UNIVERSITÉ DE MONTRÉAL

OPTIMISATION DES MOUVEMENTS DES CONTENEURS DANS UN TERMINAL MARITIME

KHAOULA CHEBLI

DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÉS SCIENCES APPLIQUÉES

(GÉNIE INDUSTRIEL)

DÉCEMBRE 2011

© Chebli Khaoula, 2011.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

OPTIMISATION DES MOUVEMENTS DES CONTENEURS DANS UN TERMINAL MARITIME

présenté par : CHEBLI Khaoula

en vue de l'obtention du diplôme de : Maîtrise és sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. GAMACHE Michel, ing., Ph. D, président

M. LANGEVIN André, Ph.D., membre et directeur de recherche

Mme CHEN Lu, Ph.D., membre et codirectrice de recherche

M. FRAYET Jean-Marc, ing., Ph. D., membre

DÉDICACE

Je dédie ce mémoire

À mes parents

À mes frères

À toute ma famille

À Aymen

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma profonde reconnaissance à mon Directeur de recherche, M. André Langevin pour sa patience, ses idées pertinentes qui ont permis l'avancement de mon mémoire. Je tiens particulièrement à le remercier pour ses qualités humaines.

Mes remerciements s'adressent également à Mme Lu Chen pour ses conseils et ses remarques fort utiles qui ont permis de bien mener ce projet.

Je tiens également à adresser mes remerciements les plus profonds à M. Michel Gamache et M. Jean-Marc Frayet qui ont accepté d'examiner et de juger ce travail.

Je remercie également la Mission Universitaire de Tunisie en Amérique du Nord (MUTAN) pour son soutien financier.

Finalement, j'adresserai mon dernier remerciement, à mes parents qui m'ont soutenue durant toutes mes années d'études et qui m'ont toujours encouragée à aller plus loin. Je vous aime.

RÉSUMÉ

De nombreuses recherches ont montré l'importance et la valeur des problèmes de planification et d'optimisation dans un terminal maritime. Dans ce mémoire, on s'intéresse au problème d'optimisation des mouvements des conteneurs dans le cas d'exportation. Les séquences de fonctionnement des portiques de cour et des camions sont prises en considération en même temps. En outre, on prend en compte les interférences qui peuvent exister entre les portiques de cour. En fouillant dans les travaux de littérature sur les problèmes de planification des portiques de cour, on ne trouve pas un travail qui examine les mouvements non productifs et les interférences possibles entre ce type de portique simultanément, ce qui sera un point d'innovation dans notre travail.

Le problème de planification des opérations de chargement des conteneurs est d'abord formulé en programme linéaire mixte. La fonction objectif minimise le temps de complétion des opérations de manutention par les portiques de cour. Le modèle mathématique est basé sur plusieurs hypothèses, tenant compte des deux phénomènes d'interférence et des mouvements non productifs.

Pour résoudre le problème, une approche heuristique de type Recherche Adaptative à Large Voisinage (*ALNS*) est développée. Cette méthode a la capacité de résoudre les problèmes d'optimisation dans un terminal à conteneurs. En effet, la méthode *ALNS* est jugée efficace quelque soit la taille du problème : 10, 20 et 100 conteneurs. Les données utilisées pour tester l'approche sont fictives et on a généré plusieurs instances en variant le nombre de conteneurs et/ou le nombre d'équipements de manutention disponibles. Les tests ont permis d'évaluer l'efficacité de l'algorithme *ALNS*. Plusieurs scénarios ont été utilisés où on a combiné des heuristiques de retrait et d'insertion. Les résultats des tests nous ont montré la qualité des solutions générées par la méthode *ALNS*.

Mots clés : terminal à conteneurs, optimisation, planification des opérations, recherche adaptative à large voisinage, programme linéaire mixte.

ABSTRACT

Most of the researchers have shown the importance and the value of scheduling and optimization problems in a maritime terminal. In this work, we focus on the optimization problem for loading operations of outbound containers. Thus, the sequencing of each yard crane and of each yard truck is studied at the same time. Furthermore, we consider the possibility of potential interferences between yard cranes and rehandles which significantly influence the performance of yard cranes. In the literature about yard crane scheduling problems, there is no work that combines interference between yard cranes and rehandles at the same time which is the innovation point of our work.

The scheduling problem for loading operations is formulated as a mixed linear program model. The objective function is to minimize the makespan of loading operations by yard cranes. The mathematical model is based on various assumptions and it includes the potential interferences and the rehandle.

A heuristic method is developed for solving this problem, namely *Adaptive Large Neighborhood Search* (*ALNS*). This method has the potential to handle with optimization problems in a container terminal. In fact, the ALNS method is deemed to be efficient with different scale problems: 10, 20 and 100 containers. The data are fictitious and many instances are built by varying the number of containers or/and the number of equipments (yard crane and yard truck) in each time. Computational tests are made to evaluate the efficiency of the developed algorithm (*ALNS*) for that we used multiple strategies where we made different combinations of removal and insertion heuristics. These numerical results show the quality of solutions produced by *ALNS* method.

Keywords: container terminal, optimization, scheduling operations, adaptive large neighborhood search, mixed linear program.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE DES MATIÈRES vii

LISTE DES TABLEAUX x

LISTE DES FIGURES xii

LISTE DES SIGLES ET ABRÉVIATIONS xiii

LISTE DES ANNEXES xiv

CHAPITRE 1 INTRODUCTION GÉNÉRALE 1

CHAPITRE 2 ÉTUDE BIBLIOGRAPHIQUE 8

2.1 Introduction 8

2.2 Les travaux liés à la planification des équipements de manutention portuaire 8

2.3 Le problème d'interférence et des mouvements non productifs 27

2.3.1 Problème d'interférence entre les portiques de cour 27

2.3.2 Problème des mouvements non productifs « *Rehandle* » 29

2.4 Conclusion 29

CHAPITRE 3 FORMULATION MATHÉMATIQUE 31

3.1 Introduction 31

3.2 Les hypothèses du modèle 31

3.3 Le modèle mathématique 32

3.3.1 Les paramètres du modèle 32

3.3.2 Les variables de décision du modèle 33

3.3.3 Le modèle mathématique 35

3.4 Exemple 38

3.5 Solution 45

3.6 Conclusion 48

CHAPITRE 4 MÉTHODE DE RÉSOLUTION 49

4.1 Présentation de la méthode 49

4.1.1 Recherche à large voisinage 49

4.1.2 Recherche Adaptative à Large Voisinage 50

4.1.3 La méthode ALNS développée par Pisinger et Ropke (2007) 52

4.1.4 Modifications de Laporte et al. (2010) 58

4.2 La méthode ALNS appliquée au problème d'optimisation des mouvements des conteneurs : 62

4.2.1 Conception de la méthode ALNS 62

4.2.2 Pseudo-code de l'algorithme 66

4.2.3 La solution initiale 67

4.2.4 Mécanismes de retrait et d'insertion 67

4.2.5 Critères d'acceptation et d'arrêt 69

4.2.6 Ajustement des poids et des scores 69

4.3 Tests et résultats 71

4.3.1 Présentations des instances tests 71

4.3.2 Comparaison des scénarios 73

4.3.3 Application de la méthode de ALNS pour les instances de grande taille (INS9 : cas de 100 conteneurs) 82

CHAPITRE 5 CONCLUSION GÉNÉRALE 87

BIBLIOGRAPHIE 89

ANNEXES 95

LISTE DES TABLEAUX

Tableau 2-1 : Résumé des travaux de recherche pertinents traitant la planification des portiques de cour 12

Tableau 2-2 : Résumé des travaux de recherche pertinents traitant la planification des portiques de cour (Suite) 13

Tableau 2-3 : Résumé des travaux de recherche pertinents traitant la planification des véhicules de transport 17

Tableau 2-4 : Résumé des travaux de recherche pertinents traitant la planification simultanée des portiques de cour et des véhicules de transport 25

Tableau 2-5 : Résumé des travaux de recherche pertinents traitant la planification simultanée des portiques de cour et des véhicules de transport (Suite) 26

Tableau 3-1 : Destination des conteneurs sur le quai 41

Tableau 3-2 : Données d'ordre général 42

Tableau 3-3 : Le temps de manutention des portiques de cour 42

Tableau 3-4 : Le temps de transport vers la destination sur le quai des conteneurs t_i pour chaque camion 43

Tableau 3-5 : Le temps de retour à vide des camions d'un portique de quai à une baie dans la cour de stockage 44

Tableau 4-1 : Caractéristiques des instances générées 72

Tableau 4-2 : Comparaison entre les différentes méthodes de calcul étudiée pour les instances INS1 et INS2 73

Tableau 4-3 : Qualité de la solution et le temps d'exécution pour l'instance INS2 76

Tableau 4-4 : Qualité de la solution et le temps d'exécution pour l'instance INS4 78

Tableau 4-5 : Qualité de la solution et le temps d'exécution pour l'instance INS6 79

Tableau 4-6 : Qualité de la solution et le temps d'exécution pour l'instance INS7 80

Tableau 4-7 : Comparaison des instances avec le scénario 12 (ALNS) 81

Tableau 4-8 : Qualité des solutions par rapport au nombre de conteneurs retirés pour les scénarios A, B et ALNS 85

LISTE DES FIGURES

Figure 1-1 : Les différentes constituantes d'un bloc (Zhang et al., 2002) 2

Figure 1-2 : Vue schématique d'un terminal (Vis et Koster, 2003) 3

Figure 1-3 : Les moyens de stockage des conteneurs dans un terminal (Steenken et al., 2004) 4

Figure 1-4 : Les moyens de transport dans un terminal (Steenken et al., 2004) 5

Figure 1-5 : Le cycle de processus de déchargement des conteneurs, Lee et al. (2009a) 5

Figure 2-1 : La collision causée par les portiques de cour, Liang et al. (2011) 28

Figure 3-1 : Les localisations des conteneurs dans l'aire de stockage 40

Figure 3-2 : Présentation de la solution avec AMPL 46

Figure 3-3 : Les séquences de fonctionnement des portiques de cour 47

Figure 3-4 : Les séquences de fonctionnement des camions 47

Figure 3-5 : Les séquences de fonctionnement des différents équipements de manutention 48

Figure 4-1 : L'algorithme général d'ALNS (Pisinger and Ropke, 2007) 53

Figure 4-2 : Algorithme ALNS, Laporte et al. (2010) 60

Figure 4-3 : Exemple de destruction et réparation dans la méthode ALNS 63

Figure 4-4 : Conception de la méthode heuristique proposée 65

Figure 4-5 : Variation de la qualité de la solution en fonction du nombre des conteneurs retirés pour les scénarios A, B et l'algorithme ALNS 86

LISTE DES SIGLES ET ABRÉVIATIONS

ALNS	Adaptive Large Neighborhood Search
LNS	Large Neighborhood Search
QC	Quay Crane
YC	Yard Crane
YT	Yard Truck

LISTE DES ANNEXES

ANNEXE 1 : Programmation avec AMPL……95

ANNEXE 2 : Qualité de la solution pour les instances : INS1, INS3, INS5 et INS8…….....104

CHAPITRE 1 INTRODUCTION GÉNÉRALE

Avec l'évolution du phénomène de conteneurisation, les ports maritimes ont connu de grands développements des techniques de manutention. Un terminal maritime à conteneurs se décompose en deux grandes zones, chacune étant caractérisée par ses propres opérations de manutention et ses équipements. En effet, dans la partie quai, les bateaux sont chargés/déchargés par des portiques de quai. Tandis que dans la partie terrestre, appelée encore la cour, cette zone possède comme équipements les portiques de cour. Un autre équipement, qui est le véhicule de transport, assure la liaison entre ces deux zones.

En pratique, afin d'améliorer l'efficacité des opérations portuaires, il est primordial d'identifier et de résoudre une série de problèmes d'optimisation comme : la planification des déplacements des camions de transport, la planification des opérations des portiques de cour, la planification des opérations des portiques de quai et l'allocation des conteneurs dans les zones de stockage.

Toutefois, dans un terminal à conteneurs, ces derniers sont stockés dans l'aire de stockage (cour) qui est séparée en blocs. La figure 1-1 illustre une configuration d'un bloc dans l'aire de stockage où un bloc n'est qu'une grille formée de tronçons adjacents horizontaux formant les baies et des tronçons adjacents verticaux formant les rangées. Les conteneurs sont stockés en piles composés de plusieurs niveaux appelés encore étages. La position d'un conteneur dans la cour est caractérisée par une adresse spécifique formée du bloc, baie, rangée, étage. Le nombre d'étages maximal dépend de l'équipement de manutention mis en disposition dans le terminal (*Steenken et al.*, 2004). Par ailleurs, en se référant au travail de *Chen et Langevin* (2011), un bloc est généralement composé de 6 lignes (rangées). Chaque ligne est formée de 20 baies ou plus qui peuvent atteindre 4 à 5 conteneurs de hauteur (étages).

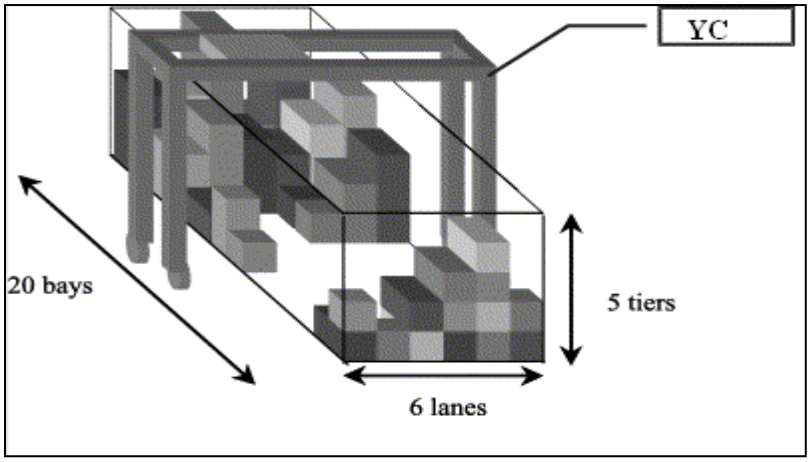

Figure 1-1 : Les différentes constituantes d'un bloc (Zhang et al., 2002)

Le processus général d'un terminal à conteneurs peut être décrit comme une séquence d'opérations à partir de l'arrivée des porte-conteneurs jusqu'au départ des conteneurs du port ou vice-versa comme le montre la figure 1-2.

Les conteneurs destinés à l'exportation arrivent au port par camion, sont répartis entre les blocs et stockés dans une aire de stockage. Après une certaine période de temps, les conteneurs sont retirés des blocs avec les portiques de cour *(Yard Cranes)* et sont transportés par les véhicules *(Yard Truck)* vers les quais où ils sont prélevés par des portiques de quai *(Quai Cranes)* et chargés sur les navires. Dans la situation d'importation, quand un navire arrive au terminal, les conteneurs importés doivent être déchargés par les portiques de quai. Ensuite, ils sont placés sur des véhicules qui vont les amener jusqu'aux aires de stockage. Après un certain temps, les conteneurs quittent les aires de stockage et ils seront transportés par véhicules ou d'autres types de transport terrestre (trains, camions).

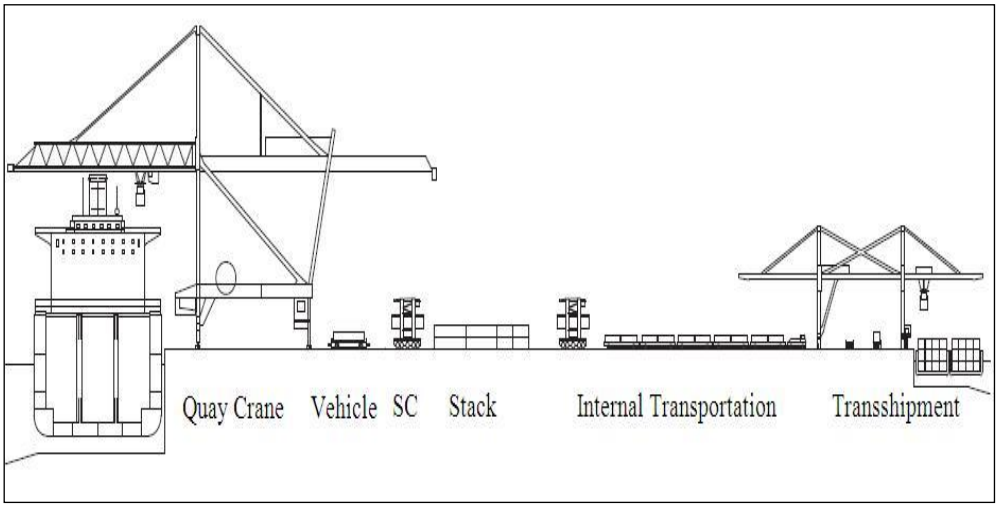

Figure 1-2 : Vue schématique d'un terminal (Vis et Koster, 2003)

Pour les équipements de manutention, la sélection de ces équipements est très importante et elle influence beaucoup la performance d'un terminal à conteneurs. Pour cela, plusieurs auteurs ont mis l'accent sur l'impact des équipements de manutention sur la compétitivité des terminaux.

- Les équipements de manutention des conteneurs (les portiques) : On distingue généralement deux types de portiques (*Steenken et al.,*2004) à savoir les portiques de cour (*Stacking crane*/*Yard Crane*) et les portiques de quai (*Quay crane*) (figure 1-3). Le deuxième type de portique s'occupe essentiellement du chargement et du déchargement des porte-conteneurs. Les deux grandes catégories des portiques de quai sont : « *Single-trolley* » et « *Dual-trolley* », la première est conduite par l'homme « man driven » alors que la deuxième est automatique. La performance des portiques de quai dépend largement de la catégorie (*Steenken et al.,* 2004). Le deuxième type de portiques est les portiques de cour dont la mission est le chargement (déchargement) des conteneurs dans les blocs ainsi que sur les camions. Les types des portiques de cours sont multiples, on trouve par exemple les RMG (*Rail Mounted Gantry*), RTG (*Rubber Tired Gantries*) et OBC (*Overhead Bridge Cranes*).

1- Portique de cour 2- Portique de quai

Figure 1-3 : Les moyens de stockage des conteneurs dans un terminal (Steenken et al., 2004)

- <u>Les équipements de transport des conteneurs</u> : Un autre équipement indispensable dans un port maritime est celui des camions. À coté des camions simples et ordinaires, on trouve plusieurs autres formes de véhicules de transport à savoir les *AGV* (*Automated Guided Vehicules*) et les *SC* (*Straddle carriers*) qui sont présentés dans la figure 1-4. La fonction principale de ces camions est la liaison entre l'aire de stockage et les quais par le transport des conteneurs destinés à l'exportation ou à l'importation. Les *AGV* demandent des investissements importants et ont la capacité de charger un conteneur de 40' ou bien deux conteneurs de 20' (*Steenken et al., 2004*). Pour les *SC*, la principale caractéristique est leur capacité de manutentionner les conteneurs dans la cour, en plus de leur fonction de base qui est le transport des conteneurs à partir de la zone de stockage vers les quais ou vice-versa. On peut noter parmi les propriétés de ces moyens de transport, qu'ils sont très flexibles et dynamiques et qu'ils sont conduits par l'homme « man driven » contrairement aux *AGV*. Les *SC* sont capables de lever les conteneurs pour les stocker et c'est pour cela qu'ils sont aussi appelés *ALV* (*Automated Lifting Vehicules*).

1- AGV 2- ALV

Figure 1-4 : Les moyens de transport dans un terminal (Steenken et al., 2004)

Lee et al. (2009a) ont fait une description du cycle de déchargement des conteneurs importés par les camions à partir du quai vers la cour (Figure 1-5). Le processus de chargement (déchargement) d'un conteneur de sa localisation dans la zone de stockage vers le navire est référé comme une tâche. Le temps nécessaire pour terminer toutes les tâches est appelé le temps de complétion (makespan).

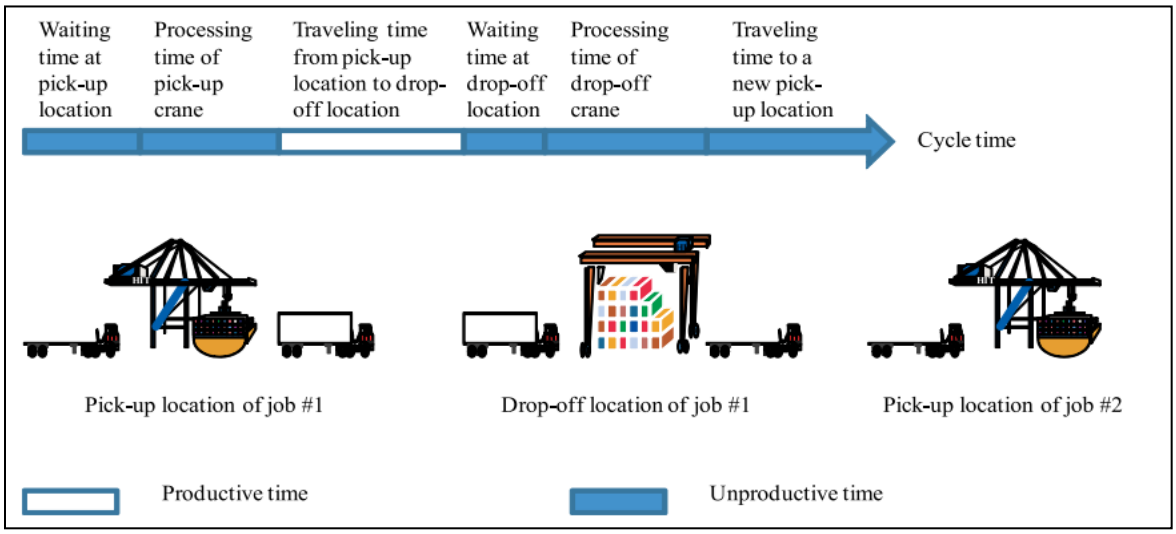

Figure 1-5 : Le cycle de processus de déchargement des conteneurs, Lee et al. (2009a)

Le temps de traitement d'une tâche peut être divisé en 6 étapes, dans une situation d'importation (*Lee et al.*,2009a):

- Temps d'attente au point de collecte
- Temps de traitement au point de collecte
- Temps de transport du point de collecte au point de dépose
- Temps d'attente au bloc de la cour pour être servi
- Temps de traitement au point de dépose
- Temps de retour à vide du point de dépose au point de ramassage d'une nouvelle tâche.

Par analogie, les 6 étapes formant le temps total de traitement d'une tâche dans le cas d'export sont définies ci-dessous:

- Temps d'attente au point de collecte
- Temps de traitement au point de collecte
- Temps de transport du point de collecte vers le point de dépose
- Temps d'attente sur le quai
- Temps de traitement au point de dépose
- Temps de retour à vide du point de dépose au point de ramassage d'une nouvelle tâche

Dans ce travail on va se limiter à la planification des opérations de deux équipements portuaires ; à savoir les portiques de cour et les véhicules de transport tout en essayant de savoir comment les auteurs ont analysé les opérations relatives à ces équipements et sur quoi ils se sont basés pour résoudre les problèmes qui en découlent.

L'objectif général de ce projet est de planifier les opérations de manutention des conteneurs par les portiques de cour. Le présent travail s'intéresse aussi à la synchronisation des opérations de manutention simultanées par les portiques de cour et les camions tenant compte de la possibilité d'interférence entre les portiques de cour et des mouvements non productifs. Ce problème est formulé en programme linéaire mixte puis il est résolu à l'aide d'une méthode heuristique (*Recherche Adaptative à Large Voisinage*) dans le but de minimiser le temps de complétion des

opérations de manutention des conteneurs destinés à l'exportation. Finalement, on effectue des tests sur des différentes instances afin de valider cette méthode.

Pour mener à bien ce travail, nous avons jugé utile de diviser ce mémoire en quatre chapitres.

Ce chapitre (premier chapitre) est une présentation des opérations portuaires et description des équipements de manutention.

Le deuxième chapitre est consacré à une synthèse bibliographique générale des travaux théoriques et expérimentaux sur la planification des équipements de manutention dans un port maritime ainsi que sur les problèmes d'interférence et de remanutention des conteneurs.

Le troisième chapitre, quant à lui, présente une formulation mathématique du problème de planification des portiques de cour et des camions simultanée. De plus, il donne une présentation des hypothèses prises en considération lors de la modélisation. Enfin, une modélisation de ce problème est faite avec le logiciel AMPL Studio.

Dans le quatrième chapitre, la méthode de résolution heuristique de type recherche adaptative à large voisinage « *Adaptative Large Neighborhood Search (ALNS)* » sera abordée. On a généré différentes instances pour effectuer quelques tests afin de valider la méthode utilisée et des discussions qui consolident et complètent certains points tirés de la recherche bibliographique sur le problème d'optimisation des opérations portuaires.

Pour conclure, un récapitulatif des principaux points et contributions apportés par ce travail de maîtrise sera présenté ainsi que de suggestions ouvrant sur de futurs projets de recherches dans le cinquième chapitre.

CHAPITRE 2 ÉTUDE BIBLIOGRAPHIQUE

2.1 Introduction

Dans ce chapitre, nous passons en revue des travaux de recherche pertinents à notre travail. Nous commençons en première partie par une présentation des travaux liés à la planification des équipements de manutention portuaire. Dans ce contexte, nous examinons d'abord les travaux liés à la planification des mouvements des portiques de cour dans les terminaux à conteneurs. Ensuite, nous envisageons mettre l'accent sur les travaux traitant le problème de planification des tâches des camions de transport entre la cour et le quai. Enfin, nous donnons une récapitulation des travaux analysant la planification simultanée des mouvements des portiques de cour et les déplacements des véhicules surtout lorsqu'il s'agit d'une opération d'exportation (*outbound*).

La seconde section sera consacrée aux problèmes relatifs à l'interférence et aux mouvements non productifs des portiques de cour appelés « *rehandle* ».

2.2 Les travaux liés à la planification des équipements de manutention portuaire

2.2.1 La planification des portiques de cour dans un terminal maritime

Les portiques de cour effectuent un certain nombre d'opérations : soulever et déposer les conteneurs sur les véhicules et recevoir les conteneurs à exporter pour leur empilement dans les blocs de cour (*Chen et al., 2007*). D'après (*Li et Tang, 2009*), les opérations des portiques de cour consistent essentiellement dans le levage et le déplacement des conteneurs, soit à l'exportation, soit à l'importation, à partir de la cour vers les camions ou vice-versa.

Le travail de *Kim & Kim* (1997) aborde la planification des séquences de chargement des conteneurs à exporter dans un terminal maritime ; pour cela, ils ont suggéré un algorithme optimal de routage « *Optimal Routing algorithm* ». L'objectif est de minimiser le temps total de la manutention des conteneurs par les portiques de cour. Les contributions de ce travail sont : la détermination du nombre optimal de conteneurs à ramasser dans chaque baie ainsi que la route optimale qu'un portique de cour va parcourir.

Une extension du travail de *Kim & Kim* (1997) a conduit à un nouveau travail *Kim & Kim* (1999). Ces auteurs ont résolu le problème de planification des opérations d'un seul portique de cour lors du chargement des conteneurs d'exportation, tout en proposant un modèle de programmation linéaire mixte pour déterminer le nombre de conteneurs à charger et l'ordre de visite des baies de la cour par les portiques. Ce modèle est basé sur quatre hypothèses suivantes :

1- Un seul portique est considéré.
2- Tous les conteneurs à charger sont localisés dans un seul bloc ou bien dans des blocs adjacents.
3- Le programme de travail et l'implantation de la cour sont déjà donnés.
4- Seulement les conteneurs d'un même groupe sont localisés et empilés dans une baie de cour ; cette méthode permet l'allocation de l'espace aux conteneurs.

Le travail de *Zhang et al.* (2002) aborde l'efficacité des opérations dans la cour. Ces opérations, selon les auteurs, dépendent largement de la productivité des équipements de manutention et spécifiquement les portiques de cour de type RTGC (*Rubber tyred gantry cranes*). L'objectif à atteindre dans ce travail est de déterminer le temps et l'itinéraire des mouvements des portiques dans le but de minimiser les retards dans la cour. Le problème est formulé comme un programme linéaire mixte et il est résolu à l'aide de la relaxation lagrangienne. Les auteurs abordent le problème de déploiement des portiques de cour. Le modèle développé se base sur plusieurs hypothèses observées de la pratique, on cite à titre d'exemple :

1- Tous les portiques RTGC ont la même capacité. Cette capacité est mesurée en minutes.
2- Limitation du nombre de portiques par bloc (maximum deux portiques dans un bloc) et cela vue la taille limitée des blocs et les dangers de collision potentiels entre les portiques.
3- Chaque RTGC peut être relocalisé au plus une seule fois dans chaque période de planification (4 heures).

En outre, *Linn et Zhang* (2003) ont étudié le problème de déploiement dynamique des portiques de cour afin de trouver la fréquence de chargement/déchargement optimale des portiques minimisant le surplus de charge de travail. Ce problème est modélisé sous la forme d'un programme linéaire mixte et un algorithme heuristique est développé pour le résoudre.

De son côté, *Ng* (2005) a développé un modèle de programmation linéaire mixte (MIP) pour résoudre le problème de planification des portiques de cour en prenant en compte l'interférence entre les portiques et il a proposé un algorithme heuristique pour résoudre ce type de problème. Dans le même ordre d'idées, *Ng et Mak* (2005) ont étudié le problème de planification des portiques de cour afin de minimiser la somme des temps d'attente. Le modèle mathématique est sous la forme d'un programme linéaire en nombres entier et pour le résoudre; *Ng* et *Mak* ont utilisé l'algorithme de séparation et d'évaluation progressive (*Branch & Bound*).

Lee et al (2007) ont traité le problème de planification de deux portiques de cour utilisés pour le chargement des conteneurs d'exportation sur les camions. Cette planification a permis de déterminer simultanément l'ordre de visite des baies de conteneurs ainsi que le nombre de conteneurs à enlever à chaque visite par les deux portiques. Le modèle mathématique est basé sur certaines hypothèses. *Lee et al.* ont utilisé un algorithme de recuit simulé (*Simulated Annealing*) pour le résoudre.

Guo et al. (2008) ont mis l'accent sur le problème de gestion des enchaînements des portiques de cour en temps réel. L'intégration des données en temps réel dans le système de gestion des portiques de cour permet une meilleure utilisation des ressources du port maritime et par conséquent d'améliorer la productivité globale de ce port. La définition de ce problème se base sur deux hypothèses qui sont :

1- Chaque véhicule transporte un seul conteneur.
2- On a des informations précises et en temps réel sur l'arrivée des véhicules et sur l'emplacement des stockages.

Afin de résoudre ce problème, les auteurs ont proposé un modèle de simulation.

Zhang et Jiang (2008) considèrent qu'avec une planification des portiques non efficace, les camions vont attendre dans la cour pour le chargement des conteneurs et par la suite les portiques de quai seront inactifs en attendant les camions, ce qui va ralentir le fonctionnement de l'enchaînement des opérations portuaires. Ce qui leur permet de dire que l'étude des problèmes de planification des équipements de manutention (portiques de cour, véhicules de transport, portiques de quai) améliore la productivité des terminaux à conteneurs.

D'après *Li et al.* (2009), une planification efficace pour la réduction du temps d'attente causé par les portiques de cour est essentielle dans l'augmentation du débit d'un terminal à conteneurs. C'est pour cela qu'ils ont développé un modèle de planification d'un seul portique de cour ou bien de plusieurs tout en tenant compte de certaines hypothèses lors de la formulation :

1- Le temps d'une tâche de manutention pour tous les portiques de cour est de 3 minutes.
2- 20-30 mouvements en 2 heures (fenêtre de temps) sont utilisés dans les scénarios testés.
3- chaque bloc de la cour contient 40-60 emplacements de conteneurs.

Pour trouver une solution à ce problème, les auteurs ont proposé un algorithme heuristique et un algorithme appelé « *rolling-horizon* ».

Nishimura et al. (2009) se sont focalisé sur l'arrangement de l'opération de stockage des conteneurs de transbordement dans la cour afin d'assurer l'efficacité des opérations de manutention des navires dans un terminal avec de très grands porte-conteneurs. L'objectif de ce travail est d'aborder le problème de planification de stockage de conteneurs de transbordement qui sont traités dans un très grand port. Un modèle d'optimisation a été proposé par les auteurs pour une étude des flux des conteneurs des navires. Une heuristique basée sur la relaxation lagrangienne est formulée. Afin d'évaluer la qualité de cette approche heuristique des tests expérimentaux ont été faits.

À la fin de cette section, nous passons à un résumé des travaux de recherche les plus pertinents décrivant le problème de planification des portiques de cour (Tableaux 2-1 et 2-2).

Tableau 2-1 : Résumé des travaux de recherche pertinents traitant la planification des portiques de cour

Auteurs	**Problème traité & ses caractéristiques**	**Méthode(s) de résolution**	**Contributions clés**
Kim et Kim (1997)	- Planification des séquences du chargement des conteneurs à exporter dans un port maritime - Minimisation du temps total de manutention des conteneurs par les portiques de cour	Algorithme optimal de routage	- Détermination du nombre optimal des conteneurs à collecter dans chaque baie et de la route optimale des portiques de cour
Kim et Kim (1999)	Planification des opérations d'un seul portique de cour pour le chargement des conteneurs d'exportation.	Programme linéaire mixte	- Détermination du nombre de conteneurs à charger et de l'ordre de visites des baies par les portiques de cour
Zhang et al. (2002)	- Déploiement des portiques de cour (RTGC) - Efficacité des opérations de manutention dans la cour	- Programme linéaire mixte - Relaxation Lagrangienne	- Détermination du temps et itinéraires des mouvements des portiques - Minimisation des retards dans la cour
Linn et Zhang (2003)	Déploiement dynamique des portiques de cour	- Programme linéaire mixte - Algorithme Heuristique	Déterminer la fréquence de chargement/déchargement optimale des portiques de cour pour minimiser le surplus de charge de travail
Ng et Mak (2005)	Planification des portiques de cour pour la manutention d'un ensemble d'opérations de chargement/déchargement des conteneurs	Algorithme de Branch & Bound	Minimiser la somme des temps d'attente

Tableau 2-2 : Résumé des travaux de recherche pertinents traitant la planification des portiques de cour (Suite)

Auteurs	**Problème traité & ses caractéristiques**	**Méthode(s) de résolution**	**Contributions clés**
Lee et al. (2007)	Planification de deux portiques de cour pour le chargement/déchargement des conteneurs d'exportation sur les camions	Algorithme de recuit simulé (SA)	- Détermination de l'ordre de visite des baies et du nombre de conteneurs à ramasser à chaque visite pour les deux portiques de cour
Guo et al. (2008)	Gestion des enchaînements des portiques de cour en temps réel	Modèle de simulation	-Meilleure utilisation des ressources et amélioration de la productivité globale du port
Nishimura et al. (2009)	- Planification de stockage de conteneurs de transbordement. - Minimisation du temps total de manutention des conteneurs à partir des navires vers les blocs, et de la cour vers leurs localisations sur le quai plus le temps d'attente pour le chargement sur le navire.	Relaxation lagrangienne	Amélioration de l'efficacité des opérations de manutention des navires
Li et al. (2009)	Planification d'un seul ou de plusieurs portiques de cour	Algorithme « *Rolling-horizon* »	- Réduction du temps d'attente causé par les portiques - Augmentation du débit du terminal

2.2.2 La planification des véhicules de transport dans un terminal maritime

Les véhicules les plus couramment utilisés dans les ports maritimes sont les simples camions. Avec le développement technologique, les moyens de transport sont devenus de plus en plus automatisés tels que les AGV (*Automated Guided Vehicle*) et les ALV (*Automated Lifting Vehicle*). Ces véhicules et les camions de transport de façon générale ont pour mission le transport des conteneurs à partir de l'aire de stockage, après leur chargement par les portiques de cour, vers le quai où ils seront déchargés par les portiques de quai. C'est le cheminement lorsqu'il s'agit une opération d'exportation. De même, les camions transportent les conteneurs d'importation à partir du quai vers les aires de stockage. Pour certains types de moyens de transport, tels que les ALV, la fonctionnalité principale dépasse la mission de transport vers le stockage des conteneurs (rappelons que les ALV jouent un double rôle : le transport et la manutention des conteneurs).

Evers et Koppers (1996) ont mis l'accent sur le contrôle du trafic des véhicules AGV dans un terminal à conteneurs. Ce contrôle est une activité cruciale dans la performance du système. Le scénario proposé par les auteurs, pour le contrôle de la circulation des AGV, consiste en une technique de modélisation et de configuration du terminal et il est réalisé par un modèle de simulation. Ce scénario permet de diminuer la quantité de communication à condition que ces véhicules envoient des rapports de position à proximité des zones contrôlées. Le scénario mis en place pour le contrôle du trafic augmente les performances du système d'information utilisé pour contrôler les AGV.

Bish (2003) a traité plusieurs problèmes en même temps. L'objectif principal de ce travail est d'analyser l'efficacité des politiques communes des véhicules permettant le partage des véhicules entre les différents navires. *Bish* a mis en lumière deux niveaux de problème. Le premier est relatif à l'assignation des localisations de stockage pour les conteneurs, la répartition des conteneurs sur les véhicules et l'ordonnancement des opérations de chargement et déchargement des portiques de quai. Le deuxième problème est consacré à la localisation des portiques de cour dans l'aire de stockage et la détermination des séquences des localisations desservies par chaque portique de cour. L'auteur a développé une heuristique simple appelée heuristique *TLS* (*transshipment problem based list scheduling heuristic*) qui fournit des solutions pour des

problèmes de grande taille. Les études expérimentales ont montré que cette heuristique est assez efficace dans un cadre réel.

Une comparaison entre deux moyens de transport, ALV (*Automated Lifting Vehicles*) et AGV (*Automated Guided Vehicles*), est faite dans le travail de *Vis et Harika* (2004). Selon ces deux auteurs, le choix d'un certain type d'équipement doit être fait en effectuant une étude de faisabilité et une analyse économique sur les différents types d'équipements. L'objectif de ce travail est de discuter l'effet de l'utilisation de chaque moyen de transport sur le temps de déchargement d'un navire à l'aide d'une étude de simulation. Le choix d'un certain type d'équipement se base sur des critères tels que le temps de déchargement d'un navire, le degré d'occupation ou le nombre de véhicules nécessaires.

Kim et Bae (2004) considèrent que les opérations de transport dans un terminal jouent un rôle important dans la synchronisation des autres opérations surtout celles assurées par les portiques. Leur étude discute comment affecter les tâches de transport des conteneurs aux AGV afin de synchroniser les opérations des équipements de manutention dans un environnement dynamique. Ce problème a été formulé comme un modèle de programmation linéaire mixte avec les hypothèses suivantes :

1- Chaque AGV sert plus d'un portique.
2- Tous les AGV sont les mêmes dans leurs fonctionnements et ils ne transportent qu'un seul conteneur à la fois.
3- Le temps entre le chargement d'un conteneur sur un AGV et le ramassage d'un autre à partir d'un AGV en attente est négligeable.
4- La congestion entre les AGV n'est pas prise en compte.

Pour la résolution de ce type de problème, les auteurs ont proposé un algorithme heuristique dont la performance a été testée avec une étude de simulation.

Bish et al (2005) ont mis l'accent sur le problème de répartition des véhicules aux conteneurs afin de minimiser le temps total nécessaire pour servir un navire. Ce temps est entre autres le temps total qu'il faut pour décharger tous les conteneurs du navire et en charger de nouveaux. L'objectif de la recherche est de trouver des politiques simples pour la répartition des véhicules de transport

afin de minimiser la valeur du temps d'achèvement de la dernière tâche pour ce problème de répartition. Les auteurs ont développé un algorithme heuristique (algorithme glouton) pour arriver à des résultats proches de l'optimum.

Ng et al. (2007) ont abordé le problème de planification de la flotte des camions de transport en vue de minimiser le temps de complétion. Le problème de planification a été formulé comme un programme linéaire mixte, il s'agit d'un problème NP-difficile et il a été résolu par l'algorithme génétique.

Dans un autre travail, *Huynh et Walton* (2008) ont utilisé un système d'affectation des camions qui entrent au terminal venant des fournisseurs pour contrôler leurs arrivées dans un terminal à conteneurs. En effet, les auteurs ont examiné les effets de la limitation de l'arrivée des camions (principe de la fenêtre de temps) sur le temps de rotation de ces véhicules venant de l'extérieur du port et sur l'exploitation des portiques. En outre, ils ont proposé une méthodologie pour aider les exploitants des terminaux à déterminer le nombre optimal de camions à accepter. La méthode qui a été proposée, qui est une combinaison d'un modèle mathématique et d'un modèle de simulation, cherche une solution qui soit bénéfique pour les exploitants du terminal maritime.

Grâce à un modèle de simulation, *Huynh* (2009) a démontré que les différentes règles de planification des équipements et surtout des véhicules influencent l'utilisation des ressources d'un terminal et le temps la rotation des camions dans les opérations terrestres. Une telle étude peut influencer les décisions des exploitants portuaires. Ce travail vise une évaluation des performances des différentes règles de planification dans plusieurs scénarios ainsi que des principaux facteurs qui agissent sur la performance des ces règles.

Les travaux de recherche précédemment cités sont présentés dans le tableau 2-3.

Tableau 2-3 : Résumé des travaux de recherche pertinents traitant la planification des véhicules de transport

Auteurs	**Problème traité & ses caractéristiques**	**Méthode(s) de résolution**	**Contributions clés**
Evers et Koppers (1996)	Contrôle du trafic des AGV dans un terminal	Modèle de simulation	- Diminution de la quantité de communication - Augmentation de la performance des systèmes d'information utilisés dans le terminal
Bish (2003)	- Planifier le déplacement du chargement de chaque conteneur sur un véhicule - Planifier les opérations de chargement et de déchargement des portiques de quai	Heuristique TLS (*transshipment problem based list scheduling heuristic*)	- Déterminer l'efficacité des politiques communes d'un ensemble de véhicules partagés entre les navires - Minimiser le makespan
Vis et Harika(2004)	Comparaison entre deux moyens de transport AGV et ALV	Modèle de simulation (Aréna 3.5)	- Effets de l'utilisation des AGV ou ALV sur le temps de déchargement des navires - Choix d'un équipement en fonction de plusieurs critères : nombre de véhicules nécessaire, temps d'attente des portiques, coût d'achat.
Kim et Bae (2004)	Affectation des tâches de transport des conteneurs aux AGV	- Modèle de simulation - Algorithme heuristique	Synchronisation des opérations des équipements de manutention dans un environnement dynamique
Bish et al (2005)	- Répartition des véhicules aux conteneurs - Minimisation du makespan	Algorithme glouton	Détermination des politiques simples pour la répartition des véhicules de transport
Ng et al. (2007)	- Planification de la flotte des camions de transport - Minimisation du makespan	Algorithme génétique	Trouver une planification efficace pour le problème à l'aide de l'heuristique
Huynh et Walton (2008)	- Affectation des camions venant de l'extérieur - Contrôle de l'arrivée des camions au terminal	Modèle de simulation	- Effets de la limitation de l'arrivée des camions sur le temps de rotation et sur l'exploitation des portiques - Détermination du nombre optimal de camions à accepter

2.2.3 La planification des portiques de quai dans un terminal maritime

La planification des portiques de quai a pour objectif la détermination d'une séquence d'opérations de chargement et/ou de déchargement afin par exemple de minimiser le temps total. La minimisation du temps de complétion, appelé communément le makespan, est très importante étant donné qu'elle définit et précise le temps de départ des porte-conteneurs (*Kim, 2008*).

Le problème de planification des portiques de quai (*Quay-Crane Scheduling Problem, QCSP*) a commencé avec le travail de *Daganzo* (1989) par une formulation de programme linéaire mixte avec un nombre fixe de portiques. Une solution exacte du problème est trouvée dans le cas d'un problème de petite taille alors que pour les problèmes de grande taille, l'auteur propose une méthode heuristique. Un autre travail de *Daganzo* (1990) a traité le *QCSP* dans un terminal polyvalent qui sert deux types de trafic ; une ligne régulière et une ligne à la demande (tramping). L'objectif est de trouver une simple formulation analytique de la productivité d'un terminal polyvalent et de calculer le taux d'utilisation des postes d'amarrage.

Kim et Park (2004) ont eu comme objectif la réduction du makespan ; le temps de complétion des portiques de quai pour un problème de planification de portiques formulé dans un modèle de programmation linéaire mixte. Pour cela, ils ont utilisé un algorithme de *Branch & Bound* et la procédure *GRASP* (*Greedy Randomized Adaptive Search Procedure*) pour résoudre le problème. L'algorithme de *Branch & Bound* est exact mais il n'est pas efficace en termes de temps de calcul par rapport au *GRASP*.

Dans le cadre du travail de *Moccia et al* (2006), les auteurs ont traité le problème de planification des portiques de quai dans le but de minimiser le temps de complétion ainsi que le temps d'inactivité, comme la séquence d'opérations de chargement ou de déchargement des conteneurs par les portiques de quai sur les navires assignés. Le temps d'inactivité peut être issu de l'interférence entre ces portiques. Ils ont proposé une formulation du problème de planification des portiques de quai comme un problème de tournées de véhicules (VRP). Pour cela, ils ont choisi de résoudre les problèmes de petite taille avec CPLEX (*Branch & Bound*) et avec un algorithme de *Branch & Cut* pour les problèmes de grande taille.

Sammarra et al. (2007) ont traité le même problème de planification inspiré du travail de *Moccia et al.* (2006) comme un problème VRP. Ce problème est décomposé en deux parties : problème de routage et problème d'ordonnancement, ce dernier étant un problème NP-difficile. Les auteurs

ont développé un algorithme de recherche Tabou pour le problème de routage et une technique de recherche locale pour le problème de planification.

Dans le même axe, *Lee et al.* (2008) ont eux aussi étudié le problème de planification des portiques de quai par la détermination d'une séquence de traitement des conteneurs par ces équipements. La spécificité de cette étude est de tenir compte de l'interférence qui peut se produire entre les portiques de quai. Pour résoudre le problème, les auteurs ont proposé un modèle de programmation linéaire mixte. De plus un algorithme génétique pour obtenir des solutions proches de l'optimum a été fourni.

2.2.4 La planification simultanée des portiques de cour et des véhicules de transport

Dans cette section, on va prendre en considération les travaux qui examinent les problèmes de planification de deux équipements de manutention en même temps et plus précisément les portiques de cour et les véhicules de transport dans un terminal à conteneurs.

Dans une étude regroupant les mouvements des portiques de cour et les déplacements des camions de transport, *Huynh et al.* (2004) ont élaboré un modèle de simulation permettant l'analyse du temps de rotation des camions tout en tenant compte de la disponibilité des portiques de cour. La réduction du temps de rotation permet la diminution des coûts de transport terrestre dans le port. Une telle étude permet de trouver le nombre optimal de portiques de cour pour atteindre le temps de rotation voulu des camions et la quantité d'espace de stockage optimale. Pour simplifier la modélisation des mouvements des camions et des portiques de cour, les auteurs ont introduit certaines hypothèses :

1- Tous les conteneurs sont de même longueur (40 pieds).
2- Tous les blocs sont de même longueur (800 pieds).
3- Il y a 20 baies le long du bloc pour le chargement et le déchargement.
4- Il n'y a pas de collisions, ni entre camions ni entre les camions et les portiques.
5- Les camions transportent un seul conteneur à la fois.

Afin d'obtenir une solution à ce modèle, les auteurs ont utilisé l'algorithme de recherche Tabou.

Les camions et les portiques de cour sont les équipements les plus utilisés pour la manutention des conteneurs dans un terminal. Dans ce contexte, *Ng* et *Ge* (2006) ont étudié l'interaction entre

les opérations de ces deux équipements, dans leur travail, ils ont développé une méthode pour une planification efficace des tâches des camions et des portiques de cour dans le but d'améliorer les opérations de chargement et de déchargement des conteneurs sur le quai, maximiser le taux de manutention et minimiser les retards des navires. Pour résoudre ce type de problème de planification, *Ng* et *Ge* ont proposé un algorithme heuristique appelé heuristique *floue*. Un modèle de simulation a été mis en place afin d'évaluer l'efficacité de l'algorithme heuristique, les auteurs ont démontré que l'algorithme *flou* est très performant par rapport à un ensemble de règles de planification.

Chen et al. (2007), dans leur travail, ont essayé d'améliorer la coordination entre les différents équipements de manutention et par conséquent augmenter la productivité d'un terminal maritime. Dans le but de minimiser le temps de complétion, les auteurs ont développé un modèle mathématique sous la forme de programme linéaire mixte pour planifier les opérations de plusieurs types d'équipements mis en place. Ce modèle se base sur un certain nombre d'hypothèses :

1- Les opérations de chargement des navires commencent dès que les opérations de déchargement sont finies.
2- Dans un bloc de cour, il ne faut pas mélanger les conteneurs à importer avec ceux à exporter. Ainsi, un portique de cour a une seule fonction : soit la collecte soit la dépose d'un conteneur.
3- Tous les camions ont la même unité de capacité, c'est à dire qu'ils ne peuvent manutentionner qu'un seul conteneur à la fois.
4- Les temps de fonctionnement du véhicule et les temps de voyage sont déterministes et ils sont les mêmes pour les voyages chargés et à vide.
5- Comme la route n'est pas considérée comme une source de contraintes, les conflits ne sont pas pris en compte.
6- Le temps de transfert du conteneur entre les portiques et les véhicules n'est pas inclus dans le temps de processus.

Pour résoudre ce problème de planification des opérations des divers équipements de manutention, on a fait appel à l'algorithme de recherche Tabou.

Lei et al. (2008) ont mis en lumière le rôle de la planification des mouvements des portiques de cour et des déplacements des camions dans l'amélioration de l'efficacité d'un terminal à conteneurs. Contrairement à la planification traditionnelle des camions de transport basée sur l'opération de file d'attente, le travail de *Lei et al.* a examiné le temps de transport des camions et le temps de fonctionnement des portiques de cour durant les opérations simultanées de chargement et de déchargement des conteneurs. Ce problème, visant la minimisation du temps de déplacement des camions, a été résolu avec l'algorithme Tabou et un autre algorithme appelé *Immune Algorithm*.

Zhang et *Jiang* (2008) ont examiné un problème plus complexe : il s'agit de la planification des équipements de manutention (portiques de cours, les camions et les portiques de quai) en même temps pour améliorer la productivité d'un terminal à conteneurs. Cette planification a pour but de déterminer les séquences d'envois et chargement/déchargement des conteneurs et aussi les séquences des conteneurs transportés par les camions afin de maximiser l'utilisation des portiques de cour et des portiques de quai. Un modèle de simulation est développé pour valider que la planification dynamique est efficace. Ce modèle est fondé sur deux hypothèses.

1- Chaque équipement (portique de cour, camion, portique de quai) peut seulement exécuter une tâche à la fois.
2- Les plans de stockage et d'arrimage sont connus à l'avance et il n'y a pas d'interférence entre les itinéraires.

S'inspirant d'autres travaux antérieurs qui étudient le problème de planification de deux opérations en même temps, *Li et Tang* (2009) donnent une description du problème intégré de la planification des mouvements des véhicules et la planification d'allocation des conteneurs dans les blocs. Le présent travail prend en compte l'affectation des conteneurs aux véhicules et leur allocation dans les blocs. La fonction objectif du modèle mis en place est de minimiser la durée totale (temps de complétion) de toute l'opération de déchargement des conteneurs. Pour résoudre le problème, les auteurs ont passé d'une solution initiale basée sur les hypothèses suivantes :

1- Le conteneur déchargé est affecté au camion qui le transporte vers la destination au plutôt c'est à dire avec un temps de voyage minimal.

2- On attribue le conteneur en question au bloc contenant le portique de cour permettant son

déchargement du camion avec un minimum de temps.

Afin d'obtenir une solution à ce problème, l'algorithme recherche Tabou a été utilisé et a permis d'obtenir des solutions très efficaces par rapport à CPLEX.

Lee et al. (2009a) ont associé deux sous problèmes séparés dans un terminal à conteneurs. Ils ont en effet proposé un modèle intégré de la planification des déplacements des camions de transport et de l'allocation de la zone de stockage à des conteneurs destinés à l'importation. Ce problème est formulé comme une programmation linéaire mixte qui considère ces deux sous-problèmes. La formulation mathématique se base sur un ensemble d'hypothèses telles que :

1- Seul le processus de déchargement des conteneurs entrant est pris en considération.
2- Les blocs sont répartis en fonction des navires pour le stockage des conteneurs.
3- La planification des portiques de quai fournie le temps de disponibilité pour le transport des conteneurs vers la cour.
4- Le temps de stockage dans les blocs est déterminé à partir des caractéristiques des conteneurs. Ainsi que le nombre des portiques de la cour dans chaque bloc.
5- Le temps de voyage des camions de transport (du navire vers le bloc de la cour) est calculé comme le temps du voyage sur le chemin le plus court.
6- Dans le cas où les portiques de cours sont occupés, les camions doivent attendre au bloc.

L'objectif de cette étude est de réduire le temps de transport et le temps d'attente des véhicules, sur le quai et dans la cour, de façon à minimiser le temps de complétion des opérations de déchargement des conteneurs. Les algorithmes Glouton et Génétique ont été utilisés pour résoudre ce problème. Ces algorithmes ont permis d'obtenir une solution proche de la solution optimale obtenue par CPLEX pour un problème de petite taille.

Dans le même contexte que le travail précédent, *Lee et al.* (2009b) ont essayé de montrer que l'amélioration de l'efficacité des opérations des terminaux à conteneurs est issue d'une bonne intégration du problème de planification des camions de transport et du problème de planification d'allocation de stockage des conteneurs simultanément. Le but de la recherche est de minimiser les retards et le temps total de transport des camions. Le problème de planification se base lors de sa formulation sur quatre conditions potentielles :

1- Un camion transporte une demande de chargement sur le navire après une autre demande de chargement.
2- Un camion transporte une demande de déchargement sur le navire après une demande de chargement.
3- Un camion transporte une demande de déchargement sur le navire après une autre demande de déchargement
4- Un camion transporte une demande de chargement sur le navire après une demande de déchargement.

Lee et al. ont mis en place l'algorithme d'insertion hybride (HIA) afin de résoudre le problème. Cet algorithme HIA a atteint des solutions proches de l'optimum pour les deux sous-problèmes : la planification des véhicules de transport et la planification de stockage dans un terminal maritime.

D'après le travail récent de *Cao et al.* (2010), les recherches dans la planification des portiques de cour et des camions de transport sont quasi-absentes vu que les recherches se concentrent essentiellement sur un seul problème de l'ensemble des opérations portuaires. Selon les auteurs, la synchronisation des équipements (portiques de cour et véhicules de transport) permet l'amélioration de la productivité des terminaux à conteneurs. Un modèle mathématique est développé afin d'atteindre l'objectif des auteurs qui est de minimiser le temps de complétion de l'opération de chargement des conteneurs dans la cour destinés à l'exportation. Ce modèle prend la forme d'une programmation linéaire mixte en se basant sur les hypothèses suivantes :

1- Seules les opérations de chargement des conteneurs d'exportation sont prises en considération.
2- L'emplacement de chaque conteneur est donné.
3- La localisation des portiques de cour est donnée.
4- Chaque conteneur a un temps de ramassage spécifique et ce temps est le même quelque soit le portique de cour.
5- Après avoir terminé la tâche actuelle, les portiques de cour et les camions peuvent se déplacer vers le point d'origine de la tâche qui suit.

6- La vitesse de déplacement du portique de cour est différente de la vitesse de déplacement du camion.
7- La capacité d'un camion est égale à 1, c'est à dire effectuer une seule tâche.
8- Pas d'interférence entre les camions.
9- Les conteneurs peuvent être manutentionnés dans n'importe quel ordre.
10- Les conteneurs sont disponibles à partir de la cour dans n'importe quel ordre.
11- Pas d'interférence entre les portiques de cour.

Cao et al. ont choisi la méthode de décomposition de Benders pour la résolution du problème de planification des camions et des portiques de cour. Cette méthode a conduit à de meilleures solutions lors de la résolution de problèmes d'optimisation multi-étapes. La décomposition de Benders peut être appliquée aux modèles intégrés dans les opérations de terminal à conteneurs. Selon les auteurs, la résolution du problème d'ordonnancement des opérations des portiques de cour et des camions est une nouvelle idée permettant l'amélioration de l'efficacité des opérations portuaires.

Une récapitulation de cette section relative aux travaux de planification simultanée des portiques et des camions est présentée sous la forme d'un tableau (Tableaux 2-4 et 2-5).

Tableau 2-4 : Résumé des travaux de recherche pertinents traitant la planification simultanée des portiques de cour et des véhicules de transport

Auteurs	**Problème traité & ses caractéristiques**	**Méthode(s) de résolution**	**Contributions clés**
Huynh et al. (2004)	- Minimiser le temps de rotation des camions - Modéliser les mouvements des camions et des portiques de cour	- Modèle de simulation - Algorithme de recherche par Tabou	Trouver le nombre optimal des portiques de cours pour atteindre le temps de rotation voulu des camions
Ng et *Ge*(2007)	- Interaction entre les opérations - Maximiser le taux de manutention - Minimiser les retards des navires	- Algorithme Heuristique *floue* - Modèle de simulation	Améliorer les opérations de chargement/ déchargement des conteneurs sur le quai
Chen et al.(2007)	Minimiser le makespan ou le temps que prennent les équipements pour servir un certain nombre de navires	Algorithme de recherche par Tabou	- Planifier les mouvements de plusieurs types d'équipements de manutention dans un terminal. - Augmenter la productivité d'un terminal maritime
Lei et al. (2008)	- Planifier les mouvements des portiques de cour et des déplacements des camions - Minimiser le temps de déplacement des camions	*Immune Algorithm*	Améliorer de l'efficacité d'un terminal à conteneurs
Zhang et Jiang (2008)	- Planifier les différents équipements de manutention en même temps - Déterminer les séquences de chargement / déchargement des conteneurs et les séquences des conteneurs transportés par les camions.	Modèle de simulation	- Améliorer la productivité d'un terminal à conteneurs - Maximiser l'utilisation des portiques de cour et des portiques de quai

Tableau 2-5 : Résumé des travaux de recherche pertinents traitant la planification simultanée des portiques de cour et des véhicules de transport (Suite)

Auteurs	**Problème traité & ses caractéristiques**	**Méthode(s) de résolution**	**Contributions clés**
Li et Tang (2009)	Minimiser le *makespan* de l'opération de déchargement	- Recherche par Tabou - Algorithme Glouton	Présenter une planification intégrée de l'affectation des conteneurs aux camions et l'allocation des conteneurs aux blocs de stockage
Lee et al (2009a)	Minimiser le *makespan* des opérations de déchargement	- Algorithme Génétique - Algorithme Glouton	Développer un modèle intégré de la planification des déplacements des camions et de l'allocation de la zone de stockage des conteneurs d'importation
Lee et al (2009b)	Minimiser la somme pondérée du temps de voyage des camions et du retard total des demandes de chargement et de déchargement	Algorithme d'insertion hybride (HIA)	Améliorer l'efficacité des opérations d'un terminal à conteneurs
Cao et al. (2010)	- Ordonner les opérations des portiques de cour et des camions - Minimiser le *makespan* de l'opération de chargement des conteneurs exportation	Décomposition de Benders	Améliorer la productivité des terminaux à conteneurs

2.3 Le problème d'interférence et des mouvements non productifs

2.3.1 Problème d'interférence entre les portiques de cour

Le problème d'interférence est bien lié au problème de planification de multiples portiques de cour dans un port maritime vu que ces portiques partagent la voie de circulation dans la cour et le déplacement de l'un peut bloquer le déplacement de l'autre. Dans ce cadre, *Ng* (2005) a considéré l'interférence inter-portiques de cour lors de la planification des mouvements de ce type d'équipements. Ce phénomène est défini comme étant un blocage physique qui doit être examiné dans le cadre de la planification des portiques de cour. D'après *Ng*, le problème d'interférence inter-portiques de cour est un problème compliqué et il nécessite le développement d'une méthode efficace pour la résolution du problème de planification.

L'interférence entre les portiques de cour peut avoir des effets sur la planification des portiques de quai. Dans ce contexte, *Jung et al.* (2006) ont mis en lumière la relation entre la planification des portiques de quai et la synchronisation des opérations des portiques de cour. Leur travail a pour objectif la minimisation du temps de rotation des navires et il s'adresse à l'ordonnancement des portiques de quai tenant compte des progressions des opérations de manutention au niveau de la cour. En effet, tout retard dans les opérations des portiques de cour affecte les opérations des portiques de quai.

Li et al. (2009) ont proposé un modèle pour la planification des portiques de cour tout en tenant compte de certaines contraintes relatives à l'interférence entre les portiques de cour, des distances fixes de séparation entre ces portiques et des opérations simultanées de stockage/récupération des conteneurs. Le modèle a pour objectif la minimisation d'une combinaison linéaire des collectes anticipées, des stockages et collectes tardifs. Le problème d'interférence entre les portiques de cour est développé dans les contraintes de la formulation mathématique.

Chen et Langevin (2011) ont traité le problème d'interférence d'une façon implicite. Le travail traite le problème de planification de plusieurs portiques de cour lors des opérations de chargement des conteneurs. L'objectif de ce travail est d'améliorer l'efficacité de traitement des conteneurs destinés à l'exportation par les portiques de cour. Les auteurs ont tenu compte des interférences potentielles qui peuvent se produire. En effet, ils ont intégré ce phénomène dans le

problème de planification des portiques. Une hypothèse relative à l'interférence est développée dans le modèle mathématique. Cette hypothèse prend en considération deux types d'interférence qui sont la « collision » et le « croisement ». Le premier type d'interférence se produit lorsque deux portiques de cour dans un même bloc essaient de ramasser des conteneurs stockés dans des baies adjacentes. Selon *Chen et Langevin* une distance minimale de cinq baies doit être respectée pour éviter une collision potentielle entre deux portiques de cour. Le second type est le croisement qui se présente lorsque le conteneur suivant à traiter par un portique de cour donné est localisé de l'autre côté d'un autre portique de cour. Afin d'éviter toute interférence possible, les auteurs ont développé plusieurs contraintes dans la formulation mathématique.

Le travail de *Liang et al*. (2011) a étudié le fonctionnement de plusieurs portiques de cour dans le cadre de chargement/déchargement des conteneurs. Selon les auteurs, les portiques de cour doivent laisser une distance de sécurité pour éviter les accidents. En effet, dans un même bloc, plusieurs portiques de cour peuvent travailler simultanément; dans cette situation un risque de collision entre les portiques de cour et même entre les camions peut se présenter. Dans la figure 2-1, le portique de cour 2 (*YC2*) veut se déplacer vers la baie 01 après avoir traité un conteneur localisé dans la baie 07 mais il est bloqué par le portique de cour 1 (*YC1*) qui est entrain de traiter un conteneur localisé dans la baie 03. Donc le portique de cour 2 doit attendre jusqu'à ce que le portique de cour 1 termine sa tâche.

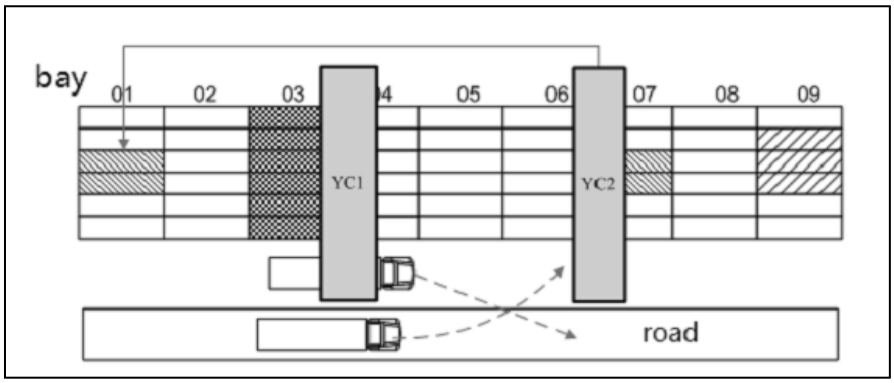

Figure 2-1 : La collision causée par les portiques de cour, Liang et al. (2011)

2.3.2 Problème des mouvements non productifs « *Rehandle* »

Kim (1997) propose dans son travail une méthodologie permettant l'estimation du nombre prévu des mouvements non productifs pour ramasser un conteneur quelconque ainsi que le nombre total de tous les mouvements non productifs des conteneurs dans une baie avec une configuration d'empilage initiale. Selon *Kim*, la hauteur et la largeur d'une baie jouent un rôle très important dans la détermination du nombre moyen de remanipulation ainsi que dans la conception de la configuration de stockage. Pour l'estimation du nombre prévu des mouvements non productifs dans le prochain ramassage, l'auteur a considéré plusieurs configurations de piles de conteneurs pour arriver à une formule approximative permettant l'estimation du nombre total prévu des mouvements non productifs.

Imai et al. (2006) ont mis l'accent sur le problème des mouvements non productifs des portiques lors de chargement/ déchargement des conteneurs dans un port maritime. Les auteurs se sont intéressés au stockage des conteneurs sur un navire afin d'en assurer la stabilité ainsi qu'aux mouvements non productifs des conteneurs dans les blocs dans le but de minimiser leur nombre. Ce problème est formulé comme un programme en nombres entiers multi-objectifs. *Imai et al*. ont utilisé la méthode de pondération en vue d'obtenir des solutions non dominées. En s'inspirant des travaux d'*Imai et Miki* (1989) et *Imai et al.* (2002), les auteurs ont pris les mouvements non productifs en considération dans la formulation grâce à une estimation du nombre des mouvements non productifs, puis ils ont introduit l'idée de la probabilité afin d'étudier cette estimation. La difficulté de cette dernière est liée à la récupération aléatoire des conteneurs, un calcul exact est basé sur une séquence de chargement prédéfinie. Pour résoudre le problème de chargement des conteneurs sur le navire ainsi celui du problème de remanutention, *Imai et al.* ont utilisé l'algorithme génétique.

2.4 Conclusion

On trouve des travaux qui étudient un seul sous-système de tout l'ensemble des opérations portuaires et d'autres qui intègrent plusieurs sous-problèmes. Dans ce cadre, les analyses et les études sont diversifiées et multiples vu la diversification des opérations et des équipements mis en place.

Dans le cadre de la planification des véhicules de transport et des équipements de manutention (portiques de cour et portiques de quai), cela est plus facile si chaque équipement est étudié à part. Cependant, l'analyse se complique en regroupant deux éléments ou plus. Par la suite, les travaux diminuent en étudiant les opérations des différents équipements simultanément. De ce fait, les camions et les portiques de cour sont les équipements les plus utilisés dans la partie terrestre du terminal.

CHAPITRE 3 FORMULATION MATHÉMATIQUE

3.1 Introduction

Dans la première partie de ce chapitre, nous présentons les hypothèses du modèle prises en compte dans le cadre de la modélisation. Dans la seconde partie, nous aborderons la formulation mathématique du problème d'optimisation des mouvements des conteneurs et nous présenterons ses différentes composantes : paramètres, variables de décision, fonction objectif et les contraintes utilisées dans notre travail. Puis, on va présenter une situation en fonction des données fictives pour une meilleure compréhension du problème. Par la suite, pour résoudre ce problème on a fait appel au logiciel de programmation AMPL Studio. Enfin, une représentation graphique contenant le chemin des différents équipements de manutention sera présentée.

3.2 Les hypothèses du modèle

Lors de l'élaboration d'un modèle mathématique, il faut prendre en compte un certain nombre d'hypothèses. Ces dernières vont être intégrées d'une façon ou d'une autre dans le modèle. Nos hypothèses sont les suivantes :

1. On se limite aux opérations de chargement des conteneurs d'exportation (*outbound*).
2. La localisation des conteneurs est donnée.
3. Une cour est formée de plusieurs blocs adjacents.
4. Il y a jusqu'à deux portiques de cour dans un bloc.
5. Les déplacements des portiques de cour entre les blocs sont possibles, au maximum un seul déplacement.
6. Pour chaque conteneur, on connait sa destination sur le quai (le portique de quai).
7. On considère les mouvements simultanés des portiques de cour et des camions.
8. Tous les conteneurs d'une baie sont destinés au même navire.
9. On prend en compte la possibilité d'interférence des portiques de cour dans un même bloc.
10. Les conteneurs sont classés en groupes avec un ordre et une priorité dans la manutention de chaque groupe. L'ordre est connu d'avance.
11. Les mouvements non productifs des conteneurs (Rehandle) sont pris en compte.

12. On ne considère pas le temps de déchargement des camions, il s'agit d'un déchargement instantané.
13. La vitesse des portiques de cour est de 36 km/h (*Lu Chen*, 2006).
14. On s'intéresse uniquement aux conteneurs de 40 pieds.

3.3 Le modèle mathématique

3.3.1 Les paramètres du modèle

- Les indices utilisés dans le modèle sont les suivants :

  i, j : Indices relatifs aux conteneurs ; $i, j \in C$.

  $i = 0$: Conteneur fictif.

  m : Indice des portiques de cour ; $m \in M$.

  n : Indice des camions ; $n \in N$.

  q : Indice des portiques de quai ; $q \in Q$.

  b : Indice du bloc ; $b \in B$.

- Les ensembles et les paramètres utilisés dans le modèle :

  C : Ensemble des conteneurs à traiter dans la cour.

  M : Ensemble des portiques de cour (YC) disponibles dans la cour.

  N : Ensemble des camions (YT) disponibles dans la cour.

  B : Ensemble des blocs dans la cour.

  Q : Ensemble des portiques de quai (QC) disponibles sur le quai.

  M_0^b : Nombre initial de portiques de cour dans un bloc b ; ($b \in B$).

  L : Ensemble des localisations pour le stockage des conteneurs d'exportation (localisation = une adresse formée du numéro du bloc, baie, rangée et étage).

  l_i : Localisation d'un conteneur i dans la cour de stockage.

  a_i : Numéro de la baie dans un bloc pour un conteneur i dans une localisation l_i.

b_i : Numéro du bloc dans la cour pour un conteneur i dans une localisation l_i.

e_i : Numéro de l'étage pour un conteneur i dans une localisation l_i.

r_i : Numéro de la rangée pour un conteneur i dans une localisation l_i.

h_1 : Temps de manutention d'un conteneur.

h_2 : Temps de manutention d'un conteneur pour le prendre et le remettre à sa place (*rehandle*) ; on suppose que $h_2 = 2 * h_1$.

P : Ensembles des paires de conteneurs avec une relation de priorité. On parle ici d'un ordre de priorité des conteneurs qui doit être respecté lors du processus de manutention par les portiques de quai.

U : Ensemble des paires de conteneurs qui ne peuvent pas être traitées en même temps à cause de l'interférence. À titre d'exemple, deux conteneurs dans la même baie ne peuvent pas être traités en même temps par deux portiques de cour parce qu'ils sont trop près l'un de l'autre.

T : Grande valeur.

k_{ij} : Temps de parcours d'un portique de cour du conteneur i vers le conteneur j quelle que soit la localisation de i et j ($l_i \neq l_j$).

t_i : Temps de transport du conteneur i par un camion, connu a priori, de sa localisation li à sa destination sur le quai ; ($i \in C$).

tv_{qa} : Temps de retour à vide d'un camion, du portique de quai q vers la baie a.

3.3.2 Les variables de décision du modèle

d_i^{YC} : Temps de début de prélèvement du conteneur i par un portique de cour ; ($i \in C$).

d_i^{YT} : Temps de départ d'un camion transportant le conteneur i ;($i \in C$).

$X_{ijm} = 1$, si le portique de cour m traite le conteneur j juste après le conteneur i ; 0 sinon.

$Y_{ijn} = 1$, si le camion n charge le conteneur j juste après le chargement du conteneur i ; 0 sinon.

$Z_{bb'm} = 1$, si le portique de cour m se déplace du bloc b vers le bloc b' ; 0 sinon ($b \neq b'$).

$V_{ij} = 1$, si le traitement du conteneur *j* commence après la fin du traitement du conteneur *i* par un portique de cour.

p_i : Temps de traitement du conteneur *i* par un portique de cour qui comprend le temps de prélèvement du conteneur (h_1) et le temps pour enlever et remettre les conteneurs au-dessus (ayant le même a_i et r_i ; de plus $e_i < e_j$) (h_2), sachant qu'on a supposé que $h_2 = 2 * h_1$. Le temps p_i est calculé comme suit ;

$$p_i = h_1 + \sum_{\substack{j \in C \\ a_i = a_j \\ r_i = r_j \\ e_i < e_j}} h_2 \times V_{ij}$$

où $\sum_{\substack{j \in C \\ a_i = a_j \\ r_i = r_j \\ e_i < e_j}} h_2 \times V_{ij}$ est le temps non productif des portiques de cour (*rehandle*).

La variable p_i peut être divisée en deux parties p_i^1 et p_i^2 qui sont respectivement le temps de manutention du conteneur *i* et de son chargement sur le camion et le temps de remettre le(s) conteneur(s) qu'on a enlevé à sa (leur) place(s).

D'où $p_i = p_i^1 + p_i^2$ avec ;

$$p_i^1 = h_1 + \sum_{\substack{j \in C \\ a_i = a_j \\ r_i = r_j \\ e_i < e_j}} h_1 \times V_{ij}$$

$$p_i^2 = \sum_{\substack{j \in C \\ a_i = a_j \\ r_i = r_j \\ e_i < e_j}} h_1 \times V_{ij}$$

3.3.3 Le modèle mathématique

Minimiser $\{\underset{i}{\text{Max}}\ d_i^{YT}\}$ (1)

Sous contraintes :

$$\sum_{\substack{i=0 \\ i\neq j}}^{|C|} \sum_{m=1}^{|M|} X_{ijm} = 1 \ ; \forall\, j \in C \quad (2)$$

$$\sum_{i=0}^{|C|} \sum_{n=1}^{|N|} Y_{ijn} = 1 \ ; \forall\, j \in C \quad (3)$$

$$\sum_{j=1}^{|C|} X_{0jm} = 1; \forall\, m \in M \quad (4)$$

$$\sum_{j=1}^{|C|} Y_{0jn} = 1; \forall\, n \in N \quad (5)$$

$$\sum_{\substack{j=0 \\ i\neq j}}^{|C|} X_{ijm} = \sum_{\substack{j=0 \\ i\neq j}}^{|C|} X_{jim} \ ; i \in C, m \in M \quad (6)$$

$$\sum_{\substack{j=0 \\ i\neq j}}^{|C|} Y_{ijn} = \sum_{\substack{j=0 \\ i\neq j}}^{|C|} Y_{jin} \ ; i \in C, n \in N \quad (7)$$

$$p_i^1 = h_1 + \sum_{\substack{j\in C \\ a_i=a_j \\ r_i=r_j \\ e_i<e_j}} h_1 \times V_{ij} \ ; i \in C \quad (8)$$

$$p_i^2 = \sum_{\substack{j\in C \\ a_i=a_j \\ r_i=r_j \\ e_i<e_j}} h_1 \times V_{ij} \ ; i \in C \quad (9)$$

$$d_j^{YC} \geq d_i^{YC} + (p_i^1 + p_i^2) + k_{ij} + T\,(X_{ijm} - 1); \ i \in C, j \in C, m \in M \quad (10)$$

$$d_j^{YC} \geq d_i^{YT} + k_{ij} + T\,(X_{ijm} - 1); \ i \in C, j \in C, m \in M \quad (11)$$

$$d_j^{YT} + t_j \geq d_i^{YT} + t_i; \ \forall\, (i, j) \in P \quad (12)$$

$$d_j^{YC} \geq d_i^{YC} + (p_i^1 + p_i^2) + T\,(V_{ij} - 1); \ i \in C, \ j \in C, (i, j) \in U \quad (13)$$

$$d_j^{YT} \geq d_i^{YT} + t_i + tv_{q_i a_j} + T\,(Y_{ijn} - 1);\ i \in C, j \in C, n \in N \tag{14}$$

$$d_i^{YT} \geq d_i^{YC} + p_i^1;\ i \in C \tag{15}$$

$$Z_{b_i b_j m} \geq X_{ijm}\ ;\ i \in C, j \in C, m \in M, b_i \neq b_j \tag{16}$$

$$M_0^{b'} + \sum_{\substack{b \in B \\ | b \neq b'}} \sum_{m=1}^{|M|} Z_{bb'm} \leq 2;\ b' \in B, m \in M \tag{17}$$

$$\sum_{b' \in B} \sum_{\substack{b \in B \\ | b \neq b'}} Z_{bb'm} \leq 1;\ m \in M \tag{18}$$

$$d_j^{YC} - d_i^{YC} - (p_i^1 + p_i^2) \leq V_{ij} T;\ i \in C, j \in C \tag{19}$$

$$d_i^{YC} + (p_i^1 + p_i^2) - d_j^{YC} \leq (1 - V_{ij}) T\ ;\ i \in C, j \in C \tag{20}$$

$$V_{ij} + V_{ji} = 1;\ \forall\ (i, j) \in U \tag{21}$$

$$\sum_{m=1}^{M} X_{ii'm} \leq V_{ij} + V_{ji}, \forall i, j, i' \mid b_i = b_j = b_{i'} \text{ et } a_i < a_j < a_{i'} \tag{22}$$

$$\sum_{m=1}^{M} X_{jj'm} \leq V_{ij} + V_{ji}, \forall i, j, j' \mid b_i = b_j = b_{j'} \text{ et } a_{j'} < a_i < a_j \tag{23}$$

$$\sum_{m=1}^{M} X_{ii'm} \leq V_{ij} + V_{ji} + T\,(1 - u_{iji'j'})\ ;\ \forall i, j, i', j' \mid b_i = b_{i'} = b_j = b_{j'} \text{ et } a_i < a_{j'} < a_{i'} < a_j \tag{24}$$

$$\sum_{m=1}^{M} X_{jj'm} \leq V_{ij} + V_{ji} + T u_{iji'j'}\ ;\ \forall\ i, j, i', j' \mid b_i = b_{i'} = b_j = b_{j'} \text{ et } a_i < a_{j'} < a_{i'} < a_j \tag{25}$$

$$d_i^{YC}, d_i^{YT}, p_i^1, p_i^2 \geq 0 \tag{26}$$

$$X_{ijm}, Y_{ijn}, Z_{bb'm}, V_{ij}, u_{iji'j'} \in \{0, 1\} \tag{27}$$

Dans la fonction objectif (1), on cherche à minimiser le temps de complétion (makespan). Les contraintes (2) et (3) garantissent que chaque conteneur est affecté à un seul portique de cour et à un seul camion.

Les contraintes (4) et (5) garantissent que les équipements qui sont respectivement les portiques de cour et les camions sortent une seule fois du point de départ. Les contraintes (6) et (7) assurent la conservation des flux pour chaque équipement de manutention.

La contrainte (8) définit le temps de manutention du conteneur *i* et de son chargement sur le camion. Tandis que, la contrainte (9) définit le temps de remise en place des conteneurs déjà enlevés.

La contrainte (10) permet d'assurer l'ordre de traitement des conteneurs par chaque portique de cour. Si un portique de cour *m* traite le conteneur *j* juste après le conteneur *i* alors le portique *m* va prendre un temps de déplacement k_{ij} pour aller chercher le conteneur *j* suivant.

La contrainte (11) permet de présenter l'ordre de traitement des conteneurs par chaque portique de cour en fonction du temps de départ des camions transportant les conteneurs. La contrainte (12) signifie que lorsque les conteneurs *i* et *j* forment une paire de conteneurs avec une relation de priorité alors le conteneur *i* est traité avant le conteneur *j* dés son arrivée sur le quai.

La contrainte (13) définit la séquence de traitement des conteneurs pour chaque portique de cour au cas où V_{ij} = 1 (le traitement du conteneur *j* est fait après le traitement du conteneur *i*).

La contrainte (14) garantit l'ordre d'enchaînement des conteneurs par camion. La contrainte (15) détermine la séquence de traitement des conteneurs par camion.

Dans la contrainte (16), la variable $Z_{b_i b_j m}$ prend la valeur 1 si le portique de cour *m* traite le conteneur *j* juste après le conteneur *i* tel que les deux conteneurs *i* et *j* sont stockés dans deux blocs différents. La contrainte (17) permet de vérifier l'hypothèse que dans un même bloc, on ne peut pas avoir plus de 2 portiques de cours : si le nombre de portique initial $M_0^{b\prime}$ = 2, pas de mouvement des portiques entre les blocs ; si $M_0^{b\prime}$=1, un seul portique de cour *m* au plus peut se déplacer entre les blocs et si $M_0^{b\prime}$= 0, le nombre maximal de portiques *m* qui peuvent se déplacer entre les blocs est 2. La contrainte (18) assure qu'au maximum un seul déplacement d'un portique de cour *m* peut se produire vers un autre bloc (une autre destination).

Les contraintes (19) et (20) définissent aussi la variable V_{ij}. V_{ij} = 1 si le traitement du conteneur *j* commence après que le traitement du conteneur *i* est achevé.

La contrainte (21) garantit qu'il n'y a pas de collision entre les portiques de cour, en d'autres termes, c'est que les conteneurs *i* et *j* ne sont pas traités en même temps.

Les contraintes (22) – (25) assurent que le phénomène d'interférence ne va pas se produire. On suppose :

- Les conteneurs *i* et j sont stockés dans le même bloc et $a_i < a_j$
- Les conteneurs *i* et *j* sont manutentionnés, simultanément, respectivement par les portiques de cour *m* et *m'* donc $V_{ij} + V_{ji} = 0$.

On parle d'interférence dans deux situations possibles (*Chen et Langevin, 2011*) et qui sont décrites ci-dessous :

a) Les contraintes (22) et (23) décrivent la première situation d'interférence. Le portique de cour *m* est bloqué par le portique de cour *m'* et il ne peut pas aller chercher les conteneurs se trouvant à droite du conteneur *j*. La même chose pour le portique *m'* qui se trouve bloqué par *m* et il ne peut pas aller chercher les conteneurs à gauche du conteneur *i*.
b) La situation où les deux portiques de cour peuvent ne pas se déplacer dans une direction opposée c'est ce que les contraintes (24) et (25) empêchent. Ces deux contraintes introduisent une nouvelle variable au modèle $u_{iji'j'}$.La variable $u_{iji'j'}$ = 1 si les portiques de cour *m* et *m'* fonctionnent en même temps (en d'autres termes, *m* va se déplacer de *i* vers *i'* et *m'* de *j* vers *j'*). La variable $u_{iji'j'}$ = 0 sinon ; avec *i*, *i'*,*j* et *j'* appartenant au même bloc et $a_i < a_j < a_{i'} < a_{j'}$.

La contrainte (26) garantit que les variables de décision sont positives. La contrainte (27) signifie que les variables de décision sont binaires.

3.4 Exemple

Dans le cadre d'une présentation d'une solution préliminaire du problème traité, on essaye de proposer un petit exemple avec des données fictives afin de décrire les enchaînements des équipements dans un terminal à conteneurs.

Dans cet exemple, on suppose qu'il s'agit de :

- 2 blocs : avec 4 baies, 3 rangées, 3 étages
- 2 portiques de cour : *YC1* et *YC2*
- 3 camions : *YT1*, *YT2* et *YT3*
- 10 conteneurs distribués dans les deux blocs

La localisation de chaque conteneur est comme suit :

C1 :{bloc1, baie 1, rangée1, étage1}

C2:{bloc1, baie 2, rangée2, étage1}

C3 :{bloc1, baie 3, rangée3, étage1}

C4 :{bloc1, baie 4, rangée2, étage2}

C5 :{bloc1, baie 4, rangée2, étage1}

C6 :{bloc2, baie 5, rangée3, étage1}

C7 :{bloc2, baie 7, rangée2, étage1}

C8 :{bloc2, baie 7, rangée1, étage1}

C9 :{bloc2, baie 8, rangée3, étage1}

C10 :{bloc2, baie 8, rangée2, étage1}

Les localisations des conteneurs sont représentées à la figure 3-1, il s'agit ici de la partie de stockage des conteneurs : la cour.

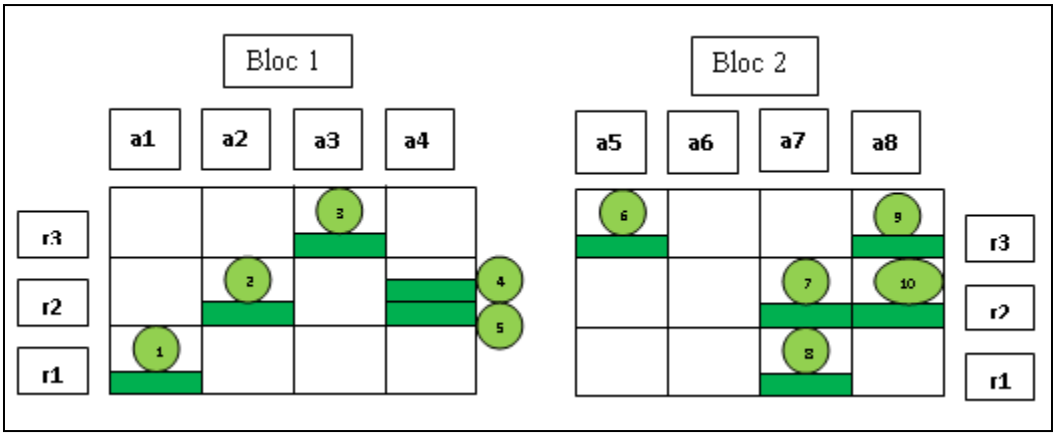

Figure 3-1 : Les localisations des conteneurs dans l'aire de stockage

On suppose que les conteneurs sont regroupés. Ces groupes suivent un ordre de priorité à leur tour. Dans cet exemple, on suppose qu'on a 4 groupes A, B, C et D.

Les données relatives aux groupes sont détaillées comme suit :

Groupe A : C1, C4, C5 et C8

Groupe B : C2 et C7

Groupe C : C9 et C10

Groupe D : C3 et C6

Les groupes de conteneurs A et B seront manutentionnés par le portique de quai 2 afin d'être transportés par le porte-conteneurs 2. Alors que les groupes de conteneurs C et D seront manutentionnés par le portique de quai 1 et qui seront par la suite transportés par le porte-conteneurs 1 (Tableau 3-1).

En effet, lors de la manutention des conteneurs par les portiques de quai, ces derniers doivent suivre un ordre, cela pour plusieurs raisons : facilitation du processus de déchargement des conteneurs dans le terminal de la destination, caractéristiques particulières de la marchandise dans les conteneurs, etc. Dans notre cas, les conteneurs appartenant au groupe B doivent être manutentionnés avant les conteneurs du groupe A. De même, les conteneurs du groupe C doivent être traités avant les conteneurs du groupe D par les portiques de quai.

On note qu'il est possible que la priorité sur les groupes de conteneurs soit transférée en priorité sur les conteneurs. De plus, en vertu d'une des hypothèses du modèle développé, la destination des conteneurs est connue sur le quai (sur quel porte-conteneurs ils seront chargés). Dans l'exemple choisi, la destination des conteneurs sur le quai et le portique de quai par lequel ils seront chargés sont représentés comme suit :

Tableau 3-1 : Destination des conteneurs sur le quai

$\mathbf{C_i}$	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
QC	2	2	1	2	2	1	2	2	1	1

On peut résumer les données d’ordre général dans le tableau 3-2 :

Tableau 3-2 : Données d'ordre général

Paramètres	**Valeurs**
Nombre total de conteneurs	10
Nombre total de portiques de cour	2
Nombre total de portiques de quai	2
Nombre total de camions	3
Nombre de baies par bloc	4
Nombre de rangées par bloc	3
Nombre d`étages maximal	3

Les autres données relatives au temps de manutention des portiques de cour, au temps de transport et au temps de retour à vide des camions sont présentées respectivement dans les tableaux 3-3, 3-4 et 3-5.

Tableau 3-3 : Le temps de manutention des portiques de cour

Paramètres	**Valeurs (en sec)**
Temps de manutention d’un conteneur (h_1)	20
Temps pour déplacer un conteneur et le remettre à sa place (h_2)	40

Tableau 3-4 : Le temps de transport vers la destination sur le quai des conteneurs t_i pour chaque camion

Conteneur C	**Destination sur le quai**	**Temps de transport (min)**	**Temps de transport (sec)**
C1	QC 2	3	180
C2	QC 2	3	180
C3	QC 1	2	120
C4	QC 2	3	180
C5	QC 2	3	180
C6	QC 1	2	120
C7	QC 2	2	120
C8	QC 2	2	120
C9	QC 1	3	180
C10	QC 1	3	180

Tableau 3-5 : Le temps de retour à vide des camions d'un portique de quai à une baie dans la cour de stockage

Portique de Quai	**Baie**	**Temps tv_{qa} (sec)**
QC1	a1	60
QC1	a2	60
QC1	a3	60
QC1	a4	60
QC1	a5	90
QC1	a6	90
QC1	a7	90
QC1	a8	90
QC2	a1	90
QC2	a2	90
QC2	a3	90
QC2	a4	90
QC2	a5	60
QC2	a6	60
QC2	a7	60
QC2	a8	60

3.5 Solution

Après la présentation des différentes catégories de données, on a eu recours au logiciel AMPL Studio dans le but de coder le modèle mathématique et présenter une solution au problème. Dans une première étape, on a écrit le modèle mathématique dans un fichier *mod*. Par la suite, les données exposées dans la section ci-dessus sont présentées dans un fichier *dat*. On a fait appel à CPLEX pour la résolution de ce programme. Les différents fichiers sont présentés dans l'annexe 1.

La solution obtenue avec le logiciel de programmation AMPL (Figure 3-2), nous a permis d'avoir plusieurs informations sur le problème en question comme la valeur du «makespan», la séquence de traitement de chaque portique de cour et de chaque camion. Le temps de complétion pour traiter 10 conteneurs est de 560 secondes.

La séquence de chaque équipement est comme suit :

YC1: C7 – C8

YC2: C10 – C9 – C6 – C2 – C1 – C3 – C5– C4

YT1: C7 – C6 – C3 – C4

YT2:C10 – C2 – C5

YT3: C9 – C1 – C8

À cet effet, la figure 3-2 illustre le fonctionnement de chaque équipement dans la cour.

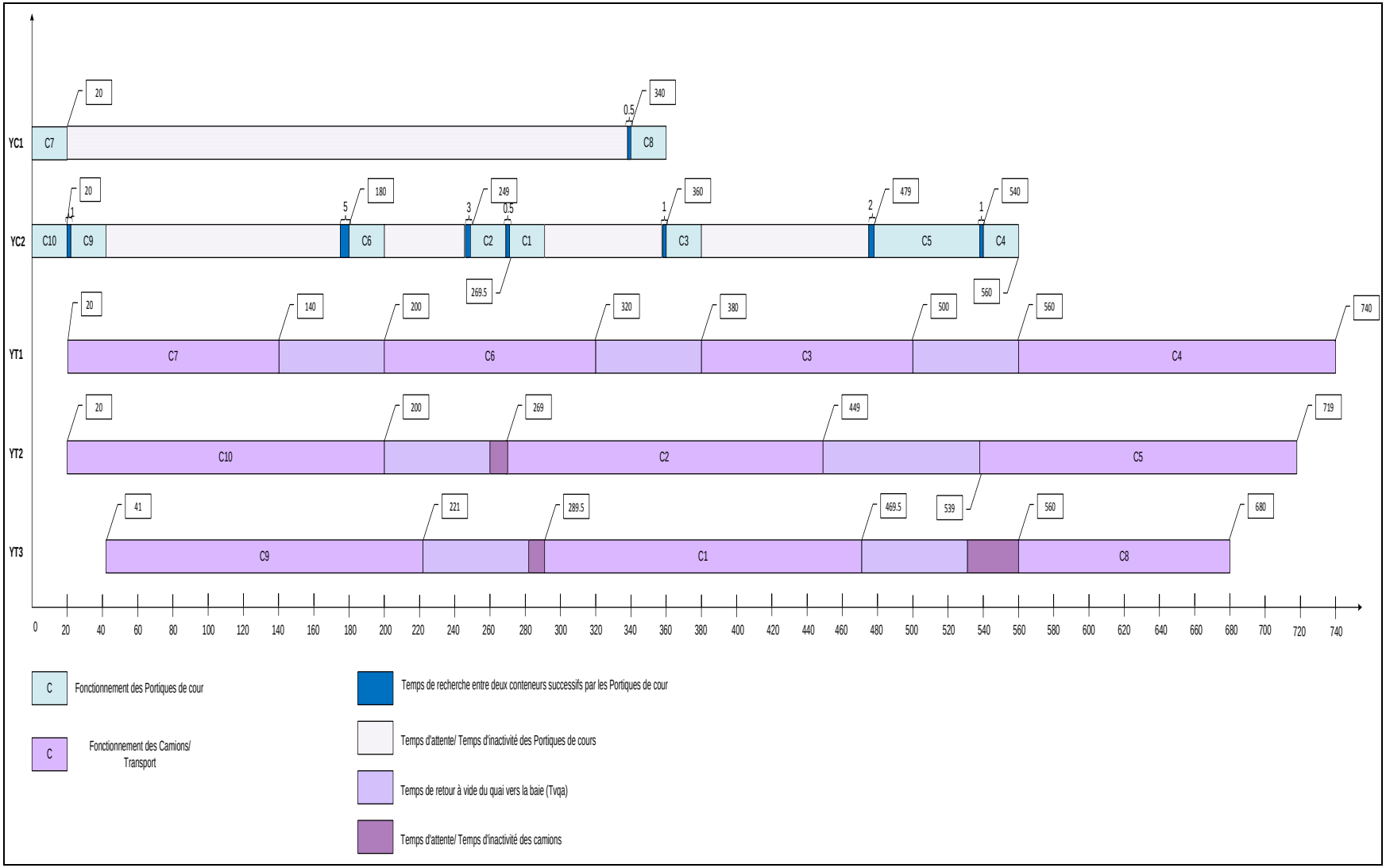

Figure 3-2 : Présentation de la solution avec AMPL

La solution obtenue par AMPL nous permet de déterminer le fonctionnement de chaque équipement. La séquence de travail de chacun (portique de cour et camion) est présentée graphiquement (Figure 3-3 et 3-4). La figure 3-5 présente le réseau de parcours des deux équipements dans le même graphique.

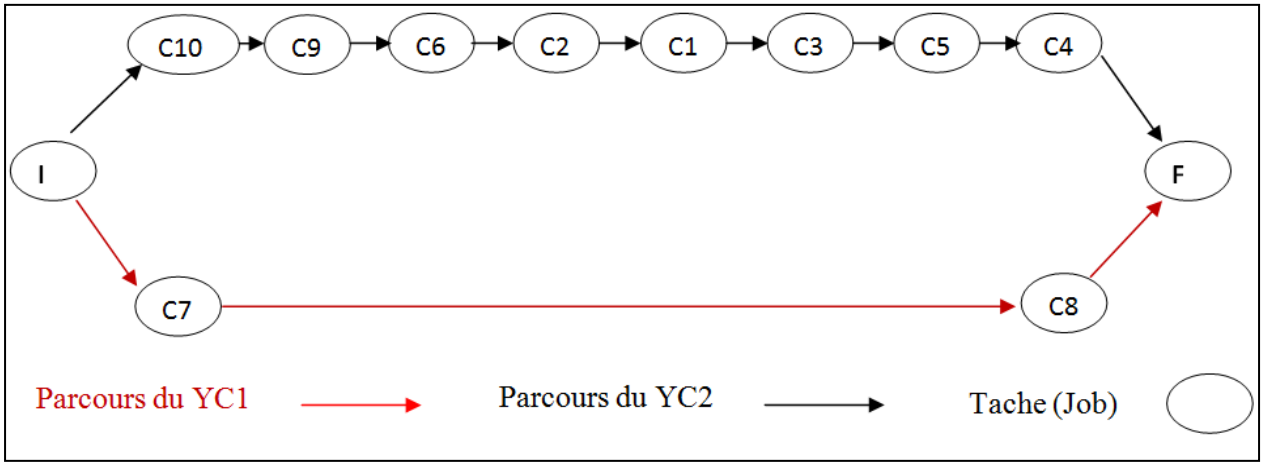

Figure 3-3 : Les séquences de fonctionnement des portiques de cour

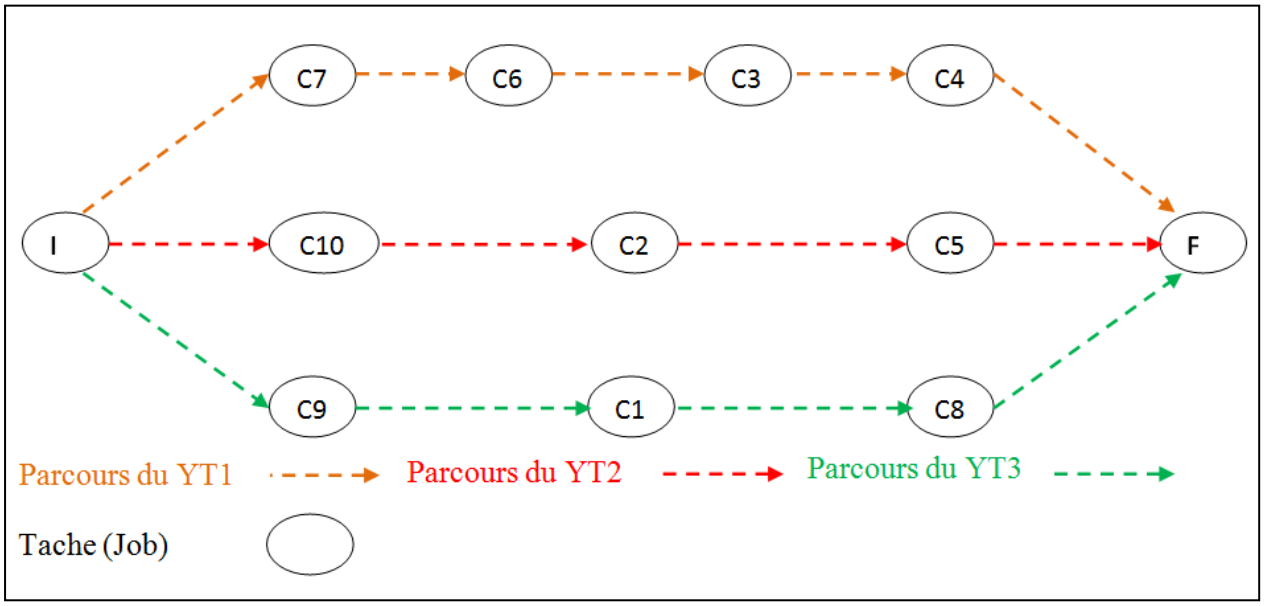

Figure 3-4 : Les séquences de fonctionnement des camions

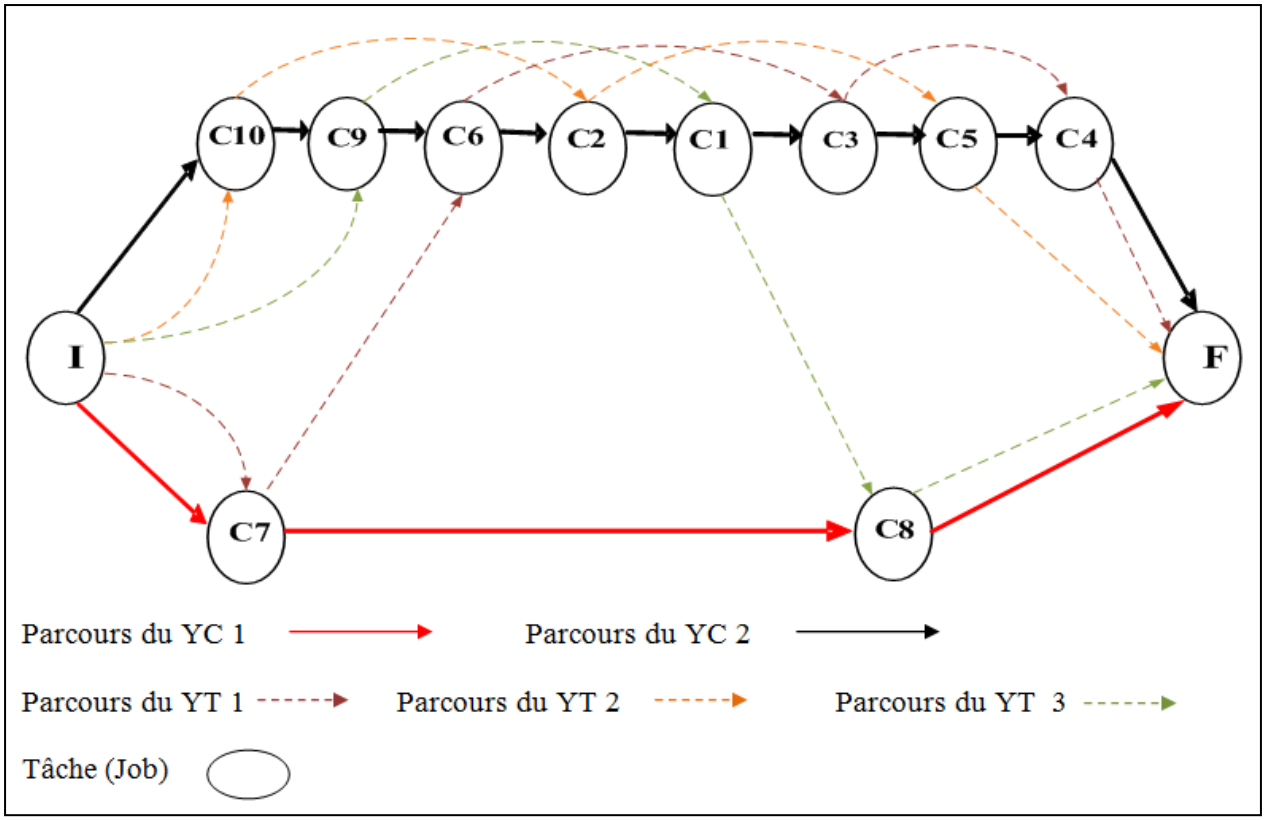

Figure 3-5 : Les séquences de fonctionnement des différents équipements de manutention

3.6 Conclusion

Dans cette partie du travail, on a présenté une formulation mathématique du problème développé dans les parties antécédentes. Cette formulation nous a permis d'intégrer les hypothèses du problème ainsi que les objectifs à atteindre. Dans un deuxième temps, et avec des données fictives, on a résolu le problème avec le logiciel AMPL. On a obtenu une solution avec un temps de complétion (560 secondes) pour le traitement des 10 conteneurs présentés dans l'exemple ainsi que les séquences de traitement des différents équipements de manutention mis en place dans le terminal maritime. La résolution avec AMPL est la première approche de résolution du problème. Ensuite et dans le cadre du chapitre suivant, on va essayer de résoudre le problème avec une deuxième approche de résolution qui est la méthode de recherche adaptative à large voisinage (*ALNS*).

CHAPITRE 4 MÉTHODE DE RÉSOLUTION

Dans ce chapitre, on met l'accent sur la méthode de résolution (*ALNS*). Premièrement, on décrit et on présente cette méthode en se basant sur une étude bibliographique. Deuxièmement, on applique la méthode *ALNS* sur le problème traité tout au long de ce travail. Finalement, une codification de la méthode heuristique avec un langage de programmation et une analyse des résultats sont faites dans la dernière partie de ce chapitre.

4.1 Présentation de la méthode

L'heuristique de recherche adaptative à large voisinage, peut se traduire par « *Adaptative large neighborhood search (ALNS)* ». Il s'agit d'une extension de l'heuristique de recherche à large voisinage « *Large neighborhood search LNS* » qui a été proposée par *Shaw* (1998). Il est donc préférable, voire nécessaire, d'étudier plus en profondeur en première partie la méthode *LNS* puis en seconde partie le fonctionnement de la méthode *ALNS*. La troisième partie met en exergue le travail de *Pisinger et Ropke* (2007) alors que la dernière partie porte sur les modifications faites par *Laporte et al.* (2010).

4.1.1 Recherche à large voisinage

La méthode *LNS* a donné de bons résultats pour les problèmes relatifs à la planification et aux problèmes de transport. C'est une méthode qui explore un voisinage complexe via plusieurs heuristiques, ce qui rend possible la découverte de meilleures solutions à chaque itération. Le principe de la méthode *LNS* est qu'une solution initiale possible pour le problème traité est améliorée graduellement par sa destruction partielle puis par la réparation de cette solution dans le but d'obtenir une meilleure solution.

Ropke et Pisinger (2006) ont énuméré quelques exemples de problèmes qui peuvent être résolus à l'aide de la méthode *LNS*, on trouve par exemple : le problème de voyageur de commerce (*Traveling Salesman Problem TPS*) et le problème de tournées de véhicules avec une contrainte de capacité (*Capacitated Vehicule Routing Problem CVRP*). Les deux auteurs ont donné une définition d'un voisinage d'une solution $x \in X$ comme $N(x) \subseteq X$; avec : X l'ensemble des solutions réalisables, x est une solution réalisable, N est une fonction qui associe à x un sous ensemble de solution.

Un exemple simple de voisinage pour le problème de voyageur de commerce est le voisinage *2-opt*. Avec la méthode *LNS*, le voisinage est défini implicitement par la méthode de destruction et la méthode de réparation (*Ropke et Pisinger,* 2006).

Pour la méthode de destruction, elle détruit une partie de la solution actuelle alors que la méthode de réparation reconstruit la solution détruite. L'idée principale derrière l'heuristique *LNS* est que le voisinage large permet à l'heuristique de naviguer facilement dans l'espace de solution même s'il s'agit d'un cas fortement contraint, tandis que dans un petit voisinage, la navigation dans l'espace de solution devient plus difficile. En effet, *LNS* est basée sur un processus continu de relaxation et de re-optimisation. Deux facteurs peuvent influencer le déroulement de la recherche : comment choisir les clients à supprimer ainsi que le processus utilisé pour les réinsérer. *Shaw* (1998) a utilisé deux algorithmes relatifs à ces deux facteurs dans les problèmes de tournées des véhicules. Il a montré dans son travail que *LNS* est plus compétitive en termes de performance et en termes de capacité à générer de meilleures solutions qui n'ont pas été trouvées par d'autres méthodes.

4.1.2 Recherche Adaptative à Large Voisinage

Dans le cadre de notre mémoire, on a choisi de développer une méthode heuristique de type recherche adaptative à large voisinage « *Adaptative Large Neighborhood Search (ALNS)* ». Cette méthode heuristique a été proposée par *Ropke* et *Pisinger* (2006) et il s'agit d'une extension de l'heuristique de recherche à un voisinage local développé par *Shaw* (1998).

La méthode *ALNS* a été testée pour la première fois sur le problème de collecte et de livraison avec fenêtres de temps en 2006 par *Ropke* et *Pisinger*. L'heuristique *ALNS* est composée d'un certain nombre de sous-heuristiques et elle est facile à appliquer dans la plupart des problèmes de planification et de transport.

Il semble que l'heuristique *ALNS* est basée non seulement sur la méthode *LNS* présentée par *Shaw* (1998) mais aussi sur le principe de Détruire et Recréer (*Ruin and Recreate Principle*) présenté par *Schrimpf et al.* (2000).

Dans le même ordre d'idée, la méthode *ALNS* possède plusieurs propriétés (*Pisinger et Ropke*, 2007). En effet, l'algorithme *ALNS* explore un espace de solutions large d'une façon structurée. Il s'agit d'un algorithme robuste et qui est capable de s'adapter à des caractéristiques diversifiées et

des exemples variés. De plus, *ALNS* s'adapte spécifiquement à des problèmes fortement contraints. L'heuristique *ALNS* se base sur la recherche locale où participent plusieurs algorithmes simples pour la modification d'une solution en cours. À chaque itération, on choisit un algorithme pour détruire la solution actuelle et un autre est choisi pour la réparer. À la fin, on arrive à une nouvelle solution qui sera acceptée si elle vérifie certains critères.

Dans cette section, nous passons en revue des travaux de recherche pertinents qui enrichirent le développement de la méthode. C'est dans ce contexte que nous citons quelques travaux récents. D'abord, le travail de *Pisinger* et *Ropke* (2007) où les auteurs ont utilisé l'heuristique *ALNS* pour plusieurs exemples de problèmes de tournées de véhicules. Ils ont appliqué cette méthode sur le problème de tournées de véhicules avec fenêtres de temps (*Vehicle Routing Problem with Time Windows VRPTW*), le problème de tournées de véhicules avec contraintes de capacité (*Capacitated Vehicule Routing Problem CVRP*), le problème de tournées de véhicules avec de multiples dépôts (*Multi-Depot Vehicle Routing Problem MDVRP*) ainsi que d'autres. Les deux auteurs ont énuméré quelques avantages de l'approche *ALNS*, on cite à titre d'exemple que cette approche permet d'avoir des solutions de meilleure qualité et qu'il s'agit d'un algorithme robuste. Les études expérimentales appliquées aux différents types de problèmes de tournées de véhicules ont donné des résultats prometteurs.

Ensuite, dans le travail de *Muller* (2009), l'auteur a présenté une application de l'algorithme *ALNS* au problème d'ordonnancement de projets avec contraintes de ressources (*Resource-Constrained Project Scheduling Problem, RCPSP*).

Un problème similaire est celui du problème de tournées sur les arcs avec une contrainte de capacité et une demande stochastique (*CARPSD*) traité dans le travail de *Laporte et al.* (2010). Ce travail aborde le problème de collecte des ordures et il fait appel à l'heuristique de recherche adaptative à large voisinage. Les auteurs montrent avec les résultats expérimentaux la supériorité de cette méthode heuristique.

Dans *Azi et al.* (2010), les auteurs résolvent le problème de tournées de véhicules avec des voyages multiples. L'objectif de ce travail est de maximiser le nombre de clients à desservir et de minimiser la distance totale parcourue par les véhicules. Pour résoudre ce problème, les auteurs ont proposé l'algorithme adaptatif de recherche à large voisinage.

Le travail de *Salazar-Aguilar et al.* (2011) utilise l'algorithme de recherche adaptative à large voisinage pour le problème de synchronisation de tournées de véhicules pour des opérations de déneigement. L'algorithme *ALNS* est divisé en deux sous-algorithmes relatifs aux deux phases de la méthode qui sont respectivement la phase de construction et la phase d'amélioration. D'après les auteurs, cet algorithme est très efficace pour la résolution du problème de déneigement traité et les résultats expérimentaux ont prouvé cette efficacité.

Ribeiro et Laporte (2011) ont abordé le problème de tournées de véhicules avec capacité cumulative. Ce problème est une variation du problème classique de tournées de véhicules avec capacité. L'objectif principal des auteurs est de minimiser la somme des temps d'arrivée aux clients. Pour ce problème, l'heuristique *ALNS* a été utilisée.

D'une autre manière, la spécificité de l'*ALNS* est de choisir un certain nombre d'heuristiques de retrait et d'insertion afin d'intensifier et de diversifier la recherche et qu'elle peut être appliquée à plusieurs problèmes d'optimisation difficiles (*Pisinger et Ropke*, 2007).

4.1.3 La méthode ALNS développée par Pisinger et Ropke (2007)

Le travail de *Pisinger et Ropke* (2007) a mis en relief l'heuristique *ALNS* appliquée aux cinq variantes du problème de tournées de véhicules qui sont respectivement : le problème de tournées de véhicules avec fenêtres de temps, le problème de tournées de véhicules avec contrainte de capacité, le problème de tournées de véhicules avec multiples dépôts, le problème de tournées de véhicule avec dépendance de site et le problème de tournées ouvertes de véhicules.

Tous les problèmes cités ci-dessus ont été résolus à l'aide de la méthode *ALNS*. Parmi les avantages de cette méthode : l'obtention des solutions de meilleure qualité ainsi que la robustesse de son algorithme (*Pisinger et Ropke*, 2007). De même, cette méthode peut être appliquée à une large classe de problèmes d'optimisation difficiles.

La figure 4-1 montre le pseudo-code de l'algorithme *ALNS*. D'abord, on présente les paramètres de l'algorithme, par la suite on analyse l'algorithme *ALNS* général développé par *Pisinger et Ropke* (2007).

La notation utilisée est la suivante :

x : Solution réalisable.

x' : Solution après modifications (destruction & amélioration), solution temporaire.

x^* : Solution optimale.

π_i : Score de l'heuristique i utilisé (performance passée de l'heuristique i).

N^- : Voisinage de destruction développé.

N^+ : Voisinage de réparation développé.

$f(x)$: La valeur de la fonction objectif de la solution x.

Recherche Adaptative à Large Voisinage

1. Construire une solution faisable x ; soit $x^* := x$
2. Répéter
3. Choisir un voisinage de destruction N^- et un voisinage de réparation N^+ à l'aide du principe de *roulette* basé sur les scores obtenus précédemment $\{\pi_i\}$
4. Générer une nouvelle solution x' en utilisant les heuristiques correspondantes aux voisinages de destruction et de réparation choisis
5. Si x' peut être accepté alors soit $x := x'$
6. Mettre à jour les scores π_i du N^- et N^+
7. Si $f(x) < f(x^*)$ soit $x^* := x$
8. Jusqu'à la rencontre du critère d'arrêt
9. Retourner x^*

Figure 4-1 : L'algorithme général d'ALNS (Pisinger and Ropke, 2007)

La ligne 1 : Il est primordial de donner une solution initiale faisable x au problème, à cette étape là, la solution x est la solution optimale x^*.

De la ligne 2 à la ligne 8, il s'agit d'une boucle principale au problème : répéter les étapes à chaque itération jusqu'à ce qu'on satisfasse le critère d'arrêt. Dans la ligne 3, on choisit une heuristique de destruction ce qui génère un voisinage N^- puis une heuristique de réparation qui génère un voisinage N^+. Le choix dans cette étape se base sur le mécanisme de *roulette* qui est une probabilité de choisir une heuristique (de destruction ou de réparation) j en fonction des scores passés de l'heuristique i (π_i) parmi l'ensemble de ω heuristiques.

Après la destruction et l'amélioration qui consistent à une modification de la solution initiale, une nouvelle solution x' est générée (ligne 4). À la ligne 5, on décide si on va accepter la solution x', si cette dernière est une solution faisable alors x' sera notre solution actuelle. À ce niveau, il est nécessaire de mettre en lumière le critère d'acceptation d'une solution proposée par les auteurs. En effet, on accepte la solution x' étant donnée la solution actuelle x suivant une probabilité :

$$\boxed{e^{-(f(x')-f(x))/T}}$$

où T est un paramètre appelé *température*. Les auteurs utilisent un taux de refroidissement exponentiel standard et ils commencent par un taux de départ T_{start} qui est calculé en inspectant la solution initiale. Ce taux diminue en fonction de l'expression $T = T*c$ avec taux de refroidissement, $0 < c < 1$.

Une mise à jour des scores π_j des heuristiques de destruction et de réparation est faite à l'étape 6. À ce niveau-là, durant les M itérations, le score est réinitialisé et les probabilités de choisir une heuristique (destruction/amélioration) sont recalculées. Les probabilités de choix des heuristiques peuvent aussi être calculées comme la somme pondérée des scores durant les M itérations. Le score π_j est utilisé dans la formulation du principe de *roulette*.

Principe de *Roulette*: $\dfrac{\pi_{j_j}}{\sum_{i=1}^{\omega} \pi_i}$

À la ligne 7, si la valeur de la fonction objectif de x est meilleure que la valeur de la fonction objectif de x^* alors x sera la solution optimale.

Pisinger et Ropke (2007) poursuivent leur recherche par l'énumération de plusieurs opérateurs de retrait et d'autres d'insertion. Premièrement, on cite les opérateurs de retrait suivants:

▪ **Retrait aléatoire** : C'est une heuristique simple dont le principe est de sélectionner *q* requêtes aléatoirement et de les retirer de la solution.

▪ **Pire Retrait** (*Worst removal*) : Pour *Pisinger et Ropke* (2007), l'heuristique du pire retrait consiste à choisir un nombre de requêtes qui sont très couteuses ou bien celles qui endommagent la structure de la solution *x*.

On suppose une solution *x* et une requête *i*. On définit $f'(x, i)$ comme le coût de la solution après le retrait de *i*. On définit aussi $\Delta f_{-i} = f(x) - f'(x, i)$.

L'heuristique pire retrait choisit à chaque itération une nouvelle requête *i* ayant une large valeur de Δf_{-i} jusqu'au retrait de toutes les *q* requêtes. L'heuristique de retrait est aléatoire et elle est contrôlée par un paramètre aléatoire ρ qui détermine le degré du hasard dans l'heuristique du fait qu'il introduit un peu du hasard dans les choix des requêtes.

▪ **Retrait de Shaw** (*Related removal*) : Cette méthode de retrait peut prendre plusieurs appellations telles que : retrait de Shaw, retrait selon la distance et retrait en fonction de la mesure de « *relatedness* ». En effet, une mesure doit être fixée pour permettre une meilleure insertion dans le but d'obtenir une meilleure planification. L'algorithme se base sur une mesure de liaison entre deux requêtes appelée « *relatedness* » r_{ij}. Cette mesure est définie comme la distance entre deux requêtes *i* et *j* et elle est utilisée pour le retrait des sommets. Il est à noter que cette heuristique sélectionne au départ une requête *i* au hasard. Chaque requête *i* est formée d'un nœud de collecte *i* et d'un nœud de livraison *i + n*. La formule du « *relatedness* » r_{ij} est la suivante :

$$r_{ij} = \frac{1}{D}(d'(i, j) + d'(i, j+n) + d'(i+n, j) + d'(i+n, j+n))$$

Avec $d'(u, v)$ est la mesure de la distance entre deux nœuds, il est défini comme suit ;

$$d'(u,v) = \begin{cases} d_{uv} & \text{si } u \text{ et } v \text{ ne sont pas localisés dans un terminal} \\ 0 & \text{si } u \text{ et } v \text{ sont localisés dans un terminal} \end{cases}$$

Le dénominateur *D* est non nul et *D* peut être, par exemple, le nombre de sites de collecte et de livraison différents du terminal. Le processus commence par une sélection la requête *i* au hasard.

Le processus prend fin lorsque les *q* requêtes seront choisies, tout en choisissant les requêtes les plus liées (en termes de distance). L'algorithme se base sur un paramètre de contrôle du processus aléatoire ρ le même que celui dans l'algorithme du pire retrait (cité ci-dessus).

▪ **Retrait d'un groupe** : Dans cette heuristique, l'opérateur de retrait de groupe consiste à retirer des groupes de requêtes consécutives à partir de quelques tournées. L'objectif est de retirer un groupe entier, parmi les groupes formant les requêtes d'une tournée, à la fois. Il s'agit d'une variation de l'heuristique de retrait de Shaw, d'ailleurs on peut utiliser le même algorithme que l'heuristique de retrait de Shaw avec une petite modification. La définition des groupes se base sur l'algorithme de *Kruskal* pour le problème de recouvrement à poids minimal (*Pisinger et Ropke*, 2007). Le principe de cet algorithme est de choisir d'abord un groupe au hasard, puis on retire les requêtes se trouvant dans le groupe choisi.

▪ **Retrait basé sur le temps** (*Time-oriented removal*) : Il s'agit d'une dérivée de l'heuristique de retrait de Shaw basée sur la variable de temps. Le principe de cette heuristique est de retirer les requêtes qui sont desservies à peu près au même temps. Les auteurs ont défini une distance de temps entre deux requêtes *i* et *j* (Δt_{ij}).

$$\Delta t_{ij} = \left| t_{pi} - t_{pj} \right| + \left| t_{di} - t_{dj} \right|$$

Avec : t_{pi} et t_{di} qui sont respectivement le temps de collecte et le temps de livraison de la requête *i* dans la solution actuelle. Le processus commence par une sélection au hasard d'une requête *ř* et les *B* requêtes proches de *ř* en fonction de la distance r_{ij} définie par l'heuristique retrait de Shaw. Au début, on choisit un sous ensemble de requêtes qui sont géographiquement proches. Parmi les *B* requêtes sélectionnées, on sélectionne (*q-1)* requêtes proches à *ř* selon Δt_{ij} . Ce processus est contrôlé par un paramètre aléatoire *ρ* comme l'heuristique de retrait de Shaw.

▪ **Retrait des paires de nœuds historique** (*Historical node-pair removal*) : Un autre nom de cette heuristique est « *neighbor graph removal heuristic* ». Le principe de cette heuristique est qu'à chaque paire de nœuds (*u, v*) ∈ A, on associe un poids $f^*_{(u,v)}$ qui indique la valeur de la meilleure solution parmi les solutions qui utilisent l'arête (*u, v*). Au début de l'algorithme, le poids $f^*_{(u,v)}$ de toutes les arêtes de la solution est égal à l'infini puis il est mis à jour à chaque fois

qu'on trouve une nouvelle solution où les poids des arêtes (u, v) peuvent être améliorés. Ces poids peuvent être utilisés pour retirer des requêtes mal placées. L'heuristique simple de retrait calcule le coût d'une requête (i, $i+n$) dans la solution actuelle par la sommation des poids des arêtes incidentes à i et $i+n$. On retire la requête la plus coûteuse et le processus est répétitif et il s'arrête lorsque les q requêtes seront retirés.

▪ **Retrait des paires de requêtes historique** (*Historical request-pair removal*) : Cette heuristique peut prendre le nom de « *request graph removal* ». Les auteurs ont introduit un poids $h_{(a,b)}$ pour chaque paire de requêtes $(a, b) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$. Ce poids indique le nombre de fois où les requêtes a et b sont servies par le même véhicule dans les B meilleures solutions uniques observées durant la recherche. Initialement, le poids $h_{(a, b)}$ est mis à zéro, à chaque fois une nouvelle solution unique est observée. Dans cette heuristique, les poids sont incrémentés ou décrémentés en fonction des solutions entrantes et sortantes des solutions de la liste top-B. La valeur de B a été fixée selon l'expérimentation à 100.

Deuxièmement, on cite les heuristiques d'insertion qui sont principalement : insertion à moindre coût (*Basic greedy heuristic*) et insertion avec regrets (*Regret heuristic*).

▪ **Insertion à moindre coût** (*Basic greedy heuristic*) **:** Comme son nom l'indique, il s'agit d'une heuristique simple où on insère dans la solution les requêtes déjà retirées avec un minimum de coût. En effet, l'emplacement de la nouvelle requête a pour objectif la minimisation du coût additionnel à la solution tout en tenant compte des contraintes du problème (capacité, fenêtre de temps, etc.). Il s'agit en d'autres termes d'insérer la requête i à l'endroit le moins cher de la tournée k. L'heuristique se base sur la formulation suivante :

$$\boxed{(i, k) := \arg \min_{i \in U,\, k \in R} \Delta f_{i,k}}$$

Avec :

k : nombre de tournées partielles, $k \in R$.

U : nombre de requêtes non placées dans la solution.

$\Delta f_{i,k}$: différence de la valeur de la fonction objectif avant insertion et après insertion de la requête i dans la position la moins chère dans la tournée k.

On insère la requête *i* à l'endroit introduisant le coût le moins faible jusqu'à qu'on termine l'insertion des *q* requêtes retirées.

▪ **Insertion avec regrets** (*Regret heuristic*) : D'une manière générale, plus la différence de coût (valeur de la fonction objectif) est grande plus il est favorable d'insérer *i* dans sa meilleure position dans la solution le plus tôt possible sinon on risque de subir un coût d'insertion plus cher plus tard. Par ailleurs, à chaque itération, l'heuristique avec regret choisit d'insérer la requête *i* dans sa position la moindre coûteuse en fonction de la formule suivante :

$$\boxed{i := \arg\max_{i \in U} \left(\sum_{h=2}^{q} \Delta f_i^h - \Delta f_i^1 \right)}$$

Avec ;

Δf_i^h : Changement de la valeur de la fonction objectif avec l'insertion de la requête *i* dans la $h^{ième}$ meilleure position la moins chère.

4.1.4 Modifications de Laporte et al. (2010)

Laporte et al. (2010) ont appliqué la méthode *ALNS* au problème des tournées sur les arcs avec une contrainte de capacité avec une demande stochastique dans le contexte de collecte des ordures. Le principe de cette méthode est de sélectionner *q* arêtes qui vont être retirées de la solution avec l'heuristique de retrait puis en un second temps les réinsérer avec l'heuristique d'insertion. À chaque itération, le choix d'un type d'heuristique est basé sur le principe de *roulette* comme dans le travail de *Pisinger et Ropke* (2007). La spécificité de ce travail consiste en une pénalisation de la fonction objectif pour faire face aux *m* solutions non faisables.

Les auteurs ont développé l'algorithme d'*ALNS* associé à ce problème, ils ont fait appel à d'autres algorithmes tels que « *Stochastic Path Scanning* » et l'algorithme *RRT* «*Record-to-Record Travel* ». L'algorithme *ALNS* est présenté dans la figure 4-2 ci-dessous.

On commence en premier lieu par la description des paramètres de cet algorithme :

τ : une solution réalisable.

τ' : un voisinage de la solution τ.

τ_{feas} : la meilleure solution du problème.

f (τ') : la valeur de la solution *τ'*.

ρ : coefficient auto-ajustable de pénalité « *self-adjusting penalty coefficient* », ce coefficient est égal à 1 à l'état initial, puis à chaque 10 itérations il est multiplié par $2^{[(b/5)-1]}$ avec *b* = le nombre de *m* solutions infaisables trouvées durant les 10 dernières solutions.

Q : segment composé d'un certain nombre d'itérations; *Q*=50 itérations.

record : La valeur de la fonction objectif *f** de la meilleure solution actuelle par l'algorithme *RRT* (*Record-to-Record Travel*).

deviation : un paramètre positif unique de l'algorithme *RRT* δ, avec δ = 0.01 *f**.

$\omega_{i,j}$: le poids de l'heuristique *i* sur le segment *j*; le poids de toutes les heuristiques au départ est égal à 1 puis il subit des mises à jour.

$\pi_{i,j}$: score de l'heuristique *i* obtenu au segment *j*. Au départ de chaque *Q* itérations (segment), les scores des heuristiques sont mis à 0. Par la suite, le score de l'heuristique augmente par σ_1, σ_2 ou σ_3 ; soit par 30, 10 ou 6 respectivement et cela dépendamment de la solution, entre autres σ_1 : si la solution produite après l'opération de retrait-insertion est une nouvelle meilleure solution faisable, σ_2 : si la solution produite améliore la solution courante et σ_3 : si la solution produite est pire que la solution courante mais elle est acceptée.

$\theta_{i,j}$: nombre de fois que l'heuristique *i* a été utilisé au segment *j*.

La solution initiale utilisée par l'algorithme *ALNS* a été développée par l'algorithme « *Stochastic Path Scanning* ». Cet algorithme est une version de la méthode « *Path Scanning* » proposée par *Golden et al.* (1983).

Pour le critère d'acceptation appelé encore critère de *RRT* d'une solution, il est basé sur la relation suivante : on accepte la solution *τ'* si $f(\tau') < f^* + \delta$. De plus, si $f(\tau') < f^*$, alors une mise à jour de la valeur de *f** est faite. Tandis que pour les critères d'arrêt, le processus prend fin si la solution ne s'améliore pas durant une longue période ou bien durant un nombre d'itérations fixé dès le début.

En second lieu, les auteurs ont donné une description des deux heuristiques H^- et H^+ qui vont modifier la solution actuelle et générant une nouvelle et qui sont respectivement : heuristique de retrait et heuristique d'insertion.

Initialiser le coefficient d'auto-ajustement de pénalité ρ, les poids de toutes les heuristiques de retrait et d'insertion et leurs scores pour le premier segment ;

Soit *feas* = faux ;

Générer une solution initiale τ avec « *Stochastic Path Scanning* » ;

Soit *record* = $f(\tau)$ et initialiser *deviation* ;

Si τ est faisable

Soient *feas* = *vrai* et $\tau_{feas} = \tau$;

Répéter

Choisir une heuristique de retrait H^- et une heuristique d'insertion $H+$ à l'aide du *principe de roulette* basé sur les poids du *segment* courant ;

Générer une nouvelle solution τ' à partir de τ en utilisant H^- et H^+ ;

Si τ' est faisable

Si *feas* = *faux*

Soit $\tau_{feas} = \tau'$ et *feas* = *vrai ;*

Sinon

Si $f(\tau') < f(\tau_{feas})$

Soit $\tau_{feas} = \tau'$;

Si τ' est accepté par le critère RRT

Soit $\tau = \tau'$;

Si $f(\tau') <$ record

Mettre à jour *record* et *deviation* ;

Mettre à jour les données de H^- et H^+ (scores, nombre de fois on les a utilisé) ;

Mettre à jour ρ si nécessaire (itération courante est mise à jour) ;

Si on arrive à la fin du *segment*, mettre à jour les poids de toutes les heuristiques de retrait et d'insertion et réinitialiser leurs scores pour le segment suivant ;

Jusqu'à la rencontre du critère d'arrêt ;

Si *feas* = *vrai*

retourner τ_{feas} ;

sinon

Aucune solution admissible identifiée.

Figure 4-2 : Algorithme ALNS, Laporte et al. (2010)

D'abord, *Laporte et al.* (2010) ont proposé 4 types d'heuristiques de retrait :

▪ **Pire retrait déterministe** (*Deterministic Worst Removal*): il s'agit d'une autre version de l'heuristique du pire retrait (*Worst Removal*) développée par *Pisinger et Ropke* (2007). Les deux heuristiques ont les mêmes principes de base.

▪ **Retrait aléatoire révisé** (*Revised Random Removal*) : cette heuristique mélange deux heuristiques de retrait en même temps qui sont l'heuristique de retrait aléatoire et l'heuristique pire déterministe. L'algorithme fonctionne en 3 étapes : la première consiste à sélectionner un nombre de requêtes à retirer $\vartheta = q$. L'étape 2 se concentre sur le retrait de l'arête avec un minimum de différence de coût avant retrait et après retrait. À la fin, le processus de termine s'il ne reste pas d'arêtes à retirer $\vartheta = 0$.

▪ **Pire retrait probabiliste** (*Probabilistic Worst Removal*) : l'objectif de cette heuristique de retrait est de choisir une arête « critique » desservie et qui semble être mal placée dans la solution courante puis la retirer.

▪ ***Towards-feasibility removal*:** l'objectif de cette heuristique est de se diriger vers une solution optimale ou bien de la maintenir par le retrait des arêtes utilisées par la route.

Ensuite, il est nécessaire de choisir une heuristique d'amélioration. En effet, et par rapport aux heuristiques proposées par *Pisinger et Ropke* (2007), *Laporte et al.* (2010) ont développé d'autres heuristiques qui sont citées ci-dessous :

▪ **Insertion gloutonne déterministe** (*Deterministic Greedy Insertion*) : cette heuristique a le même principe que l'heuristique d'insertion à moindre coût (*Basic greedy heuristic*) proposée par *Pisinger et Ropke* (2007). Cette heuristique est formée de deux étapes : la première consiste à une insertion des requêtes dans la solution si le coût d'insertion est nul. Il s'agit d'un test pour savoir si on est bien dans le chemin le plus court. Tandis qu'à la deuxième étape, on effectue des calculs qui portent sur la minimisation des coûts d'insertion; une mise à jour est faite après chaque insertion. Ce processus prend fin lorsqu'on termine l'insertion des χ arêtes.

▪ ***Towards-Feasibility Insertion*** : l'objectif de cette heuristique est d'insérer les arêtes dans le but de se diriger vers une solution optimale ou bien de la maintenir. Deux situations sont

possibles, tout en comparant *m* et *m(τ)* qui sont respectivement le nombre de tournées et le nombre de tournée dans la solution actuelle τ ($m \leq m(\tau)$ ou bien $m > m(\tau)$).

▪ **Insertion par tri** (*Sorting Insertion*) : il s'agit de trier la liste des arêtes retirées χ dans un ordre décroissant en fonction des valeurs de la demande sur les arêtes. De même, les auteurs trient la liste des tournées en fonction de leurs probabilités d'échec dans un ordre décroissant. La position de l'arête dans la tournée est sélectionnée si elle permet la minimisation des coûts. Après chaque insertion, la liste des tournées est mise à jour.

4.2 La méthode ALNS appliquée au problème d'optimisation des mouvements des conteneurs :

Dans cette section, on applique la méthode *ALNS* à notre problème en se basant sur les travaux de la littérature de *Pisinger* et *Ropke* (2007), Laporte et al. (2010) ainsi que d'autres.

4.2.1 Conception de la méthode ALNS

On commence d'abord par un exemple illustrant le principe de l'heuristique *ALNS*. Ainsi, la figure 4-3 montre une représentation de la méthode de destruction et la méthode de réparation. Soit une solution initiale illustrée en 4-3 *a*. Le dessin en 4-3 *b* représente la solution après l'opération de destruction où on a supprimé aléatoirement les trois conteneurs C4, C6 et C10 de la solution initiale et le dessin en 4-3 *c* montre la solution après l'opération de réparation.

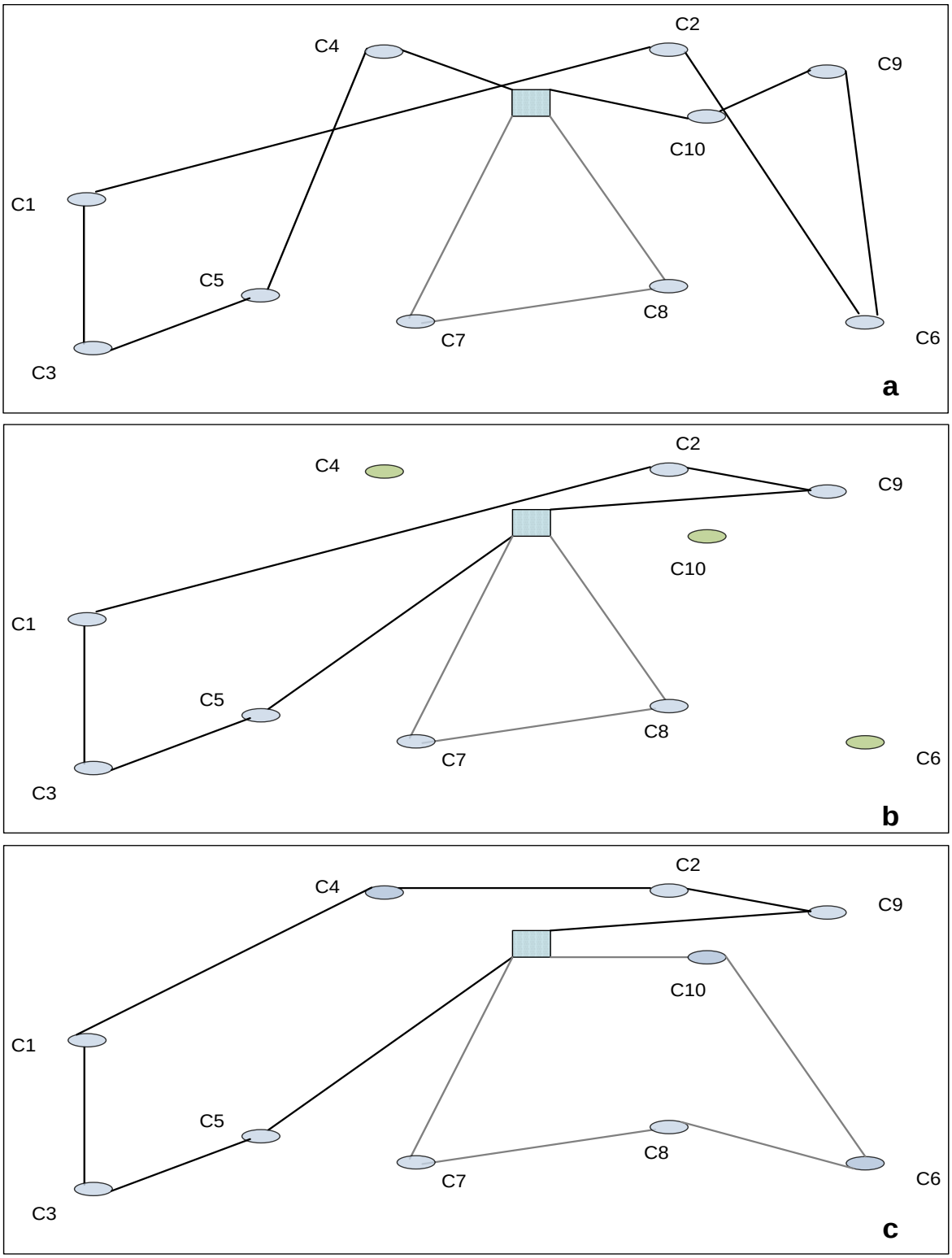

Figure 4-3 : Exemple de destruction et réparation dans la méthode ALNS

Une présentation simplifiée de la méthode proposée comporte trois niveaux, comme le montre la Figure 4-4, et chacun d'eux est développé sous forme d'un ou plusieurs algorithmes. Le premier niveau relatif à l'algorithme 1 où on a utilisé la méthode d'insertion à moindre coût qui s'occupe de créer une solution initiale faisable à notre problème. Le deuxième et le troisième niveau sont relatifs, respectivement, à un ensemble d'algorithmes de destruction et d'amélioration, à chaque fois le processus est basé sur le principe de *roulette*. À la fin, une nouvelle solution est générée. Cette solution va être acceptée ou rejetée selon des critères d'acceptation Ce processus va se répéter jusqu'on rencontre un critère d'arrêt.

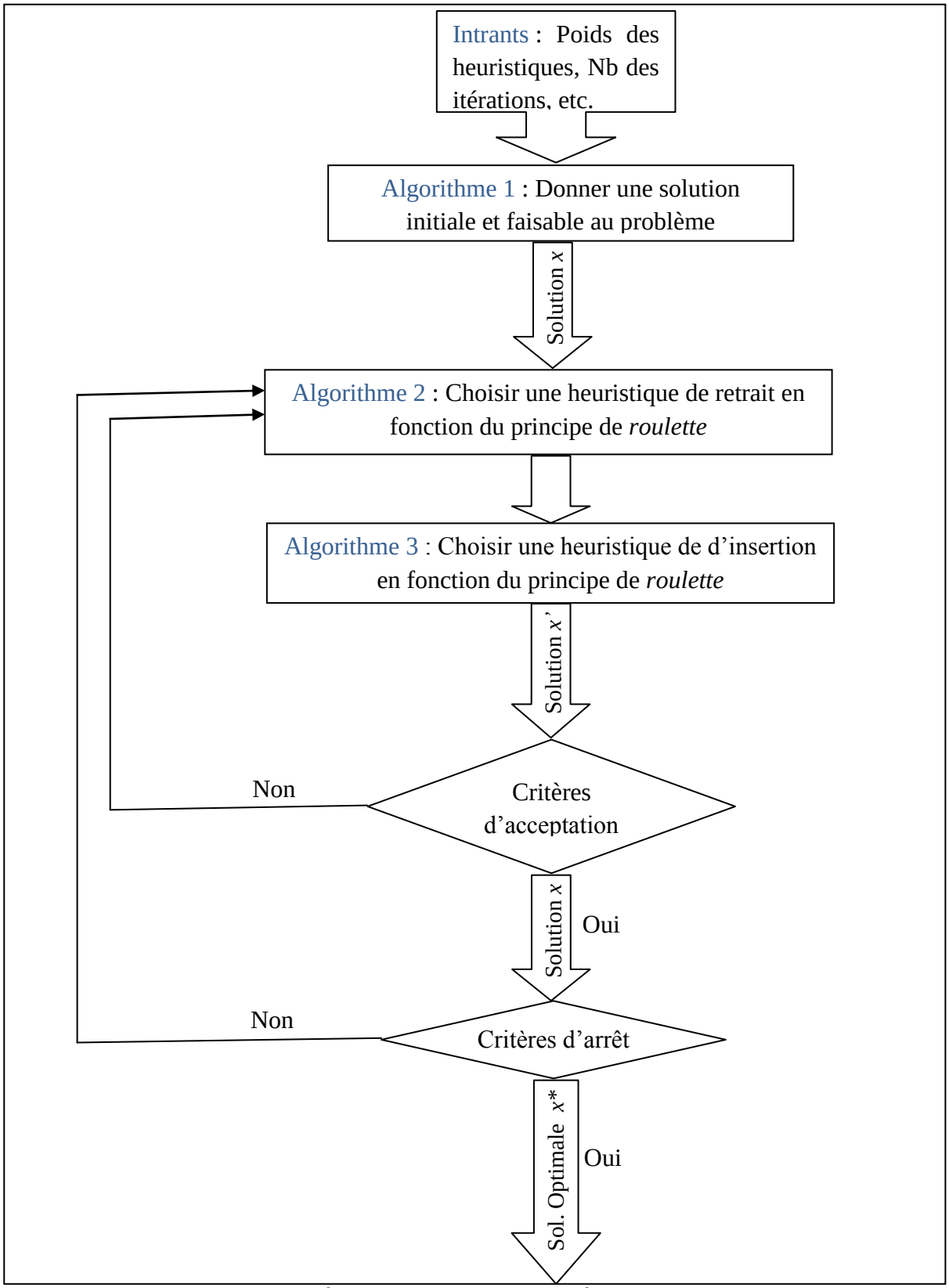

Figure 4-4 : Conception de la méthode heuristique proposée

4.2.2 Pseudo-code de l'algorithme

Dans cette partie, le pseudo-code général du fonctionnement de la méthode heuristique est présenté.

Étape 0: Intrants : solution initiale faisable *x*, initialisation des poids des heuristiques $\omega_i^- = 1$ et $\omega_i^+ = 1$ $(\forall i)$

Étape 1: $x^* = x$

Étape 2 : Répéter

Étape 3 : Choisir heuristique de destruction H^- et heuristique d'insertion H^+ avec le principe de *roulette*

Étape 4 : Une nouvelle solution générée x' ; $H^+(H^-(x)) = x'$

Étape 5 : **Si** x' est acceptée en utilisant critères d'acceptation

alors $x = x'$

Étape 6 : **Si** $f(x') < f(x)$ **alors** $x^* = x$

Étape 7 : Mise à jour des poids des heuristiques ω_i^- et ω_i^+

Étape 8 : Jusqu'à un critère d'arrêt est rencontré

Étape 9 : Retourner x^*

- Les paramètres de l'algorithme général sont les suivants :

x : solution initiale faisable pour le problème.

x' : solution générée après la destruction et la réparation.

*x** : solution optimale.

f(x) : valeur de la fonction objectif de la solution *x* (valeur du makespan).

H^- et H^+ sont respectivement les heuristiques de destruction et d'amélioration et elles sont choisies en fonction du principe de *roulette*.

Principe de *roulette*: Soit une heuristique *i* avec un poids passé ω_i et nous avons *H* heuristiques, on choisit l'heuristique *j* avec une probabilité :

$$\frac{\omega_j}{\sum_{i=1}^{H} \omega_i}$$

Le principe de roulette s'applique pour les heuristiques de destruction et d'amélioration (H^- et H^+) dont leurs poids sont respectivement ω_i^- et ω_i^+. Ces poids sont initialisés à 1 puis mis à jour à chaque segment d'itérations.

4.2.3 La solution initiale

La méthode *ALNS* nécessite une solution initiale pour son fonctionnement. La création d'une solution initiale est faite à l'aide de l'algorithme classique d'insertion à moindre coût dont le principe consiste à insérer à chaque itération un conteneur qui minimise le temps de complétion de l'opération de manutention. L'algorithme essaye de trouver le meilleur parcours en testant les différentes configurations possibles en donnant par la suite une solution réalisable. Il n'est pas nécessaire que le temps de complétion de la solution initiale soit minimum, car cette solution sera par la suite améliorée à l'aide de la méthode *ALNS*. Dans la solution initiale, il faut qu'on vérifie une condition nécessaire, si deux conteneurs sont traités par le même portique de cour dans un ordre relatif et ils sont transportés par le même camion alors le même ordre relatif doit être respecté dans la phase de transport aussi, comme le montre l'exemple suivant :

YC_i	C1	C8	**C2**	**C5**	C7
YT_j	**C2**	C6	**C5**	C10	C9

4.2.4 Mécanismes de retrait et d'insertion

i. Les opérateurs de destruction

L'objectif de ces opérateurs est de détruire la solution actuelle par le retrait d'un certain nombre d'éléments. Pour cerner la diversité des heuristiques de destruction (retrait) présentées dans plusieurs travaux qui abordent les mécanismes de destruction d'une solution actuelle, on va s'intéresser à quelques opérateurs jugés pertinents pour nous.

- **Retrait Aléatoire :** Le principe de l'heuristique du retrait aléatoire (*Random Removal*) consiste à choisir un ensemble de conteneurs d'une façon aléatoire et qui vont être retirés par la

suite de la liste de tous les conteneurs de la solution actuelle. Ce type d'heuristique se base sur un mode aléatoire. D'une façon spécifique, il s'agit de sélectionner un nombre q de conteneurs puis de les retirer de la solution x.

Dans le retrait aléatoire, on va présenter deux méthodes : un retrait séparé des conteneurs et un retrait de groupe de conteneurs.

-**Retrait aléatoire séparé de conteneurs** : Le principe de cette heuristique est de sélectionner au hasard q conteneurs de la solution initiale (avec $q \geq 1$). Les q conteneurs sélectionnés pour le retrait ne sont pas successifs.

- **Retrait aléatoire d'un groupe de conteneurs :** Le principe de cette heuristique de retrait consiste à retirer des groupes de conteneurs reliés à partir de quelques itinéraires des portiques de cour. L'objectif est de retirer un groupe entier à la fois formé de q conteneurs avec $q \geq 2$.

▪ **Pire retrait :** Cette heuristique consiste à choisir un nombre de conteneurs qui sont très coûteux dans la solution x. On définit $f'(x, i)$ comme le coût de la solution après le retrait du conteneur i. Soit $\Delta f_{-i} = f(x) - f'(x, i)$, la différence de la valeur de la fonction objectif avant retrait et après retrait du conteneur i. On choisit à chaque itération un nouveau conteneur i ayant une large valeur de Δf_{-i} jusqu'au retrait de toutes les q conteneurs.

ii. Les opérateurs d'insertion

▪ **Insertion à moindre coût :** Après la destruction de la solution, l'heuristique d'insertion vient pour la repérer. Il s'agit d'une heuristique simple où on va insérer dans la solution les conteneurs déjà retirés un après l'autre dans l'endroit introduisant le coût le moins faible. L'algorithme effectue des calculs qui portent sur la minimisation des coûts d'insertion, après chaque insertion. Plus précisément, il s'agit d'une minimisation de la différence de la valeur de la fonction objectif avant et après insertion puis d'insérer le conteneur dans la position la moins chère. Ce processus prend fin jusqu'à on termine l'insertion des q conteneurs retirés.

▪ **Insertion avec regrets :** La méthode d'insertion à moindre coût nous ne permet pas de savoir l'impact de l'insertion d'un conteneur sur les autres conteneurs à insérer au plus tard. L'insertion avec regret vient pour remédier à cette difficulté. En effet, le principe de cette méthode est de maximiser la différence de la valeur de la fonction objectif (makespan) entre la

première meilleure position et la deuxième meilleure position. D'une manière générale, plus la différence de la valeur de la fonction objectif est grande plus il est favorable d'insérer *i* dans sa meilleure position dans la solution le plus tôt possible sinon on risque de subir un coût d'insertion plus cher plus tard. Le processus prend fin lorsqu'on termine l'insertion des *q* conteneurs retirées.

Les algorithmes d'insertion ont été codifiés prenant compte les ordres de priorité dans le traitement de conteneurs qui doivent être respectés. De même, les heuristiques d'insertion doivent satisfaire la condition d'ordre développée dans le paragraphe 4.2.3 dans le but d'avoir une solution finale faisable.

4.2.5 Critères d'acceptation et d'arrêt

Le critère le plus utilisé dans la littérature et qui permet d'accepter la solution générée *x'* comme une solution actuelle *x* du problème est à la base de la méthode du *Recuit Simulé*. L'acceptation se fait suivant une probabilité. $e^{-(f(x')-f(x))/T}$

avec :

x' : la nouvelle solution.

x : la solution actuelle.

$T > 0$: paramètre appelé température. Au début, $T = T_{star}$ avec $T_{star} = f(x).(1-w)$ où *x* est la solution de départ et *w* est un paramètre. Plus *w* est grand, plus la chance est grande d'accepter des solutions moins bonne que la solution de départ. Dans notre travail, on suppose que $w = 1.05$ (1.05 dans le travail de Azi et al., 2010). Le paramètre température *T* subit des mises à jour à chaque itération, *T* devient $T * c$ où $0 < c < 1$ est appelé le *taux de refroidissement*. La probabilité d'accepter une solution moins bonne devient faible vu que le paramètre *T* diminue avec le nombre des itérations. Dans l'exécution, on a supposé que $c = 0.98$ (0.99975 dans le travail de Azi et al ,2010).

Pour le critère d'arrêt, le processus prend fin lorsqu'on atteint un nombre d'itérations; $I = 100$.

4.2.6 Ajustement des poids et des scores

La recherche dans ce travail est devisée en segments d'itérations, chaque segment est égal à 50 itérations ($Q = 50$ itérations). La sélection d'une heuristique de retrait ou d'insertion est faite de

manière indépendante à l'aide du principe de *roulette*. Rappelons qu'il est important de choisir un mécanisme de sélection d'un opérateur de retrait *j* en se basant sur une probabilité calculée de la manière suivante :

$$\frac{\omega_j}{\sum_{i=1}^{H} \omega_i} \quad \text{où } H = \left| H^+ \cup H^- \right|$$

La sélection d'un opérateur d'insertion *j* est faite de la même manière. Avec ω_i est le poids de l'heuristique *i* et *H* est l'ensemble des heuristiques utilisées dans la recherche.

Toutefois, les poids des heuristiques de destruction et d'insertion sont initialisés à 1 puis ils sont mis à jour à chaque segment d'itérations *Q*. Cela veut dire que tous les opérateurs *i*, que ce soit de retrait ou d'insertion, ont le même poids au début de chaque segment $\omega_i = {}^{1}/_{|H|}$.

À la fin de chaque segment *Q*, les poids des heuristiques sont mis à jour à l'aide des scores δ obtenus dans le segment d'itérations précédent.

En se référant au travail de *Azi et al.* (2010), les poids au début d'un segment donné (*seg)* sont en fonction de ceux dans le segment précédent (*seg*-1). Un segment (*seg*) est formé de *t* itérations. La mise à jour des poids peut être calculée comme suit :

$$\omega_i^{seg} = r\, \omega_i^{seg-1} + (1-r)\delta_i^{seg-1}$$

où *r* est un paramètre arbitraire ($0 \leq r \leq 1$). Lors de l'exécution, on suppose $r = 0.1$ (*Laporte et al*., 2010) Les scores δ de toutes les heuristiques sont initialisés à 0 au début de chaque segment. Après l'utilisation de l'heuristique d'insertion *i* et l'obtention d'une nouvelle solution, les scores sont augmentés à chaque itération *t* sous les conditions suivantes : σ_1, σ_2 et σ_3

$$\delta_i^{t+1} = \delta_i^t + \begin{cases} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{cases}$$

Tels que :

$\sigma_1 = 15$, si la nouvelle solution produite est meilleure et globale.

$\sigma_2 = 10$, si la nouvelle solution produite améliore la solution courante.

$\sigma_3 = 6$, si la nouvelle solution produite n'améliore pas la solution courante mais elle est acceptée.

4.3 Tests et résultats

Tous les algorithmes de la méthode *ALNS* ont été développés en C/C++ avec Microsoft Visual C++ 2010 Express Edition et exécutés sur un ordinateur Intel(R) Core™ i7 CPU 950@3.07 GHz et avec 6.00 Go de mémoire. Rappelons que le problème de petite taille a été résolu avec CPLEX 12.2.0.0.

Au départ on fixe les paramètres à utiliser dans l'algorithme *ALNS* [w; c ; Q ; I; r ; σ_1 ; σ_2 ; σ_3] qui sont respectivement [1.05; 0.98; 50;100; 0.1; 15; 10; 6]. Notons que le nombre des itérations I et la taille du segment Q sont variables dépendamment de la taille du problème (I=10; Q = 2 pour les problèmes de petite taille et I=100; Q= 50 pour les problèmes de grande taille)

4.3.1 Présentations des instances tests

On a testé notre algorithme sur plusieurs instances. Pour cela, on a généré trois séries d'instances avec 10, 20 et 100 conteneurs. Le nombre des blocs et des portiques de cour est variable, soit 2 ou 4. On fait varier à chaque fois le nombre de blocs et/ou le nombre des équipements de manutention disponibles dans le port pour arriver enfin à 9 instances au total dont les caractéristiques sont décrites dans le tableau 4-1.

Tableau 4-1 : Caractéristiques des instances générées

Instances	**Nb de conteneurs *n***	**Nb de blocs**	**Équipements**	
			Nb des portiques de cour	**Nb des camions**
INS1	10	2	2	2
INS2	10	2	2	3
INS3	20	2	2	2
INS4	20	2	2	3
INS5	20	2	2	4
INS6	20	4	4	2
INS7	20	4	4	3
INS8	20	4	4	4
INS9	100	4	4	3

Vu que ce problème d'optimisation des mouvements des conteneurs, utilisant ce type d'heuristiques, n'a pas été traité dans la littérature dans le passé, il n'existe pas de données comparatives dans la littérature.

Le problème a été résolu au départ par AMPL studio et CPLEX 12.2.0.0., ce qui nous a permis d'effectuer une comparaison entre les méthodes pour les instances INS1 et INS2.

Tableau 4-2 : Comparaison entre les différentes méthodes de calcul étudiée pour les instances INS1 et INS2

Instances	INS1			INS2		
Méthodes	CPLEX	Heuristique pour la solution initiale	ALNS	CPLEX	Heuristique pour la solution initiale	ALNS
Valeur de la fonction objectif	860	980	860	560	650	560
Écart (%)	-	13.95	**0**	-	16.07	**0**

Dans le tableau 4-2, on a fait une comparaison entre les différentes méthodes utilisées à savoir CPLEX, heuristique de la solution initiale à notre problème (heuristique classique d'insertion au moindre coût qui nous permet d'avoir une solution qui servira de solution initiale pour la méthode heuristique *ALNS*) et la méthode *ALNS*. Pour les deux instances INS1 et INS2, On observe qu'il n'y a pas de variation de la valeur de la fonction objectif (temps de complétion) en utilisant CPLEX et *ALNS* et que la valeur de la solution est égale à 860 et 560 pour les instances INS1 et INS2 respectivement (l'écart est nul), il s'agit ici d'une meilleure solution. Tandis qu'on observe une augmentation de la valeur de la fonction objectif pour les instances INS1 et INS2 en comparant l'heuristique de la solution initiale avec CPLEX et cette augmentation est de 13.95 % et 16.07% pour les deux instances INS1 et INS2 respectivement.

4.3.2 Comparaison des scénarios

Cette section est consacrée à présenter différents scénarios générés par la combinaison d'un ou plusieurs heuristiques de retrait H_i^- et d'insertion H_j^+ afin de mesurer la performance de la méthode.

Soit la notification suivante des opérateurs de retrait/insertion :

H_1^- : Retirer un seul conteneur aléatoirement.

H_2^- : Retirer deux conteneurs séparés aléatoirement.

H_3^- : Retirer trois conteneurs séparés aléatoirement.

H_4^- : Retirer un groupe de deux conteneurs (une séquence de conteneurs).

H_5^- : Retirer un groupe de trois conteneurs (une séquence de conteneurs).

H_6^- : Retrait pire de trois conteneurs.

H_1^+ : Insertion à moindre coût.

H_2^+ : Insertion avec regret.

Cette combinaison nous a permis de créer un ensemble de scénarios qu'on va tester au départ sur des petites instances dont le nombre de conteneurs est de 10 et 20. Vu la petite taille des instances, dans cette partie, le nombre d'itérations est fixé à 10 et le nombre d'exécutions est égal à 10. L'objectif est de comparer les différents scénarios pour une instance donnée. Les scénarios suivants ont été générés :

Scénario 1: H_1^-& H_1^+

Scénario 2 : H_2^- & H_1^+

Scénario 3: H_3^- & H_1^+

Scénario 4: H_4^- & H_1^+

Scénario 5: H_5^- & H_1^+

Scénario 6: H_2^- H_4^- & H_1^+

Scénario 7: H_1^-& H_2^+

Scénario 8: H_2^-& H_2^+

Scénario 9: H_3^-& H_2^+

Scénario 10: H_2^- & H_4^-& H_2^+

Scénario 11: H_6^- & H_1^+ & H_2^+

Scénario 12: $Tous$. Ce dernier regroupe trois opérateurs de retrait et deux opérateurs d'insertion. Les opérateurs de retrait sont le retrait de q conteneurs séparés aléatoirement, le retrait de groupe de q conteneurs aléatoirement et le retrait pire de q conteneurs. On fixe pour ce scénario q = 3. Les deux opérateurs d'insertion sont l'insertion au moindre coût et l'insertion avec regret. Ce regroupement de 3 H^- et de 2 H^+ forme la méthode *ALNS* avec un maximum d'heuristiques à utiliser en même temps. Le scénario 12 nous permet d'évaluer la performance de la méthode *ALNS*.

L'Algorithme *ALNS* a été exécuté 10 fois pour chaque instance de petite taille (10 et 20 conteneurs) avec un total de 20 itérations par exécution et un segment est composé de 2 itérations. Durant ces exécutions, on détermine la moyenne des solutions pour chaque scénario, ainsi que le temps d'exécution moyen. On conserve aussi la meilleure solution observée dans les différents scénarios pour chaque instance. Ces paramètres permettent de déterminer la qualité de la solution, le temps d'exécution T (en seconde) et le temps T^* (en seconde) qui indique le moment où l'algorithme a trouvé sa meilleure solution. La qualité de la solution pour chaque scénario est définie comme étant l'écart entre la solution moyenne et la meilleure solution et elle est exprimée en pourcentage en utilisant la formule suivante :

$$\frac{moyenne\ des\ solutions - meilleure\ solution}{meilleure\ solution} * 100$$

Un meilleur scénario est donc celui qui possède la valeur de la qualité de solution la plus faible et qui est plus proche de zéro (la moyenne des solutions dans les 10 exécutions et la meilleure solution trouvée dans les différents scénarios sont égales).

Quatre instances sont sélectionnées (INS2, INS4, INS6 et INS7) afin d'examiner l'efficacité de la méthode proposée et cela par la comparaison, principalement, de la qualité de la solution et du temps d'exécution T.

Les tableaux 4-3, 4-4, 4-5 et 4-6 regroupent les résultats. La première colonne présente les différents scénarios examinés et qui sont générés suite à une combinaison des opérateurs de retrait et d'insertion. La deuxième indique la valeur de la qualité de solution (%). La troisième colonne indique le temps d'exécution moyen T exprimé en secondes. La dernière colonne contient le temps T^* exprimé en secondes.

Tableau 4-3 : Qualité de la solution et le temps d'exécution pour l'instance INS2

Scénarios	Qualité[1] (%)	Temps d'exécution T (sec)	Temps T* (sec)
Scénario 1	5,59	2.70	1.40
Scénario 2	5.20	3.45	1.87
Scénario 3	2.91	10.71	4.83
Scénario 4	5.32	3.49	0.72
Scénario 5	2.79	4.69	2.64
Scénario 6	2.16	5.32	2.57
Scénario 7	4.73	2.58	1.01
Scénario 8	2.39	6.93	4.33
Scénario 9	3.96	10.35	5.72
Scénario 10	1.48	8.72	3.87
Scénario 11	0.79	11.73	4.52
Scénario 12 (ALNS)	**0.00**	11.32	6.75

À la lumière des résultats obtenus pour chaque scénario dans le cas de l'instance INS2 (tableau 4-3), nous pouvons conclure que le scénario 12 regroupant les différents opérateurs de retrait et d'insertion obtient les solutions de meilleure qualité (≈ 0%) même s'il a un temps d'exécution un peu élevé par rapport autres scénarios (11.32 secondes). Ce temps, plus au moins élevé,

[1] L'écart avec la solution optimale

s’explique par le nombre important d’algorithmes qui sont traités en même temps. Si on compare la stratégie 11 et la stratégie 12, la stratégie 11 est moins bonne en termes de qualité de solution générée (0.79%) et en termes du temps d’exécution (11.73 secondes) comparée à la stratégie 12 (11.32.secondes). Le scénario le plus pire, en termes de qualité de solution générée, est le scénario 1 (5.59 %).

Le tableau 4-4 rapporte les résultats sur l’instance INS4. En comparant les scénarios, on trouve que l’algorithme *ALNS* (stratégie 12) requiert un temps moyen d’exécution de 43.74 secondes pour un écart moyen de 1.63%. Si on compare les scénarios 1, 2 et 3, où on a utilisé l’algorithme d’insertion à moindre coût comme l’opérateur d’amélioration, et les scénarios 7, 8 et 9, où on a utilisé l’algorithme d’insertion avec regret comme étant l’opérateur d’amélioration, on trouve que l’algorithme d’insertion avec regret permet une amélioration moyenne de la qualité d’environ 10 % par rapport à l’algorithme d’insertion à moindre coût dans le cas de l’instance INS4. Ce qui s’explique par l’utilité forte de l’opérateur d’insertion avec regret.

Tableau 4-4 : Qualité de la solution et le temps d'exécution pour l'instance INS4

Scénarios	Qualité (%)	Temps d'exécution T (sec)	Temps T* (sec)
Scénario 1	7.50	7.95	4.70
Scénario 2	4.83	12.47	8.89
Scénario 3	4.36	33.73	23.13
Scénario 4	3.64	12.86	10.01
Scénario 5	4.75	16.07	8.83
Scénario 6	4.58	20.44	16.36
Scénario 7	5.85	8.02	6.06
Scénario 8	4.97	22.81	14.45
Scénario 9	4.30	43.44	30.62
Scénario 10	3.68	34.32	28.82
Scénario 11	3.55	44.01	29.28
Scénario 12 (ALNS)	**1.63**	43.74	30.66

D'après le tableau 4-5, on peut tirer une amélioration de plus de 57 % entre le meilleur scénario et le pire scénario. Le meilleur scénario est relatif à l'algorithme *ALNS* avec une qualité moyenne de 1.78 % alors que le pire scénario, scénario 7, est celui possédant la valeur de qualité la plus élevée (4.15%). Ce scénario a toutefois l'avantage d'être en moyenne 5 fois plus rapide que *ALNS* pour cette catégorie d'instance (INS6).

Tableau 4-5 : Qualité de la solution et le temps d'exécution pour l'instance INS6

Scénarios	Qualité (%)	Temps d'exécution *T* (sec)	Temps *T** (sec)
Scénario 1	3.72	7.71	4.28
Scénario 2	3.72	13.46	8.76
Scénario 3	3.22	31.39	18.81
Scénario 4	3.58	14.07	6.72
Scénario 5	3.34	18.49	10.43
Scénario 6	3.45	19.63	11.21
Scénario 7	4.15	8.32	4.41
Scénario 8	3.27	22.03	11.28
Scénario 9	3.02	41.59	23.93
Scénario 10	3.22	33.89	20.04
Scénario 11	2.80	43.94	20.77
Scénario 12 (ALNS)	**1.78**	43.41	24.02

Le tableau 4-6 rapporte les résultats sur l'instance INS7, le scénario 11 fait appel à l'algorithme de retrait pire comme son opérateur unique de retrait des conteneurs et il utilise les deux opérateurs d'insertion développés dans ce travail. Ce scénario obtient une solution moyenne qui est à 2.56 % de la meilleure solution connue dans un temps de 42.45 secondes. Ce scénario est le plus proche du meilleur scénario qui n'est que la méthode *ALNS*. Cette dernière possède la solution dans un temps moyen de 44.78 secondes de calcul pour l'instance INS7 et avec une qualité moyenne de solution de 1.92%.

Tableau 4-6 : Qualité de la solution et le temps d'exécution pour l'instance INS7

Scénarios	Qualité (%)	Temps d'exécution T (sec)	Temps T* (sec)
Scénario 1	3.96	7.38	5.04
Scénario 2	3.24	26.56	7.56
Scénario 3	3.21	31.14	15.77
Scénario 4	4.25	13.05	6.46
Scénario 5	2.94	18.08	6.77
Scénario 6	3.97	19.67	8.24
Scénario 7	4.21	8.58	4.12
Scénario 8	3.72	21.86	11.15
Scénario 9	2.48	39.91	20.33
Scénario 10	1.96	33.66	22.22
Scénario 11	2.56	42.54	23.14
Scénario 12 (ALNS)	**1.92**	44.78	23.62

La qualité de la solution générée par chaque scénario pour le reste des instances est présentée dans l'annexe 2.

On fixe maintenant un scénario (scénario 12 explorant tous les opérateurs de retrait et d'insertion) et on fait varier les instances. Pour effectuer cette comparaison, on a choisi les instances avec 20 conteneurs et un nombre d'équipements variable (portiques de cour et camions) ainsi que le nombre des blocs. D'après les tests effectués, on peut dresser le tableau suivant (tableau 4-7).

Tableau 4-7 : Comparaison des instances avec le scénario 12 (ALNS)

Instances	Qualité de la solution (%)	Temps d'exécution (sec)
INS4	1.63	43.74
INS6	1.78	43.41
INS7	1.92	44.78

Comparant INS4 et INS7, la qualité de la solution se dégrade avec l'augmentation du nombre de portiques de cour de 2 à 4 portiques de cour ainsi que l'augmentation du nombre de blocs de 2 à 4 blocs. La qualité moyenne de la solution pour la méthode *ALNS* passe de 1.63% à 1.92%. Dans le cas où on varie seulement le nombre des camions (de 2 à 3 camions), dans les instances INS6 et INS7, on observe aussi une dégradation de la qualité moyenne de la solution d'environ 7.3% suite à l'ajout d'un seul camion. L'augmentation du nombre des équipements ne conduit pas automatiquement à une amélioration de la solution. En d'autres termes, pour un nombre bien déterminé de conteneurs, il faut un nombre « optimal » d'équipements permettant la minimisation du temps de manutention. De même, l'augmentation du nombre de portiques de cour et/ou de camions peut être sans utilitée. D'un autre côté, l'augmentation du nombre des blocs conduit à l'augmentation de la distance de parcours des équipements, surtout des portiques de cour lors de son déplacement entre blocs pour aller chercher les conteneurs.

La rapidité de calcul pour les trois instances est presque la même. On peut donc en conclure que le nombre d'équipements de manutention dans un port maritime influence bien la qualité de la solution générée par l'algorithme *ALNS*.

4.3.3 Application de la méthode de ALNS pour les instances de grande taille (INS9 : cas de 100 conteneurs)

En premier lieu, on va s'intéresser à présenter les différents scénarios utilisés dans cette section. On a généré deux scénarios (A et B) qui vont être comparés à l'algorithme *ALNS* développé dans ce travail.

Le scénario A exploite seulement un seul opérateur de destruction H^- et un seul opérateur de réparation H^+. L'opérateur de destruction est basé sur le retrait aléatoire de q conteneurs de la solution initiale. Puis, l'opérateur d'insertion avec regret (H^+) est utilisé pour la reconstruction de la solution déjà détruite. Le scénario A n'est autre qu'une extension des scénarios 7, 8 et 9 étudiés dans la section 4.3.2, où on a retiré 1, 2 et 3 conteneurs respectivement d'une manière aléatoire et on les a réinsérés de nouveau à l'aide de l'opérateur d'insertion avec regret.

Le scénario B, quant à lui, est composé de trois opérateurs dont un est un opérateur de retrait et les deux autres sont des opérateurs d'insertion. Plus précisément, l'opérateur de pire retrait (H^-) s'occupe de retirer q conteneurs ; ensuite les deux opérateurs d'insertion, insertion à moindre coût (H_1^+) et insertion avec regret (H_2^+), vont essayer d'insérer les q conteneurs dans les meilleures positions afin d'améliorer la solution. Le scénario B est très similaire au scénario 11 étudié dans le cas des problèmes de petite taille (section 4.3.2). Rappelons que ce dernier est une combinaison de l'opérateur retrait pire de trois conteneurs et deux opérateurs d'insertion (insertion à moindre coût et insertion avec regret). Le scénario 11 a donné de bons résultats dans le cas des instances de petite taille.

Le scénario B diffère de la méthode *ALNS*, dans le sens où aucun ajustement des poids et des scores n'est nécessaire, comme c'est le cas pour la méthode *ALNS*.

Finalement, l'algorithme *ALNS* utilise les trois opérateurs de retrait et les deux opérateurs d'insertion avec les différentes mises à jour nécessaires.

Plus spécifiquement, les trois opérateurs de retrait sont les suivants :

H_1^- : Retirer q conteneurs aléatoirement.

H_2^- : Retirer une séquence (groupe) de q conteneurs de façon aléatoire.

H_3^- : Retrait pire de q conteneurs.

Les deux opérateurs d'insertion sont les suivants :

H_1^+ : Insertion à moindre coût les q conteneurs retirés.

H_2^+ : Insertion avec regret des q conteneurs retirés.

Une comparaison entre les scénarios A et B, et la méthode *ALNS* est faite afin d'étudier l'efficacité de la méthode développée (*ALNS*).

Les algorithmes développés pour les scénarios A et B et l'algorithme *ALNS* ont été appliqués sur l'instance INS9 (100 conteneurs, 4 portiques de cour et 3 camions). Ils ont été exécutés 3 fois avec un total de 100 itérations pour chaque exécution. Pour l'algorithme *ALNS*, le segment Q est composé de 50 itérations. Rappelons qu'au départ de chaque segment une mise à jour des poids est faite.

Au cours des expérimentations, on a gardé les mêmes paramètres utilisés dans le cas des instances de petite taille sauf en ce qui a trait au nombre d'itérations I et la taille du segment Q. Le vecteur [$w; c ; Q ; I; r ; \sigma_1 ; \sigma_2 ; \sigma_3$] prend les valeurs [1.05; 0.98; 50;100; 0.1; 15; 10; 6].

Dans un second lieu, le nombre de conteneurs à retirer est un facteur important et il a un impact direct sur la qualité de la solution. En effet, un petit nombre de conteneurs retirés ne permet pas d'explorer un large voisinage et la solution tend vers des minima locaux. Cependant, avec un très grand nombre de conteneurs, la reconstruction de la solution détruite sera difficile. Pour cela, on va étudier, par la suite, l'effet du nombre de conteneurs à retirer sur la qualité de la solution en utilisant l'instance INS9.

Le tableau 4-8 regroupe les résultats. La première colonne indique le numéro du test réalisé, la deuxième le nombre de conteneurs à retirer à chaque itération. Pour le nombre de conteneurs à retirer, on va se limiter seulement, dans le cadre de notre recherche, à l'ensemble {1, 5, 9, 13, 17, 21}. À titre d'exemple, pour la ligne 3, il s'agit de retirer 9 conteneurs selon l'opérateur de retrait utilisé dans chaque situation. Pour l'algorithme *ALNS*, le premier opérateur de retrait va s'occuper de retirer 9 conteneurs aléatoirement, le deuxième va retirer une séquence formée de 9 conteneurs à partir de la solution initiale, tandis que le troisième va effectuer un retrait pire de 9 conteneurs.

Les troisième, quatrième et cinquième colonnes contiennent les résultats du scénario A, soit la meilleure solution connue pour ce scénario, la moyenne des solutions obtenues après les trois

exécutions et la qualité de la solution exprimée en pourcentage. Les trois colonnes suivantes présentent les résultats du scénario B. Enfin, les trois dernières colonnes contiennent les résultats de l'algorithme *ALNS*. La qualité de la solution est l'écart en pourcentage de la moyenne des solutions obtenues après les trois exécutions par rapport à la meilleure solution connue dans les trois situations (scénario A, scénario B et *ALNS*).

À partir des résultats présentés dans le tableau 4-8, on remarque que l'algorithme *ALNS* donne de meilleurs résultats pour les différentes options quelque soit le nombre de conteneurs à retirer pour l'instance INS9.

Pour le scénario A, retirer 9 conteneurs de la solution initiale est la meilleure option car elle permet d'avoir une meilleure qualité (0.45%) par rapport aux autres options. Alors que pour le scénario B et la méthode *ALNS*, le retrait de 21 conteneurs est considéré comme la meilleure option vu qu'elle permet d'obtenir la meilleure qualité (0.31 % pour scénario B et 0.02 % pour l'algorithme *ALNS*).

Cependant, on observe une très grande différence pour le retrait d'un seul conteneur où la qualité de la solution est médiocre en comparaison avec celles obtenues en retirant plus d'un conteneur et ce quel que soit le scénario A (3.27 %), le scénario B (8.19 %) et l'algorithme *ALNS* (1.44%).

En se limitant aux options de retrait étudiées, on remarque que le retrait de 17 conteneurs permet d'avoir la meilleure valeur de la fonction objectif (temps de complétion = 5711.50 secondes) pour traiter 100 conteneurs avec 4 portiques de cour et 3 camions dans un port maritime.

La qualité de la solution obtenue par l'heuristique *ALNS* est très intéressante (figure 4-5). L'écart tend vers zéro et qui est pratiquement toujours la meilleure par rapport aux qualités des deux autres scénarios. Ce qui prouve l'efficacité de la méthode développée dans le cadre de ce travail.

Tableau 4-8 : Qualité des solutions par rapport au nombre de conteneurs retirés pour les scénarios A, B et ALNS

		Scénario A			Scénario B			ALNS		
Nº	Nombre de conteneurs à retirer	Meilleure	Moyenne	Qualité (%)	Meilleure	Moyenne	Qualité (%)	Meilleure	Moyenne	Qualité (%)
1	1	5951.50	5980.50	3.27	6265.00	6265.00	8.19	**5790.50**	5874.00	**1.44**
2	5	5780.00	5872.00	2.29	5970.50	5970.50	4.00	**5740.50**	5777.50	**0.64**
3	9	5742.00	5758.33	0.45	5790.00	5831.33	1.72	**5732.50**	5752.00	**0.34**
4	13	5762.00	5766.17	0.77	5942.50	5942.50	3.85	**57220**	5730.00	**0.13**
5	17	5744.00	5748.33	0.64	6007.50	6007.50	5.18	**5711.50**	5731.50	**0.35**
6	21	5780.00	5789.67	0.96	5752.50	5752.50	0.31	**5734.50**	5735.83	**0.02**

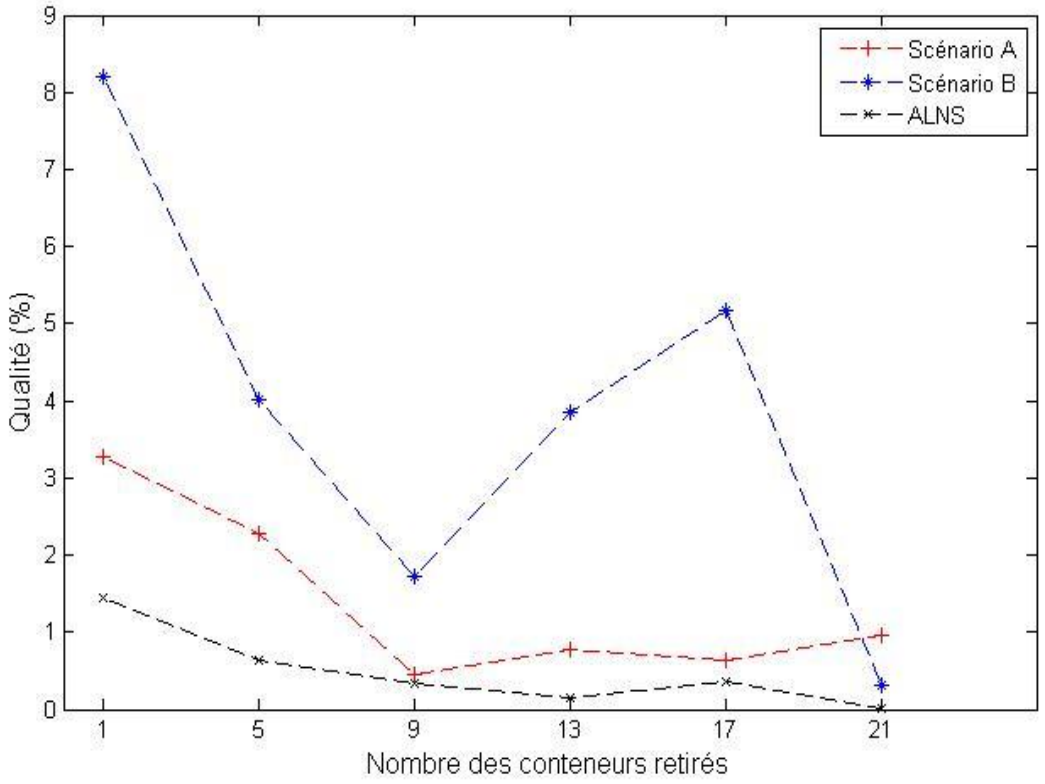

Figure 4-5 : Variation de la qualité de la solution en fonction du nombre des conteneurs retirés pour les scénarios A, B et l'algorithme ALNS

CHAPITRE 5 CONCLUSION GÉNÉRALE

Le travail présenté dans ce mémoire avait pour objectif la planification des opérations de manutention des conteneurs destinés à l'exportation dans un terminal maritime par les portiques de cour, en essayant de minimiser le temps de complétion pour le traitement des ces conteneurs. Le travail traite également de la synchronisation entre les équipements de manutention, plus principalement les portiques de cour et les véhicules de transport. On a introduit dans le problème de planification des mouvements des conteneurs deux phénomènes jugés importants par les chercheurs et qui ont une influence sur le temps de processus de manutention. Ces deux phénomènes sont l'interférence entre les portiques de cour et les mouvements de remanutention des conteneurs par les portiques de cour (*rehandle*). Le premier peut exister lors des déplacements des portiques entre les blocs et le deuxième existe dans le cas où les portiques de cour déplacent des conteneurs pour accéder à des conteneurs spécifiques.

On a présenté dans le premier chapitre une description des opérations portuaires et cela par la présentation des différents équipements de manutention (les différents types, le fonctionnement et les rôles dans le port), principalement les portiques de cour, les moyens de transport et les portiques de quai.

Le second chapitre fût consacré à une récapitulation des travaux dans la littérature portant sur le problème d'optimisation dans le cadre des terminaux à conteneurs. La première partie du chapitre regroupe les travaux étudiant la planification des opérations de manutention par les différents équipements portuaires. La seconde présente une vision d'ensemble sur les travaux étudiant l'interférence entre les portiques de cour et les mouvements non productifs.

Le problème d'optimisation a été formulé en programme linéaire mixte et dont l'objectif est de minimiser le temps de complétion (*makespan*). Le modèle développé a été résolu en premier lieu, à l'aide de AMPL Studio et CPLEX 12.2.0.0 pour un problème de petite taille (10 conteneurs). Une méthode heuristique a été développée pour résoudre ce problème dans le chapitre suivant.

Le chapitre quatre est formé essentiellement de deux grandes parties. La première est une récapitulation de travaux de littérature qui traitent cette méthode. Tandis que la deuxième est une partie expérimentale, où on a testé la méthode *ALNS* sur plusieurs instances formées de 10, 20 et

100 conteneurs afin d'évaluer sa performance à traiter ce type de problème. Pour les instances de petite taille, une comparaison entre différents scénarios (qui regroupent quelques opérateurs de retrait et quelques opérateurs d'insertion) est faite dans le but de montrer que l'approche heuristique *ALNS* est plus efficace surtout en termes de qualité moyenne de la solution générée. La valeur de cette qualité, dans la plupart des instances traités, est très faible ce qui implique que la solution générée par *ALNS* est très proche de la meilleure solution connue dans l'expérimentation quelle que soit la taille du problème. Pour un problème de grande taille (cas de 100 conteneurs) et en faisant varier le nombre de conteneurs à retirer à chaque itération, l'algorithme *ALNS* développé s'est avéré plus performant que les deux scénarios alternatifs (scénarios A et B) en termes de qualité de la solution.

En conclusion, en s'inspirant de ce sujet, plusieurs travaux futurs peuvent être développés et plusieurs problématiques peuvent se poser. En effet, à partir du modèle mathématique, une nouvelle modélisation est possible pour combiner les différentes opérations de manutention des trois équipements simultanément à savoir les portiques de cour, les camions et les portiques de quai afin de minimiser le temps de complétion pour le traitement des conteneurs destinés à l'exportation. Une autre voie possible est de traiter les opérations de manutention des conteneurs par les portiques de cour et les camions simultanément dans les deux situations d'exportation et d'importation. Les tests établis ont conduit à proposer une troisième voie. Cette dernière est d'étudier l'effet de la conception de la cour (nombre de blocs, nombre des baies, etc.) sur le temps de complétion en utilisant la méthode *ALNS*. Il s'agit de garder le même objectif de notre travail, en variant à chaque fois le nombre de blocs, le nombre de baies, le nombre de portiques de cour et de camions afin de trouver la conception optimale d'un port maritime.

BIBLIOGRAPHIE

Azi, N., Gendreau, M., & Jean-Yves, P. (2010). An adaptative large neighborhood search for a vehicule routing problem with multiple trips. *CIRRELT 8*.

Bish, E. K. (2003). A multiple-crane-constrained scheduling problem in a container terminal.*European Journal of Operational Research 144*, 83-117.

Bish, E. K., Chen, F. Y., Leong, Y. T., Nelson, B. L., Ng, J. W. C., & Simchi-Levi, D. (2005). Dispatching vehicles in a mega container terminal. *OR Spectrum 27*, 491-506.

Cao, J. X., Lee, D.-H., Chen, J. H., & Shi, Q. (2010). The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. *Transportation Research Part E: Logistics and Transportation Review, 49*(3), 344-353.

Chen, L. (2006). Optimization for the operational problem of container handling system in a maritime terminal.Université de Nantes and Shanghai Jiao Tong University, France, China.

Chen, L., Bostel, N., Dejax, P., Cai, J., & Xi, L. (2007). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *European Journal of Operational Research, 181*, 40-58.

Chen, L., & Langevin, A. (2011). Multiple yard cranes scheduling for loading operations in a container terminal. *Engineering Optimization*, 1–17.

Chen, P., Fu, Z., Lim, A., & Rodrigues, B. (2004). Port Yard Storage Optimization. *Automation Science and Engineering, IEEE Transactions on, 1*(1), 26 - 37

Côté, J.-F. (2009). Une heuristique à grand voisinage pour un problème de confection de tournée pour un seul véhicule avec cueillettes et livraisons et contrainte de chargement. Université de Montréal, Montréal.

Daganzo, C. F. (1989). The crane scheduling problem. *Transportation Research B, 23*, 159–175.

Daganzo, C. F. (1990). The productivity of multipurpose seaport terminals. *Transportation Science, 24*, 205–216.

Evers, J. J. M., & Koppers, S. A. J. (1996). Automated guided vehicle traffic control at a container terminal. *Transportation Research A, 30*(1), 21-34.

Golden, B. L., Dearmon, J. S., & Baker, E. K. (1983). Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research, 10*(1), 47-59.

Guo, X., Huang, S. Y., Hsu, W. J., & Low, M. Y. H. (2008). *Yard crane dispatching based on real time data driven simulation for container terminals*. Paper presented at the Proceedings of the 40th Conference on Winter Simulation.

Huynh, N. (2009). Reducing Truck Turn Times at Marine Terminals with Appointment Scheduling. *Transportation Research Record: Journal of the Transportation Research Board* (2100), 47-57.

Huynh, N., & Walton, C. M. (2008). Robust Scheduling of Truck Arrivals at Marine Container Terminals. *Journal of Transportation Engineering, 134*(8), 347-353.

Huynh, N., Walton, C. M., & Davis, J. (2004). Finding the Number of Yard Cranes Needed to Achieve Desired Truck Turn Time at Maritime Container Terminals. *Transportation Research Record, 1873*, 99-108.

Imai, A., & Miki, T. (1989). A heuristic algorithm with expected utility for an optimal sequence of loading containers into a containerized ship. *Journal of Japan Institute of Navigation, 80*, 117–124.

Imai, A., Nishimura, E., Papadimitriou, S., & Sasaki, K. (2002). The containership loading problem. *International Journal of Maritime Economics, 4*, 126-148.

Imai, A., Sasaki, K., Nishimura, E., & Papadimitriou, S. (2006). Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European Journal of Operational Research, 171*, 373-389.

Jung, D. H., Park, Y.-M., Lee, B. K., Kim, K. H., & Ryu, K. R. (2006). *A Quay Crane Scheduling Method Considering Interference of Yard Cranes in Container Terminals*. Paper presented at the 5th Mexican International Conference on Artificial Intelligence.

Kim, K. H. (1997). Evaluation of the number of rehandles in container yards. *Computers & Industrial Engineering, 32*(4), 701-711.

Kim, K. H. (2008). Operational Issues in Modern Container Terminals. In P. A. Ioannou (Ed.), *Intelligent Freight Transportation* (pp. 51–69).

Kim, K. H., & Bae, J. W. (2004). A Look-Ahead Dispatching Method for Automated Guided Vehicles in Automated Port Container Terminals. *Transportation Science, 38*(2), 224-234.

Kim, K. Y., & Kim, K. H. (1997). A routing algorithm for a single transfer crane to load export containers onto a containership. *Computers & Industrial Engineering, 33*, 673-676.

Kim, K. H., & Kim, K. Y. (1999). An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science 33*(1), 17-33.

Kim, K. H., & Park, Y. M. (2004). A crane scheduling method for port container terminals. *European Journal of Operational Research, 156*, 752–768.

Laporte, G., Musmanno, R., & Vocaturo, F. (2010). An Adaptive Large Neighbourhood Search Heuristic for the Capacitated Arc-Routing Problem with Stochastic Demands. *Transportation Science 44*(1), 125-135.

Lee, D.-H., Cao, Z., & Meng, Q. (2007). Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *International Journal of Production Economics, 107*(1), 115–124.

Lee, D.-H., Cao, J. X., & Shi, Q. X. (2009a). Synchronization of yard truck scheduling and storage allocation in container terminals. *Engineering Optimization, 41*(7), 659–672.

Lee, D.-H., Cao, J. X., Shi, Q., & Chen, J. H. (2009b). A heuristic algorithm for yard truck scheduling and storage allocation problems. *Transportation Research Part E 45 45*, 810-820.

Lee, D.-H., Wang, H. Q., & Miao, L. (2008). Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E, 44*, 124-135.

Lei, L., Junqing, S., & Mei, H. (2008). *On Route Optimization of CTs in Container Terminal Based on Pool Strategy*. Paper presented at the Control Conference, 2008. CCC 2008. 27th Chinese.

Li, K., & Tang, L. (2009). *A tabu search algorithm for the integrated truck scheduling and storage allocation in quay*. Paper presented at the Computational Intelligence and Design, 2009. ISCID '09. Second International Symposium on.

Li, W., Wu, Y., Petering, M. E. H., Goh, M., & Souza, R. D. (2009). Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research 198*, 165-172.

Liang, C., Ma, X., & Chen, M. (2011). Study on Yard Crane Scheduling with Multiple Container Flows in a Container Terminal. *Journal of Quality, 18*(4), 375-392.

Linn, R. J., & Zhang, C.-Q.(2003). A heuristic for dynamic yard crane deployment in a container terminal.*IIE Transactions, 35*, 161-174.

Moccia, L., Cordeau, J. F., Gaudioso, M., & Laporte, G. (2006). A Branch-and-Cut Algorithm for the Quay Crane Scheduling Problem in a Container Terminal. *Naval Research Logistics (NRL), 53*(1), 45-59.

Muller, L. F. (2009). *An Adaptive Large Neighborhood Search Algorithm for the Resource-constrained Project Scheduling Problem*. Paper presented at the Proceedings of the VIII Metaheuristic International Conference (MIC2009).

Ng, W. C. (2005). Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research 164*, 64–78.

Ng, W. C., &Ge, Y. (2006). *Scheduling Landside Operations of a Container Terminal Using a Fuzzy Heuristic*. Paper presented at the Industrial Informatics, 2006 IEEE International Conference on

Ng, W. C., &Mak, K. L. (2005). Yard crane scheduling in port container terminals.*Applied Mathematical Modelling, 29*, 263–276.

Ng, W. C., Mak, K. L., & Zhang, Y. X. (2007). Scheduling trucks in container terminals using a genetic algorithm. *Engineering Optimization, 39*(1), 33-47.

Nishimura, E., Imai, A., Janssens, G. K., & Papadimitriou, S. (2009). Container storage and transshipment marine terminals. *Transportation Research Part E 45*, 771-786.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*, 2403 – 2435.

Pisinger, D., & Ropke, S. (2010). Large Neighborhood Search. In G. Michel & P. Jean-Yves (Eds.), *Handbook of Metaheuristics* (2nd ed., Vol. 146, pp. 399-420): International Series in Operations Research & Management Science.

Ribeiro, G. M., & Laporte, G. (2011). An Adaptive Large Neighborhood Search Heuristic for the Capacitated Vehicle Routing Problem. *CIRRELT, 2.*

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science, 40*(4), 455-472.

Salazar-Aguilar, M. A., Langevin, A., Laporte, G., & Cai, J. (2011). Synchronized Arc Routing for Snow Plowing Operations. *Computers & Operations Research. CIRRELT 29*.

Sammarra, M., Cordeau, J.-F., Laporte, G., & Monaco, M. F. (2007). A tabu search heuristic for the quay crane scheduling problem. *J Sched, 10*, 327–336.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record Breaking Optimization Results Using the Ruin and Recreate Principle.*Journal of Computational Physics, 159*, 139–171.

Shaw, P. (1998). *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems.* Paper presented at the Proceedings of the Fourth International Conference 60 on Principles and Practice of Constraint Programming, Berlin.

Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research – a classification and literature review. *OR Spectrum, 26*, 3-49.

Vis, I. F. A., & Harika, I. (2004). Comparison of vehicle types at an automated container terminal. *OR Spectrum 26*, 117-143.

Vis, I. F. A., & Koster, R. d. (2003). Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research 147*, 1-16.

Zhang, C., Wan, Y.-w., Liu, J., & Linn, R. J. (2002). Dynamic crane deployment in container storage yards. *Transportation Research Part B 36*, 537–555.

Zhang, H.-L., & Jiang, Z.-B. (2008). Simulation Studies of Heuristic approaches for dynamic scheduling of container terminal operations. *International Journal of Modelling and Simulation, 28*(4), 410-422.

ANNEXE 1 – Programmation avec AMPL

- Fichier.mod

```
###  SETS              ###

param nb_conteneurs;        # Nombre de Conteneurs
param nb_YC;        # Nombre de portiques de cour
param nb_camions;       # Nombre de camions
param nb_blocs;     # Nombre de blocs
param nb_quais;   #Nombre de paortiques ede quai


set C:=1..nb_conteneurs;         # Conteneur
set Cplus:= C union {0};        #ensemble des conteneurs y compri le conteneur fictif 0
set M:=1..nb_YC;        # Portiques_de_Cour
set N:=1..nb_camions;       # Camions
set B:=1..nb_blocs;      # Blocs
set Q:=1..nb_quais;    #Portiques_de_quais
set adresse;                  # Ensemble de localisations
set P within {C,C};     # paire de Contenuers avec une relation de priorité
set U within {C,C};     # paire de Contenuers qui ne peuvent pas être traité en même temps à cause de l'intérférnece
set A;

###  PARAMETERS        ###


param T := 10000;           #Grande Valeur
param MB{B} >=0;                #nombre initial de portique de cour dans un bloc b
param h1 >= 0;              # Constante Temps de manutention d'un conteneur
param h2 >=0;           # Constante temps de déplacement d'un conteneur et de le rendre à sa place
param t{C} >= 0;            # Temps de transport du conteneur i par un camion (fixe)
param k{C, C} >= 0;             # Temps de parcours d'un portique de cour m du conteneur i vers conteneur j
param L{C, adresse};          #Adresse de chaque conteneur
param d {i in C} >0;       #numéro de bloc
param e {i in C} >0;  #numéro de l'étage
param a {i in C} >0;   #numéro de la baie
param q {i in C} >0;   #destination sur le quai
param r {i in C} >0; #numéro de la rangé
param tvqa {Q,A} integer >= 0; # Matrice de Temps de transport à vide entre un QC q et une baie a



###  VARIABLES         ###

var dYC {C} >= 0;        # temps de début de prélèvement du conteneur i par un YC
var dYT {C} >= 0;        # temps de début de prélèvement du conteneur i par un YT
var V {i in C, j in C:i<>j} binary;        # le traitement de j juste après i.
var Z {b in B, f in B, m in M} binary ;             # la portique m se déplace du bloc b vers b'(f)
var X{i in Cplus,j in Cplus, m in M:i<>j} binary ;             # Séquence des conteneurs par les portiques de cour
var Yijn{i in Cplus,j in Cplus, n in N:i<>j} binary ;              #Séquence des conteneurs par les camions
var u{i in C, j in C, s in C, o in C} binary;          #nouvelle variable pour l'interference
var p {i in C};  # Temps de traitement total des conteneurs par les portiques de cour
var p1{i in C};  # temps pour déplacer conteneur i et de le charger sur camion
var p2{i in C};  #Temps de remettre les conteneurs à leurs places
var Makespan;


###  OBJECTIVES        ###


minimize MaxTime: Makespan;
```

```
###  CONSTRAINTS       ###

#1- contrainte relative à la fonction objective
subject to Completion_Time {i in C}:
     Makespan >= dYT[i];

#2- Chaque conteneur doit être assigné a un seul portique de cour
    subject to AffectYC{j in C}:
       sum{i in Cplus,m in M: i<>j} X[i,j,m] = 1;

#3- Chaque portique doit sortir une seule fois
    subject to SortYC{m in M}:
       sum{j in C} X[0,j,m] = 1;

#3prime- Chaque camion doit sortir une seule fois
    subject to SortYT{n in N}:
       sum{j in C} Yijn[0,j,n] = 1;

#4- Conservation du flux
    subject to Flux{i in C, m in M}:
       sum{j in Cplus: i<>j} X[i,j,m] = sum{j in Cplus: i<>j} X[j,i,m];

#4prime- Conservation du flux des camions
    subject to Flux_camions{i in C, n in N}:
       sum{j in Cplus: i<>j} Yijn[i,j,n] = sum{j in Cplus: i<>j} Yijn[j,i,n];

#5- Chaque conteneur doit être assigné a un seul camion
    subject to AffectYT{j in C}:
       sum{i in Cplus, n in N:i<>j} Yijn[i,j,n] = 1;

#6- Contrainte de définition du temps de traitement p1 pour la fin du conteneur sur le camion

    subject to temps_traitement1{i in C}:
    p1[i]= sum {j in C : i<>j and a[i]= a[j] and r[i] = r[j]and e[i] < e[j]} h1 * V[i,j] +h1;

#6prime- Contrainte de définition du temps de traitement p2 pour remettre les conteneurs retirés

    subject to temps_traitement2{i in C}:
    p2[i]= sum {j in C : i<>j and a[i]= a[j] and r[i] = r[j]and e[i] < e[j]} h1 * V[i,j] ;

#6bis- Contrainte de définition du temps de traitement avec Rehandle

    subject to temps_traitement {i in C}:
    p[i]= p1[i] + p2[i];

#7- l'ordre de traitement des conteneurs par chaque YC

    subject to Contrainte4 {m in M,i in C, j in C: i <> j}:
    dYC[j] >=  dYC[i] + p1[i] + p2[i] + k[i,j] + (X[i,j,m]- 1)* T;

#8- l'ordre de traitement des conteneurs par chaque YC en fonction des moments de débuts des YTs

    subject to Contrainte6prime {m in M,i in C, j in C: i <> j}:
    dYC[j] >=  dYT[i] + k[i,j] + (X[i,j,m]- 1)* T;

#9- Relation de priorité

   subject to Contrainte5 {(i,j) in P: i <> j}:
    dYT[j] + t[j] >=  dYT[i] + t[i];
```

```
#10- sequence de traitement iet j appartiennent à U
    subject to Contrainte6 {(i,j) in U: i<>j}:
     dYC[j]>= dYC[j] + p1[i] + p2[i]+ (V[i,j]-1)*T;

#11- l'ordre de traitement des conteneurs par chaque camion

   subject to Contrainte7 {n in N,i in C, j in C: i <> j}:
    dYT[j] >=  dYT[i] + t[i] + tvqa[q[i],a[j]] + (Yijn[i,j,n]- 1)* T;

#12-  l'ordre de traitement des conteneurs par chaque camion
    subject to Contrainte8 {i in C}:
    dYT[i] >= dYC[i] + p1[i];

#13- YC m  traite le conteneur j juste aprés le conteneur i
    subject to Contrainte9 {i in C, j in C, m in M : d[i] <>d[j]}:
     Z[d[i],d[j],m] >= X[i,j,m];

#14- Vérification de l'hypothèse du nombre des YC dans chaque bloc, nombre max de YC par bloc

    subject to contrainte10 {f in B, m in M}:
       sum{b in B:b<>f} Z[b,f,m] + MB [f] <= 2;

#15-Le déplacement d'un YC juste vers une seule destination
    subject to Contrainte11 {m in M}:
     sum {b in B, f in B: b <> f} Z[b,f,m] <=1;

#16-description de la variable V
    subject to contrainte12 {i in C, j in C:i<>j}:
  dYC[j] - dYC[i] - p1[i] - p2[i]<= V[i,j]*T;

#17- description de la variable V

    subject to contrainte13{i in C, j in C:i<>j}:
   dYC[i]- dYC[j] + p1[i]+ p2[i]<= (1 - V[i,j])*T;

#18- Garantie pas de collision entre les YC

   subject to Contrainte14 {(i,j) in U}:
    V[i,j]+ V[j,i] = 1;

#19- Garantie de non interférence 1

  subject to contrainte15 {i in C, j in C, s in C: i<>j and i<>s and d[j]= d[s] and a[s]> a[j] > a[i]}:
  sum {m in M} X[i,s,m]<= V[i,j] + V[j,i];

#20- Garantie de non interférence 2
 subject to contrainte16 {i in C, j in C, o in C : i<>j and j<>o and d[i] = d[o] and a[o]< a[i] < a[j]}:
  sum {m in M} X[j,o,m]<= V[i,j] + V[j,i];

#21- Garantie de non interférence 3

 subject to contrainte17 {i in C, j in C, s in C, o in C : d[i] = d[j]and  d[i]= d[s]and d[s]= d[o] and d[j] = d[o]
            and a[i]< a[o] and a[o] < a[s] and a[s]< a[j]}:

  sum {m in M} X[i,s,m]<= V[i,j] + V[j,i] + T* (1- u[i,j,s,o]);

#22- Garantie de non interférence 4

 subject to contrainte18 {i in C, j in C, s in C, o in C: d[i] = d[j]and  d[i]= d[s]and d[s]= d[o] and d[j] = d[o]
          and a[i]< a[o] and a[o] < a[s] and a[s]< a[j]}:

   sum {m in M} X[j,o,m]<= V[i,j] + V[j,i] + T * u[i,j,s,o];
```

- Fichier.dat

```
##Set##
set P :=
(2,1)
(2,4)
(2,8)
(2,5)
(7,5)
(7,1)
(7,4)
(7,8)
(9,6)
(9,3)
(10,6)
(10,3);     # paire de Contenuers avec une relation de priorité

set U :=
(4,5)
(9,10)
(8,7);      # paire de Contenuers qui ne peuvent pas être traité en même temps à cause de la colision

set A:= 1 2 3 4 5 6 7 8;  # ensemble de baies dans les différents blocs


##PARAMETRES ##

param nb_conteneurs:=10;        # Nombre de Conteneurs
param nb_YC:= 2;        # Nombre de portique de cours
param nb_camions:=3;        # Nombre de camions
param nb_quais:=2;          # Nombre de portique de quai
param nb_blocs:= 2;     # Nombre de blocs
param MB:= 1 1, 2 1;                #nombre initial de portique de cour dans un bloc b
param h1 := 20;             # Constante Temps de manutention d'un conteneur
param d:=
1   1
2   1
3   1
4   1
5   1
6   2
7   2
8   2
9   2
10  2;

param r:=             #nombre de la rangée d'un conteneur
1   1
2   2
3   3
4   2
5   2
6   3
7   2
8   1
9   3
10  2;
```

```
param e:=             #numéro de l'étage du conteneur
1   1
2   1
3   1
4   2
5   1
6   1
7   1
8   1
9   1
10  1;

param a:=  #numéro de la baie
1   1
2   2
3   3
4   4
5   4
6   5
7   7
8   7
9   8
10  8;

param tvqa :=         # Matrice de Temps de transport à vide entre un QC q et une baie a
1 1 60
1 2 60
1 3 60
1 4 60
1 6 90
1 7 90
1 8 90
1 5 90
2 1 90
2 2 90
2 3 90
2 4 90
2 5 60
2 6 60
2 7 60
2 8 60;

param q:=  #destination sur le quai
1   2
2   2
3   1
4   2
5   2
6   1
7   2
8   2
9   1
10  1;
```

```
param t:= # Temps de transport du conteneur i par un camion vers sa distination sur le quai(fixe)
1   180
2   180
3   120
4   180
5   180
6   120
7   120
8   120
9   180
10  180;
param k (tr):           # matrice Temps de parcours d'un portique de cour m du conteneur i vers conteneur j

    1   2   3   4   5   6   7   8   9   10  :=
1   0   0.5 1   2   3   3   6   6   8   9
2   0.5 0   1   2   3   3   6   6   8   9
3   1   1   0   1   2   2   5   5   7   8
4   2   2   1   0   1   1   4   4   6   7
5   3   3   2   1   0   0.5 3   3   5   6
6   3   3   2   1   0.5 0   3   3   5   6
7   6   6   5   4   3   3   0   0.5 2   3
8   6   6   5   4   3   3   0.5 0   2   3
9   8   8   7   6   5   5   2   2   0   1
10  9   9   8   7   6   6   3   3   1   0;
```

- Le rapport de la solution

```
    AmplStudio Modeling System - Copyright (c) 2003-2010, Datumatic Ltd
    ____________________________________________________________________

MODEL.STATISTICS

Problem name              :AMPL-model 31-08
Pathname                  :C:\Program Files (x86)\AmplStudio Modeling Sy
                          :stem 1.6.J\Bin\AMPL-Model-31-08\
Model Filename            :"AMPL-model 31-08.mod"
Data  Filename            :"AMPL-model 31-08.dat"
Date                      :9:15:2011
Time                      :00:24
Constraints               :1170        :  Nonzeros
S_Constraints             :1130
Variables                 :10699       :  Nonzeros


SOLUTION.RESULT

'Optimal solution found'

CPLEX 12.2.0.0: optimal integer solution within mipgap or absmipgap; objective 560
219113865 MIP simplex iterations
7216084 branch-and-bound nodes
absmipgap = 0.0528926, relmipgap = 9.4451e-05
642 cover cuts
25 flow-cover cuts
128 clique cuts
22 Gomory cuts
427 implied-bound cuts
40 zero-half cuts
211 mixed-integer rounding cuts


DECISION.VARIABLES

          Variable       Activity      U bound    ReducedCost
     ____________________________________________________________
1        'dYC[1]'            269.5     Infinity          0
2        'dYC[2]'            249       Infinity          0
3        'dYC[3]'            360       Infinity          0
4        'dYC[4]'            540       Infinity          0
5        'dYC[5]'            479       Infinity          0
6        'dYC[6]'            180       Infinity          0
8        'dYC[8]'            340       Infinity          0
9        'dYC[9]'             21       Infinity          0
11       'dYT[1]'            289.5     Infinity          0
12       'dYT[2]'            269       Infinity          0
13       'dYT[3]'            380       Infinity          0
14       'dYT[4]'            560       Infinity          0
15       'dYT[5]'            539       Infinity          0
16       'dYT[6]'            200       Infinity          0
17       'dYT[7]'             20       Infinity          0
18       'dYT[8]'            560       Infinity          0
19       'dYT[9]'             41       Infinity          0
20       'dYT[10]'            20       Infinity          0
22       'V[1,3]'              1              1          0
23       'V[1,4]'              1              1          0
24       'V[1,5]'              1              1          0
27       'V[1,8]'              1              1          0
```

30	'V[2,1]'	1	1	0
31	'V[2,3]'	1	1	0
32	'V[2,4]'	1	1	0
33	'V[2,5]'	1	1	0
36	'V[2,8]'	1	1	0
41	'V[3,4]'	1	1	0
42	'V[3,5]'	1	1	0
60	'V[5,4]'	1	1	0
66	'V[6,1]'	1	1	0
67	'V[6,2]'	1	1	0
68	'V[6,3]'	1	1	0
69	'V[6,4]'	1	1	0
70	'V[6,5]'	1	1	0
72	'V[6,8]'	1	1	0
75	'V[7,1]'	1	1	0
76	'V[7,2]'	1	1	0
77	'V[7,3]'	1	1	0
78	'V[7,4]'	1	1	0
79	'V[7,5]'	1	1	0
80	'V[7,6]'	1	1	0
81	'V[7,8]'	1	1	0
82	'V[7,9]'	1	1	0
87	'V[8,4]'	1	1	0
88	'V[8,5]'	1	1	0
93	'V[9,1]'	1	1	0
94	'V[9,2]'	1	1	0
95	'V[9,3]'	1	1	0
96	'V[9,4]'	1	1	0
97	'V[9,5]'	1	1	0
98	'V[9,6]'	1	1	0
100	'V[9,8]'	1	1	0
102	'V[10,1]'	1	1	0
103	'V[10,2]'	1	1	0
104	'V[10,3]'	1	1	0
105	'V[10,4]'	1	1	0
106	'V[10,5]'	1	1	0
107	'V[10,6]'	1	1	0
109	'V[10,8]'	1	1	0
110	'V[10,9]'	1	1	0
115	'Z[2,1,1]'	1	1	0
116	'Z[2,1,2]'	1	1	0
122	'X[1,3,2]'	1	1	0
140	'X[2,1,2]'	1	1	0
166	'X[3,5,2]'	1	1	0
198	'X[4,0,2]'	1	1	0
206	'X[5,4,2]'	1	1	0
222	'X[6,2,2]'	1	1	0
251	'X[7,8,1]'	1	1	0
277	'X[8,0,1]'	1	1	0
290	'X[9,6,2]'	1	1	0
316	'X[10,9,2]'	1	1	0
331	'X[0,7,1]'	1	1	0
338	'X[0,10,2]'	1	1	0
359	'Yijn[1,8,3]'	1	1	0
379	'Yijn[2,5,2]'	1	1	0
405	'Yijn[3,4,1]'	1	1	10000
456	'Yijn[4,0,1]'	1	1	0
487	'Yijn[5,0,2]'	1	1	0
495	'Yijn[6,3,1]'	1	1	10000
534	'Yijn[7,6,1]'	1	1	10000
578	'Yijn[8,0,3]'	1	1	0

581	'Yijn[9,1,3]'	1	1	0
613	'Yijn[10,2,2]'	1	1	0
657	'Yijn[0,7,1]'	1	1	0
665	'Yijn[0,9,3]'	1	1	0
667	'Yijn[0,10,2]'	1	1	0
990	'u[1,4,3,2]'	1	1	0
1090	'u[1,5,3,2]'	1	1	0
10669	'p[1]'	20	20	0
10670	'p[2]'	20	20	0
10671	'p[3]'	20	20	0
10672	'p[4]'	20	20	0
10673	'p[5]'	60	60	0
10674	'p[6]'	20	20	0
10675	'p[7]'	20	20	0
10676	'p[8]'	20	20	0
10677	'p[9]'	20	20	0
10678	'p[10]'	20	20	0
10679	'p1[1]'	20	20	0
10680	'p1[2]'	20	20	0
10681	'p1[3]'	20	20	0
10682	'p1[4]'	20	20	0
10683	'p1[5]'	40	40	0
10684	'p1[6]'	20	20	0
10685	'p1[7]'	20	20	0
10686	'p1[8]'	20	20	0
10687	'p1[9]'	20	20	0
10688	'p1[10]'	20	20	0
10693	'p2[5]'	20	20	0
10699	Makespan	560	Infinity	0

ANNEXE 2 : Qualité de la solution pour les instances : INS1, INS3, INS5 et INS8 (%)

Scénarios	INS1	INS3	INS5	INS8
Scénario 1	5.05	3.66	6.56	8.03
Scénario 2	5.42	3.02	4.25	7.26
Scénario 3	5.07	2.06	3.46	7.54
Scénario 4	4.10	2.76	3.67	7.30
Scénario 5	3.91	2.89	4.37	7.78
Scénario 6	3.84	2.38	5.01	6.05
Scénario 7	3.59	4.79	4.47	8.03
Scénario 8	2.70	2.71	5.73	6.96
Scénario 9	1.15	2.34	3.88	6.11
Scénario 10	1.75	4.64	3.43	7.65
Scénario 11	2.44	2.66	1.20	7.11
Scénario 12 (ALNS)	**0.76**	**1.73**	**1.27**	**4.45**