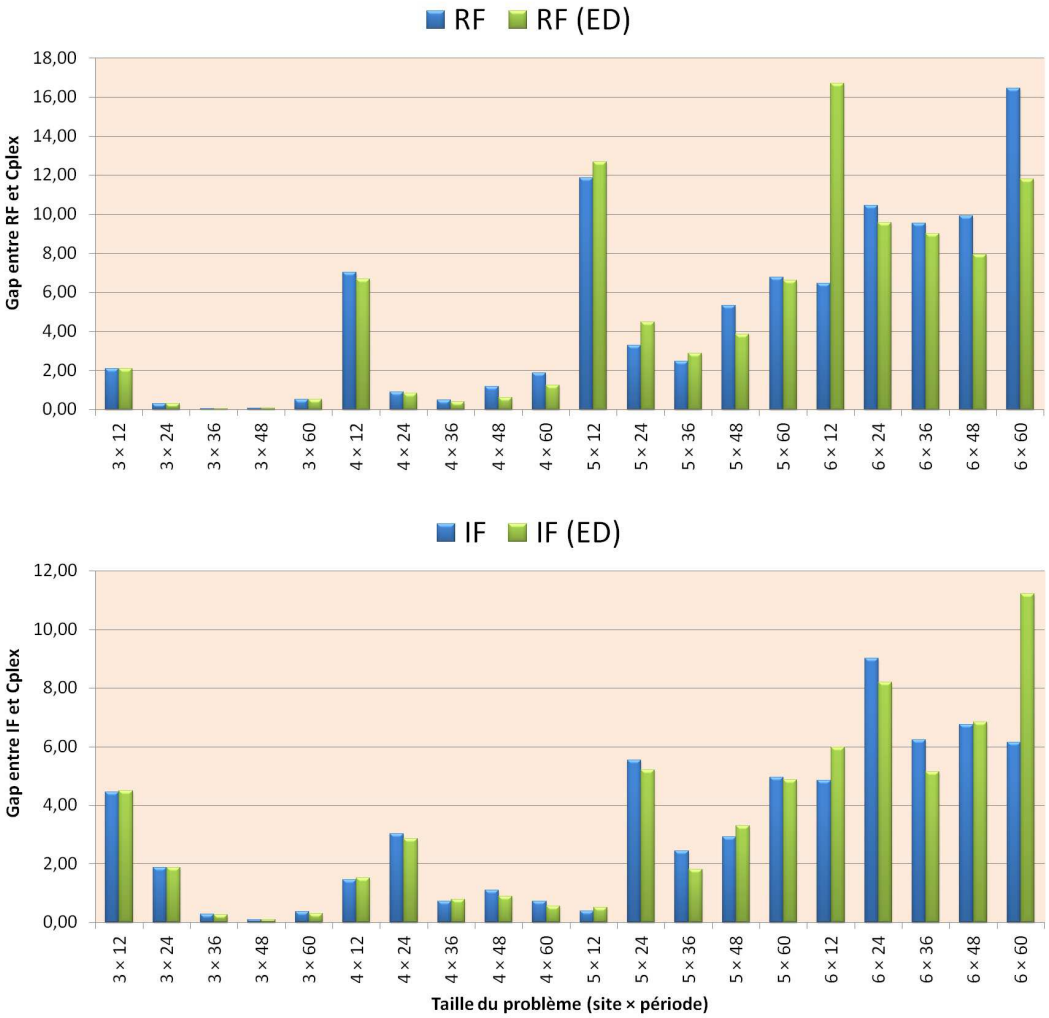


FIGURE 3.3 – Evolution du gap en fonction de la taille du problème.

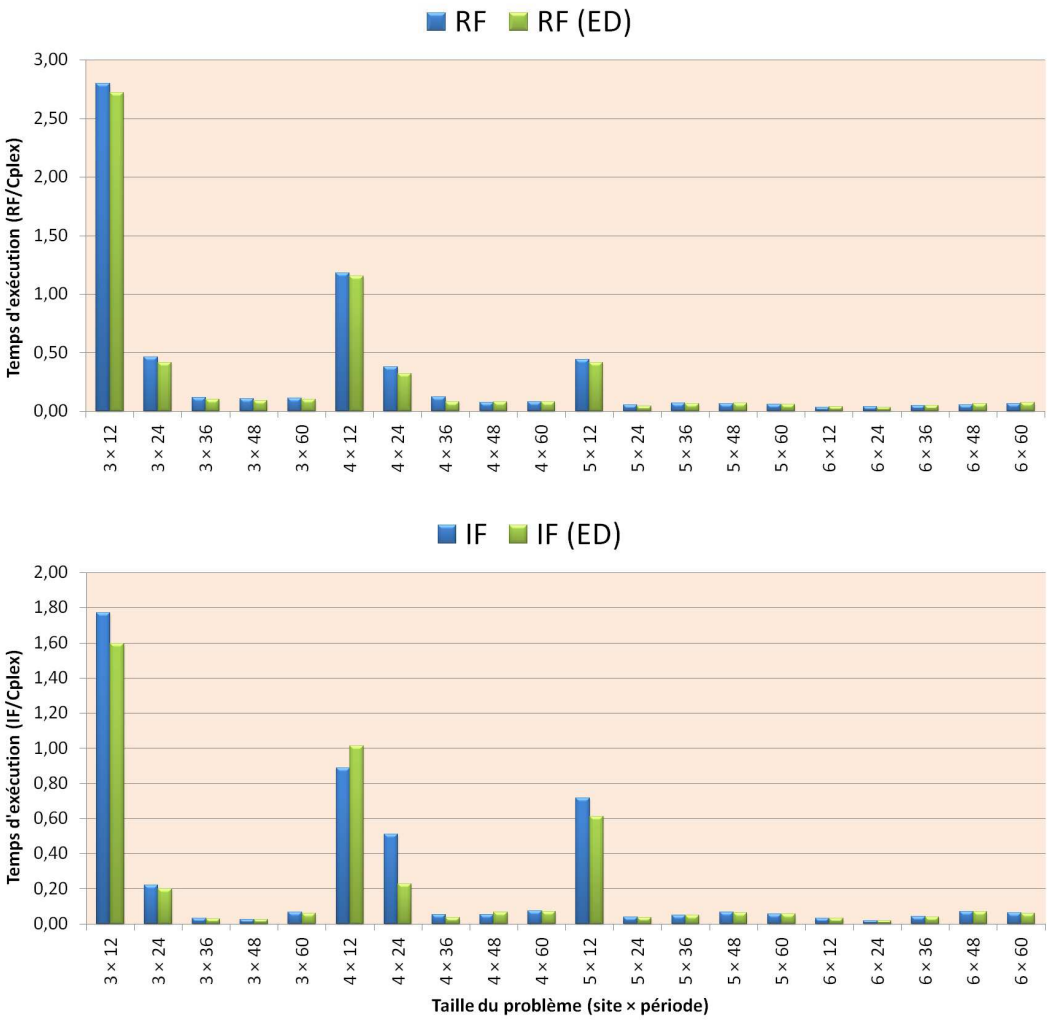


FIGURE 3.4 – Evolution du temps d'exécution en fonction de la taille du problème.

Le tableau (3.9) présente les solutions pour le problème muni des deux contraintes ($N_A_2_P_1_A_ED_ST$). En comparant avec les autres résultats, nous voyons que la combinaison de contraintes de temps de reconfiguration et de contraintes d'expédition directe des produits a favorisé l'heuristique *insert-and-fix* au lieu de l'heuristique *relax-and-fix*.

Bien sûr que toutes ces heuristiques ont des courbes de performance différentes les unes des autres. Une évaluation plus exhaustive des paramètres va surement fournir les bonnes formules à appliquer à chaque type de problème. Cependant, pour avoir une comparaison équitable de performance de toutes les heuristiques pour tous les problèmes, nous avons gardé celles du jeu d'essai pour le calibrage. Dans la sous-section suivante, nous allons voir que le comportement de ces quatre heuristiques changent pour le cas particulier de production dédiée (S-LSVRP).

TABLE 3.9 – Solutions de RF, IF, FRF et DV pour ($N_A_2_P_1_A_ED_ST$).

	Relax-and-Fix		***Insert-and-Fix***		***Fractional RF***		***Diving***	
Inst	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**	**gap (%)**	**tps (s)**
3×12	2,05	2,60	4,47	1,73	2,39	2,50	0,28	1,83
3×24	0,28	0,45	1,92	0,20	1,69	0,32	0,43	0,50
3×36	0,09	0,12	0,34	0,07	0,49	0,09	0,42	0,16
3×48	0,10	0,08	0,12	0,02	0,26	0,05	0,69	0,06
3×60	0,25	0,06	0,22	0,04	0,37	0,05	3,29	0,10
4×12	6,70	0,91	1,52	0,83	0,76	0,87	0,62	0,76
4×24	0,83	0,30	2,88	0,20	3,48	0,48	0,39	0,41
4×36	0,74	0,07	0,79	0,03	1,71	0,05	8,00	0,13
4×48	1,28	0,04	1,11	0,03	1,11	0,04	-	0,11
4×60	2,16	0,04	0,68	0,03	2,03	0,04	-	0,16
5×12	6,78	0,91	0,33	0,68	0,49	0,58	3,01	0,48
5×24	2,20	0,09	4,77	0,06	3,67	0,06	16,36	0,14
5×36	1,44	0,06	2,36	0,03	3,37	0,05	-	0,10
5×48	3,35	0,06	3,09	0,04	6,27	0,05	-	0,07
5×60	5,32	0,05	4,40	0,03	8,37	0,04	-	0,03
6×12	7,33	0,08	7,29	0,05	6,53	0,05	10,75	0,08
6×24	9,52	0,04	5,47	0,01	10,55	0,02	-	0,05
6×36	6,90	0,04	6,99	0,03	6,96	0,04	-	0,03
6×48	7,70	0,04	9,34	0,04	17,62	0,05	-	0,02
6×60	10,16	0,05	10,34	0,03	17,85	0,04	-	0,02

3.3.3 Résultats pour le cas particulier (S-LSVRP)

Les deux tableaux suivants (3.10 et 3.11) concernent les résultats fournis par l'heuristique de fusion de séquences de production (PSF), conçue pour traiter ce cas particulier de production dédiée, pour les groupes d'instances en nomenclature de série ($N_S_1_P_1_A$) et en nomenclature d'assemblage ($N_A_1_P_1_A$), respectivement. Les quatre premières colonnes des tableaux correspondent au problème sans les contraintes d'envoi direct. Elles présentent les gaps entre la solution fournie par Cplex et l'heuristique de fusion de séquences de production pour le problème complet, ainsi que les gaps pour la composante production, les gaps pour la composante transport et le temps utilisée par la PSF. Les quatre dernières colonnes dans chaque tableau s'adressent à l'utilisation des contraintes d'envoi direct.

En analysant les gaps pour la solution globale, nous remarquons que l'heuristique PSF est plus efficace pour les problèmes de taille moyenne. Le rassemblement des séquences de production limite le réseau de transport en condensant, de manière « aveugle », les décisions au début de l'horizon de production. Cette opération est intéressante lorsque nous avons suffisamment de périodes dans l'horizon pour construire un bon plan de transport qui équilibre les coûts globaux. Moins nous avons de périodes dans le problème plus il est difficile d'obtenir cet équilibre. Ce phénomène se présente même en utilisant les contraintes d'exécution directe. D'un autre côté, c'est évident que plus le nombre de niveaux d'assemblage augmente, le plus le problème d'assemblage devient combinatoire. Cela explique la chute de performance de PSF par rapport aux problèmes à 5 et 6 sites d'assemblage.

Les figures (3.5 et 3.6) montrent l'évolution des gaps entre Cplex et PSF, et aussi le temps d'exécution de la méthode, par rapport à la taille du problème traité et l'ajout des contraintes d'envoi direct de produits.

TABLE 3.10 – Solution par PSF ($N_S_1_P_1_A$).

	Envoi normal				*Envoi direct*			
Inst	**gap sol**	**gap prd**	**gap trs**	**tps**	**gap sol**	**gap prd**	**gap tsp**	**tps**
3×12	1,78	3,09	1,64	29,91	1,58	3,42	1,35	36,84
3×24	3,84	0,81	4,39	2,24	3,35	0,88	3,84	2,19
3×36	1,24	2,21	1,14	0,36	1,24	2,21	1,14	0,44
3×48	1,65	2,30	1,54	0,18	1,44	2,15	1,33	0,19
3×60	1,36	2,82	1,25	0,03	1,36	2,73	1,26	0,03
4×12	0,73	2,08	0,54	200,96	0,34	2,54	0,07	188,43
4×24	3,26	6,36	2,80	4,11	1,91	5,94	1,37	4,81
4×36	2,90	6,10	2,61	0,34	1,06	5,97	0,59	0,57
4×48	5,31	9,95	4,96	0,08	1,55	10,00	0,76	0,15
4×60	2,37	7,39	1,84	0,03	1,39	7,89	0,70	0,10
5×12	0,97	1,03	0,98	43,75	0,29	2,80	0,07	113,92
5×24	2,81	-0,02	3,09	0,40	1,39	-0,21	1,55	1,65
5×36	3,82	2,14	4,04	0,08	1,08	1,75	1,02	0,07
5×48	2,69	-1,95	3,22	0,25	1,61	-1,12	1,94	0,12
5×60	4,37	-1,44	4,97	0,20	1,29	0,64	1,37	0,13
6×12	0,67	3,25	0,42	6,34	0,46	3,83	0,15	23,72
6×24	6,57	-2,40	7,52	0,63	2,17	-0,58	2,44	3,82
6×36	2,47	-2,36	2,99	0,18	1,77	-1,37	2,09	0,08
6×48	5,83	-1,47	6,54	0,11	1,67	0,16	1,81	0,09
6×60	8,50	0,73	9,31	0,83	1,54	0,86	1,62	0,27

TABLE 3.11 – Solution par PSF ($N_A_1_P_1_A$).

	Envoi normal				*Envoi direct*			
Inst	**gap sol**	**gap prd**	**gap trs**	**tps**	**gap sol**	**gap prd**	**gap tsp**	**tps**
3×12	3,10	0,68	3,38	0,77	2,66	0,77	2,96	0,75
3×24	3,69	-1,20	4,20	2,27	3,84	-1,23	4,39	2,19
3×36	2,58	1,19	2,52	7,09	2,41	0,61	2,49	6,26
3×48	1,94	0,29	1,92	19,90	2,01	-0,02	2,01	21,65
3×60	1,69	0,93	1,66	47,31	1,59	2,81	1,42	47,70
4×12	6,76	-0,72	7,29	0,68	6,47	-0,24	6,95	0,61
4×24	6,35	-0,29	6,66	3,18	5,25	-0,01	5,47	3,23
4×36	4,64	0,72	4,79	8,44	3,39	1,34	3,29	10,06
4×48	3,08	3,64	2,80	37,16	2,27	3,40	2,07	33,43
4×60	2,08	4,20	1,87	52,36	2,04	3,99	1,87	57,41
5×12	11,10	-1,45	11,71	1,30	10,71	-1,64	11,33	1,30
5×24	5,20	-2,15	5,83	8,43	4,28	-6,88	5,38	8,15
5×36	5,69	5,31	5,23	21,14	5,08	4,03	4,74	19,54
5×48	5,95	7,46	5,94	41,64	4,61	8,04	4,41	40,82
5×60	3,74	2,78	3,48	60,56	3,74	2,09	3,77	90,11
6×12	5,63	2,33	5,68	25,52	4,87	1,48	5,00	29,06
6×24	11,06	7,12	10,38	7,46	11,44	4,53	10,94	7,79
6×36	12,22	7,31	12,63	20,33	11,05	11,86	10,89	16,68
6×48	16,40	22,46	15,83	31,51	20,17	35,99	18,61	42,07
6×60	8,57	17,29	7,81	57,15	8,55	19,29	7,49	61,33

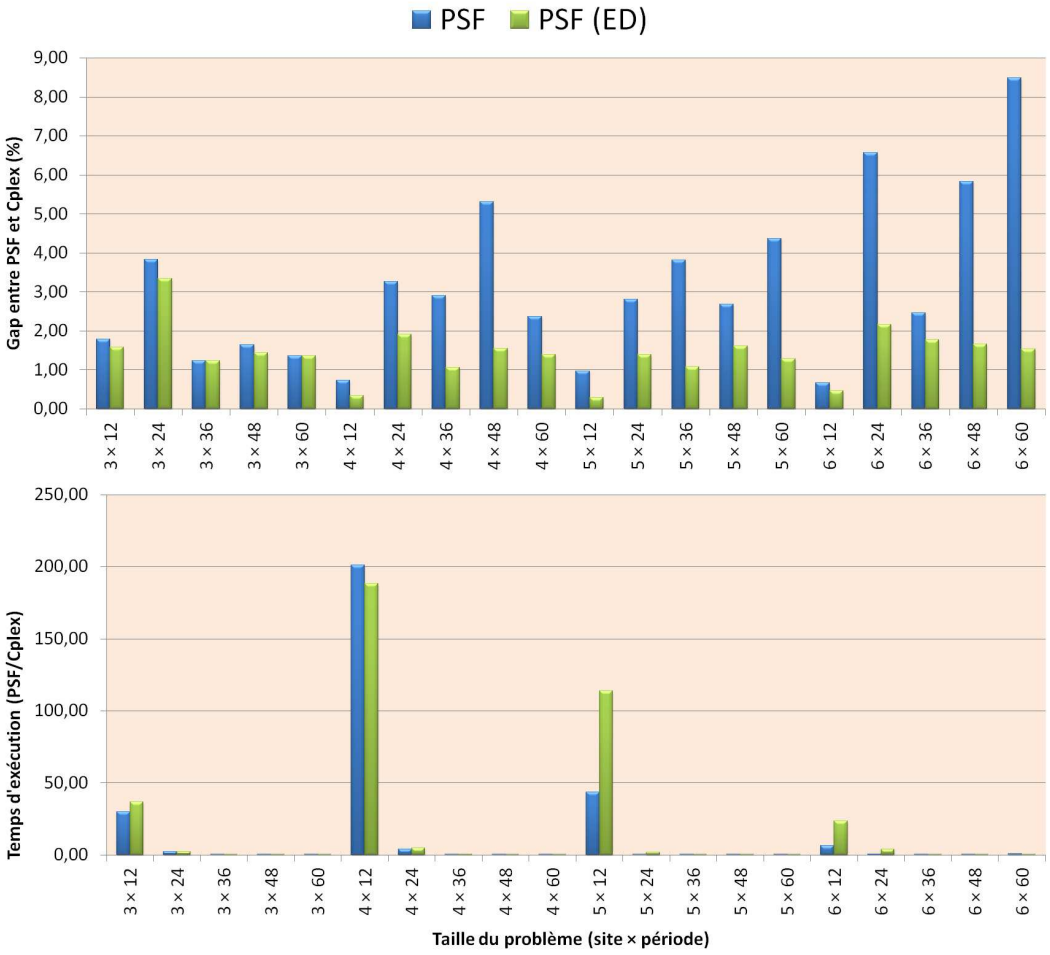


FIGURE 3.5 – Evolution du gap et du temps d'exécution en fonction de la taille du problème (N_S_1_P_1_A).

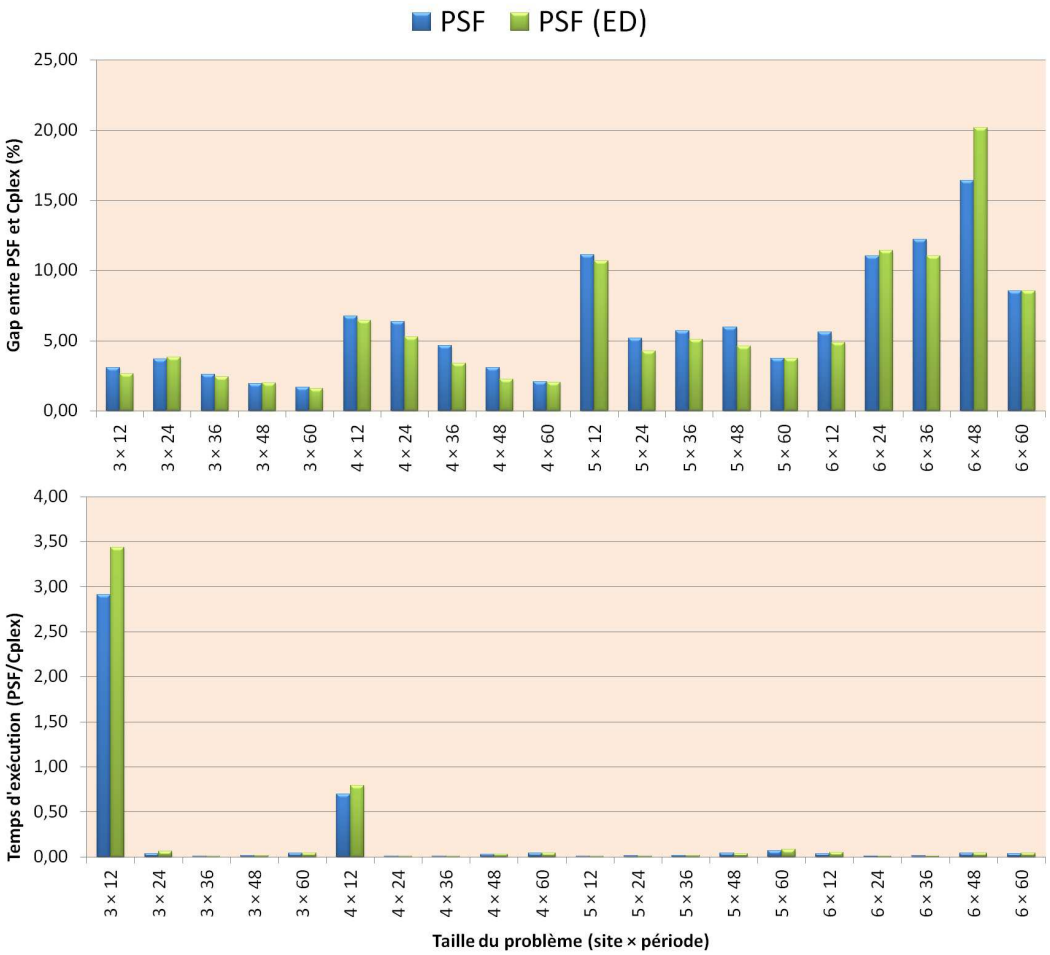


FIGURE 3.6 – Evolution du gap et du temps d'exécution en fonction de la taille du problème ($N_A_1_P_1_A$).

Une autre remarque concerne le gap individuel de chaque partie du problème (production et transport) : PSF est plus performante sur la production, parce que le problème de transport est traité en seconde partie. Avoir un gap de production plus petit que le gap de transport est normal. Cependant, ces gaps se rapprochent quand les contraintes sur le transport sont imposées.

Remarque sur le problème à 2 sites d'assemblage final :

L'heuristique de fusion de séquences de production s'est montrée inapte a résoudre les instances de type $N_A_1_P_2_A$. Son incapacité de traiter ce type de problème vient de l'imposition d'une prise de décision hors de ses compétences : le choix du site d'assemblage. En pratique, PSF suit une construction de solution qui prend en compte un seul site pour faire l'assemblage final. Cette opération met en risque tant la planification de la production comme le problème de transport résiduel pour cette situation.

Ainsi, nous concluons que PSF est une méthode qui résoud des problèmes de

type S-LSVRP à un seule site d'assemblage. Elle fournit de bonnes solutions en un temps d'exécution considérable, surtout pour le problème en nomenclature en série et avec restriction de transport.

Résultats fournis par *Relax-and-Fix*, *Insert-and-Fix*, *Fractional Relax-and-Fix* et *Diving*

Nous avons également testé les heuristiques basées sur le programmation entière mixte (MIP) dans toutes nos instances de type S-LSVRP. Évidement l'efficacité de ces heuristiques reste supérieure à PSF car ces méthodes ont été conçues pour traiter le cas général qui est plus combinatoire. Alors nous allons présenter 3 tableaux correspondant aux problèmes $N_S_1_P_1_A$, $N_A_1_P_1_A$ et $N_A_1_P_2_A$. Dans chaque tableau nous avons aussi les variantes des problèmes avec les contraintes d'expédition directe.

Le premier tableau (3.12) montre les résultats pour le premier type de problème (nomenclature en série, 1 produit final en 1 seul site d'assemblage). En analysant ce tableau nous pouvons constater que, à part *diving*, ce problème est résolu quasiment à moins de 1% pour toutes les instances de taille moyenne et grande.

Le deuxième tableau (3.13) concerne les problèmes en nomenclature d'assemblage où nous avons un seul produit final fabriqué sur un seul site d'assemblage. Les résultats sont assez bons, mais ceux des plus grosses instances (6 sites en 48 et 60 périodes) sont moins intéressants. Afin d'obtenir des meilleures solutions pour ces problèmes, il faut augmenter le temps de traitement de chaque itération des méthodes.

Le dernier (3.14) tableau concerne les problèmes en nomenclature d'assemblage avec un seul produit final dont la demande est partagée sur 2 sites d'assemblage. Cette nouvelle contrainte ajoute une nouvelle décision assez importante au niveau de complexité du problème.

TABLE 3.12 – Solutions de RF, IF, FRF et DV pour (N_S_1_P_1_A).

	Relax-and-Fix				***Insert-and-Fix***				***Fractional RF***				***Diving***			
	Envoi normal		***Envoi direct***		***Envoi normal***		***Envoi direct***		***Envoi normal***		***Envoi direct***		***Envoi normal***		***Envoi direct***	
Inst	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**
3×12	0,15	9,70	0,13	13,39	1,66	8,94	1,59	10,35	1,72	11,74	1,66	11,30	0,16	5,40	0,10	5,97
3×24	0,10	1,15	0,11	1,02	1,55	0,71	1,29	0,69	1,73	1,01	1,43	0,97	0,11	1,38	0,11	0,90
3×36	0,02	0,41	0,03	0,55	0,20	0,18	0,26	0,21	0,23	0,22	0,23	0,34	0,14	0,48	0,13	0,51
3×48	0,01	0,12	0,01	0,12	0,07	0,04	0,07	0,04	0,18	0,09	0,18	0,08	0,17	0,16	0,10	0,15
3×60	0,05	0,11	0,04	0,10	0,07	0,05	0,07	0,04	0,18	0,05	0,18	0,04	0,42	0,13	0,26	0,12
4×12	0,59	27,03	0,01	29,53	3,34	20,49	3,42	19,51	3,37	22,53	3,45	24,67	0,03	8,90	0,03	8,40
4×24	0,11	2,10	0,10	1,86	3,74	1,61	2,43	1,12	2,84	1,91	1,40	1,50	0,03	1,53	0,03	1,06
4×36	0,11	0,73	0,13	0,76	0,60	0,30	0,63	0,27	0,54	0,40	0,46	0,51	0,17	0,68	0,13	0,70
4×48	0,14	0,15	0,07	0,19	0,62	0,07	0,26	0,10	0,47	0,10	0,29	0,14	1,09	0,19	0,16	0,16
4×60	0,12	0,07	0,01	0,12	0,29	0,04	0,22	0,04	0,47	0,04	0,43	0,07	2,13	0,10	0,50	0,09
5×12	0,04	15,95	0,01	28,42	0,43	10,59	0,43	21,79	0,43	19,12	0,44	33,88	0,00	5,39	0,04	9,93
5×24	0,05	0,37	0,10	0,91	1,01	0,23	0,96	0,69	1,00	0,30	0,70	0,74	0,27	0,52	0,12	0,74
5×36	0,09	0,19	0,03	0,11	0,37	0,09	0,44	0,04	0,45	0,12	0,52	0,06	0,63	0,27	0,24	0,12
5×48	0,11	0,21	0,03	0,13	0,11	0,15	0,16	0,07	0,22	0,20	0,19	0,11	1,37	0,42	0,72	0,20
5×60	0,28	0,17	0,04	0,13	0,10	0,16	0,11	0,12	0,26	0,19	0,20	0,13	2,06	0,44	0,67	0,26
6×12	0,26	17,69	0,00	56,74	0,16	12,76	0,33	39,33	0,16	15,23	0,31	51,68	0,05	6,81	0,06	20,02
6×24	0,12	0,34	0,15	0,80	0,99	0,17	1,17	0,50	1,12	0,25	0,99	0,68	0,29	0,46	0,05	0,85
6×36	0,19	0,15	0,08	0,07	0,47	0,09	0,53	0,03	0,53	0,15	0,52	0,05	2,10	0,33	0,39	0,07
6×48	0,45	0,14	0,17	0,10	0,33	0,12	0,40	0,08	0,50	0,15	0,45	0,09	3,61	0,32	1,36	0,22
6×60	0,37	0,24	0,27	0,15	0,17	0,23	0,13	0,13	0,33	0,26	0,38	0,15	3,38	0,65	2,61	0,61

TABLE 3.13 – Solutions de RF, IF, FRF et DV pour (N_A_1_P_1_A).

	Relax-and-Fix				***Insert-and-Fix***				***Fractional RF***				***Diving***			
	Envoi normal		***Envoi direct***		***Envoi normal***		***Envoi direct***		***Envoi normal***		***Envoi direct***		***Envoi normal***		***Envoi direct***	
Inst	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**
3×12	0,24	0,47	0,21	0,40	1,58	0,24	1,70	0,23	1,33	0,36	1,71	0,32	0,15	0,35	0,15	0,32
3×24	0,16	5,99	0,16	5,00	1,09	1,45	1,08	1,30	1,10	2,38	1,10	2,20	0,17	19,84	0,17	19,67
3×36	0,08	65,21	0,05	58,32	0,17	14,39	0,13	11,59	0,47	19,87	0,47	18,49	0,44	98,43	0,27	99,83
3×48	0,07	152,54	0,07	147,47	0,09	53,97	0,09	47,89	0,21	73,31	0,20	70,43	0,70	224,04	0,56	174,24
3×60	0,10	138,99	0,12	132,72	0,07	112,66	0,05	107,27	0,15	116,26	0,19	108,26	1,24	264,39	1,16	246,28
4×12	1,27	1,28	1,06	0,73	5,96	0,48	6,24	0,38	6,37	0,65	6,26	0,44	0,17	1,43	0,15	1,05
4×24	0,51	41,89	0,51	32,87	2,58	9,32	2,67	9,30	2,86	15,74	2,73	11,34	1,82	97,25	0,88	73,86
4×36	0,31	89,79	0,15	83,80	0,70	60,42	0,80	51,64	1,12	71,08	1,13	62,65	4,15	218,43	3,15	182,11
4×48	0,68	116,57	0,25	110,13	0,10	104,96	0,05	97,82	0,40	123,81	0,45	120,18	4,18	287,14	3,49	252,86
4×60	1,12	98,19	1,06	101,24	0,36	96,96	0,19	100,87	0,81	115,08	0,72	118,27	-	-	5,44	418,54
5×12	1,08	29,88	1,07	24,16	3,82	16,97	3,67	9,43	3,74	15,73	4,25	16,24	2,03	89,71	1,41	66,21
5×24	0,25	76,98	0,18	73,49	0,57	57,74	0,44	53,90	1,19	66,66	1,03	60,10	-	-	4,88	156,52
5×36	1,15	97,27	0,88	95,08	0,63	77,54	0,95	74,57	1,13	95,29	0,81	92,99	-	-	-	-
5×48	2,54	74,96	5,28	78,52	4,75	61,85	2,09	64,49	2,95	74,51	2,93	76,92	-	-	-	-
5×60	0,47	50,40	7,76	54,59	63,44	45,60	18,24	48,49	16,07	56,88	14,86	60,35	-	-	-	-
6×12	2,19	42,99	2,51	37,98	2,81	28,52	2,62	27,02	3,24	31,04	2,88	24,74	4,49	111,12	2,95	83,55
6×24	2,74	63,55	1,73	62,65	1,87	40,20	2,17	41,66	2,31	52,41	2,32	55,74	-	-	-	-
6×36	3,40	93,26	4,57	83,96	2,71	57,05	2,65	52,17	7,42	73,07	3,08	67,29	-	-	-	-
6×48	17,43	63,41	11,49	61,64	14,09	39,87	5,37	39,65	20,89	51,86	16,66	50,90	-	-	-	-
6×60	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

TABLE 3.14 – Solutions de RF, IF, FRF et DV pour (N_A_1_P_2_A).

	Relax-and-Fix				***Insert-and-Fix***				***Fractional RF***				***Diving***			
	Envoi normal		***Envoi direct***		***Envoi normal***		***Envoi direct***		***Envoi normal***		***Envoi direct***		***Envoi normal***		***Envoi direct***	
Inst	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**	**gap**	**tps**
3×12	0,00	54,54	0,00	58,78	1,59	42,40	1,59	47,92	1,59	49,32	1,59	59,59	0,00	27,95	0,00	22,91
3×24	0,00	46,52	0,00	58,21	0,10	33,71	0,10	42,89	0,28	61,17	0,28	48,69	0,07	18,96	0,07	19,56
3×36	0,43	38,57	0,43	38,27	0,03	29,24	0,03	23,91	0,03	32,62	0,03	34,70	0,07	12,47	0,07	11,01
3×48	0,01	45,96	0,02	44,69	0,04	32,89	0,04	29,08	0,07	41,65	0,07	41,65	0,05	12,85	0,05	14,03
3×60	0,00	40,76	0,23	33,26	0,00	24,92	0,00	18,95	0,02	27,44	0,02	34,32	0,50	10,21	0,50	8,56
4×12	0,07	8,72	0,03	8,30	1,61	7,42	1,67	5,68	0,58	8,03	0,59	9,67	0,21	4,77	0,21	4,11
4×24	0,17	9,00	0,19	12,69	0,30	5,81	0,30	8,80	0,40	10,88	0,37	8,31	0,26	4,23	0,27	4,98
4×36	0,07	4,25	0,10	8,09	0,15	2,73	0,13	5,41	0,78	7,96	0,49	3,71	0,14	1,81	0,17	2,50
4×48	0,15	19,56	0,01	22,39	0,07	12,36	0,06	14,45	0,09	20,79	0,07	18,77	0,47	3,23	0,41	3,91
4×60	0,22	0,93	0,07	0,92	0,15	0,73	0,21	0,65	0,09	0,82	0,10	0,86	1,47	0,52	1,19	0,35
5×12	1,37	5,75	0,97	2,42	2,22	3,56	2,53	1,44	2,54	2,16	2,28	5,22	0,34	3,24	0,34	2,25
5×24	0,87	1,38	0,52	1,15	2,56	0,62	2,16	0,61	2,90	0,86	3,59	0,87	1,50	1,14	0,70	1,27
5×36	0,34	5,41	0,61	8,24	0,60	3,22	0,52	4,99	0,60	6,65	0,34	4,24	1,82	2,03	1,28	3,43
5×48	1,34	0,27	0,91	0,19	0,56	0,22	0,38	0,13	0,50	0,17	0,94	0,26	-	-	7,03	0,24
5×60	1,65	0,71	1,41	1,66	1,25	0,49	1,61	1,14	1,43	1,53	1,49	0,67	-	-	-	-
6×12	2,01	0,98	1,53	0,81	2,52	1,30	5,20	0,60	3,44	0,54	2,33	1,32	1,26	2,10	1,24	1,84
6×24	1,40	0,61	0,58	0,83	1,68	0,37	1,68	0,48	2,70	0,57	1,52	0,48	-	-	-	-
6×36	1,25	0,65	1,38	0,76	0,56	0,41	0,19	0,28	2,81	0,48	3,08	0,48	-	-	-	-
6×48	15,56	0,28	11,87	0,42	2,64	0,15	6,95	0,35	8,64	0,23	7,51	0,18	-	-	-	-
6×60	7,60	0,31	10,83	0,11	4,53	0,18	2,24	0,09	5,04	0,08	6,62	0,24	-	-	-	-

Comme considération finale sur les heuristiques *relax-and-fix*, *insert-and-fix*, *fractional relax-and-fix* et *diving*, la première semble être globalement la plus efficace sur des problèmes de petite et moyenne tailles. Elle fournit des solutions avec un gap assez petit en un temps raisonnable. L'*insert-and-fix* est aussi efficace, mais ses solutions perdent en qualité quand le problème comprendre un nombre petit de périodes. Normalement, avoir davantage de périodes permet de mieux anticiper les demandes. En fait avoir peu de périodes est plus compliqué car chaque décision a alors un impact beaucoup plus élevé et toute erreur se traduit par une solution dégradée. L'heuristique FRF est une moyenne entre les deux premières, et exprime dans la plupart des cas une bonne performance. *Diving* montre des bons résultats pour les instances avec nomenclature en série. En ce que concerne l'envoi directe, les heuristiques donnent des meilleures solutions en un meilleur temps pour les instances de moyenne et grande taille.

3.4 Difficultés rencontrées dans l'implémentation

L'implémentation des méthodes pour résoudre le LSVRP est complexe car il faut gérer la coordination d'un problème qui agrège production et transport. En particulier ils ne partagent pas les mêmes notions sur certains aspects, comme le temps des opérations par exemple. Nous avons cité ci-dessous quelques difficultés inhérentes à l'élaboration des modèles ainsi qu'au développement des méthodes.

Génération des instances

Il existe une certaine flexibilité dans la production parce que le modèle affecte une demande fixe uniquement aux sites d'assemblage des produits finaux. Il subsiste cependant la difficulté d'identifier les demandes dépendantes, qui sont celles des sites qui fabriquent les composants. Ce problème provient de la capacité de production ($M_{i,t}$) disponible à la période t, sur chaque site i du système, ainsi que du taux de consommation de composants pour chaque site ($A_k^{k'}$, k et k' impliqués dans la production de i). Si la consommation des ressources est grande, cela peut induire une augmentation de la demande sur les niveaux précédents de production. Par conséquent, la capacité de production peut ne pas suffire. Pour éviter cette situation, nous pouvons jouer sur le stock initial pour assurer un plan de production qui satisfasse la demande finale.

Dans les situations réelles, un stock résiduel subsiste à la fin de l'horizon de planification. C'est le cas lorsque nous souhaitons anticiper des demandes pour les périodes au-delà de l'horizon, en supposant qu'elles sont déjà connues (elles peuvent aussi être estimées).

Notre modèle ne travaille pas avec des demandes stochastiques mais il est possible de réaliser un contrôle de l'excès de stock en utilisant une pénalité à la fin de l'horizon de production. Le stock au début et à la fin de l'horizon est exprimé par

des contraintes présentées dans le chapitre 2.

Flot de véhicules

Une autre difficulté rencontrée concerne la modélisation du problème de tournée de véhicules dans le LSVRP. Les modèles VRP lient normalement les arcs des tournées aux véhicules, ce qui permet un suivi de chaque véhicule du système mais qui ajoute une nouvelle dimension au problème car les dates de passage sont inconnues. Notre modèle simplifie cet aspect en assumant une caractéristique de problème de flot liée à la discrétisation du temps et à la transformation du graphe en un graphe temporel. De plus nous avons décidé d'utiliser des variables entières, et non binaires, pour exprimer la quantité de véhicules par arc ($g_{ij,t}$). Ceci permet que plusieurs véhicules fassent le même trajet tout en respectant les contraintes de flot de véhicules et les contraintes de tournée du modèle VRP.

Limitations de l'heuristique *Production Sequences Fusion* et *Diving*

En ce qui concerne les heuristiques implémentées, PSF présente des limitations : comme elle a été conçue pour traiter les problèmes mono-produit (un seul type de produit par site), elle ne peut pas résoudre la variante multi-site d'assemblage du S-LSVRP. Ainsi, elle n'est pas capable de répartir efficacement la demande en composant sur l'ensemble des sites qui les fabriquent. Même en faisant un calcul des coûts de production et de transport pour chaque possibilité d'acheminement, il reste la difficulté de coordonner les opérations de fusion. Pour cela il serait nécessaire de réaliser un suivi précis de chaque production et une exploration des nombreuses possibilités de groupement des produits.

En ce que concerne le *diving*, le temps de calcul est le point sensible pour cette heuristique. Vu qu'elle travaille avec un nombre petit de variables par itération, l'augmentation de la taille du problème va augmenter aussi le nombre d'itérations et, par conséquent, le temps global de calcul. Nous pouvons limiter le nombre de secondes disponibles pour chaque pas de la méthode, mais cette limitation peut causer l'infaisabilité de la solution. De l'autre coté, augmenter le temps de traitement implique une durée globale assez grande pour rivaliser avec les méthodes de calcul en nombre entier.

Chapitre 4

Schéma lagrangien pour le LSVRP mono-niveau avec transfert

4.1 Objectifs

Le but de ce chapitre est de présenter un problème particulier de *Lot-Sizing* mono-niveau avec multi-production et transfert, que nous nommons ici de « 1-LSVRP », et d'exposer comment ce problème peut se décomposer, à l'aide d'un schéma lagrangien, en composants élémentaires qui sont des problèmes de localisation ou des problèmes de flots.

Nous travaillons dans un graphe $G = (V, E)$ où l'ensemble des sommets V représente les sites de fabrication plus le dépôt des véhicules, et E est l'ensemble des arcs représentant les liaisons entre chaque pair de site. Nous posons $V = V_s \cup V_0$ où V_s est l'ensemble des sites i et V_0 est le dépôt. $V_s = V_p \cup V_d$ dont V_p est l'ensemble des sites producteurs et V_d est l'ensemble des sites demandeurs de sort que $V_p \cap V_d = \emptyset$.

Comme nous traitons le graphe G comme une réseau dynamique dans le temps, chaque sommet V est associé à une période t de temps ($t = 0, \ldots, T$) sur l'horizon de planification, où $t = 0$ est la période de préparation et $T + 1$ de finalisation. Cependant nous avons fait quelques modification dans le modèle dans le but de traiter le problème comme un multiflot. Ces changements impliquent des simplifications et aussi de nouvelles hypothèses :

Niveaux de production : le problème traité est mono-niveau, c'est-à-dire qu'il n'y a pas de consommation de produits composants (k') pour fabriquer un autre produit (k). Cette « simplification » exclue l'utilisation des contraintes de consommation de composants ainsi que d'autres contraintes, variables et données qui sont liées à ces contraintes (taux de consommation, contraintes d'entrée/sortie de composants dans les sites, etc.).

Stock : dans ce problème nous avons ajouté une capacité de stockage pour chaque site et pour chaque période ($S_{i,t}$) pour avoir un contrôle plus précis du flot de produits sur les sites.

Transport : cette version du problème considère qui les sites de fabrication sont des îlots de production concentrés dans une usine et liés par des AGVs (*Automated-Guided Vehicles*), sans considérer les problématiques (délais, blocage, etc.) liées au transport, ceci plutôt vu comme une opération de transfert instantané entre les îlots. De ce fait, et aussi dû au traitement du problème comme un multiflot, nous avons ignoré la plupart des contraintes de tournée de véhicules, en gardant uniquement celles de capacité de flot ($W_{ij,t}$).

4.2 Présentation du problème

Nous supposons donné un ensemble V_s de sites producteurs, un ensemble V_d de sites demandeurs, un ensemble U de produits, un espace-temps de T périodes. A chaque instant t, la demande en produit $k \in U$ de l'agent $i \in V_d$ est connue et égale à $D^k_{i,t}$. Les produits le sont par les agents producteurs : chaque opération de fabrication correspond à une unité de temps ; les produits peuvent être échangés de façon instantanée entre les agents producteurs et être transmis de la même façon aux demandeurs. Un produit peut par ailleurs être stocké chez un agent producteur, entre un instant t et un instant $t+1$, avant d'être transmis à un demandeur ou à un autre producteur. Les demandeurs jouent le rôle d'agents terminaux, les producteurs jouant à la fois un rôle de producteur, de stockeur et d'émetteur.

D'autres modèles de production et transfert ont été déjà discutés dans le chapitre 1, comme par exemple les travaux de Bouldia *et al.* [15, 16, 17, 18, 19] et Hahm et Yano [59]. Cependant, l'étude qui nous allons présenter dans ce chapitre prend comme base celui de Sambasivan et Yahya [86] qui traite de la planification de production en plusieurs sites, sur plusieurs périodes, où le transport est considérée instantané entre sites. Ce modèle a été choisi comme référence par le fait de que les auteurs utilisent aussi une approche lagrangienne comme méthode de résolution, ce que donne des bons résultats et montre que le modèle est bien adapté à ce type de méthode.

4.2.1 Données et variables du problème

Soit $k \in U$ un produit, i et $j \in V_s$ des sites et $t = 0, \ldots, T$ des périodes de temps, nous avons comme données du problème les coefficients suivants :

- $D^k_{i,t}$ sont les demandes connues pour les produits k ;
- $Q^k_{i,t}$ est le coût fixe de production attribué à chaque site producteur sur chaque période t et pour chaque produit k ;
- $P^k_{i,t}$ est le coût de production pour chaque unité de k fabriqué en chaque site sur chaque période ;
- $H^k_{i,t}$ est le coût unitaire de stockage attribué à chaque produit en chaque site sur chaque période ;

- $J^k_{ij,t}$ est la coût de transfert par unité du produit k du site i vers le site j à la période t ;
- $M^k_{i,t}$ est le taux de consommation de capacité de production par le produit k ;
- $S^k_{i,t}$ est le taux de consommation de capacité pour stocker le produit k ;
- $W^k_{ij,t}$ est le taux de consommation de capacité pour le transfert du produit k ;
- $M_{i,t}$ est la capacité totale de production ;
- $S_{i,t}$ est la capacité totale de stockage ;
- $W_{ij,t}$ est la capacité totale de transfert.

Le problème posé est donc de décider de la production et des modes de stockage et de transfert de façon à minimiser les coûts globaux pour ces trois opérations, tout en satisfaisant les contraintes de capacité et les demandes. Nous noterons Z ce problème.

Plus précisément, les variables induites ici sont :

- $y^k_{i,t}$, à valeurs dans $0, 1$ qui dit si un producteur i est actif ou non par rapport au produit k (autrement dit, ce sont les variables de *setup*) ;
- $x^k_{i,t}$, à valeurs dans les entiers (ou, suivant le cas, fractionnaires) positifs ou nuls, qui fixe la production en k par le producteur i à l'instant t ;
- $s^k_{i,t}$, à valeurs dans les entiers (ou, suivant le cas, fractionnaires) positifs ou nuls, qui fixe le stock en k par l'agent i entre les instants t et $t + 1$;
- $f^k_{ij,t}$, à valeurs dans les entiers positifs ou nuls (ou, suivant le cas, fractionnaire), qui fixe le transfert en k depuis l'agent i vers l'agent j, à l'instant t.

4.2.2 Coûts

Les coûts globaux qui doivent être minimisés peuvent s'exprimer en une fonction :

$$\sum_{k\in U}\sum_{i\in V_p}\sum_{t=0}^{T}(Q^k_{i,t} + P^k_{i,t}\, x^k_{i,t} + H^k_{i,t}\, s^k_{i,t} + \sum_{j\in V_s} J^k_{ij,t}\, f^k_{ij,t}) \tag{4.1}$$

Cette fonction exprime que, pour chaque produit $k \in U$, chaque site $i \in V_p$ et chaque période de temps $t = 0, \ldots, T$, peut produire une quantité $x^k_{i,t}$ non nulle implique d'une composition d'un coût fixe de démarrage (*setup*, $Q^k_{i,t}$) et des coûts par unité produite ($P^k_{i,t}$). De même, stocker une quantité $s^k_{i,t}$ entre l'instant t et l'instant $t + 1$, chez l'agent producteur i implique des coûts unitaires de stockage ($H^k_{i,t}$), et transférer une quantité $f^k_{ij,t}$ depuis l'agent i (producteur) vers l'agent j (producteur ou demandeur), implique des coûts de transfert ($J^k_{ij,t}$) associés à chaque quadruplet (k, i, j, t).

4.2.3 Capacités

Toute fabrication, stockage ou transfert a une limite imposée qui peut être traduite, dans une situation réelle, en disponibilité de main-d'œuvre et/ou de matière première, en limitations de volume des containers de transport, en disponibilité d'espace dans un entrepôt, etc. Notre modélisation pour ce problème comprends trois types de capacités : de production, de stockage et de transfert. Ces capacités peuvent être exprimées sous forme de contraintes :

$$\sum_{k \in U} M_{i,t}^k \, x_{i,t}^k \leq M_{i,t} \quad \forall i \in V_p, t = 1, \ldots, T \tag{4.2}$$

$$\sum_{k \in U} S_{i,t}^k \, s_{i,t}^k \leq S_{i,t} \quad \forall i \in V_p, \forall t = 1, \ldots, T \tag{4.3}$$

$$\sum_{k \in U} W_{ij,t}^k \, f_{ij,t}^k \leq W_{ij,t} \quad \forall i \in V_p, \forall j \in V_s, \forall t = 1, \ldots, T \tag{4.4}$$

Les contraintes (4.2) limitent la production totale à une quantité $M_{i,t}$. Chaque produit a un taux de consommation de cette capacité, exprimé par $M_{i,t}^k$. Les contraintes (4.3) limitent le stock entre périodes à une quantité totale de $S_{i,t}$ produits. Chaque produit a un taux $S_{i,t}^k$ de consommation de la capacité totale de stockage. Analoguement aux autres deux contraintes, (4.4) limite le transfert entre sites à une quantité maximale $W_{ij,t}$, celle-ci consommé à un taux $W_{ij,t}^k$ par chaque produit.

4.3 Reformulation du problème sous une forme de flot/graphe

4.3.1 Graphe de base

Nous travaillons sur un graphe orienté $G = (V, E)$ où V sont les nœuds et E les arcs. V est divisé en deux ensembles : V_s et V_0, respectivement les sites et le dépôt de véhicules. V_s est aussi partagé en deux sous-ensembles : V_p de sites producteurs et V_d de sites demandeurs. Comme mentionné précédemment, le graphe est temporel et à chaque site i est indexé une période de temps $t = 0, \ldots, T$. Le dépôt (V_0) est aussi soumis au temps, les véhicules pouvant partir à n'importe quelle période. Cependant, pour la représentation du graphe comme un graphe de flot, nous nommons V_0 de « *Source* » au début de l'horizon de planification (à la période 0, dont les véhicules partent pour faire les transferts) et « *Puits* » à la fin de l'horizon (à $t = T + 1$, dont les véhicules reviennent au point de départ).

La structure de graphe orienté sur V est définie par un ensemble d'arcs $E = E_p \cup E_s \cup E_t \cup E_d$, où :

– $E_p = (Source,\ i),\ i \in V_p$, sont les arcs de production ;

- $E_s = (i,\ i')$, i et $i' \in V_p$, sont les arcs de stockage, de sorte que i' représente le site i à la période $t+1$;
- $E_t = (i,\ j)$, $i \in V_p$, et $j \in V_s$, sont les arcs de transfert ;
- $E_{st} = E_s \cup E_t$ sont les arcs de stockage/transfert ;
- $E_d = (i,\ Puits)$, $i \in V_d$, sont les arcs de demandes.

Tout arc producteur $g \in E_p$ est muni de coûts fixes : $Q_g^k = Q_{i,t}^k$, tel que $k \in U$, $i \in V_p$, $t = 0, \ldots, T$. De même, tout arc demandeur $g \in E_d$ est muni de coefficient de demande $D_g^k = D_{i,t}^k$, tel que $i \in V_d$.

Nous pouvons munir chaque arc g de type production, stockage ou transfert dans G de coûts variables C_g^k, qui sont associés de façon naturelle aux coûts variables $H_{i,t}^k$, $P_{i,t}^k$ et $J_{ij,t}^k$, tel que $i \in V_p$ et $j \in V_d$.

4.3.2 Graphe réduit

Pour simplifier la résolution du problème sur la forme de multiflot, nous allons constituer un graphe réduit H sur V par les arcs en $E^* = E_p \cup E_{st}^* \cup E_d$. Nous posons la fermeture transitive des arcs de stockage et de transfert du graphe orienté G, $E_{st}^* = (i,\ j)$, tels que $j \in V_d$ se trouve dans une période de temps t' supérieur ou égale à la période t de $i \in V_p$.

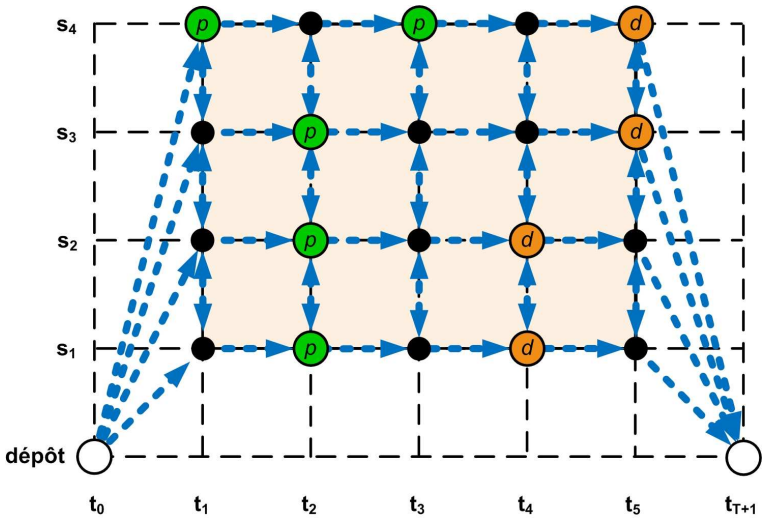
Pour tout arc $g^* \in E_{st}^*$, nous avons C_g^{*k} comme la distance, au sens des plus courts chemins, nécessaire pour aller de i à j, en se déplaçant au moyen des arcs g de E, pourvus des coûts C_g^k, augmenté de la quantité $C_{(Source,i)}^k$.

Pour illustrer la transformation du graphe G en un graphe réduit H, considérons la figure (4.1). En (4.1(a)) nous avons un graphe complet où les arcs en traits représentent les possibilités de transfert des produits entre les sites s_n $(n = 1,\ \ldots, 4)$ en une période précise $t_m\,(m = 0, \ldots, T+1)$. Ceux-ci sont présentés comme des cercles. Les cercles identifiés par « p » sont les sites en production et ceux identifiés par « d » sont les sites demandeurs. Les petits cercles noirs sont les sites sans production ni demandes. Les cercles en blanc sont la représentation des sommets *Source* et *Puits*.

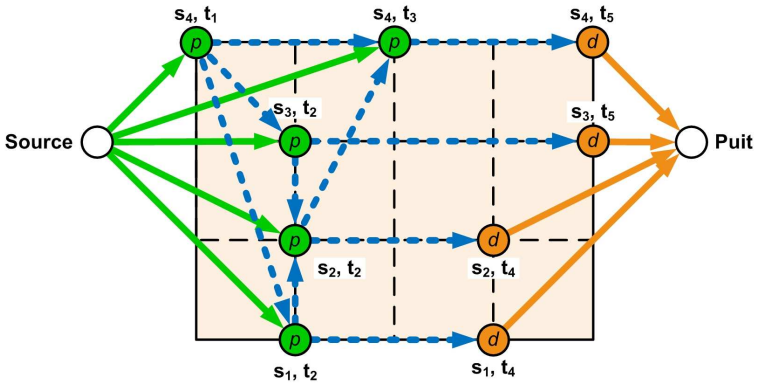
Une fois qui nous avons trouvé les distances minimales entre les sites, nous pouvons reformuler le graphe en considérant uniquement les sites en activités et les arcs représentant les relations entre eux. La figure (4.1(b)) montre le graphe H réduit. Pour représenter le flot des produits, nous avons des arcs pleins (qui sortent de *Source* quand il y a production et qui rentrent au *Puits* pour les demandes). De même, pour exprimer le flot de stockage/transfert des produits nous considérons les arcs en traits qui lient les sites actifs.

4.3.3 Reformulation multiflot du problème de *lot-sizing*

Sur le graphe orienté $G = (V, E)$, nous allons rechercher un multiflot $w = (w^k,\ k \in U)$, dont la sémantique sera la suivante :



(a) Graphe complet.



(b) Graphe réduit.

FIGURE 4.1 – Réduction du graphe G au graphe H.

- $w^k_{(Source,\, i)} = x^k_{i,t},\ i \in V_p$;
- $w^k_{(i,\, Puits)} = D^k_{i,t},\ i \in V_d$;
- $w^k_{(i,\, i')} = s^k_{i,t}$, i et $i' \in V_p$, dont i' est le sommet i à la période $t+1$;
- $w^k_{(i,\, j)} = f^k_{ij,t}$, $i \in V_p$, et $j \in V_s$;

Ainsi qu'un vecteur y, indexé sur les arcs de E_p, de sémantique :

- Pour tout arc $g \in E_p$, $y^k_g = y^k_{i,t}$.

Suivant le cas, le multiflot w sera entier ou fractionnaire (suivant la façon dont sont mesurés les différents produits). Cette sémantique sur le multiflot w et le vecteur y, se traduit sous la forme des contraintes suivantes :

$$\sum_{k \in U} M^k_g\, w^k_g \leq M_g \quad \forall g \in E_{st} \cup E_p \tag{4.5}$$

$$w_g^k = D_g^k \quad \forall k \in U, \forall g \in E_d \tag{4.6}$$

$$y_g^k \to w_g^k = 0 \quad \forall g \in E_p \tag{4.7}$$

Les contraintes (4.5) limitent la capacité sur les arcs. Les contraintes (4.6) définissent le flot sur les arcs de demande. Les contraintes (4.7) imposent un flot nul sur les opérations de *setup*.

Le coût à minimiser est la combinaison d'une quantité linéaire, dans laquelle les coûts C_g^k sont associés aux coûts variables $P_{i,t}^k$, $H_{i,t}^k$ et $J_{ij,t}^k$; et d'une quantité de coûts fixes de *setup*, associée à tous les arcs producteurs $g \in E_p$, tels que les valeurs w_g soient non nulles, et dans lesquelles pour chaque $i \in V_p$, $t = 1, \ldots, T$, nous avons $Q_g^k = Q_{i,t}^k$:

$$\sum_{k \in U} (\sum_{g \in E_p} Q_g^k \, y_g^k + \sum_{g \in E_{st}} C_g^k \, w_g^k) \tag{4.8}$$

Dans la suite de ce chapitre, nous ne considérerons plus le problème Z que sous cette forme de « multiflot ».

4.3.4 Le cas sans capacités de stockage/transfert et le réseau réduit H

Dans le cas où les capacités sur les arcs de stockage et de transfert sont infinies, nous voyons que les routages d'un producteur vers un demandeur peuvent se faire par le plus court chemin pour les coûts C. Nous déduisons que le problème peut se reformuler en se restreignant au réseau réduit H :

Sur ce graphe orienté $H = (V, E^*)$, nous allons rechercher un multiflot $w = (w^k, k \in U)$, dont les flots de stockage et de transfert sont unifiés :

- $w_{(i,\,j)}^k$ est la quantité de k routée depuis la production en $i \in V_p$ à l'instant t vers le site demandeur $j \in V_d$ à l'instant $t' \geq t$.

Cette sémantique sur le multiflot w est soumisse aux contraintes (4.5), (4.6) et (4.7), et le coût à minimiser sera la combinaison d'une quantité linéaire et d'une quantité *setup* :

$$\sum_{k \in U} (\sum_{g \in E_p} Q_g^k \, y_g^k + \sum_{g \in E_{st^*}} C_g^{*k} \, w_g^k) \tag{4.9}$$

4.4 Le problème sans capacité : problème de localisation

4.4.1 Problème de localisation mono-produit

Soit X un ensemble, dont chaque élément x est muni d'un poids $B_x \geq 0$ et dans lequel chaque couple (x, y) est muni d'un poids $I_{(x,y)} \geq 0$, de telle sorte que les

poids $I_{(x,x)}$ sont nuls. Nous considérons implicitement que toute paire (x, x) est un arc dont les poids associés sont nuls. Le problème de localisation induit consiste à chercher un sous-ensemble Y de X qui minimise la quantité :

$$\sum_{x \in Y} B_x + \sum_{x \in X} \min_{y \in Y} I_{(y,x)} \tag{4.10}$$

Un tel ensemble Y étant calculé, nous identifions, pour chaque $x \in X$, un élément x', qui est un des éléments $y \in Y$ tel que $I_{(y,x)} = \min_{y \in Y} I_{(y,x)}$, et, pour chaque élément $y \in Y$, l'ensemble $R_y = \{x \mid y = x'\}$.

Pour un tel couple (x, y), nous considérons que y dessert x et que x est desservi par y.

4.4.2 Un algorithme simple pour le problème de localisation mono-produit

Pour trouver quelles sont les sites fournisseurs du système de production, nous pouvons utiliser un algorithme simples de localisation. Alors, la procédure *LOC-FUSION* (LF) suivant cherche le sous-ensemble Y de X qui minimise les coûts impliqués :

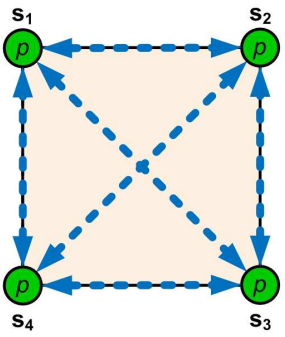
LOC-FUSION(X, B, I)

1. Initialiser : $Y \leftarrow X$; $Stop \leftarrow false$;
2. Pour tout $x \in X$, poser $x' \leftarrow x$;
3. Pour tout $y \in Y$, poser $R_y = \{x \in X \mid x' = y\}$;
4. Tant que $Stop = false$, faire
5. Chercher s'il existe y et $y' \in Y$ tels que :
6. $\sum_{x \in R_{y'}} I_{(y,x)} < B_{y'} + \sum_{x \in R_{y'}} I_{(y',x)}$;
7. Si y et y' existent, alors
8. $Y \leftarrow Y - \{y'\}$;
9. Pour tout $x \in R_{y'}$, faire
10. $x' \leftarrow y'' \in Y \mid I_{(y'',x)} = \min$;
11. Insérer x dans $R_{y''}$;
12. Sinon
13. Chercher s'il existe $y \in (X - Y)$ et $y' \in Y$ tel que :
14. $B_y + \sum_{x \in R_{y'}} I_{(y,x)} < B_{y'} + \sum_{x \in R_{y'}} I_{(y',x)}$;
15. Si y et y' existent alors
16. $Y \leftarrow Y - \{y'\} + \{y\}$;

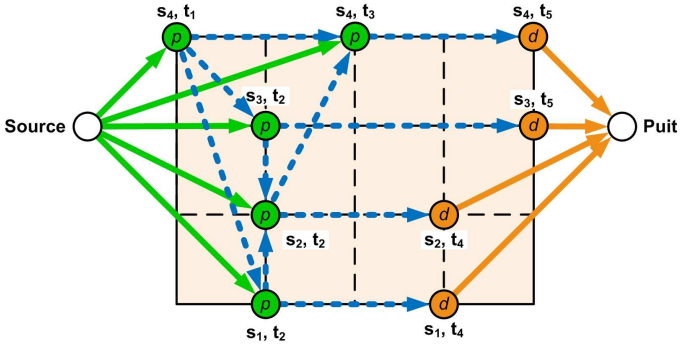
17. Pour tout $x \in R_{y'}$, faire
18. $x' \leftarrow y'' \in Y \mid I_{(y'',x)} = \min$;
19. Insérer x dans $R_{y''}$;
20. Sinon ;
21. $Stop \leftarrow true$;

Pour illustrer le fonctionnement de la procedure LF, prenons le graphe réduit H de la figure (4.1(b)) et ajoutons des poids B_{s_x} aux sommets qui représentent les sites producteurs (s_x), tel que pour l'ensemble $X = [s_1 \quad s_2 \quad s_3 \quad s_4]$, nous avons $B_X = [4 \quad 6,5 \quad 2,5 \quad 1]$. Ajoutons aussi des poids $I_{(s_x,\, s_y)}$ aux arcs que représentent le plus court chemin entre deux sites, tel que :

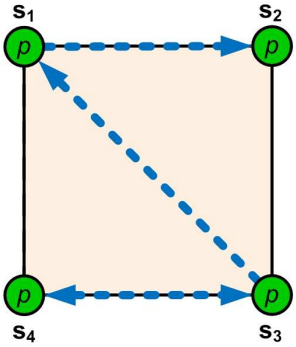
$$I_{(s_x,\, s_y)} = \begin{pmatrix} 0 & 2 & 7 & 10 \\ 3 & 0 & 5 & 8 \\ 7 & 4 & 0 & 5 \\ 12 & 9 & 5 & 0 \end{pmatrix}$$



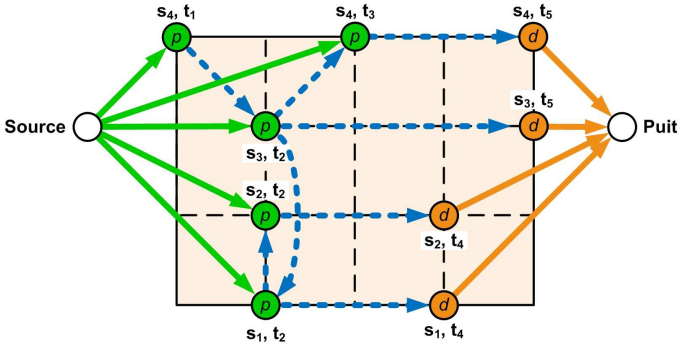
(a) Possibilités de transfert.



(b) Graphe réduit H.



(c) Schéma de fournissement définit par LF.



(d) Graphe H après *LOC-FUSION*.

FIGURE 4.2 – Exemple *LOC-FUSION*.

La figure (4.2) montre le résultat pour une exécution de la méthode à partir des données citées précédemment. (4.2(a)) exprime toutes les possibilités de transfert

entre les sites producteurs du graphe réduit H (4.2(b)). Après la résolution du problème de localisation, LF trouve le sous-ensemble $Y \cup X$ tel que $Y = [s_1 \quad s_3 \quad s_4]$ comprend tous les sites fournisseurs. (4.2(c)) exprime la relation entre les sites fournisseurs et desservis. La solution du problème est exprimée en (4.2(d)) où uniquement le site s_2 n'est pas un fournisseur.

Cet algorithme peut être rendu non déterministe en effectuant chaque recherche de couple (y, y') par double balayage circulaire à partir d'un couple de sommets tirés aléatoirement. Alors, soit $N \in \mathbb{N}$ le nombre de itérations de la méthode, nous en déduisons le schéma de traitement *GRASP-LOC-FUSION* (GLF) suivant :

GRASP-LOC-FUSION(N, X, B, I)
1. Pour $i = 1 \dots N$, faire
2. *LOC-FUSION*(X, B, I) ;
3. Conserver le meilleur résultat obtenu ;

4.4.3 Le problème sans capacité vu comme un problème de localisation

Nous supposons ici que l'ensemble des contraintes de capacité sont relâchées. Nous notons alors $Z_{(Relax)}$ ce problème.

$Z_{(Relax)}$ comprend tous les problèmes de localisation $Z^k_{(Relax)}$, traités séparément pour chaque produit $k \in U$, avec X, B^k, I^k définis comme suit :

- $X = V_s$;
- Pour tout $i \in V_p$, le poids sur le site est exprimé par $B^k_i = Q^k_{(Source,\, i)}$;
- Pour tout $i \notin V_p$, $B^k_i = +\infty$;
- Pour tout $k \in U$, et tout arc $g = (i, j) \in E^*_{st}$, le coût du plus court chemin entre deux sites est exprimé comme $I^k_{(i,j)} = C^{*k}_g \; D^k_{(j,\, Puits)}$;
- Pour tout autre $g = (i, j) \in X$, $I^k_{(i,j)} = 0$.

4.5 Traitement des contraintes de capacités de production

Nous travaillons dans cette section sur la modélisation qui correspond au cas où les capacités des arcs de stockage/transfert du graphe G sont infinies (vu dans la section 4.3.4). Cette modélisation ne met donc en jeu que le graphe réduit H.

4.5.1 Maximisation du lagrangien

La relaxation Lagrangienne qui dérive alors de la relaxation des contraintes de capacité s'écrit comme ci-dessous :

$$L(w,\lambda) = \sum_{k\in U}(\sum_{g\in E_p} Q_g^k\, y_g^k + \sum_{g\in E_{st}^*} C_g^{*k}\, w_g^k) + \sum_{g\in E_p} \lambda_g(\sum_{k\in U} M_g^k\, w_g^k - M_g) \quad (4.11)$$

Le vecteur lagrangien $\lambda = (\lambda_g, g \in E_p)$ est ici indexé sur les arcs producteurs, et l'optimisation de ce lagrangien se ramène donc à une famille séparée de problèmes de localisation $Z^k_{\lambda\,(Relax)}$ tels que décrit en (4.4), avec, pour chaque $k \in U$:

- $X = V_s$;
- Pour tout $i \in V_p$, le poids sur le site est exprimé comme $B^k_{i,\lambda} = Q^k_{(Source,\,i)}$;
- Pour tout $i \notin V_p$, $B^k_{i,\lambda} = +\infty$;
- Pour tout couple $(i,j) \in E^*_{st}$, i producteur, et j demandeur, le poids sur l'arc entre ces deux sites est exprimé par $I^k_{(i,j),\lambda} = (\lambda_{(Source,\,i)} + C^{*k}_{(i,j)})\; D^k_{(j,\,Puits)}$.

A l'issue de la résolution d'un tel problème $Z^k_{\lambda(Relax)}$, nous considérons un ensemble Y^k_λ producteur, pour chaque $i \in Y^k_\lambda$ un ensemble desservi $R^k_{i,\lambda}$ et, pour chaque demandeur j, un nœud producteur $x'^k_{j,\lambda}$. Nous déduisons un multiflot w défini sur H en posant :

- Pour tout nœud i producteur, $w^k_{(Source,\,i)} = \sum_{j\in R^k_{i,\lambda}} D^k_{(j,\,Puits)}$;
- Pour tout i producteur, $j \in R^k_{i,\lambda}$, $w^k_{(i,j)} = D^k_{(j,\,Puits)}$;
- Pour tout autre couple (i,j) arc de E^*_{st}, $w^k_{(i,j)} = 0$;
- Pour tout arc g demandeur, $w^k_g = D^k_g$.

En fournissant un nombre $N \in \mathbb{N}$ d'itérations et le graphe réduit H, le schéma général *LAGRANGE-SANS-TRANSFERT* (LST) de traitement de ce lagrangien s'écrit comme ci dessous :

LAGRANGE-SANS-TRANSFERT(N, H) ;

1. Initialiser : λ, $Stop \leftarrow false$;
2. Tant que $Stop = false$, faire
3. Pour chaque k, traiter $Z^k_{\lambda(Relax)}$ via *GRASP-LOC-FUSION* $(N, X, B^k_\lambda, I^k_\lambda)$;
4. Calculer la quantité Pas ;
5. $\Delta \leftarrow (\sum_{k\in U} M^k_g\, w^k_g - M_g)$, pour $g \in E_p$;

6. $\lambda \leftarrow \lambda + Pas\,\Delta$;
7. Mettre à jour $Stop$;

La mise à jour de $Stop$ et le calcul de Pas peuvent s'effectuer en utilisant certaines des formules classiquement associées à la gestion des lagrangiens, telles que présentées par exemple en [57].

Incrémentalité : une exécution efficace de l'instruction LST suppose de mettre en œuvre une notion d'incrémentalité, c'est-à-dire le fait de pouvoir traiter le problème de localisation associé à λ à partir de la solution obtenue pour une valeur de λ relativement proche. Il faut donc, une fois passée la phase d'initialisation, remplacer l'appel à *GRASP-LOC-FUSION* par un appel à une procédure de type « transformation locale ». Nous pouvons par exemple envisager d'utiliser la procédure *LOC-TRANSFO* (LT) suivante :

LOC-TRANSFO(Y)

1. Initialiser : $Stop \leftarrow false$, $Z \leftarrow y$;
2. Tant que $Stop = false$, faire
3. Chercher $z \in Z$ et $x \in (X - Z)$ tel que x au lieu de z améliore Z ;
4. Si x et z ont été trouvés, alors
5. $Z \leftarrow Z + \{x\} - \{z\}$;
6. Sinon
7. $Stop \leftarrow true$;
8. *LOC-TRANSFO*(Z) ;

Par exemple, nous supposons ici l'existence d'un seul produit, de 2 centres de production A et B, et de 2 centres de demandes X et Y. Les demandes émanant de X et Y sont de 10 chacune. Les données relatives aux coûts de production et de transport pour A et B (respectivement) sont présentés dans le tableau (4.1).

Table 4.1 – Exemple 1 : tableau de coûts.

	site A	**site B**
Coût fixe de production	70	1
Coût variable	4	5
Capacité de production	22	12
Consommation de capacité	1	1
Coût de transfert vers le site X	3	2
Coût de transfert vers le site Y	1	7

Le problème de localisation associé au lagrangien induit par ces données fait apparaitre 3 solution possibles :

- Production totale sur A : A produit pour X et Y. Le coût induit est :
  $L(w,\lambda) = 1\times 70+((4+3)\times 10+(4+1)\times 10)+\lambda_{(Source,\,A)}(10+10)-\lambda_{(Source,\,A)}22$
  , ce qui donne la valeur de $190 - 2\lambda_{(Source,\,A)}$;
- Production totale sur B : B produit pour X et Y. Le coût induit est :
  $L(w,\lambda) = 1\times 1+((2+5)\times 10+(7+5)\times 10)+\lambda_{(Source,\,B)}(10+10)-\lambda_{(Source,\,B)}12$
  , ce qui donne la valeur de $191 + 8\lambda_{(Source,\,B)}$;
- Production partagée entre A et B : A produit pour X et B produit pour Y. Le coût induit est :
  $L(w,\lambda) = (1 \times 70 + 1 \times 1)((4 + 1) \times 10 + (7 + 5) \times 10) + (\lambda_{(Source,\,A)}(10) + \lambda_{(Source,\,B)}10) - (\lambda_{(Source,\,A)}\ 22 + \lambda_{(Source,\,B)}\ 12)$
  , ce qui donne la valeur de $191-12\lambda_{(Source,\,A)}-2\lambda_{(Source,\,B)}$. Cette optimisation renvoie $\lambda_{(Source,\,A)} = \lambda_{(Source,\,B)} = 0$.

4.5.2 Lissage

Le processus de maximisation du Lagrangien s'achève en principe avec l'obtention d'un vecteur $\lambda_{(max)}$, et, pour tout $k \in U$, d'un ensemble $V^k_{p\,(max)} \subset V_p$ de sommets producteurs. Le multiflot flot $w_{(max)}$ ne satisfait pas forcément les contraintes de capacités. Il s'ensuit qu'il faut procéder, à partir de $\lambda_{(max)}$, au calcul d'un « bon » multiflot w associé.

Nous pouvons alors calculer dans le graphe $H = (V, E^*)$, en utilisant par exemple Cplex, un flot f fractionnaire tel que :

- Pour tout arc $g \in E^*_p$, $\sum\limits_{k\in U} M^k_g\, f^k_g \leq M_g$;
- Pour tout arc $g \in E^*_p$ tel que i n'est pas dans $V^k_{p\,(max)}$, $f^k_g = 0$;
- Pour tout arc $g \in E^*_d$, et tout $k \in U$, $f^k_g \leq D^k_g$;
- $\sum\limits_{k\in U} f^k_{(Puits,\,Source)}$ est maximal ;
- Le coût $\sum\limits_{\substack{k\in U\\ g\in E^*}} C^{*k}_g\, f^k_g$ est minimal, la condition précédente étant satisfaite.

Dans le cas où les unités de production sont des nombres entiers, nous pouvons de surcroit lisser f, en calculant par exemple, pour tout $k \in U$, un flot entier r^k, défini sur H, et tel que :

- Pour tout arc g, $r^k_g \leq \lfloor f^k_g \rfloor$;
- $r^k_{(Puits,\,Source)}$ est maximal.

Nous remplaçons alors f^k par r^k, pour tout $k \in U$.

Le multiflot f ayant été calculé, deux possibilités peuvent se présenter :

1) f satisfait les contraintes de demande, c'est-à-dire que sur chaque arc $g \in E_d$, et pour tout k, nous avons $f_g^k = D_g^k$. Nous pouvons alors considérer que nous l'avons terminé et que f est une solution du problème d'origine (Z) ;
2) f ne satisfait pas les contraintes de demandes. Nous pouvons considérer donc qu'il y a des demandes résiduelles non satisfaites, qu'il faut essayer de satisfaire en tenant compte du fait que certains producteurs sont déjà occupés. De ce fait, nous voyons que nous devrions alors, pour satisfaire ces demandes résiduelles, considérer un problème de *lot-sizing* du type de départ, avec les paramètres suivants :

- Producteurs : pour tout sommet producteur $i \in V_{p\,(max)}^k$, nous posons $Q_{(Source,\,i)}^k = +\infty$, ce qui signifie que i ne peut plus produire k ; la capacité de production associée à un arc $g = (Source,\, i)$ devient $M_g - \sum_{k \in U} M_g^k\, f_g^k$;
- Demandeurs : pour tout sommet demandeur i, la nouvelle demande en k associée à l'arc $g = (i, Puits)$ devient la demande résiduelle $D_g^k - f_g^k$.

Nous pouvons alors reprendre le processus de calcul sur le problème résiduel ainsi défini, ce qui se traduit sous la forme du schéma algorithmique exposé en section suivante (4.5.3).

4.5.2.1 Producteurs désactivés

Nous pouvons remarquer que le fait de manier des coûts fixes infinis ou des capacités nulles ou très petites peut avoir des effets pervers au niveau du lagrangien, et conduire au calcul dans $V_{p\,(max)}$ de sites producteurs qui ne sont de ce fait en mesure de répondre à aucune demande. Afin de palier à ces inconvénients, nous introduisons la notion de producteur « activé » ou « désactivé » : un producteur désactivé, pour un produit donné $k \in U$, ne pourra plus être choisi comme élément d'un ensemble $V_{p\,(max)}$. A l'issue du processus de calcul de $V_{p\,(max)}$ conformément au schéma ci-dessus, seront déclarés désactivés, pour un produit k donné :

- Tous les éléments de $V_{p\,(max)}^k$;
- Tous les éléments dont la capacités résiduelle $M_g - \sum_{k \in U} f_g^k\, M_g^k$, ne leur permet plus de satisfaire aucune demande en k (soit parce que cette capacité est nulle, soit du fait d'une éventuelle contrainte d'intégrité sur les valeur f^k, parce que cette capacité est trop petite).

4.5.3 Schéma algorithmique général

La procédure algorithmique *LOT-SIZING-SANS-TRANSFERT* (LSST) opère sur le graphe H, et les données qui vont avec, en $N \in \mathbb{N}$ itérations. Elle est exprimée

comme :

LOT-SIZING-SANS-TRANSFERT(H, N) :

1. Initialiser : $w \leftarrow 0$; $Stop \leftarrow false$; $M_{(aux)} \leftarrow M$; $D_{(aux)} \leftarrow D$; $Q_{(aux)} \leftarrow Q$;
2. Tant que $(D_{(aux)} \neq 0)$ ET $(Stop = false)$ faire
3. Appliquer *LAGRANGE-SANS-TRANSFERT*(N, H) en utilisant les quantités $M_{(aux)}$, $D_{(aux)}$, $Q_{(aux)}$, C et déduire un multiflot f de manière décrite dans la section (4.5.2) ainsi que, pour chaque produit k, un ensemble producteur $V^k_{p\,(max)}$;
4. Si f a été trouvé
5. $w \leftarrow w + f$;
6. Pour tout arc $g = (i, Puits), i \in V_d$, et tout $k \in U$ faire
7. $D^k_{g\,(aux)} \leftarrow D^k_{g\,(aux)} - f^k_g$;
8. Pour tout arc $g = (Source, i)$, $i \in V^k_{p\,(max)}$, i producteur, faire
9. $M_{g\,(aux)} \leftarrow M_{g\,(aux)} - \sum_{k \in U} M^k_g \, f^k_g$;
10. $Q^k_i \leftarrow +\infty$;
11. Sinon
12. $Stop \leftarrow true$;

Le résultat global est fourni par w en cas de succès. En cas d'échec, la demande n'a pu être satisfaite.

4.6 Traitement des contraintes de capacités en deux étapes

Nous allons, dans cette section, étudier un schéma de décomposition lagrangienne destiné au problème général, c'est-à-dire au problème mettant en jeu le graphe $G = (V, E)$. Ce schéma à 2 niveaux de relaxation consiste à d'abord relaxer les contraintes de capacités sur les arcs de transfert/stockage, ce qui induit la formulation d'un problème relâché du type de celui traité dans la section (4.5), puis à traiter celui-ci selon le schéma décrit dans la même section, c'est-à-dire par relaxation des contraintes de capacités sur les arcs de production.

4.6.1 Maximisation du lagrangien

Nous relaxons donc les contraintes de capacités sur les arcs de transfert/stockage. Le lagrangien induit s'écrit :

$$L^*(w,\mu) \quad = \quad \sum_{\substack{g\in E_p \\ k\in U}} (Q_g^k \;\; y_g^k \;+\; C_g^k \;\; w_g^k) \;+\; \sum_{g\in E_{st}} \mu_g \;\; (\sum_{k\in U} M_g^k \;\; w_g^k \;-\; M_g) \quad (4.12)$$

Le vecteur lagrangien $\mu = (\mu_g, g \in E_{st})$ est ici indexé sur les arcs de transfert/stockage.

Le vecteur w est supposé satisfaire les contraintes de capacité sur les arcs de production ainsi que les contraintes de demandes. Moyennant ceci, l'optimisation de ce lagrangien, se ramène donc à un problème $Z_{\mu\,(Relax)}$ tel que traité dans la section (4.5), restreint de fait au graphe orienté $H = (V, E^*)$, avec :

- Demandes et capacités de production dérivées de façon naturelle ;
- Pour tout $k \in U$, et tout arc $g = (i, j)$ dans E^*_{st}, $C^{*k}_{g,\,\mu} = C^k_{(Source,\,i)} + L^k_{(i,j),\mu} =$ distance de i à j, au sens des plus courts chemins dans le graphe G, chaque arc $f = (u, u')$ de E étant muni d'une longueur $C^k_f + M^k_f\,\mu_f$.

L'algorithme LSST décrit dans la section (4.5.3) peut donc être appliqué au traitement du problème $Z_{\mu\,(Relax)}$.

Cet algorithme ne nous fournit à priori qu'un vecteur w^* indexé sur les arcs du graphe orienté H. Nous pouvons le transformer en un vecteur flot canonique w défini sur le graphe G en routant chaque quantité w^{*k}_g, $g = (i, j)$ dans E^*_{st}, le long d'un plus court chemin $\Gamma^k_{g,\,\mu}$ de i à j pour le système de distance L^k_μ.

Soit $N \in \mathbb{N}$ le nombre d'itération et G le graphe du problème, le schéma général de traitement *LAGRANGE-COMPLET*(LC) s'écrit alors :

LAGRANGE-COMPLET(G, N)

1. Initialiser : $\mu \leftarrow 0$; $Stop \leftarrow false$; $Echec \leftarrow false$;
2. Pour chaque $k \in U$, faire
3. Calculer le tableau des distances et chemins L^k_μ et Γ^k_μ ;
4. Tant que $Stop = false$ ET $Echec = false$, faire
5. Traiter $Z_{\mu\,(Relax)}$ via *LOT-SIZING-SANS-TRANSFERT*(H, N);
6. Si $Z_{\mu\,(Relax)}$ a été traité, alors
7. Déduire un flot canonique w à partir de la solution w^* obtenue pour $Z_{\mu\,(Relax)}$, conformément à la procédure (4.6.1) ;
8. Calculer la quantité Pas ;
9. Poser $\Delta \leftarrow$ le vecteur $(\sum_{k\in U} M^k_g\, w^k_g - M_g,\, g \in E_{st})$;

10. $\mu \leftarrow \mu + Pas\ \Delta$;

11. Mettre à jour, par propagation de contraintes, les systèmes de distances L^k_μ et chemins Γ^k_μ, $k \in U$:

12. Mettre à jour $Stop$;

13. Sinon

14. $Echec \leftarrow true$;

La mise à jour de $Stop$ et le calcul de Pas peuvent s'effectuer en utilisant certaines des formules classiquement associées à la gestion des lagrangiens, telles que présentées par exemple en [57].

4.6.1.1 Mise à jour du système de distance L et chemins Γ

Nous pouvons supposer que les chemins Γ^k_μ, sont identifiés, pour tout couple (i, j), par la donnée de l'antécédent de j dans le chemin $\Gamma^k_{(i,j),\,\mu}$. Nous propageons les règles suivantes, qui s'appliquent à un tableau de longueur L, défini sur les arcs d'un graphe $K = (X, E)$, à un tableau de distance L^*, défini sur tous les couples de sommets de G, et supposé à terme représenter les distances au plus courts chemins associés au longueurs définies par L, et à un tableau d'antécédents A :

$$L^*_{(x,y)} + L_{(y,z)} < L^*_{(x,z)} \;\Big|\; \begin{array}{l} L^*_{(x,z)} = L^*_{(x,y)} + L_{(y,z)} \\ A_{(x,z)} = y \end{array}$$

$$\begin{array}{l} L^*_{(x,y)} + L_{(y,z)} > L^*_{(x,z)} \\ A_{(x,z)} = y \end{array} \;\Big|\; L^*_{(x,z)} = L^*_{(x,y)} + L_{(y,z)}$$

Incrémentalité : une exécution efficace de l'instruction *LAGRANGE-SANS-TRANSFERT* suppose de mettre en œuvre une notion d'incrémentalité, c'est-à-dire le fait de pouvoir traiter le problème de localisation associé à μ à partir de la solution obtenue pour une valeur de μ relativement proche. Il faut donc, une fois passée la phase d'initialisation, remplacer l'appel à LSST par un appel à une procédure de type « transformation locale ». Nous pourrons par exemple adapter LSST en utilisant la procédure *LOC-TRANSFO* décrite dans la section (4.5.1).

Par ailleurs, il convient de remarquer que le test d'échec ne concerne en fait que la première exécution de la boucle, puisque associé au fait de savoir si les capacités de production permettent de satisfaire la demande dès lors que nous ne prenons pas en compte les capacités de transfert et de stockage.

4.6.2 Lissage

Le processus de maximisation du lagrangien s'achève en principe avec l'obtention d'un vecteur $\mu_{(max)}$, indexé sur les arcs de transfert/stockage, et d'un vecteur flot $w_{(max)}$ associé. Ce vecteur flot identifie, pour tout $k \in U$, un ensemble $V^k_{p\,(max)} \subset V_p$

de sommets producteurs, qui, les contraintes de capacités de transfert et stockage étant relaxées, s'avère suffisant pour satisfaire les demandes de production.

Nous pouvons alors calculer dans le graphe G, en utilisant par exemple Cplex, un flot f fractionnaire tel que :

- Pour tout arc g, $\sum_{k \in U} M_g^k \, f_g^k \leq M_g$;
- Pour tout $i \notin V^k_{p\,(max)}$, $f^k_{(Source,\, i)} = 0$;
- Pour tout arc $(j,\, Puits)$, et tout k, $f^k_{(j,\, Puits)} \leq D^k_{(j,\, Puits)}$;
- $\sum_{k \in U} f^k_{(Puits,\, Source)}$ est maximal ;
- Le coût $\sum_{\substack{k \in U \\ g \in E_{(max)}}} C_g^k \, f_g^k$ est minimal, la condition précédente étant satisfaite.

Dans le cas où les unités de production sont en nombres entiers, nous pouvons lisser en f, en calculant par exemple, pour tout $k \in U$, un flot entier r^k, défini sur G, et tel que :

- Pour tout arc g, $r_g^k \leq \lfloor f_g^k \rfloor$;
- $r^k_{(Puits,\, Source)}$ est maximal.

Nous remplaçons alors f^k par r^k, pour tout $k \in U$.

Nous déduisons, au terme de ce processus, outre le flot f, les entrées d'un problème de *lot-sizing* du type de départ, avec les quantités :

- Producteurs : pour tout sommet producteur $i \in V^k_{p\,(max)}$, nous posons $Q_i^k = +\infty$, ce qui signifie que i ne peut plus produire k ;
- Transfert/Stockage : pour tout arc g de transfert/stockage, la capacité associée à g devient $M_g - \sum_{k \in U} M_g^k \, f_g^k$;
- Demandeurs : pour tout arc $g = (i,\, Puits)$, où i est un sommet demandeur, la nouvelle demande devient $D^k_{(i,\, Puits)} - f^k_{(i,\, Puits)}$.

Dans le cas où r ne résout pas le problème de départ, nous pouvons alors reprendre le processus de résolution à partir des données de ce problème « résidu ». Ceci est alors exprimé dans la section (4.6.3) suivante.

4.6.3 Schéma algorithmique général

Ce qui précède s'exprime au travers du schéma algorithmique *LOT-SIZING-COMPLET* (LSC), qui opère sur le graphe G et les données qui vont avec :

LOT-SIZING-COMPLET(G) :

1. Initialiser : $w \leftarrow 0$; $Stop \leftarrow false$; $M_{(aux)} \leftarrow M$; $D_{(aux)} \leftarrow D$; $Q_{(aux)} \leftarrow Q$;

2. Tant que ($D_{(aux)} \neq 0$) ET $Stop = false$, faire
3. Appliquer *LAGRANGE-COMPLET* en utilisant les quantités $M_{(aux)}$, $D_{(aux)}$, $Q_{(aux)}$, C et déduire un multiflot f et, pour chaque produit k, un ensemble producteur $V^k_{p\,(max)}$;
4. Si $f = 0$, alors
5. $Stop \leftarrow true$
6. Sinon
7. $w \leftarrow w + f$;
8. Pour tout $g = (i,\ Puits)$, $i \in V_d$, et tout $k \in U$, faire
9. $D^k_{g\,(aux)} \leftarrow D^k_{g\,(aux)} - f^k_g$;
10. Pour tout $i \in V^k_{p\,(max)}$, i producteur, i, faire
11. $Q^k_i \leftarrow +\infty$;
12. Pour tout arc g, faire
13. $M_{g\,(aux)} \leftarrow M_{g\,(aux)} - \sum_{k\in U} Q^k_g\, f^k_g$;

Le résultat global est fourni par w en cas de succès. En cas d'échec, la demande n'a pu être satisfaite.

4.7 Traitement en simultané des contraintes de capacités

Le schéma décrit en (4.6) est un schéma à double détente : la relaxation des contraintes de stockage et transfert débouche sur un problème avec contraintes de production, auquel nous appliquons à son tour un processus de relaxation lagrangienne.

Nous pouvons envisager de procéder en relaxant simultanément l'ensemble des contraintes de capacités.

4.7.1 Maximisation du lagrangien

La relaxation lagrangienne induite concerne un multiflot w défini sur le graphe orienté de départ $G = (V, E)$. Le lagrangien induit s'écrit :

$$K(w,\lambda,\mu) = \sum_{\substack{g\in E_p\\ k\in U}} (Q^k_g\, y^k_g + C^k_g\, w^k_g) + \sum_{\substack{g\in E_{st}\\ k\in U}} C^k_g\, w^k_g + \sum_{g\in E_p} \lambda_g (\sum_{k\in U} M^k_g\, w^k_g - M_g) + \sum_{g\in E_{st}} \mu_g (\sum_{k\in U} M^k_g\, w^k_g - M_g). \quad (4.13)$$

Le vecteur lagrangien $\mu = (\mu_g, g \in E_{st})$ est ici indexé sur les arcs de transfert/stockage. Le vecteur λ est indexé sur l'ensemble E_p des arcs de production. Sa résolution se ramène à celle d'une famille disjointe de problèmes de localisation. En effet, nous voyons que, w pouvant être construit à l'aide de routage aux plus courts chemins, $K(w, \lambda, \mu)$ peut se réécrire :

$$K(w,\lambda,\mu) = \sum_{k\in U}(\sum_{g\in E_p} Q_g^k\, y_g^k + \sum_{g\in E_{st^*}} C_{g,\lambda,\mu}^{*k}\, w_g^k) + \sum_{i\in V_p}\lambda_i \sum_{k\in U} M^k_{(Source,\,i)}\, w^k_{(Source,\,i)} - \sum_{g\in E_p} M_g\,\lambda_g - \sum_{g\in E_{st}} M_g\,\mu_g \quad (4.14)$$

à l'intérieur duquel chaque quantité $C^{*k}_{g,\,\mu}$ est définie, pour chaque couple i producteur, j demandeur, en $g = (i,\, j)$, comme étant la distance de i à j, au sens des plus courts chemins, pour les longueurs d'arcs dans E_{st} : $C_g^k + M_g^k\,\mu_g$, augmentée de la quantité : $C^k_{(Source,\,i)} + M^k_{(Source,\,i)}\,\lambda_{(Source,\,i)}$.

L'optimisation de ce lagrangien, se ramène donc à des problèmes $Z^k_{\lambda\,\mu\,(Relax)}$, dont pour chaque k :

- $X = V_s$;
- Pour tout $i \in V_p$, $B^k_{i,\lambda,\mu} = Q^k_{(Source,\,i)}$;
- Pour tout $i \notin V_p$, $B^k_{i,\lambda,\mu} = +\infty$;
- Pour tout couple $(i,j) \in E_{st^*}$, i producteur, et j demandeur, $I^k_{(i,j),\lambda,\mu} = D^k_{(j,\,Puits)}\,(\lambda_{(Source,\,i)} + C^{*k}_{(i,\,j),\,\mu})$

L'algorithme *GRASP-LOC-FUSION* décrit précédemment (4.4.2) peut donc être appliqué au traitement du problème $Z^k_{\lambda,\mu\,(Relax)}$. A l'issue de la résolution d'un tel problème $Z^k_{\lambda,\mu\,(Relax)}$, nous déduisons un ensemble Y^k_λ producteur, pour chaque $i \in Y^k_\lambda$ un ensemble desservi $R^k_{i,\lambda,\mu}$ et, pour chaque demandeur j, un nœud producteur $x'^k_{j,\lambda,\mu}$. Nous déduisons un flot $w^k_{\lambda,\mu}$ défini sur G en posant :

- Pour tout nœud i producteur $w^k_{(Source,\,i)\,\lambda,\mu} = \sum_{j\in R^k_{i,\lambda,\mu}} D^k_{(j,\,Puits)}$;
- Pour tout i producteur, j dans $R^k_{i,\lambda,\mu}$, nous choisissons un chemin $\Gamma^k_{(i,j),\lambda,\mu}$ qui va de i à j et qui est le plus court au sens des distances C^{*k}_μ, et nous routons $D^k_{(j,\,Puits)}$ le long de ce chemin $\Gamma^k_{(i,j),\lambda,\mu}$;
- Pour tout arc g demandeur, $w^k_g = D^k_g$.

Le schéma général de traitement *LAGRANGE-SIMULTANE* (LS) s'écrit alors :

LAGRANGE-SIMULTANE :

1. Initialiser : μ, $\lambda \leftarrow 0$; Pour chaque k, calculer le vecteur C^{*k}_μ; $Stop \leftarrow false$;

2. Tant que $Stop = false$, faire
3. Traiter chaque problème $R^k_{\lambda,\mu\,(Relax)}$ via *GRASP-LOC-FUSION* ;
4. Déduire un multiflot canonique $w_{\lambda,\mu}$;
5. Calculer la quantité Pas ;
6. Poser Π = le vecteur $(\sum_{k\in U} M^k_g\, w^k_{g,\lambda,\mu} - M_g,\ g \in E_{st})$;
7. Poser Δ = le vecteur $(\sum_{k\in U} M^k_{(Source,\,i)}\, w^k_{g,\lambda,\mu} - M_g,\ g \in E_p)$;
8. $\lambda \leftarrow \lambda + Pas\,\Delta$;
9. $\mu \leftarrow \mu + Pas\,\Pi$;
10. Mettre à jour, par propagation de contraintes, les systèmes de distances C^{*k}_μ, $k \in U$;
11. Mettre à jour $Stop$;

Incrémentalité : comme dans les sections (4.5) et (4.6), une exécution efficace de l'instruction *LAGRANGE-COMPLET* suppose de mettre en œuvre une notion d'incrémentalité, c'est-à-dire le fait de pouvoir traiter le problème de localisation associé à μ, λ à partir de la solution obtenue pour une valeur de μ, λ relativement proche.

4.7.2 Lissage

Le processus de maximisation du lagrangien s'achève en principe avec l'obtention de vecteurs $\lambda_{(max)}$ et $\mu_{(max)}$, et d'un vecteur flot $w_{(max)}$ associé. Ce vecteur flot identifie, pour tout $k \in U$, un ensemble $V^k_{p\,(max)} \subset V_p$ de sommets producteurs.

Nous pouvons alors calculer dans le graphe G, en utilisant par exemple Cplex, un flot f fractionnaire tel que :

- Pour tout arc g, $\sum_{k\in U} M^k_g\, f^k_g \leq M_g$;
- Pour tout $i \notin V^k_{p\,(max)}$, $f^k_{(Source,\,i)} = 0$;
- Pour tout arc $g \in V_d$, et tout k, $f^k_g \leq D^k_g$;
- $\sum_{k\in U} f^k_{(Puits,\,Source)}$ est maximal ;
- Le coût $\sum_{\substack{g\in E_{(max)}\\ k\in U}} C^k_g\, f^k_g$ est minimal, la condition précédente étant satisfaite.

Dans le cas où les unités de production sont en nombres entiers, nous pouvons lisser en f, en calculant par exemple, pour tout $k \in U$, un flot entier r^k, défini sur G, et tel que :

- Pour tout arc g, $r^k_g \leq f^k_g$;
- $r^k_{(Puits,\,Source)}$ est maximal.

Nous remplaçons alors f^k par r^k, pour tout $k \in U$.

Nous déduisons, au terme de ce processus, outre le flot f, les entrées d'un problème de *lot-sizing* du type de départ, avec les quantités :

- Producteurs : pour tout sommet producteur $i \in V^k_{p\,(max)}$, nous posons $Q^k_i = +\infty$, ce qui signifie que i ne peut plus produire k ;
- Transfert/Stockage : pour tout arc g de transfert/stockage, la capacité associée à g devient $M_g - \sum_{k \in U} M^k_g \, f^k_g$;
- Demandeurs : pour tout sommet demandeur i, la nouvelle demande devient $D^k_{(i,\,Puits)} - f^k_{(i,\,Puits)}$.

Dans le cas où r ne résout pas le problème de départ, nous pouvons alors reprendre le processus de résolution à partir des données de ce problème « résidu ».

Le schéma algorithmique général pour cette opération s'exprime au travers du schéma *LOT-SIZING-COMPLET*, déjà présenté en (4.6.3).

4.8 Compléments

Un problème posé, qui porte sur la capacité des processus décrits en (4.6) et (4.7) à générer une solution réalisable. Un procédé de décomposition lagrangienne s'applique en effet, de façon standard, à un problème de la forme :

Z : {Trouver un objet z, dont $w_z \geq 0$, qui minimise la quantité Φ_z}.

w_z désigne ici un vecteur de quantités $w_{i,z}$, $i \in I$, contraintes à être toutes non négatives.

Le lagrangien $L(\lambda,\, z)$ s'écrit alors $\Phi_z - \lambda w_z$, où λ est un vecteur $\lambda_i, i \in I$, supposé non négatif. Le processus de résolution par décomposition lagrangienne vient alors comme suit :

- Calculer $\lambda_{(max)}$ qui maximise la quantité $Inf_z \, L(\lambda,\, z)$;
- Calculer z^* qui minimise $L(\lambda,\, z)_{(max)}$;
- Projeter z^* sur l'ensemble des éléments z tels que $w_z \geq 0$.

Le processus décrit en (4.5), (4.6) et (4.7) ne correspond pas exactement à ce schéma : il considère l'ensemble producteur $V_{p\,(max)}$, comme étant l'objet z^*, et, au lieu de le modifier de façon à rendre possible la satisfaction des demandes, il s'efforce de satisfaire au mieux les demandes à l'aide de cet ensemble producteur, induisant donc parfois l'existence d'une demande résiduelle non satisfaite. L'expérience fait alors apparaitre des situations d'échec, c'est-à-dire des situations où il n'y a pas de réponse à l'ensemble des demandes, alors même que des solutions existent. Afin

de palier cet inconvénient, plusieurs approches peuvent être envisagées, qui peuvent être combinées :

- 1ère approche : nous focalisons l'attention sur le calcul de $\lambda_{(max)}$, en s'efforçant de faire en sorte que l'ensemble producteur induit soit le plus à même de satisfaire l'ensemble des demandes ;
- 2ème approche : nous essayons d'affiner les procédures de lissage, de façon à anticiper les échecs potentiels ;
- 3ème approche : nous compensons les situations d'échec par des actions de re-routage et par la création de producteurs actifs additionnels.

4.8.1 Perfectionnement du calcul de $\lambda_{(max)}$

La quantité $Inf_z\, L(\lambda, z)$ est linéaire par morceaux en λ, et sa maximisation tend à placer l su des zones de frontières correspondant à des objets $z_\lambda = ArgInf_z L(\lambda, z)$, parfois sensiblement différents les uns des autres. Dans le cas présent, z_λ renvoie à des ensembles producteurs $V^k_{p,\lambda}$, $k \in U$. Il est important d'essayer de choisir λ, de telle sorte que ces ensembles producteurs soient les mieux à même de répondre à la demande.

Pour ce faire, nous pouvons revenir sur le fait que chaque nœud producteur ou demandeur est en fait un couple formé d'un nœud géographique $i \in V_s$ et d'un instant t, et définir, pour chaque produit $k \in U$, et tout ensemble producteur V^k_p :

- La courbe de profil K^k_t = la somme de toutes les demandes courantes en k formulées à l'instant par les nœuds de V_s ;
- La courbe de profil cumulé $K^{*k}_t = \sum_{u \leq t} K^k_u$;
- La courbe de profil $A_{V_p,t}$ = la somme des capacités de productions (les quantités $M_{i,t}$) à l'instant t pour les producteurs de V_p ;
- La courbe de profil compensé $A^*_{V_p,t} = \sum_{u \leq t} A_{V_p,t}$.

Nous considérons alors de coefficients de normalisation Y^k, $k \in U$, supposés moyenner les coefficients $M^k_{i,t}$, et nous posons :

- $X(V_p) = Inf_{t \in T}(A^*_{V_p,t} - \sum_{k \in U} Y^k\, K^{*k}_t)$ = Plausibilité de l'ensemble producteur V_p par rapport aux demandes.

Clairement, cette quantité X tend à exprimer la capacité qu'aura l'ensemble producteur V_p à satisfaire les demandes courantes.

Le processus de sélection de $\lambda_{(max)}$ qui en dérive consiste alors, au cours de la maximisation de la quantité $Inf_z L(\lambda, z)$, à tenir à jour une liste FIFO, formée des m (m est un paramètre) dernières valeurs λ générées, accompagnées de leur ensemble

producteur $V_{p,\lambda}$ associé, et, une fois λ stabilisé, de retenir comme valeur $\lambda_{(max)}$ celle des valeurs λ qui induit la plus haute valeur $X(V_{p,\lambda})$.

4.8.2 Perfectionnement des procédures de lissage

Le point principal consiste ici à faire en sorte que, au cas où des demandes résiduelles subsisteraient à l'issue de la phase de lissage, elles soient traitables au tour d'après, c'est-à-dire qu'elles demeurent à portée de nœuds producteurs qui n'ont pas été désactivés.

Considérons donc l'ensemble producteur $V_{p\,(max)}$ issu du processus de maximisation du lagrangien : de fait, chaque élément dans $V_{p\,(max)}$ est un nœud x producteur du réseau, accompagné d'un produit k. Notons, en adoptant la même convention, V'_p l'ensemble des nœuds producteurs déjà utilisés dans les phases antérieures et qu'il est donc possible de considérer comme désactivés.

Pour chaque produit $k \in U$, et chaque demande courante en k émanant d'un nœud y, nous calculons, à l'aide d'un algorithme simple de recherche de chemins, l'ensemble Q^k_y des nœuds producteurs encore actifs (pour le produit k) susceptibles d'alimenter y en k. Si Q^k_y est inclus dans $V_{p\,(max)}$, alors, nous incluons le couple (y, k) dans un ensemble PRIOR. Ceci amène à reformuler le programmes linéaires décrit en (4.6.2) en imposant que la demande induite par chaque couple (y, k) de PRIOR soit satisfaite :

Calculer dans le graphe G un flot f fractionnaire tel que :

- Pour tout arc g, $\sum_{k\in U} M^k_g\, f^k_g \leq M_g$;
- Pour tout $i \notin V^k_{p\,(max)}$, $f^k_{(Source,\,i)} = 0$;
- Pour tout arc $g = (j,\, Puits)$, et tout k, $f^k_g \leq D^k_g$;
- Pour chaque couple (y, k) dans PRIOR, $f^k_{(y,\,Puits)} = D^k_{(y,\,Puits)}$;
- $\sum_{k\in U} f^k_{(Puits,\,Source)}$ est maximal ;
- Le coût $\sum_{\substack{g\in E_{(max)}\\ k\in U}} C^k_g\, f^k_g$ est minimal, la condition précédente étant satisfaite.

4.8.2.1 Remarque :

Ce programme peut ne pas avoir de solution. Dans ce cas, nous relâchons la contrainte ainsi introduite.

4.8.3 Perfectionnement des procédures de lissage : extension

Notre approche de la procédure de lissage peut être critiquée dans la mesure où, à chaque étape, elle verrouille la façon dont sont routées les productions en direction de celles de demandes courantes qui sont satisfaites. Nous pouvons revenir sur ce point en considérant que $V_{p\,(max)}$ (qui est forcément différent de V'_p) vient s'agréger à

V'_p, et que le programme linéaire décrit en section précédente dérive de cet ensemble producteur agrégé ($V_{p\,(max)}$ est remplacé par $V'_p \cup V_{p\,(max)}$ dans la formulation de ce programme ; les capacités et les demandes sont, dans ce programme, les capacités et les demandes originelles).

4.8.4 Gestion des échecs : réparation des routages

Considérons que nous sommes arrivés, au terme du processus général décrit en section (4.6) ou (4.7), à une situation qui laisse certaines demandes résiduelles insatisfaites. Soit D une telle demande, associée à un couple (y, k). Nous cherchons alors le plus court chemin γ reliant (y, k) à un nœud producteur x, dans le graphe sont les arcs de G pris à « rebrousse-poil », et dont les coûts sont les dépassements de capacités induit par une augmentation de la valeur f^k de $\delta = D^k_{(y,\,Puits)}$. Nous satisfaisons alors cette demande résiduelle en augmentant f^k de δ le long de γ. Puis, pour chaque arc g de γ induisant un dépassement de capacité, nous nous efforçons de résorber ce dépassement en choisissant k' (éventuellement égal à k) et en construisant une chaine augmentant Γ qui relie l'extrémité de g à l'origine de g : nous cherchons dans un premier temps G qui n'induit la création d'aucun nouveau producteur, puis, en cas d'échec, nous acceptions la création d'un nouveau producteur pour k'. Le choix de k' est un choix par continuité, c'est-à-dire que nous nous efforçons, à chaque fois, de travailler avec une des ressources pour laquelle nous avons créé un nouveau producteur.

4.9 Résultats

En suite nous présentons les résultats obtenus par notre méthode appliquée à des instances du problème *single-level lot-sizing with vehicle routing problem* (1-LSVRP) de taille (produit × sites × périodes). Ces instances ont été générées avec des coûts aléatoires compris dans l'intervalle $[a,\ b]$ avec une distribution uniforme $U[a,\ b]$:

- Coût unitaire de production : $U[0,5\ ;\ 5,0]$;
- Coût unitaire de stockage : $U[0,3\ ;\ 1,5]$;
- Coût unitaire de transfert : $U[2,0\ ;\ 8,0]$;
- Coût *setup* : $U[1,0\ ;\ 3,0]$;
- Demande moyenne[12] : $U[4,0\ ;\ 12,0]$;
- Temps de *setup* : $U[0,2\ ;\ 0,5]$;
- Capacité de production : $U[0,0\ ;\ 1200,0]$;
- Capacité de stockage : $U[10,0\ ;\ 50,0]$;
- Capacité de transfert : $U[0,0\ ;\ 50,0]$;
- Taux de production : $U[1,0\ ;\ 3,0]$;
- Taux de stockage : $U[2,0\ ;\ 8,0]$;
- Taux de transfert : $U[2,0\ ;\ 8,0]$.

Chaque type de problème commence avec les multiplicateurs lagrangiens pour la partie production (λ) et la partie stockage/transfert (μ) à la valeur de 0. Chaque tableau contient deux colonnes dont la première concerne les valeurs des solutions trouvées par Cplex pour le problème sans capacité ($P1$), le problème uniquement avec capacité de production ($P2$, le même traité par [86]), le problème avec toutes les capacités ($P3$) et les valeurs fractionnaires des problèmes $P1$ et $P3$. La deuxième colonne concerne aux valeurs trouvées par notre méthode et montre la solution pour la relaxation lagrangienne (Lag), la valeur fournie par l'heuristique ($Heurist$) et le gap (en %) ente la solution du problème complet fournie par Cplex et la solution heuristique, ceci calculé comme $(Heurist - P3) \times 100/P3$.

Le 1er tableau (4.2) montre les résultats pour les instances de taille $(5 \times 3 \times 3)$ dont 1 producteur est considéré comme « écarté », cela veut dire que sa capacité de production est nulle.

TABLE 4.2 – Résultats des problèmes (5×3×3).

Solutions Cplex					**Solutions heuristiques**		
FractP1	P1	P2	P3	FractP3	Lag	Heurist	Gap (%)
327,00	470,20	835,30	840,20	720,20	614,90	844,40	0,49
423,60	573,70	776,50	778,10(1)	671,10	613,70	778,10(1)	0,00
389,02	552,27	823,60	862,30(2)	764,70	628,50	865,60(2)	0,38
448,40	651,00	774,20	794,20	648,20	683,10	799,20	0,62
438,30	644,10	818,00	838,40	688,10	699,90	838,40	0,00
Moyenne							0,30

Remarques :

Ces 5 instances ne nécessitent qu'une seule itération de la boucle globale de la méthode. Le valeurs de *P3* et *Heurist* identifiés par (1) proviennent de la même solution optimale. Autrement dit, malgré les 3 types de capacités, le groupe de producteur sélectionné pour la projection inclus les producteurs de la solution optimale. Lorsque *Heurist* et *P3* sont proches sans être identiques, comme ceux identifiés par (2), la projection est amputée d'au moins un producteur de la solution optimale.

Le tableau (4.3) montre les résultats pour les instances de type $(5 \times 4 \times 6)$, dont un producteur est écarté. Nous ajoutons aussi une limite de recherche pour la borne inférieure fournie par le lagrangien ($supMIN$), ceci ajusté à 200 itérations pour ces instances.

Remarques :

Les variables de la solution optimale « serre » aléatoirement les arcs d'acheminement (stockage compris) et, toujours de manière stochastique, certains arcs sont fermés (e.g. impossible de livrer directement le site 1 vers le site 2).

TABLE 4.3 – Résultats des problèmes (5×4×6).

Solutions Cplex					**Solutions heuristiques**		
FractP1	P1	P2	P3	FractP3	Lag	Heurist	Gap (%)
2340,10	2989,00	4335,10	4559,60	3737,40	4510,80	4620,40	1,31
2111,90	2842,20	3156,10	3282,60	2616,00	3235,30	3382,70	2,95
2448,30	3199,70	3876,00	3989,90	3268,80	3944,20	4044,70	1,35
2246,40	2951,20	3558,80	3680,80	2975,60	3620,00	3702,60	0,58
2480,70	3170,70	3736,60	3822,90	3085,20	3767,20	3822,90	0,00
Moyenne							1,24

Les prochains tableaux (4.4, 4.5 et 4.6) montrent les résultats pour des instances de tailles plus importantes. La génération de ces instances est identique à la précédente batterie hormis la différence d'une augmentation des capacités de production qui était nécessaire pour l'élargissement des types de produit et non pas du reste.

Le tableau (4.4) présent les résultats pour les instances de taille $(10 \times 4 \times 6)$ avec 2 producteurs écartés et dont les tentatives de recherche de $supMIN$ sont de 400 itérations.

TABLE 4.4 – Résultats des problèmes (10×4×6).

Solutions Cplex					**Solutions heuristiques**		
FractP1	P1	P2	P3	FractP3	Lag	Heurist	Gap (%)
4265,30	5478,00	6370,70	6772,70	5437,80	6743,10	6894,70	1,76
4409,70	5737,40	6254,00	6393,00	5404,50	6387,90	6393,00	0,00
4766,90	6169,40	7236,10	7437,60	6282,40	7423,40	7515,90	1,04
4469,80	5757,60	6732,40	6883,70	5792,90	6869,10	6883,70	0,00
4227,30	5681,80	6085,50	6363,20	5325,60	6339,40	6363,20	0,00
Moyenne							0,56

Remarques :

Les dernières mises à jour font disparaître le problème plusieurs fois (celui des multiplicateurs de Lagrange qui ne s'améliorent pas en les initialisant par une valeur nulle, nous avions *Lag* identique à *P3*). Au contraire, la gestion des multiplicateurs de Lagrange fonctionne désormais bien et il serait même intéressant de lancer de plus longs calculs afin d'estimer les chances d'obtenir à une valeur optimale avec recherche du *supMIN* à grande échelle (un nombre d'itérations très supérieur à 400).

Une autre remarque est que pour quelques instances nous avons obtenu le résultat optimal, dont quelques uns donnent une solution par heuristiques identique à celles de Cplex.

Le tableau (4.5) montre les résultats pour instances de taille $(15 \times 4 \times 6)$ dont 2 producteurs sont écartés et dont la valeur de $supMIN$ est de 100 itérations.

Les résultats pour le dernier groupe d'instances $(10 \times 5 \times 10)$ est présenté dans le tableau (4.6). Elles considèrent 2 producteurs écartés. La valeur de $supMIN$ pour

TABLE 4.5 – Résultats des problèmes (15×4×6).

Solutions Cplex					**Solutions heuristiques**		
FractP1	P1	P2	P3	FractP3	Lag	Heurist	Gap (%)
6383,40	8320,80	8837,10	9706,40	7796,70	9446,30	9970,30	2,64
7169,60	9491,50	10777,10	11868,20	9670,30	11733,20	12240,30	3,03
6318,00	8585,50	9459,10	10207,50	8025,80	9980,12	10251,40	0,42
6895,00	8797,50	10071,60	11623,80	9570,20	11377,10	11928,60	2,55
6749,80	8968,00	10706,90	11236,50	9092,60	11151,30	11257,60	0,18
Moyenne							1,77

la 1ère itération de la méthode est de 20 tentatives de recherche. Pour les itérations suivantes $supMIN = 100$.

TABLE 4.6 – Résultats des problèmes (10×5×10)

Solutions Cplex					**Solutions heuristiques**		
FractP1	P1	P2	P3	FractP3	Lag	Heurist	Gap(%)
7731,70	10419,80	12136,30	12699,80	10030,10	12627,10	12800,70	0,78
8018,00	11093,00	12112,30	13136,20	10332,40	12965,00	13232,30	0,72
8005,30	11088,00	11802,60	12410,00	9721,50	12366,80	12423,40	0,10
8022,10	11414,90	12313,00	12529,40	9697,20	12498,90	12597,20	0,53
7772,20	10951,90	11700,50	12162,50	9371,40	12100,60	12253,60	0,74
Moyenne							0,58

Vu de manière globale, l'algorithme qui nous avons développé montre que pour ces batteries d'instances, nous obtenons en moyenne des solutions de très bonne qualité car elles sont assez proche de l'optimale (inférieur à 1%).

Conclusion

Dans ce travail nous avons traité trois types de problèmes de production avec transport : production multi-niveaux, multi-sites, multi-produits avec consommation de composants et transport inter-sites réalisé par des véhicules (LSVRP) ; multi-niveaux, production multi-sites, mono-produit avec consommation de composants et transport (cas particulier S-LSVRP) ; et production mono-niveau, multi-sites avec transfert instantanée de produits entre sites (1-LSVRP). Pendant la réalisation de cette thèse nous avons approfondi le sujet grâce à une littérature abondante afin d'étudier les différents types de problèmes, leurs modèles et approches de traitement. Le résultat a été la création de méthodes efficaces pour la résolution des problèmes proposés, ce qui nous a permis de proposer plusieurs contributions dans le domaine.

Nous avons présenté dans le premier chapitre une étude bibliographique qui contenait plusieurs modèles et approches de résolution sur les problèmes de type *lot-sizing* et tournée de véhicules. Les modèles ont été choisis par rapport aux liens avec les problèmes que nous avons proposés, et nous les avons présentés par ordre de parution, tout en montrant leur progression. Pour les méthodes de résolution, nous avons mis l'accent sur celles qui travaillent sur la relaxation linéaire, étant donné qu'elle sert aussi comme base pour nos heuristiques. Nous avons aussi accordé un espace considérable de dans chapitre pour présenter les caractéristiques principales des problèmes traités dans le but de bien définir l'espace sur lequel repose notre recherche.

Dans le deuxième chapitre nous avons présenté dans un premier temps deux de nos modèles mathématiques reposant sur la programmation linéaire : le *Lot-Sizing with Vehicle Routing Problem* (LSVRP), modèle plus général qui comprend des problèmes de production sur plusieurs sites avec un problème de tournée de véhicules pour réaliser le transport des produits entre les sites ; et le *Single-item* LSVRP (S-LSVRP), cas particulier du modèle général dans lequel la production sur chaque site est dédiée à un seul produit. Agréger ces deux sujets d'optimisation (planification et transport) a été une tâche délicate car chaque problème repose sur un univers différent dont les caractéristiques ne sont pas tout à fait les mêmes. Nous avons eu recours, par exemple, à l'utilisation d'un graphe temporel discrétisé sur les périodes de temps (normalement utilisé par des problèmes de lot-sizing) pour pouvoir représenter les délais de transport inter-sites, et aussi à des simplifications pour « alléger » les modèles.

Dans un deuxième temps, nous avons présenté les heuristiques élaborées pour résoudre chacun des problèmes. Une partie sont des heuristiques basées sur la programmation entière mixte (MIP) et reposent sur la relaxation linéaire des variables de décision (de *setup* pour la production et du nombre d'arcs pour le transport). L'autre heuristique a été développée pour le cas particulier, et son principe est d'effectuer des fusions d'acheminements de la chaîne de production dans le but d'éviter les opérations de *setup*. Chaque méthode a ses avantages et désavantages, et convient à un type particulier de problème. Leur conception a été aussi l'occasion d'étudier de manière approfondie la théorie de la relaxation et la manipulation du logiciel d'optimisation Cplex, version 11, que nous avons utilisé.

Les résultats fournis par les heuristiques proposées ont été présentés dans le chapitre 3. Ils ont été comparés avec les bornes inférieures de bonne qualité fournies par Cplex et se s'ont montré très satisfaisants. Dans le même chapitre nous avons fait plusieurs analyses à propos des caractéristiques de chaque modèle et méthode développés. Nous avons comparé les heuristiques entre elles et l'efficacité de chacune par rapport au type de problème. Chaque méthode à ses points forts et faibles. En termes de qualité de solution, l'heuristique *Relax-and-Fix* s'est montré la plus efficace. Pour le cas particulier de production dédiée, l'heuristique de fusion gagne en terme de temps d'exécution et donne une solution de bonne qualité.

Le denier chapitre a présenté une autre façon de résoudre le problème de multi-production à mono-niveau sur plusieurs sites (1-LSVRP). Puisque le transport a été considéré comme un transfert instantané de produits entre sites, au lieu de considérer un schéma de tournée de véhicules avec différents délais de livraison, le problème a été traité comme un problème de multiflot de produits. Cette façon de manipuler le 1-LSVRP nous a amené à élaborer une approche basée sur la relaxation lagrangienne des contraintes de capacité. Cette technique s'est montrée aussi efficace que celles de la littérature. Ses résultats sont présentés à la fin du chapitre et sont prometteurs.

Nos travaux laissent ouvertes plusieurs perspectives de recherche. Même avec des simplifications, nos modèles restent assez complexes et produisent des problèmes très difficiles à traiter. C'est évident qu'un modèle plus complet s'approche encore plus d'un problème réel, cependant cette proximité amène un nombre exponentiel de variables à traiter. Alors, une étude approfondie doit être réalisée pour concevoir des modèles qui permettront de développer des algorithmes de plus faible complexité. Par ailleurs, il faudra prendre en compte de manière plus précise la partie transport. En effet le modèle développé dans cette thèse ne permet pas de faire un suivi précis des tournées réalisées ni du flot des produits. Cependant il est clair que cette opération risque d'augmenter encore la complexité.

Les heuristiques offrent diverses possibilités d'extension. En ce qui concerne les heuristiques « MIP », la méthode *Relax-and-Fix* peut encore être paramétrée au niveau des variables à fixer, d'organisation des groupes de variables, etc. Les mêmes extensions sont valides pour les heuristiques *Insert-and-Fix* et *Franctional Relax-and-Fix*, et nous pouvons aussi jouer avec les variables de « inactives » de ces deux

heuristiques. En effet le nombre de variables inactives influe sur la qualité de la solution.

Entre les heuristiques « MIP », l'heuristique *Diving* est celle qui donne les solutions de moins bonne qualité et les temps de calcul les moins bons. Ceci est dû au fait qu'elle est basée sur l'arrondi et la fixation unitaire des variables à chaque itération. Cette opération est très onéreuse en termes de temps d'exécution et, malgré nos tentatives de grouper les variables, le temps pris par la méthode reste assez important. Alors, une étude sur des nouveaux critères de fixation et de groupement des variables peut nous conduire à des améliorations.

Par rapport aux deux autres méthodes (fusion et relaxation lagrangienne), il se trouve que l'heuristique de fusion d'acheminements de production (PSF) peut être adaptée au cas général, mais cette tâche n'est pas une activité triviale. Il faudra ajouter d'autres critères d'évaluation de solution qui prennent en compte la combinaison produit $\times$ site $\times$ période, sans pour autant trop augmenter le temps de calcul, point fort de cette heuristique. Une autre possibilité d'extension est d'utiliser d'autres méthodes pour faire la recherche locale pour la partie production, par exemple le *local-branching*. De la même manière, pour le problème 1-LSVRP, l'affectation des sites producteurs peut être rendue plus efficace à l'aide des méthodes cités précédemment.

Une dernière partie de notre étude consiste à élaborer un modèle étendu de production et de transport en utilisant le concept de « multi-gammes d'assemblage ». En gros, un site pourra fabriquer un produit de plusieurs façons selon la disponibilité des ressources. Cette extension a été prévu pour cette thèse et un modèle préliminaire a déjà été élaboré. Cependant l'implémentation du problème ne consiste pas en un simple ajout de dimension et de contraintes. Ce problème mérite une étude bien plus approfondie, c'est pourquoi nous avons décider de laisser ce travail en perspective.

Bibliographie

[1] N. Absi et S. Kedad-Sidhoum : The Multi-item Capacitated Lot-sizing Problem with Safety and Demand Shortage Costs. *CAOR*, 36(11):2926–2936, 2009.

[2] K. Akartunali et A.J. Miller : A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research*, 193(2):396–411, 2009.

[3] A. Alfieri, S. D'Orazio et P. Brandimarte : Lp-based heuristics for the capacitated lot-sizing problem: the interaction of model formulation and solution algorithm. *International Journal of Production Research*, 40:441–458, 2002.

[4] F.P. Alvelos, J.M.V. Carvalho et C.M.O. Pimentel : Algoritmo de partição e geração de colunas para dimensionamento de lotes de produção. *Investigação Operacional*, 26:139–146, 2006.

[5] M. Arenales, V.A. Armentano et R. Morabito : *Pesquisa Operacional - Modelagem e Algoritmos*. Campus, 2006.

[6] V.A. Armentano et F.M.B. Toledo : A Lagrangean-based Heuristic for the Capacitated Lot-sizing Problem in Parallel Machines. *EJOR*, 175:1070–1083, 2006.

[7] E. Balas et N. Christofides : A restricted lagrangean approach to the traveling salesman problem. *Mathematical Programming*, 21:19–46, 1981. 10.1007/BF01584228.

[8] J.F. Bard et N. Nananukul : The integrated production-inventory-distribution-routing problem. *Journal of Scheduling*, 12(3):257–280, 2009.

[9] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh et P.H. Vance : Branch & price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[10] R. Barták : Constraint programming: In pursuit of the holy grail. *In Proceedings of Week of Doctoral Students (WDS99)*, pages 555–564, 1999.

[11] P.P. Belfiore et L.P.L. Fávero : Problema de roteirização de veículos com janelas de tempo: revisão da literatura. *XIII SIMPEP*, 2006.

[12] G. Belvaux et L.A. Wolsey : Lot-Sizing Problems: Modelling Issues and a Specialized Branch-and-Cut System bc-prod. Internal report, Center for Operations Research and Econometrics, Universite Catholique de Louvain, Belgium, 1998.

[13] G. Belvaux et L.A. Wolsey : LOTSIZELIB: A Library of Models and Matrices for Lot-Sizing Problems. Internal report, Center for Operations Research and Econometrics, Universite Catholique de Louvain, Belgium, 1999.

[14] G.R. Bitran et H.H. Yanasse : Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1174–1186, 1982.

[15] M. Boudia, M.A.O. Louly et C. Prins : Global optimization of production and distribution. *Proceedings of the International Conference on Industrial Engineering and Systems Management*, 2005.

[16] M. Boudia, M.A.O. Louly et C. Prins : Grasp for the combined optimization of production and distribution. *MIC2005: The Sixth Metaheuristics International Conference*, 2005.

[17] M. Boudia, M.A.O. Louly et C. Prins : A reactive grasp and path relinking for a combined production-distribution problem. *Comput. Oper. Res.*, 34(11): 3402–3419, 2007.

[18] M. Boudia, M.A.O. Louly et C. Prins : Fast heuristics for a combined production planning and vehicle routing problem. *Production Planning & Control: The Management of Operations*, 19(2), 2008.

[19] M. Boudia et C. Prins : A memetic algorithm with dynamic population management for an integrated production-distribution problem. *European Journal of Operational Research*, 195(3):703–715, 2009.

[20] N. Brahimi, S. Dauzere-Pérès, N.M. Najid et A. Nordli : Single item lot sizing problems. *European Journal of Operational Research*, 168(1):1–16, 2006.

[21] O. Bräysy, M. Gendreau, G. Hasle, A. Løkketangen et J.Y. Potvin : Metaheuristics for the vehicle routing problem and its extensions: a categorized bibliography. *In* B. Golden, S. Raghavan et E. Wasil, éditeurs : *The Vehicle Routing Problem - Latest Advances and New Challenges*. Springer Verlag, Heidelberg, 2008.

[22] G. Carpaneto et P. Toth : Some new branching and bounding criteria for the asymmetric travelling salesman problem. *Management Science*, 26(7):736–743, 1980.

[23] P. Chandra et M.L. Fisher : Coordination of production and distribution planning. *European Journal of Operational Research*, 72(3):503–517, 1994.

[24] G. Clarke et J.W. Wright : Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, juillet 1964.

[25] M. Comelli, M. Gourgand et D. Lemoine : A review of tactical planning models. *Journal of Systems Science and Systems Engineering*, 17:204–229, 2008. 10.1007/s11518-008-5076-8.

[26] M. Constantino : A cutting plane approach to capacitated lot-sizing with start-up costs. *Mathematical Programming*, 75:353–376, 1996. 10.1007/BF02592189.

[27] M. Constantino, T. Grant et P. Hahn : Lower bounds in lot-sizing models: a polyhedral study. *Mathematics of Operations Research*, 23:101–118, January 1998.

[28] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon et F. Soumis : *The VRP with time windows*, chapitre 7, pages 157–194. SIAM monographs on discrete mathematics and applications, 2001.

[29] G. Dantzig, R. Fulkerson et S. Johnson : Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[30] G.B. Dantzig et J.H. Ramser : The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 1959.

[31] S.A. de Araujo et M.N. Arenales : Problema de dimensionamento de lotes monoestagio com restrição de capacidade: Modelagem, método de resolução e resultados computacionais. *Pesquisa Operacional*, 20, December 2000.

[32] L.P. de Assis : Algoritmos para o problema de roteamento de veículos com coleta e entrega simultâneas. Mémoire de D.E.A., UFMG - Universidade Federal de Minas Gerais, Belo Horizonte - Brésil, Juillet 2007.

[33] Z. Degraeve et R. Jans : Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, 177(3):1855–1875, 2007.

[34] J. Desrosiers et M.E. Lübbecke : Selected topics in column generation. *Operations Research*, 53:1007–1023, 2002.

[35] C. Dillenberger, L.F. Escudero, A. Wollensak et W. Zhang : On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*, 75(2):275–286, June 1994.

[36] G. Dobson : The economic lot-scheduling problem: Achieving feasibility using time-varying lot sizes. *Operations Research*, 35:764–771, September-October 1987.

[37] A. Drexl et A. Kimms : Lot sizing and scheduling - survey and extensions. *European Journal of Operational Research*, 99(2):221–235, 1997.

[38] S.E. Elmaghraby : The economic lot scheduling problem (elsp): Review and extensions. *Management Science*, 2, February 1978.

[39] D. Erlenkotter : Ford whitman harris and the economic order quantity model. *Operations Research*, 38(6):937–946, 1990.

[40] B. Escoffier et O. Spanjaard : *Dynamic Programming*, chapitre 4, pages 71–98. ISTE–Wiley, 6 2010. isbn: 9781848211476.

[41] L.G.A. Espejo et R.D. Galvão : O uso das relaxações lagrangeana e surrogate em problemas de programação inteira. *Pesquisa Operacional*, 22:387–402, 07 2002.

[42] S. Çetinkaya, C.-Y. Lee et A.P.M. Wagelmans : A dynamic lot-sizing model with demand time windows. Tinbergen Institute Discussion Papers 99-095/4, Tinbergen Institute, 1999.

[43] S.M.T. Fatemi Ghomi, B. Karimi et J.M. Wilson : The capacitated lot sizing problem: a review of models and algorithms. *OMEGA - The International Journal of Management Science*, 31:365–378, April 2003.

[44] A. Federgruen et M. Tzur : A simple forward algorithm to solve general dynamic lot sizing models with n periods in 0(n log n) or 0(n) time. *Management Science*, 37(8):909–925, 1991.

[45] D. Feillet : A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

[46] T.A. Feo et M.G.C. Resende : Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[47] M. Florian, J.K. Lenstra et R. Kan : Deterministic production planning: Algorithms and complexity. *Management Science*, 1980.

[48] B. Gavish et S.C. Graves : The travelling salesman problem and related problems. Working paper, Operations Research Center, MIT, 1978.

[49] C.D. Gelatt, S. Kirkpatrick et M.P. Vecchi : Optimization by Simulated Annealing. *Science*, 220, 4598(4598):671–680, 1983.

[50] M. Gendreau et G. Pesant : A constraint programming framework for local search methods. *J. Heuristics*, 5(3):255–279, 1999.

[51] F. Glover, G. Jones, D. Karney, D. Klingman et J. Mote : An integrated production, distribution, and inventory planning system. *Interfaces*, 9(5):21–35, 1979.

[52] F. Glover et G.A. Kochenberger : *Handbook of Metaheuristics*, volume 57. Kluwer Academic Publishers, 2003.

[53] F. Glover et M. Laguna : *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[54] F. Glover et M. Laguna : Fundamentals of Scatter Search and Path Relinking. *Control and Cybernetics*, 29(3):653–684, 2000.

[55] M.C. Goldbarg et H.P.L. Luna : *Otimização Combinatória*. Campus, 2005.

[56] D.E. Goldberg : *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st édition, 1989.

[57] M. Gondran et M. Minoux : *Graphes et algorithmes*. Collection EDF R&D. Tec & Doc, 2009.

[58] K. Haase : *Lotsizing and scheduling for production planning.* Lecture Notes in Economics and Mathematical Systems 408. Springer-Verlag, 1994.

[59] J. Hahm et C.A. Yano : The economic lot and delivery scheduling problem: the single item case. *International Journal of Production Economics*, 28(2): 235–252, November 1992.

[60] F.W. Harris : How many parts to make at once. *Factory, The Magazine of Management*, 10(2):135–136, 152, February 1913.

[61] W. Heuvel et A.P.M. Wagelmans : Four equivalent lot-sizing models. Econometric institute report, Erasmus University Rotterdam, Econometric Institute, January 2007.

[62] H.-C. Hwang et W. Jaruphongsa : Dynamic lot-sizing model with demand time windows and speculative cost structure. *Operations Research Letters*, 34(3):251–256, 2006.

[63] J. Jaffar et J.-L. Lassez : Constraint logic programming. *In Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL'87, pages 111–119, New York, NY, USA, 1987. ACM.

[64] R. Karp : Reducibility among combinatorial problems. *In* R. Miller et J. Thatcher, éditeurs : *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[65] B.M. Khumawala, J.-M. Thizy et L.N. van Wassenhove : Comparison of exact and approximate methods of solving the uncapacitated plant location problem. *Journal of Operations Management*, 6(1):23–34, 1985.

[66] C. Kilger et H. Stadtler : *Supply Chain Management and Advanced Planning: Concepts, Models, Software and Case Studies.* Springer, 2008.

[67] G. Laporte : The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2): 231–247, 1992.

[68] F.B. Lawrence et E.P. Robinson : Coordinated capacitated lot-sizing problem with dynamic demand: A lagrangian heuristic. *Decision Sciences*, 35(1):25–53, 2004.

[69] L. Lei, S. Liu, S. Park et A. Ruszczynski : On the integrated production, inventory, and distribution routing problem. *IIE Transactions*, 38(11):955–970, November 2006.

[70] R. Levi, A. Lodi et M. Sviridenko : Approximation algorithms for the multi-item capacitated lot-sizing problem via flow-cover inequalities. *In Proceedings of the 12th international conference on Integer Programming and Combinatorial Optimization*, IPCO'07, pages 454–468, Berlin, Heidelberg, 2007. Springer-Verlag.

[71] J. Maes, J.O. McClain et L.N. Van Wassenhove : Multilevel capacitated lotsizing complexity and lp-based heuristics. *European Journal of Operational Research*, 53(2):131–148, 1991.

[72] A.S. Manne : Programming of Economic Lot Sizes. *Management Science*, 4:115–135, 1958.

[73] S.E. Merzouk : *Probleme de Dimensionnement de Lots et de Livraisons - Application au cas d'une Chaine Logistique*. Automatique et informatique, université de Technologie de Belfort-Montbéliard, 2007.

[74] B. Meyer : Hybrids of constructive metaheuristics and constraint programming: A case study with aco. *In Hybrid Metaheuristics*, pages 151–183. 2008.

[75] C.E. Miller, A.W. Tucker et R.A. Zemlin : Integer programming formulation of traveling salesman problems. *J. ACM*, 7:326–329, October 1960.

[76] D.L. Miller et J.F. Pekny : Results from a parallel branch and bound algorithm for the asymmetric traveling salesman problem. *Operations Research Letters*, 8(3):129–135, 1989.

[77] M.C.V. Nascimento : Uma Heurística GRASP para o Problema de Dimensionamento de Lotes com Múltiplas Plantas. Mémoire de D.E.A., USP - São Carlos, São Carlos - Brésil, April 2007.

[78] M.C.V. Nascimento, M.G.C. Resende et F.M.B. Toledo : GRASP Heuristic with Path-Relinking for the Multi-Plant Capacitated Lot Sizing Problem. *EJOR*, 200(3):747–754, 2009.

[79] G. Pesant et L.-M. Rousseau : *Programmation par contraintes*, chapitre 10, pages 223–246. Presses Internationales Polytechnique, 2005.

[80] D. Pisinger et S. Ropke : A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 2004:750–775, 2006.

[81] D. Pisinger et S. Ropke : A general heuristic for vehicle routing problems. *Computers & operations research*, 34:2403–2435, August 2007.

[82] Y. Pochet et L.A. Wolsey : *Production Planning By Mixed Integer Programming*. Springer, 2006.

[83] C.N. Potts et L.N. Van Wassenhove : Integrating Scheduling with Batching and Lot-Sizing: A Review of Algorithms and Complexity. *The Journal of the Operational Research Society*, 43(5):395–406, 1992.

[84] T. Prabhakar : A Production Scheduling Problem with Sequencing Considerations. *Management Science*, 21(1):34–42, 1974.

[85] J. Rogers : A computational approach to the economic lot scheduling problem. *Management Science*, 4, April 1958.

[86] M. Sambasivan et S. Yahya : A Lagrangean-based Heuristic for Multi-plant, Multi-item, Multi-period Capacitated Lot-sizing Problems with Inter-plant Transfers. *CAOR*, 32:537–555, 2005.

[87] K. Smith : Investigation of the Uncapacitated Lot-Sizing Problem with Probabilistic Constraints. Working paper, program in applied mathematics, University of Arizona, Tucson, AZ, USA, may 2008.

[88] H. Stadtler : Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. Publications of Darmstadt Technical University, Institute for Business Studies (BWL) 20204, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL), 2003.

[89] C. Suerie : *Lecture Notes in Economics and Mathematical Systems. Time Continuity in Discrete Time Models: New Approaches for Production Planning in Process Industries*, volume 552. Springer Berlin Heidelberg, 2005.

[90] The NEMI Perfect BoM Team : In Search of the Perfect Bill of Materials (BoM). White paper, National Electronics Manufacturing Initiative, Inc., march 2002.

[91] L.J. Thomas, W.W. Trigeiro et J.O. McClain : Capacitated Lot Sizing with Setup Times. *MSCIAM*, 35:353–366, 1989.

[92] F.M.B. Toledo et A.L. Shiguemoto : Lot-sizing problems with several production centers. *Pesquisa Operacional*, 25(3):479–492, December 2005.

[93] D. Toth et P. Vigo : *The Vehicle Routing Problem*. SIAM, 2002.

[94] H. Toussaint : *Algorithmique rapide pour les problèmes de tournées et d'ordonnancement*. Thèse de doctorat, Université Blaise Pascal - Clermont-Ferrand II, France, 2010.

[95] C.P.M. van Hoesel et A.P.M. Wagelmans : Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. Research Memoranda 014, Maastricht : METEOR, Maastricht Research School of Economics of Technology and Organization, 1997.

[96] A. Wagelmans, S. van Hoesel et A. Kolen : Economic lot sizing: An o(n log n) algorithm that runs in linear time in the wagner-whitin case. *Operations Research*, 40(1):145–156, 2006.

[97] H.M. Wagner et T.M. Whitin : Dynamic Version of the Economic Lot Size Model. *MSCIAM*, 5:89–96, 1958.

[98] L.A. Wolsey : Progress with single-item lot-sizing. *European Journal of Operational Research*, 86(3):395–401, November 1995.

[99] L.A. Wolsey : *Integer Programming*. Wiley-Interscience, 1998.

[100] L.A. Wolsey : Lot-sizing with production and delivery time windows. *Mathematical Programming*, 107(3):471–489, 2006.

[101] P. Zipkin : Computing optimal lot sizes in the economic lot scheduling problem. *Operatios Research*, 39(1):56–63, January-February 1991.