Ordonnancement des activités de manutention dans les terminaux portuaires

Ali Skaf

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Automatique

par

ALI SKAF

Ordonnancement des activités de manutention dans les terminaux portuaires

Thèse présentée et soutenue à Belfort, le 05 novembre 2020

Composition du Jury :

NICOD JEAN-MARC	Professeur, Ecole Nationale Supérieure de Mécanique et Microtechniques	Président
MOUKRIM AZIZ	Professeur, Université de Technologie de Compiègne	Rapporteur
SAUER NATHALIE	Professeure, Université de Lorraine	Rapporteur
MOALIC LAURENT	Maître de Conférences, Université de Haute Alsace	Examinateur
YALAOUI FAROUK	Professeur, Université de Technologie de Troyes	Examinateur
MANIER MARIE-ANGE	Maître de Conférences HDR, UTBM	Directrice de thèse
LAMROUS SID	Maître de Conférences, UTBM	Encadrant de thèse
HAMMOUDAN ZAKARIA	Maître de Conférences, Jinan University	Encadrant de thèse

N° 2020UBFCA019

école doctorale **sciences pour l'ingénieur et microtechniques**

Titre : Ordonnancement des activités de manutention dans les terminaux portuaires

Mots-clés : Grues de quai, camions, conteneurs, optimisation, ordonnancement, programmation linéaire à nombres entiers mixtes, algorithme numératif, algorithme génétique

Résumé :

L'ordonnancement des activités de manutention dans les terminaux maritimes a suscité beaucoup d'attention dans les recherches sur la gestion des opérations. Généralement, des conteneurs sont déplacés d'un port à un autre par des navires porte-conteneurs, déchargés à quai par des grues et transportés par des camions jusqu'à une zone de stockage. Pour obtenir des performances opérationnelles optimales, la coordination entre tous les équipements du port est un enjeu majeur. Nos travaux consistent à ordonnancer des opérations de chargement/ déchargement et de mise en zone de stockage de conteneurs par diverses ressources de manutention, avec comme application concrète le port de Tripoli-Liban. Ils s'articulent autour de trois scénarios. Le premier considère plusieurs grues de quai et un seul navire porte-conteneurs à décharger. Le navire est divisé en plusieurs baies, chacune contenant un nombre donné de conteneurs. Dans le deuxième scénario, nous considérons une grue de quai et un navire porte-conteneurs avec plusieurs camions de transport. Après déchargement des conteneurs du navire par la grue, ceux-ci doivent être transportés vers la zone de stockage par un camion. Enfin, dans cette zone, des chariots frontaux déchargent les conteneurs du camion vers une aire spécifique pour être livrés aux clients. Le troisième scénario prend en compte plusieurs grues, plusieurs camions de transport et deux navires porte-conteneurs afin de décharger les conteneurs du navire au lieu de stockage et vice versa. Pour ces trois variantes, nous appliquons une démarche de modélisation, élaboration d'algorithmes de résolution, tests et confrontation de nos résultats sur des instances de la littérature et issues de cas réels. Plusieurs méthodes exactes et approchée ont été explorées: la programmation linéaire, une méthode numérative, et une métaheuristique. Les avantages et inconvénients de ces méthodes sont analysés. Une conclusion sur les variantes étudiées et algorithmes développés est fournie en fin de manuscrit, et des perspectives à ces travaux sont ouvertes, avec en toile de fond l'objectif d'améliorer encore la gestion opérationnelle des manutentions dans des ports tels que le port de Tripoli-Liban.

Title: Ordonnancement des activités de manutention dans les terminaux portuaires

Keywords: Quay cranes, yard trucks, containers, optimization, scheduling, mixed integer linear programming, enumerative algorithm, genetic algorithm

Abstract:

Scheduling handling activities at maritime terminals has attracted much attention in research on operations management. Generally, containers are moved from one port to another by container vessels, unloaded by quay cranes and transported by yard trucks to a storage location. To obtain optimal operational performance, coordination between all of the port's equipments is a major issue. We aim at scheduling loading/unloading operations and placing containers in storage locations by various handling resources, with the practical application of the port of Tripoli-Lebanon. This study revolves around three scenarios. The first one considers several quay cranes and a single container vessel to unload. The vessel is divided into several bays and each bay contains a specific number of containers. In the second scenario, we consider a single quay crane and a single container vessel with several yard trucks. After being unloaded by the quay crane from the vessel, the containers are transported to the storage location by a yard truck. Then the reach-stacker cranes unload the containers from the yard truck to a specific area for delivery to customers. The third scenario includes several quay cranes, yard trucks and two vessels, in order to unload the containers from the vessel to the storage location and vice versa. To solve these three variants, we applied a modeling approach, development of solving algorithms, tests, analysis and comparison of our results on instances from the literature and real cases. Several exact and approximate methods have been explored: linear programming, enumerative method, and a metaheuristic. The advantages and disadvantages of these methods are highlighted. A conclusion on the variants studied and algorithms developed is provided at the end of the manuscript, and some perspectives for this work are opened, with the backdrop of the objective of further improving the operational management of handling in ports such than the port of Tripoli-Lebanon.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ
Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France

SOMMAIRE

REMERCIEMENTS

Tout d'abord, j'aimerais exprimer mon profond sentiment de gratitude envers ma directrice de thèse, Marie-Ange Manier, et mes encadrants Sid Lamrous et Zakaria Hammoudan. Merci pour leur soutien académique et personnel continu et leur mentorat au cours de ces trois années d'études. Merci pour leurs conseils, suggestions et dévouement tout au long de cette thèse. Ils sont parmi les rares personnes que je connais qui n'ont pas perdu un peu de leur enthousiasme étincelant pour leur travail. Travailler avec eux, c'était un défi intéressant, à la fois dur et amusant. Je voudrais remercier spécialement Zakaria Hammoudan pour sa collaboration étroite avec moi. Son soutien inconditionnel dans la résolution de nombreux détails entourant cette thèse et ses précieux commentaires sont très appréciés. Un grand merci à la Commission d'examinateurs, pour leur commentaires et suggestions constructifs qui ont conduit à une amélioration significative de la thèse.

J'ai fortement ressenti la foi incessante et le soutien de ma famille tout au long du chemin, j'aimerais profondément remercier mes parents, Ahmad et Moufida, mes frères Mouhamad et Abdallah, ma soeur Zaynab. Leur amour et leur soutien sont une source d'inspiration constante dans mes efforts pour m'améliorer. Cette thèse est dédiée à vous, spécialement à ma mère et à mon père. Cette thèse n'aurait pas été possible sans vos soutiens. De plus, je voudrais exprimer mes sincères remerciements à ma bien-aimée, Fadila et mon petit fils Ahmad pour leur amour, soutien et patience, me permettant d'atteindre mon objectif et de partager avec moi tous les moments heureux et tristes.
Merci de tout mon coeur au feu Mouhamad GHAMRAWI, qu'il repose en paix, d'être à mes côtés tous le temps et pour tout son appui.
Je voudrais également exprimer ma reconnaissance à tous mes amis et collègues du laboratoire et à l'université UTBM pour éclairer ma vie de tous les jours, spécialement Zaher AL-CHAMI, Mouhamad SLEIMAN et Mouhamad KHALIL.

LISTE DES PUBLICATIONS

Cette thèse a déjà été valorisée à travers 6 publications internationales, parues ou soumises, et contient du matériel non encore publié.

CONFÉRENCES INTERNATIONALES

1. Ali SKAF, Sid LAMROUS, Zakaria HAMMOUDAN, Marie-Ange MANIER, Exact method for single vessel and multiple quay cranes to solve scheduling problem at port of Tripoli-Lebanon. 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, Thailand, december 16-19, 2018, pp.457-461, 10.1109/IEEM.2018.8607546.
2. Ali SKAF, Sid LAMROUS, Zakaria HAMMOUDAN, Marie-Ange MANIER, Genetic algorithm to optimize unloading of large containers vessel in port of Tripoli-Lebanon. 6th 2019 International Conference on Control, Decision and Information (CoDIT 2019), Paris, France, april 23-26, 2019, pp.569-574, 10.1109/CoDIT.2019.8820367.
3. Ali SKAF, Sid LAMROUS, Zakaria HAMMOUDAN, Marie-Ange MANIER, Single quay crane and multiple yard trucks scheduling problem with integration of reach-stacker cranes at port of Tripoli-Lebanon. 2019 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2019), Bari, Italy, october 6-9, 2019, pp.852-857, 10.1109/SMC.2019.8914667.
4. Ali SKAF, Sid LAMROUS, Zakaria HAMMOUDAN, Marie-Ange MANIER, Mixed-integer linear programming model for the simultaneous unloading and loading processes in a maritime port. *Accepted in* The 2020 World Congress in Computer Science, Computer Engineering and Applied Computing (CSCE), Las Vegas, USA, july 27-30, 2020.

REVUES INTERNATIONALES

1. Ali SKAF, Sid LAMROUS, Zakaria HAMMOUDAN, Marie-Ange MANIER, Solving methods for the quay crane scheduling problem at port of Tripoli-Lebanon, soumis à RAIRO-Operations Research, 10.1051/ro/2020135.
2. Ali SKAF, Sid LAMROUS, Zakaria HAMMOUDAN, Marie-Ange MANIER, Integrated quay crane and yard truck scheduling problem at port of Tripoli-Lebanon, soumis à Computers and Industrial Engineering.

1

INTRODUCTION

1.1/ CONTEXTE

Le transport maritime est le mode de transport le plus important pour le transport de marchandises appelé communément marine marchande. Les opérations de manutention portuaire concernent les métiers de la logistique qui imposent un regard affûté et une rapidité de décision. La manutention portuaire, c'est avant tout des hommes d'expérience capables d'analyser, de concevoir et de réaliser les opérations les plus courantes, comme le chargement et le déchargement des bateaux. Pour garantir le bon déploiement de ces opérations, il est inévitable de modéliser une synchronisation en accord avec l'architecture globale du port. Il s'agit d'intégrer les activités de transport en arrière-plan liées au stockage et au déstockage de marchandises. Pour assurer le bon déroulement des opérations portuaires et éviter les embouteillages dans le port, il est inévitable de moderniser en permanence son infrastructure physique, d'investir dans le capital humain, de favoriser la connectivité du port et de mettre ses activités à niveau conformément aux normes en vigueur. Par conséquent, les opérations portuaires peuvent être définies par rapport aux installations associées et à l'ensemble des politiques, réformes et réglementations qui influent sur l'infrastructure. L'explosion au port de Beyrouth survenu le 04 Août 2020 avec 2750 tonnes de nitrate d'ammonium stockés dans un hangar constitue un exemple incontournable à citer. Le sens réglementaire comprend aussi les services de transport maritime. Plus de 80% du commerce mondial se fait par voie maritime, constituant de loin le moyen de transport de marchandises le plus important. Le transport maritime a connu une croissance annuelle d'environ 3,1% au cours des trois dernières décennies. Bien qu'il existe de nombreuses sociétés de transport maritime, la plupart d'entre elles sont petites et détiennent des parts de marché insignifiantes. Par exemple, 52% de la capacité mondiale en EVP en 2012 a été fournie par les dix plus grands opérateurs de services. L'EVP (Equivalent Vingt Pieds), est l'unité dans laquelle sont exprimés les trafics conteneurisés et la capacité des navires porte-conteneurs.

En 2015, les trois premiers opérateurs (Maersk Line, Danemark ; MSC, Suisse ; et le groupe CMA-CGM, France) fournissent un total de 5 291 145 EVP, soit environ 30% de la capacité totale en EVP de la planète. Même si les plus grandes entreprises de transport maritime sont situées dans des économies développées, leur flotte est généralement enregistrée dans les pays en développement. Panama et le Libéria, les deux principaux registres, représentent un tiers du tonnage de port en lourd dans le monde.

Les conteneurs permettent de transporter des marchandises sans les reconditionner des fournisseurs à leurs clients, et sont donc très utiles. Les navires porte-conteneurs qui

les transportent peuvent charger jusqu'à 13 000 conteneurs. Nous avons réalisé une étude sur des données fournies par une presse spécialisée dans le domaine maritime [AAPA, 2018] et [Lloyd's-List, 2018] pour superviser sur une cartographie mondiale la concentration des plus grands grands ports à conteneurs au monde (Figure 1.1). La donnée utilisée est le nombre de conteneurs exprimé en millions d'EVP. Ce nombre tient compte des conteneurs arrivant au port et partant du port, en charge ou à vide. Cette donnée est moyennée sur quatre années (2015 à 2018). Nous remarquons distinctement une dominance du marché asiatique (Chine, Singapour, Corée du sud, Hong Kong, Malaisie et Taïwan), suivie des États-Unis, Émirats arabes unis, Pays-bas, Belgique, Allemagne, Espagne, etc.

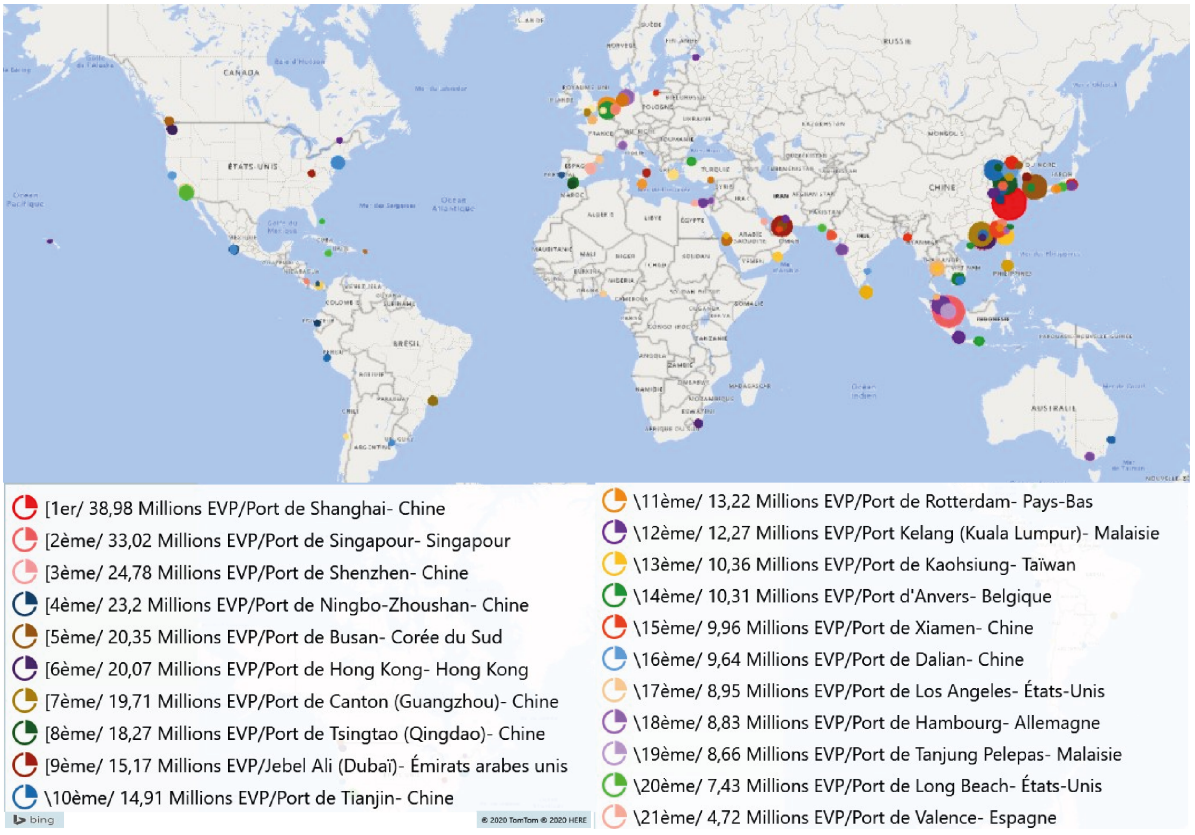

FIGURE 1.1 – Les plus grands ports à conteneurs au monde. Étude basée sur une moyenne EVP sur les années (2015 à 2018)

Les services disponibles pour les commerçants et les navires, ainsi que la qualité des services concernant la rapidité, la fiabilité, la fréquence, la sécurité, revêtent une importance croissante dans le contexte de processus de production mondialisé et de livraisons juste à temps. Il existe de grandes différences entre les pays en ce qui concerne la qualité et les coûts des services portuaires et de navigation mis à la disposition de leurs importateurs et exportateurs, par exemple le remorquage, l'amarrage, etc. Mais les infrastructures et la connectivité existantes revêtent également une importance majeure pour que les commerçants puissent amener leurs marchandises à destination.

La mondialisation croissante des échanges et la grande complexité des opérations portuaires nécessitent l'application d'un système informatique sophistiqué. Ces dernières années, la taille des navires a doublé, ce qui a compliqué la gestion des opérations

portuaires et exigé un effort logistique encore plus important. La tendance vers la fabrication juste à temps nécessite une amélioration permanente du flux d'informations et l'intégration de l'activité de transport dans le processus de production.

Le transport maritime et la sécurité portuaire sont devenus une préoccupation majeure, en particulier après les attaques terroristes du 11 septembre. Les mesures prises pour renforcer la sécurité, telles que le code ISPS (International Ship and Port Facility), ont eu un impact significatif sur le commerce et les opérations portuaires. Dans le cadre de l'initiative SOLAS (Sauvegarde de la vie humaine en mer), il s'agit d'un ensemble complet de mesures visant à renforcer la sécurité du port et des navires dans le port. Le programme SOLAS oblige les pays à évaluer en permanence leur situation en matière de sécurité qui correspond à un ensemble de mesures de sécurité à prendre par les navires, les ports et les autorités portuaires. Un défi majeur concerne le piratage dans certaines régions du monde. Bien qu'il existe un accord général sur les menaces que les pirates et les terroristes font peser sur le commerce international, de multiples parties prenantes poursuivent des intérêts différents.
L'organisation mondiale des douanes (OMD) a adopté le « Cadre de normes pour sécuriser et faciliter le commerce mondial» (Cadre SAFE) afin d'accroître la sécurité des marchandises. Ce cadre fournit des directives et des normes visant à harmoniser les opérations douanières nationales, à renforcer leur coopération et à établir des partenariats entre les douanes et les entreprises. L'un des enjeux majeurs, en particulier parmi les pays en développement, reste la reconnaissance mutuelle des opérateurs économiques agréés d'autres pays.

La CNUCED (Conférence des Nations unies sur le Commerce et le Développement, organisme créé en 1964 et qui rassemble tous les pays de l'ONU (193 pays) avec pour objectif de favoriser le développement du Tiers-monde) a une longue expérience dans la fourniture d'une assistance technique concernant les aspects commerciaux de la gestion des transports maritimes et des ports. Cela inclut les politiques et pratiques de tarification portuaire et de transport maritime. L'objectif est d'améliorer la compétitivité, en particulier des économies en développement, et, en définitive, de promouvoir et de faciliter le commerce mondial.

Comme nous l'avons évoqué précédemment, les flux de production de plus en plus tendus impactent aussi sur les flux de transport dont les temps deviennent également un enjeu majeur. Par ailleurs, les coûts du transport maritime sont un élément fondamental influant dans la balance concurrentielle entre les modes de transport. Temps et coûts sont donc des critères fondamentaux qu'il est critique de minimiser. En particulier, le temps passé à quai par chaque navire pour le déchargement et le chargement des conteneurs doit être réduit le plus possible, et ce d'autant plus que cela peut engendrer des files d'attente d'autres navires à l'entrée du port. Ainsi différents types de ressources de manutention sont nécessaires pour le chargement et le déchargement des conteneurs dans les ports tels que grues de quai, camions, chariots ...

La Recherche Opérationnelle (RO) est la discipline permettant de mettre en place des modélisations appropriées dans le contexte portuaire. C'est une discipline qui favorise le développement des outils d'aide à la décision et des méthodes scientifiques permettant de résoudre des problèmes complexes en les solutionnant vers de meilleures décisions. La résolution des problèmes d'optimisation a pour objectif selon le contexte, de minimiser ou de maximiser un ou plusieurs critères en respectant certaines conditions

dites contraintes. En tenant compte du compromis entre le temps nécessaire pour obtenir des solutions et la qualité de celles-ci, la résolution adéquate peut être utilisée avec des méthodes exactes ou approchées. Pour résoudre un même problème d'optimisation, plusieurs algorithmes peuvent être développés avec des complexités significativement distinctes. La complexité d'un problème est qualifiée par l'algorithme de plus faible complexité qui permet de le résoudre en termes de rapidité de résolution.

Les algorithmes exacts sont des méthodes qui permettent de répondre de façon exacte un problème. Dans cette lignée d'algorithmes, on pourrait privilégier l'énumération exhaustive de toutes les solutions possibles quand la complexité le permet. Ou fort heureusement, il existe des méthodes plus souples comme la programmation dynamique introduite par [Bellman, 1957]. Cette approche se base sur le principe que pour résoudre l'optimalité d'un problème, on passe par la résolution de l'optimalité de ses sous-parties. Contrairement aux algorithmes exacts, les métaheuristiques ne nécessitent pas d'analyses théoriques pour démontrer ou borner la qualité des solutions qu'elles obtiennent. Ces algorithmes peuvent être très efficaces et résoudre des problèmes de très grande taille dans des délais très concurrentiels. L'idée de tels algorithmes provient souvent de l'observation de la nature ou de la vie courante comme par exemples les algorithmes génétiques ou les colonies de Fourmis.

1.2/ OBJECTIFS ET CONTRIBUTIONS

L'objectif principal de cette thèse concerne la réduction du temps d'attente (et donc des coûts) des navires porte-conteneurs à quai (dans les terminaux portuaires), en exploitant des méthodes de Recherche Opérationnelle. Notre travail consiste à résoudre un problème d'optimisation combinatoire, faisant partie de la classe connue dans la littérature sous la dénomination *Quay Crane Scheduling Problem*. Il s'agit plus précisément d'ordonnancer les activités de transfert des conteneurs entre un navire porte-conteneurs et une zone de stockage portuaire, afin d'en minimiser la durée totale. Notons que le sujet de cette thèse a été conçu suite à des échanges avec le port de Tripoli au Liban. Même s'ils n'ont pas abouti à un partenariat officiel, ils ont permis d'identifier les besoins réels de ce type de secteur d'activités, et de générer quelques instances issues de cas réels utiles pour confronter le résultat de nos recherches.

Les principales contributions de cette thèse s'appesantissent sur le fil conducteur suivant :

— Modélisation mathématique du problème d'ordonnancement des grues de quai (*Quay Crane Scheduling Problem*) sous trois versions. Dans la première version dite de base, des grues de quai sont affectées au déchargement d'un seul navire porte-conteneurs, en tenant compte de l'évitement de collisions. Dans une deuxième version, un navire porte-conteneurs desservi par une seule grue de quai est interfacé avec plusieurs camions intégrant des contraintes temporelles. Enfin, dans une généralisation du modèle précédent, plusieurs grues de quai ainsi que plusieurs camions de transport et chariots frontaux sont intégrés dans une troisième version. La résolution algorithmique est étayée sous deux lignées d'algorithmes, exacte et métaheuristique. L'objectif formulé étant la minimisation du temps de manutention de tous les conteneurs, des expérimentations sont menées pour la validation de la qualité des solutions et de la performance de résolution.

— Résolution des modèles par méthodes exactes et approchée. La résolution algorithmique est basée sur un algorithme numératif, et un algorithme génétique. Ce modèle permet une résolution du problème d'affectation des conteneurs aux grues de quai et aux camions de transport.
— Proposition de bornes inférieures et bornes supérieures pour le critère considéré.
— Génération de jeux d'instances pour montrer la pertinence de la modélisation, et des algorithmes développés.

1.3/ STRUCTURE DE LA THÈSE

Outre cette introduction, ce mémoire comporte quatre chapitres. Le chapitre 2 dresse un état de l'art du domaine. Une première partie est dédiée au métier pour décrire les problèmes de décision et de planification dans le monde portuaire. Celle-ci traite d'abord des terminaux maritimes et des équipements avec un focus sur le port de Tripoli. Puis, elle donne une description des opérations portuaires et des problèmes d'ordonnancement dans les ports. Une seconde partie est consacrée à la littérature liée aux méthodes de résolution.

Le chapitre 3 montre une modélisation et une résolution pour l'ordonnancement de grues de quai dédiées au déchargement d'un porte conteneur. Après une modélisation mathématique, trois méthodes de résolution sont présentées dont deux exactes s'appuyant sur la programmation linéaire en nombres entiers et un algorithme numératif. Aussi, une méthode de résolution approchée basée sur un algorithme génétique est développée. Les propositions sont validées sur la base d'instances générées aléatoirement puis comparées à la littérature. Des exemples concrets liés au port de Tripoli font partie des jeux de test.

Le chapitre 4 traite le problème d'ordonnancement d'une seule grue de quai qui décharge des conteneurs d'un navire, et aliment plusieurs camions destinés à transporter la marchandise en zone de stockage. Pour ce problème, nous présentons également une résolution basée sur les mêmes méthodes qu'au chapitre 3. De plus, nous proposons des bornes inférieures et supérieures pour le critère considéré. Le protocole de validation suit le même fil conducteur que celui du chapitre 3.

Le chapitre 5 développe une extension des deux précédents. Il étudie l'ordonnancement de plusieurs grues de quai et plusieurs camions de transport, à la fois pour le chargement et le déchargement des conteneurs. Pour ce problème nous présentons une formulation mathématique de type programmation linéaire en nombre entiers et confrontons nos résultats avec ceux du port de Tripoli.

Nous terminons ce mémoire de thèse par un bilan des travaux menés et des résultats obtenus, et nous ouvrons des perspectives pour de futures recherches.

2

ÉTAT DE L'ART

2.1/ LES TERMINAUX MARITIMES

2.1.1/ CONTEXTE

Dans le domaine du transport intermodal, la voie maritime est largement utilisée pour transporter des milliers de tonnes de marchandises. Les produits sont chargés dans des conteneurs de dimensions standardisées, qui sont ensuite embarqués sur des bateaux assurant la liaison entre les ports. Le transport terrestre, permettant de lier ces ports aux origines et/ou destination des marchandises, s'effectue ensuite par trains ou camions, voir par mode fluvial. Les ports de commerce et les terminaux constituent donc le lieu d'interface terre-navires. Ils forment de larges plateformes logistiques où vont transiter les marchandises et où les conteneurs vont subir un ensemble d'opérations de manutention et de stockage.

De manière générale, de nombreux problèmes de décision/planification doivent être résolus, parmi lesquels on distingue les problèmes liés à l'arrivée des bateaux dans un port, au chargement/déchargement de ces bateaux, au transport des conteneurs vers un emplacement de stockage (et vice versa) ([Vis et al., 2003], [Steeken et al., 2004]). Décomposés en niveaux de décision, les problèmes associés sont :

- Au niveau stratégique : le dimensionnement des quais (nombre d'emplacements disponibles) ; le dimensionnement et l'implantation du stockage des conteneurs ;
- Au niveau tactique : l'affectation des emplacements de quais aux bateaux qui arrivent au port (berth allocation), la détermination du nombre de palans de manutention affectés simultanément à un bateau (pour minimiser le temps passé par le bateau au terminal), et du nombre de véhicules nécessaires pour déplacer les conteneurs dans les délais impartis ; le calcul du nombre de ressources de manutention au niveau du stock (dépôt et retrait) (*yard cranes* ou ASCs for automated cranes)
- Au niveau opérationnel : le chargement et déchargement des conteneurs par des palans de manutention (*quay crane scheduling*), en tenant compte de l'emplacement des conteneurs sur le bateau, des risques de collision des palans qui circulent sur un rail unique ; l'affectation et le routage des véhicules transportant les conteneurs entre le quai et le stock (pour minimiser la distance parcourue, les distances à vide, le temps de transport. . .) ; l'ordonnancement des ressources de manutention au niveau du stockage/déstockage des marchandises.

Dans le cadre de cette thèse, nous nous sommes intéressés plus particulièrement aux

décisions de type opérationnel, tant sur les quais qu'en arrière-plan, avec la prise en compte de la synchronisation entre les diverses activités de manutention. Les problèmes posés à ce stade s'apparentent à des problèmes rencontrés dans d'autres domaines (ordonnancement d'ateliers et tournées). Par exemple, le *Quay Crane Scheduling Problem* est similaire au *Hoist Scheduling Problem* qui concerne l'ordonnancement d'ateliers comportant plusieurs ressources de transport (palans de manutention empruntant un rail unique).

Ces types de problèmes sont des problèmes d'optimisation combinatoire dont les variantes dites de base ont été démontrées comme étant complexes. De nombreux chercheurs se sont déjà penchés sur ces questions et ont proposé des méthodes de résolution exactes ou approchées basées sur la Recherche Opérationnelle (programmation linéaire, heuristiques et métaheuristiques diverses). Pour autant, les instances complexes plus conformes à la réalité sont encore loin d'être résolues de manière satisfaisante, et de nombreuses variantes constituent encore à ce jour des terrains d'exploration prometteurs ([Boysen et al., 2016]).

Dans ce contexte, notre objectif est donc de pouvoir appréhender, modéliser et résoudre ce type de variante de manière efficace (tant en termes d'objectif(s) à atteindre qu'en termes de temps de résolution). Dans cette section, nous commençons par décrire les éléments du système physique étudié dans le cadre d'un terminal portuaire. Nous fournissons aussi à titre d'exemple quelques données concernant le port de Tripoli au Liban, qui nous a permis de générer des instances réalistes sur lesquelles tester nos algorithmes. Dans un second temps nous détaillons les processus logistiques associés à la manutention des conteneurs dans les ports et les problèmes d'optimisation associés. Nous définissons en particulier les problèmes d'ordonnancement des tâches de chargement/déchargement des bateaux et mise en stock des conteneurs, pour lesquels nous effectuons un état des lieux des recherches dans la littérature. Pour terminer ce chapitre, nous fournissons un rapide panorama des méthodes de résolutions utilisées en recherche opérationnelle, dans lesquelles nous avons puisé pour développer et proposer des algorithmes de résolution pour différentes variantes de problèmes d'ordonnancements dans les ports.

2.1.2/ LES ÉQUIPEMENTS D'UN TERMINAL PORTUAIRE

La deuxième moitié du XXème siècle a vu la modernisation du transport maritime, avec en particulier l'automatisation des opérations et la mondialisation, se traduisant notamment par l'apparition des conteneurs sur le marché et l'accélération des mouvements portuaires. Les mouvements de marchandises à l'intérieur des ports sont aujourd'hui réalisés par de multiples ressources dont nous décrivons ici succinctement une partie d'entre eux. Pour une description plus détaillée, voir le rapport de [Dubreuil, 2008].

Le système que nous étudions est un terminal porte-conteneurs (maritime), qui comporte ordinairement un quai pour l'amarrage des bateaux, des portiques, grues et chariots frontaux/cavalier, et une surface consacrée au stockage des conteneurs qui peuvent être empilés. Le terminal comporte également des réseaux de transport permettant l'intermodalité (routes et voies ferrées), qui est hors des limites de nos travaux. Notons que les plus grands terminaux porte-conteneurs sont situés dans les principaux ports mondiaux. Les terminaux à conteneurs terrestres sont eux, plutôt situés proches ou à l'intérieur des villes majeures, avec de bonnes connections ferroviaires aux terminaux porte-conteneurs.

Les conteneurs sont les entités physiques manipulées dans les ports. Ce sont des caissons métalliques parallélépipèdes conçus pour le transport de marchandises, dont les dimensions ont été normalisées au niveau international (Figure 2.1). Les conteneurs de 20 et 40 pieds de long (environ 6 et 12 mètres) sont les plus utilisés. Presque toutes les marchandises peuvent faire l'objet d'un transport par conteneur. Ce sont essentiellement des biens d'équipement et de consommation plus ou moins élaborés qui empruntent cette voie, mais le conteneur s'ouvre également aux produits en vrac si ce choix présente des avantages : logistique facilitée pour les petits lots de céréales, opérations de transvasement réduites pour les produits chimiques, possibilité de services porte à porte pour les fruits...([ISEMAR, 2002]).

FIGURE 2.1 – Conteneurs

Les conteneurs sont transportés par voie maritime sur des bateaux appelés porte-conteneurs, qui sont spécialement conçus pour contenir de grandes quantités de conteneurs (Figure 2.2). Les premiers porte-conteneurs apparaissent en 1956. Depuis leur mise en service, on assiste à un accroissement de la taille et de la capacité de ces navires. Inauguré en 2019, le MSC Gülsün est le plus grand porte-conteneurs du monde. Il atteint 400 mètres de long, dépassant en longueur la hauteur de la tour Eiffel, et il peut embarquer, par exemple, 386 millions de paires de chaussures ou 47 512 voitures. . .

FIGURE 2.2 – Porte-conteneurs

Ce nouveau mode de transport des marchandises a révolutionné le transport maritime et requiert des moyens de manutention adaptés. Le transbordement de la cargaison à bord de navires ou à quai est effectuée à l'aide de grues de quai. Il s'agit d'un type de grue portique qui peut parcourir la longueur d'un quai sur une voie ferrée (Figures 2.3 et 2.4). Ces portiques sont l'instrument obligatoire pour charger et décharger les navires.

On en met jusqu'à six par navires selon la quantité de conteneurs à manipuler. Le rythme des portiques peut varier de 20 à 35 manutentions à l'heure. Le flux est donc rapide car le navire ne reste à quai que quelques heures. Un poste équipé de quatre portiques peut ainsi manipuler près de cinq millions de tonnes de marchandises par an. D'autres engins de manutention sont également utilisés. Par exemple des camions de transport (*yard trucks*, Figure 2.5) servent à transporter les conteneurs, depuis la grue de quai vers l'entrée de la zone de stockage où des chariots cavalier (*straddle carriers*, Figure 2.9), ou des chariots frontaux (*reach-stackers*, Figure 2.8) récupèrent les conteneurs pour les placer (éventuellement en les empilant sur plusieurs hauteurs) pour quelques heures ou quelques jours, en attendant que se présentent les camions ou les trains qui doivent les emmener vers leur destination finale. On distingue aussi des véhicules à guidage automatique (Figure 2.10), des chariots à fourches (Figure 2.7), ... aussi employés pour le transport d'un seul conteneur. Néanmoins il existe également des systèmes multi-remorques capables de déplacer simultanément plusieurs conteneurs (Figure 2.6).

FIGURE 2.3 – Grue de quai

FIGURE 2.4 – Grue portique

FIGURE 2.5 – Camion de transport

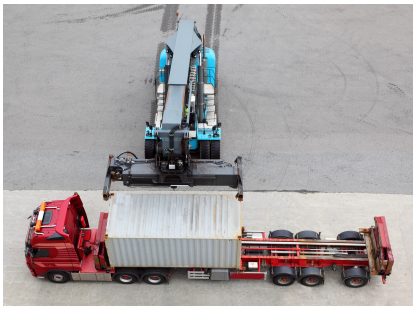

FIGURE 2.6 – Multi-remorque

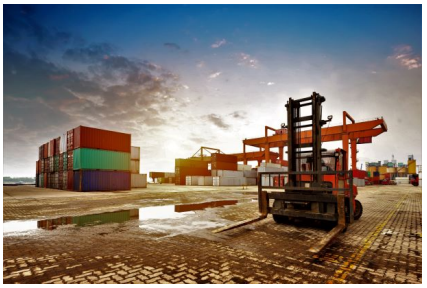

FIGURE 2.7 – Chariot à fourches

FIGURE 2.8 – Chariot frontal

FIGURE 2.9 – Chariot cavalier

FIGURE 2.10 – Véhicule à guidage automatique

Notons que les grues à conteneurs et les camions bennes sont conduits par des professionnels qualifiés. Ils sont chargés de vider les conteneurs du port ou de les charger sur des cargos. De cette façon, une équipe de plus d'une douzaine de personnes est nécessaire, au lieu de plusieurs inspecteurs pour superviser l'opération pour s'assurer que tout se passe bien. Ces inspections sont notamment liées à la détermination de la quantité et du poids, à la confirmation de l'emballage et de l'étiquetage, au scellage de la cargaison, au contrôle de la température, à la traçabilité des conteneurs, au contrôle opérationnel du chargement et du déchargement, et à la préparation des rapports et documents de suivi des opérations.

2.1.3/ CAS D'ÉTUDE : LE PORT DE TRIPOLI-LIBAN

Au début de la thèse, nous avons eu l'opportunité de visiter le port de Tripoli au Liban et de discuter avec des responsables opérationnels. Cela explique que nous avons pris par la suite ce port comme cas d'étude concret et que nous avons pu confronter nos résultats avec ses performances actuelles. Nous donnons donc ici à titre d'exemple quelques caractéristiques du port de Tripoli.

Le port de Tripoli est le deuxième port du Liban après le port de Beirut. Il possède une superficie approximative de trois millions de m^2, une surface d'eau de 2.200.000 m^2, une surface d'atterrissage de 320.000 m^2 et une surface pour le déchargement de 420.000 m^2 adjacente au port actuel, réservée au futur terminal des conteneurs et à la zone marché libre (Figure 2.11).

FIGURE 2.11 – Vues aériennes du port du Tripoli-Liban

Le port de Tripoli est indépendant sur le plan administratif et financier et il est régi par le code général des institutions publiques, conformément au décret n ° 4513. Il est géré

par un conseil d'administration composé de 5 membres élus pour 3 ans. Ce port dispose actuellement d'un quai et de 8 postes d'amarrage dont la profondeur varie de 8 à 10 mètres. Il reçoit environ 450 navires chaque année, soit une moyenne de 37 navires par mois. La plupart des navires transporte des marchandises générales et des rejets à sec tels que le fer, le bois, le sucre, divers types de haricots, de la ferraille, des véhicules et des matériaux de construction. Le port de Tripoli comprend également une zone libre avec une surface de 150.000 m^2. Actuellement, il fait l'objet de travaux d'expansion. Une nouvelle surface d'amarrage de 600 m de long est en construction pour le commerce de conteneurs, avec une zone arrière de 1 200 000 m2. Cette zone a été approuvée par le parlement libanais en tant que zone économique libre.

Le port de Tripoli présente plusieurs caractéristiques principales :

— La présence d'une zone arrière d'une surface atteignant 1 200 000 m2.
— La construction d'un nouveau poste d'amarrage d'une profondeur de 15 m et d'une longueur de 600 m dans une phase initiale, avec une extension de longueur jusqu'à 1200 m dans une seconde phase.
— La présence de zones de stockage à l'intérieur du port, de la zone libre et de la zone libre économique.
— La disponibilité de grandes zones à proximité du port pour construire des ports secs corrélés.
— L'emplacement du port à seulement 30 km de la frontière syrienne et donc accessible au Golfe et au monde arabe.
— De faibles taxes et tarifs, des frais réduits pour la main-d'oeuvre embauchée.

Le tableau 2.1 fournit quelques informations plus techniques sur le port de Tripoli. Nous présentons également une liste des services et installations de ce port.

TABLE 2.1 – Détails du port

Type	**Profondeur**
Mouillage	17,1–18,2 m
Poste de chargement	7,1–9,1 m
Profondeur de la station d'huile	17,1–18,2 m
Cale sèche	-
Taille du port	Moyen
Taille du rail	-
Type de port	Brise-lames côtier
Correctifs	Modéré
Refuge	Bien

Entrepôts et places

— 4 entrepôts pour produits de drainage sec, d'une surface de 11.000 m^2
— 10 entrepôts pour le drainage sec et le bois, d'une surface de 17. 500 m^2
— 5 verges pour stocker les véhicules, d'une surface supérieure à 10. 000 m^2
— 1 cour pour stocker les conteneurs, d'une surface de 10.000 m^2
— 1 cour à usage général, d'une surface de 15.000 m^2
— 2 verges d'une surface de 3000 m^2 pour stocker le bois de sapin

Équipement portuaire

— 6 grues mobiles d'une capacité de 125 à 165 tonnes

— 7 grues mobiles d'une capacité de 100 à 120 tonnes
— 10 grues mobiles d'une capacité de 70 à 90 tonnes
— 20 grues mobiles d'une capacité de 40 à 65 tonnes
— 11 grues mobiles d'une capacité de 25 à 38 tonnes
— 2 grues de quai
— 15 caisse pour les produits de drainage
— 24 chariots élévateurs
— 8 bulldozers
— 30 camions
— 6 camions de transport pour les conteneurs
— 2 chariots frontaux
— 4 tracteurs
— Équipement pour l'arrimage des biens de drainage sec

Autre services

— 8 points d'alimentation d'eau pour les navires en utilisant des technologies modernes
— cafétérias
— Un hôpital (en construction)
— Un bureau pour le syndicat des travailleurs

2.2/ LES OPÉRATIONS PORTUAIRES

2.2.1/ LES PROCESSUS LOGISTIQUES

Les terminaux à conteneurs dans un port maritime sont constitués de plusieurs processus logistiques connectés entre eux, de l'arrivée d'un navire porte-conteneurs jusqu'à la livraison des conteneurs vers la zone de stockage et vice-versa (Figure 2.12).

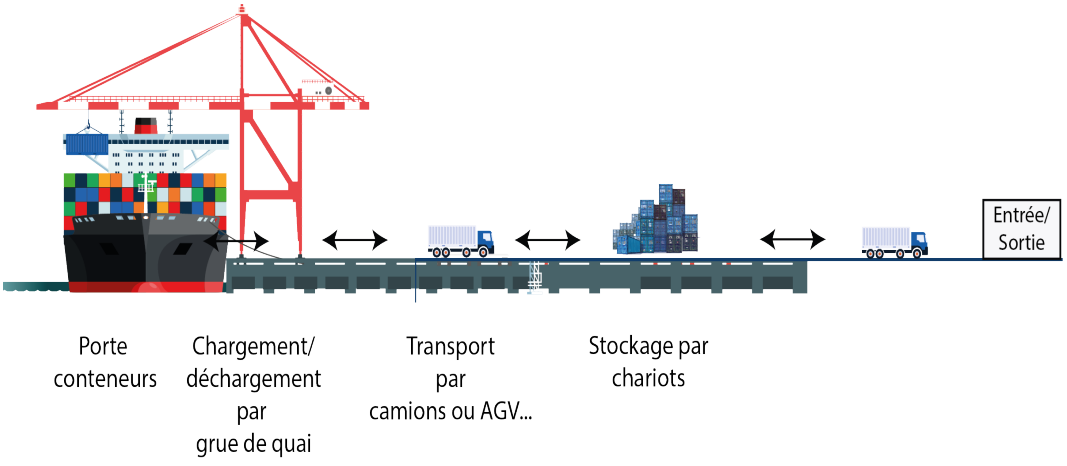

FIGURE 2.12 – Processus de chargement et déchargement des conteneurs

Les navires porte-conteneurs sont déchargés et chargés dans les terminaux maritime, le système de chargement et déchargement des conteneurs pouvant être divisé en plu-

sieurs processus : lorsque le navire arrive au port, les grues de quai (Quay cranes) déchargent les conteneurs et les déposent sur les camion de transport ; puis, les camions les transfèrent vers la station des chariots frontaux ; enfin, chaque conteneur sera déchargé du camion de transport à l'aide du chariot frontal vers la zone de stockage.

Il existe plusieurs indicateurs utiles pour évaluer la performance des processus de chargement et déchargement des conteneurs dans les terminaux maritimes :

- le nombre de matériels de manutention requis à chaque étape du processus : nombre de grues de quai, de camions, de chariots frontaux/cavaliers, etc.
- le nombre de mouvements des grues de quai.
- le temps et/ou le coût requis pour charger/décharger les conteneurs.
- le temps et/ou le coût qu'un camion de transport reste dans le terminal.
- le temps et/ou le coût qu'un conteneur reste dans le terminal avant être transporté.

Les objectifs à atteindre dans les ports maritimes sont généralement :

- de minimiser le temps total de déchargement/chargement des conteneurs (et/ou le coût total).
- d'optimiser le nombre des mouvements des grues de quai.

Pour optimiser les performances au niveau opérationnel, certaines décisions doivent être considérées afin de gérer toutes les opérations et leurs répercussions. Par exemple, le choix de l'emplacement des zones de stockage des conteneurs affecte directement l'affectation des grues de quai et ses conducteurs.

2.2.2/ PROBLÈMES DE PLANIFICATION DES ACTIVITÉS PORTUAIRES

La figure 2.13 illustre les processus dans les terminaux à conteneurs dans les ports maritimes, que nous avons définis précédemment. Dans cette partie, nous étudions plus en détail les problèmes relatifs à ces processus et rapportons quelques travaux associés.

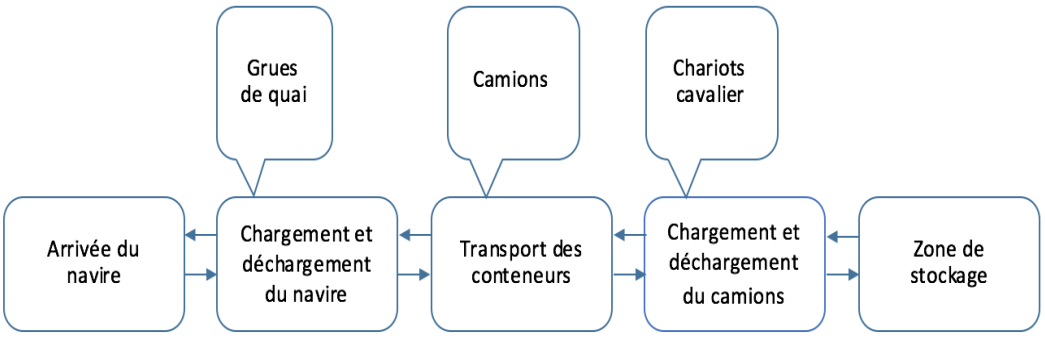

FIGURE 2.13 – Processus dans les terminaux à conteneurs

2.2.2.1/ ARRIVÉE DU NAVIRE PORTE-CONTENEURS

Lorsqu'un navire porte-conteneurs arrive au port, il doit être assigné à un poste d'amarrage (berth) pour s'attacher au quai. Le problème d'allocation des postes d'amarrage (Figure 2.14) consiste à affecter chaque navire porte conteneurs à un poste d'amarrage disponible durant un créneau horaire prévisionnel (permanence). La période dite de permanence d'un bateau dépend du temps de déchargement des conteneurs par les grues

de quai (Quay crane). Notons que l'occupation des quais par les bateaux constitue un enjeu important dans la gestion opérationnelle des ports. Le nombre limité de places dans les terminaux oblige certains navires à diminuer leur vitesse de croisière pour éviter un engorgement à l'entrée des ports. Cela leur évite de rester en attente à l'extérieur des ports tout en permettant une économie de carburant.

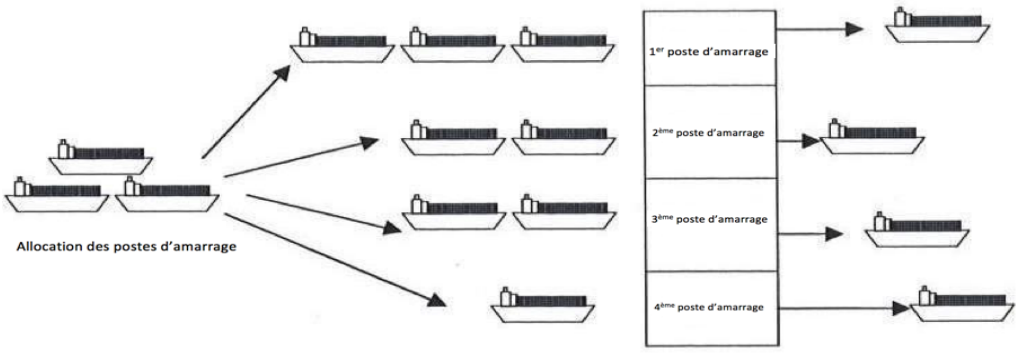

FIGURE 2.14 – Affectation des postes d'amarrages

L'objectif de l'affectation des postes d'amarrage est de prévoir le lieu et le créneau horaire nécessaire pour réaliser toutes les opérations des bateaux dans le temps accordé par la compagnie des navires porte-conteneurs et l'opérateur des terminaux ([Kim, 2008]).

Le problème scientifique associé est connu sous l'appellation *Berth Allocation Problem (BAP)*, et il a donné lieu à diverses études telles que celles citées ci-après.
([Imai et al., 1997]) ont proposé un algorithme qui affecte des postes d'amarrages aux navires porte-conteneurs, en minimisant leurs temps de stationnement et l'insatisfaction obtenue suite à une avance ou un retard d'arrivée au terminal.
[Kim et al., 2003] ont présenté un modèle mathématique MIP (Mixed Integer Linear Problem) pour l'affectation des postes d'amarrage. Leur objectif est la réduction du coût produit par une gestion des postes d'accostage non optimale. Les résultats expérimentaux prouvent que l'algorithme de recuit simulé donne des solutions identiques aux solutions optimales obtenues par le modèle MIP.
[Park et al., 2003] ont présenté un programme linéaire en nombres entiers pour ordonnancer des grues du quai et les postes d'amarrage en même temps. Ils ont divisé le problème en deux phases, la première pour l'ordonnancement des grues de quai et la deuxième pour l'ordonnancement des postes d'amarrage. Ils ont développé une méthode de programmation dynamique pour obtenir la solution optimale en première phase.
[Imai et al., 2008] et [Liang et al., 2009] ont résolu ce problème d'affectation grâce à un algorithme génétique. L'objectif de [Liang et al., 2009] est de minimiser le temps de manutention, le temps d'attente, et le temps de retard pour chaque navire porte-conteneur.
Ce problème reste d'actualité aujourd'hui, notamment dans les terminaux de transbordement automobile. Par exemple, [Dkhil et al., 2020] ont considéré le problème de définition des positions d'amarrages des navires entrant dans le terminal Ro-Ro (Roll-on/roll-of ou roulier qui est un navire utilisé pour transporter entre autres des véhicules, chargés grâce à une ou plusieurs rampes d'accès.), afin de minimiser le temps total de manutention tout en considérant plusieurs contraintes opérationnelles.

2.2.2.2/ CHARGEMENT ET DÉCHARGEMENT DE NAVIRE PORTE-CONTENEURS

En général, il est nécessaire de déterminer à l'avance un plan pour le chargement et le déchargement, qui indique la manière de positionner les conteneurs sur un navire, et qui précise quels conteneurs sont à décharger et leur positionnement sur le navire. Pour assurer un transport global efficace et rapide, les conteneurs sont souvent regroupés par type pour optimiser le temps de chargement et déchargement. Leur positionnement dans les baies des porte-conteneurs (rangées) est un problème à part entière, sur lequel des chercheurs se sont concentrés, tels que ceux cités ci-après.
D'après [Shields, 1984], les conteneurs chargés sur le navire porte-conteneur doivent satisfaire des contraintes attachées aux limites physiques du navire porte-conteneurs et des conteneurs et à l'ordre des ports destinataires.
[Avriel et al., 1993] et [Avriel et al., 1998] ont présenté un modèle mathématique pour la planification du stockage des conteneurs dans un navire porte-conteneurs. L'objectif est de minimiser le nombre de déplacements de conteneurs pour atteindre un conteneur particulier. En effet, les conteneurs à bord d'un navire sont placés dans des piles verticales, situées dans de nombreuses baies. Étant donné que l'accès aux conteneurs se fait uniquement par le haut de la pile, une situation courante est que les conteneurs désignés pour le port B doivent être déchargés et rechargés au port A (avant B) afin d'accéder aux conteneurs situés en dessous d'eux, désignés pour le port A.
[Wilson et al., 2000] a confirmé qu'il est difficile de déterminer une solution optimale du problème de chargement et déchargement des conteneurs et impossible à réaliser dans des délais acceptables. Ce type de problème est considéré comme un problème complexe. Les auteurs l'ont décomposé en deux phases : la première dépend de la capacité du navire et la deuxième dépend du nombre de conteneurs chargés et déchargés à chaque port. Ils ont proposé un algorithme par séparation et évaluation pour affecter chaque conteneur à un bloc dans le navire porte-conteneurs et la recherche Tabou pour assigner les conteneurs vers la zone de stockage. Les solutions obtenues sont réalisables mais pas toujours optimales.
[Imai et al., 2006] ont proposé un plan de stockage des conteneurs dans un navire qui répond aux contraintes de stabilité du navire et le nombre minimal de déplacements inutiles des conteneurs. Ce problème est exprimé comme un programme multi-objectifs.
[Sciomachen et al., 2007] ont développé un algorithme heuristique pour réduire le temps de chargement total. Cette méthode est comparée à une autre méthode heuristique et les résultats ont montré leur efficacité.

Une fois le problème de positionnement des conteneurs résolu, il sera alors possible de planifier l'ordre des conteneurs à charger/décharger par les grues de quai.

2.2.2.3/ TRANSPORT DES CONTENEURS

Les conteneurs doivent être transportés du navire porte-conteneurs vers la zone de stockage et vice versa. La résolution de ce problème consiste à choisir le type de véhicules de manutention pour le transport des conteneurs et à déterminer le nombre nécessaire de véhicules participants. Le routage des ressources de transport est aussi à considérer. Ici, dans la littérature on parle de *Yard Truck Scheduling Problem*. Plusieurs variantes peuvent être considérées, par exemple suivant le type de véhicule, autonome ou pas...
Plusieurs chercheurs ont traité le problème de transport des conteneurs comme [Meer, 2000], [Bish et al., 2001], [Le-Anh et al., 2010], etc. L'objectif est de minimiser la

distance totale parcourue par les véhicules, et de réduire le nombre des véhicules utilisés ainsi que leur temps de fonctionnement.

Ce type de problème n'a pas beaucoup été traité seul par les chercheurs. Il est en général couplé en amont avec le problème lié au chargement/déchargement des navires, ou en aval avec le problème de mise en stock des conteneurs. Par exemple, [Wang et al., 2015] ont développé un algorithme génétique pour résoudre simultanément l'ordonnancement des camions et l'affectation des zones de stockage (*Yard truck scheduling and storage allocation problems*, notés YTS-SAP), pour les conteneurs déchargés d'un navire.

2.2.2.4/ STOCKAGE DES CONTENEURS

Le stockage des conteneurs est un processus complexe dans un terminal maritime car les conteneurs à charger et décharger sont stockés simultanément dans la même zone. Après l'arrivée du navire porte-conteneurs, les conteneurs qui en sont déchargés sont déplacés vers la zone de stockage. Ceux qui possèdent la même longueur sont empilés les uns sur les autres. L'opération d'exportation est l'inverse de l'opération d'importation. Avant l'arrivée d'un navire, le port reçoit des conteneurs destinés à être transportés vers un autre port. Ils sont stockés provisoirement jusqu'à la date de départ du porte-conteneurs. Avant le chargement des conteneurs sur un navire, le plan de chargement est organisé en prenant en considération la destination finale, le type, et le poids de chaque conteneur en cohérence avec le poids maximal admis.

Il existe deux techniques de stockage de conteneurs : le stockage sur châssis et le stockage sur terrain. Avec la première technique, les conteneurs sont placés les uns à côté des autres et chaque conteneur est accessible de manière individuelle et directe. Avec la deuxième technique, la plus populaire, les conteneurs peuvent être empilés. En effet, pour assurer une meilleure gestion de l'espace alloué au stockage, il est intéressant de superposer les conteneurs quand c'est possible. Mais la conséquence est que pour admettre un conteneur, il faut enlever de la pile tous les conteneurs qui se trouvent au-dessus. La Figure 2.15 montre un exemple de stratégie de superposition, où pour emporter le conteneur numéro 2 qui est situé au milieu d'une pile, il faut emporter tout d'abord le conteneur numéro 3.

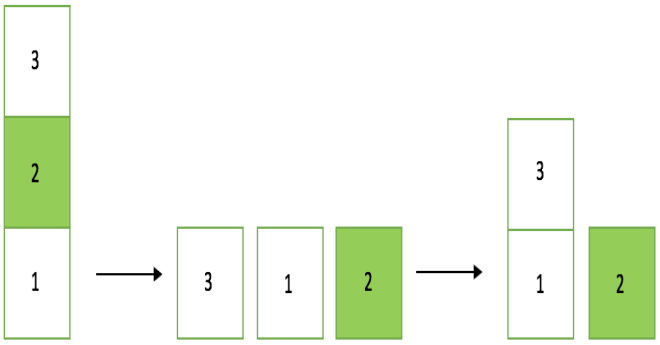

FIGURE 2.15 – Zone de stockage

La zone de stockage est divisée en plusieurs blocs et chaque bloc est constitué de plu-

sieurs baies. Chaque baie comprend un ensemble de colonnes. On identifie la position d'un conteneur par la colonne, la baie et l'étage (Figure 2.16).

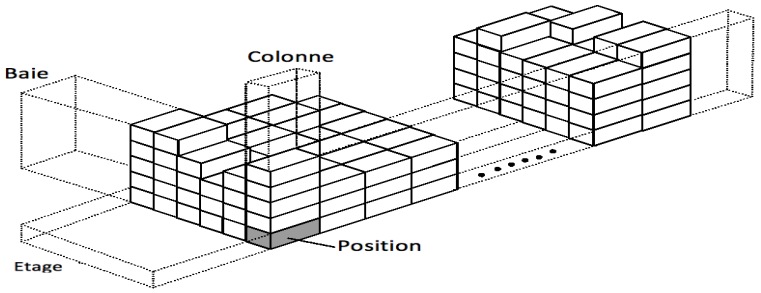

FIGURE 2.16 – Zone de stockage

La distribution primaire des conteneurs dans les zones de stockage est basée sur plusieurs normes pour simplifier le transfert des conteneurs vers les autres zones. Les conteneurs destinés aux autres ports se retrouvent généralement près des grues de quai pour minimiser la distance parcourue par les camions de transport. La figure 2.17 représente une description du terminal à conteneurs.

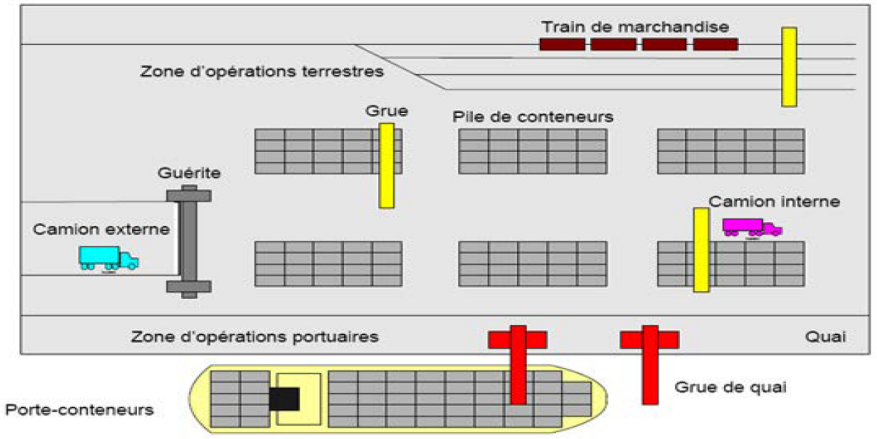

FIGURE 2.17 – Zone de stockage

Le stockage des conteneurs (*Storage Allocation Problems*) fait partie des décisions complexes associées au terminal maritime. Ainsi, l'attribution des lieux les plus appropriés aux conteneurs arrivant au port doit permettre la réduction des transferts inutiles de conteneurs lors de leur transport vers un navire, un camion. Cela réduit le temps de fonctionnement des grues et le temps que le camion passe au terminal.
Pour les conteneurs importés, la date de départ de la zone de stockage est inconnue, car l'arrivée des camions de transport est aléatoire, tandis que pour les conteneurs à exporter par voir maritime, la date de départ est connue car il sont reliés à la date du départ du navire porte-conteneurs. Cependant, le plan de stockage est prêt peu de temps avant le chargement et parfois le navire se trouve retardé pour des raisons météorologiques.

Les problèmes d'optimisation liés au stockage dans les terminaux maritimes ont été abordés par des chercheurs qui ont développé des formulations mathématiques et des méthodes heuristiques. Nous citons ci-après quelques-uns des travaux de la littérature.

[Castilho et al., 1993] établissent que pour introduire une meilleure configuration des blocs de stockage, il faut estimer le nombre de mouvements requis pour déplacer un conteneur en fonction de la stratégie de fonctionnement et de la hauteur de la pile.
[Chen et al., 2004] ont utilisé des méthodes métaheuristiques comme le recuit simulé, la recherche Tabou et l'algorithme génétique pour optimiser les zones de stockage dans un port. L'objectif est de réduire l'espace alloué au chargement dans un temps donné.
[Kumar et al., 2008] ont présenté un modèle analytique pour déterminer le temps de déchargement des conteneurs et préciser le temps d'utilisation des matériels maritimes. Ils ont appliqué ce modèle au port Suva, avec de bonnes performances.
[Bazzazi et al., 2009] ont résolu une variante étendue d'un problème d'affectation d'une zone de stockage défini par [Zhang et al., 2003], en considérant différents types et tailles de conteneurs. Ils ont développé un algorithme génétique pour résoudre le problème, en supposant que les blocs réservés pour chaque type de conteneur sont connus.
[Liu et al., 2015] ont proposé des modèles et algorithmes de type *branch and price* pour un problème général considérant des mouvements de conteneurs internes aux zones de stockage, et où les conteneurs peuvent être déplacés directement d'une pile à l'autre.

Parmi les divers problèmes scientifiques identifiés ci-dessus, nous nous sommes focalisés sur la partie ordonnancement des tâches de manutention entre le bateau et le stock. Aussi, dans la sous-section suivante, nous avons porté notre attention sur les problèmes d'ordonnancement identifiés dans les terminaux portuaires, et sur les recherches déjà menées dans ce domaine.

2.3/ PROBLÈMES D'ORDONNANCEMENT DANS LES PORTS

Nous nous intéressons particulièrement aux problèmes d'ordonnancement dans les terminaux portuaires. "Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début" ([GOTHA, 1993]). Un problème d'ordonnancement se définit par quatre types d'éléments : les tâches à ordonnancer, les ressources qui leur sont affectées, les contraintes qui les lient entre elles et aux ressources, les objectifs à atteindre. Dans notre cas, les tâches à ordonnancer sont les activités de manutention des conteneurs entre les navires et la zone de stockage. Les ressources sont les grues, camions semi-remorques et chariots frontaux/cavaliers. Les contraintes sont liées à la capacité des ressources, au besoin de synchronisation entre les ressources quand la tâche se fait en plusieurs étapes, les contraintes spatiales pour éviter les collisions entre les grues par exemple... Les objectifs peuvent être différents en fonction du problème sur lequel on se focalise. Par contre il est principalement lié au temps ou au coût.

Globalement, nous pouvons distinguer quatre catégories de problèmes dans les ports, qui seront détaillées dans les sous-sections suivantes :

- **QCSP pour *Quay Crane Scheduling Problem*** : ordonnancement des grues de quai affectées aux navires porte-conteneurs pour les opérations de chargement et/ou déchargement des conteneurs ;
- **YTSP pour *Yard Truck Scheduling Problem*** : ordonnancement des camions déplaçant les conteneurs entre le quai et les zones de stockage ; des grues sont considérées pour charger les conteneurs sur les camions ; ce type de problème consiste à étudier la synchronisation seulement entre les camions.

— **QCYTSP pour *Quay Crane and Yard Truck Scheduling Problem*** : ordonnancement des grues de quai avec intégration des camions semi-remorques pour transporter les conteneurs du quai vers la zone de stockage ;
— **QCAVSP pour *Quay Crane and Autonomous Vehicle Scheduling Problem*** : c'est une déclinaison du problème précédent dans lequel les camions sont à guidage automatique (véhicules autonomes sans chauffeur).

Pour construire le planning de travail des grues de quai, il faut déterminer quelle grue doit être affectée à quel navire porte-conteneurs et sur quelle partie. Ce planning indique la séquence de navires porte-conteneurs sur lesquels travailler, la quantité et le type d'opérations à effectuer. Il existe plusieurs normes à respecter pour créer un tel planning, en relation avec la disponibilité des grues de quai, le plan de chargement du navire porte-conteneurs et la position des conteneurs sur le terminal devant être chargés sur le navire porte-conteneurs (Kim et Park 2004).

Le problème d'ordonnancement de grues de quai (QCSP) consiste à déterminer une séquence d'exécution des opérations de chargement/déchargement des conteneurs qui minimise le temps total passé sur le navire. Pour réaliser le planning de travail, il faut tenir compte de la position des conteneurs, du nombre de conteneurs à manutentionner et éviter le croisement des grues de quai. En réalité, la taille des grues nécessite de laisser une cale libre entre les grues afin d'éviter un croisement au cours des opérations.

De nombreux chercheurs ont étudié le problème d'ordonnancement de grues de quai (QCSP) et ont proposé plusieurs méthodes de résolution, exactes et/ou approchées.
[Daganzo, 1989] est l'un des premiers à avoir étudié ce problème pour plusieurs navires. Sa suggestion était de diviser les navires en baies (Figure 2.18). Dans ce cas, une seule grue de quai est capable de travailler sur une baie à la fois. Ces grues sont libres de se déplacer d'une baie à une autre, et les porte-conteneurs ne peuvent pas partir avant que toutes leurs baies soient déchargées. Minimiser le coût du retard est l'objectif principal, avec une méthode de solution exacte et approchée.

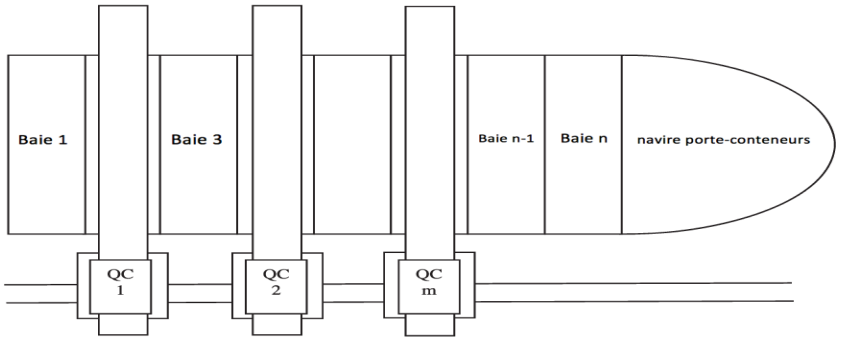

FIGURE 2.18 – Ordonnancement de grue de quai

[Peterkofsky et al., 1990] ont développé un algorithme par séparation et évaluation (*branch and bound*) pour le problème d'ordonnancement de grue de quai statique, en cas où si les grues de quai pouvaient se croiser entre eux.
Enfin, [Kim et al., 2004] ont étudié le problème d'ordonnancement de grues de quai sans contraintes d'interférence, en considérant le cas d'un seul navire porte-conteneurs. L'objectif principal était de réduire le temps total de travail de toutes les grues.

[Lim et al., 2004] ont supposé qu'un navire porte-conteneurs est une tâche. Lorsque cette tâche est affectée à la grue de quai, elle génère un profit. Les auteurs ont cherché l'affectation des tâches qui maximise le profit total. Un algorithme de programmation dynamique et une recherche taboue sont utilisés pour apporter une solution au problème.
[Steeken et al., 2004] ont décrit et classifié les principaux processus et opérations logistiques dans les terminaux à conteneurs et ils ont présenté une revue des méthodes utilisées pour leur optimisation.
[Liang et al., 2008] ont étudié le problème de la position du poste d'amarrage et déterminé le nombre de grues de quai à affecter à chaque navire. L'objectif était de réduire le temps de manutention et le temps d'attente pour chaque navire. Un algorithme génétique a été proposé pour trouver la meilleure solution possible.
[Moghaddam et al., 2008] ont présenté un nouveau modèle de programmation en nombres entiers mixtes (MIP) pour le problème d'ordonnancement et d'affectation des grues de quai.
[Bierwirth et al., 2009] ont examiné le problème d'ordonnancement des grues de quai qui sont utilisées dans les terminaux à conteneurs dans les ports maritimes pour charger et décharger les conteneurs. Ils ont présenté un modèle d'optimisation révisé pour l'ordonnancement de grues et ont proposé un algorithme par séparation et évaluation pour résoudre le problème.
Par la suite, [Wang et al., 2009] ont déterminé la séquence des opérations de déchargement, dans le but de réduire le temps de réalisation pondéré des travaux et le temps de déplacement.
[Chang et al., 2010] ont réalisé un état de l'art sur la planification des postes d'amarrage et des grues de quai. Ils terminent leur processus de tri des connaissances par la génération d'un arbre taxonomique.
[Izquierdo et al., 2011] ont proposé un algorithme à estimation de distribution (EDA) afin de résoudre un QCSP. Les EDA sont un type de techniques d'optimisation qui appartiennent au calcul évolutif. Leur fonctionnement est basé sur l'utilisation d'un modèle probabiliste, qui tente d'atteindre des régions prometteuses à travers des informations statistiques concernant les individus appartenant à la population.
[Chung et al., 2012] ont élaboré un algorithme génétique modifié pour traiter le problème et tester la fiabilité de l'algorithme proposé. Un ensemble de problèmes d'analyse comparative bien connus a été résolu.
[Hakam et al., 2012] ont élaboré un algorithme génétique pour ordonnancer deux grues de quai avec des contraintes de croisement au terminal à conteneurs de Narvik (Norvège).
[Kaveshgar et al., 2012] ont également développé un algorithme génétique pour un QCSP, qui s'est révélé être NP-complet.
[Yi et al., 2012] ont proposé un modèle général pour ce problème et ont introduit une heuristique pour le résoudre. Leur objectif est de minimiser le temps de traitement total de toutes les tâches.
[Chung et al., 2013] ont proposé un algorithme génétique pour résoudre un QCSP, qui est connu pour être l'une des tâches les plus critiques dans les opérations des terminaux.
[Guo et al., 2013] ont étudié le QCSP pour déterminer la séquence de traitement des tâches dans les baies par un ensemble de grues de quai affectées à un navire porte-conteneurs, de sorte que le temps de service du navire soit minimisé. Ils ont utilisé l'algorithme GEO (generalized extremal optimization) et MGEO (modified generalized extremal optimization) pour résoudre le problème.
[Azza et al., 2014] ont cherché à optimiser la séquence des tâches à bord du navire porte-

conteneurs. Leur objectif est de minimiser le temps passé par le navire à quai et de minimiser le coût total. La programmation linéaire en nombre entiers et un algorithme de colonie de fourmis (ACF) ont été utilisés pour résoudre le problème.
De plus, [Diabat et al., 2014] ont développé une formulation pour les problèmes d'affectation et d'ordonnancement des grues de quai, qui tient compte de la position de la grue, et ils ont proposé un algorithme génétique pour le résoudre.
[Al-Dhaheri et al., 2015] ont déterminé la séquence des opérations de déchargement d'un navire porte-conteneurs par un ensemble de grues de quai, de sorte que la date d'achèvement des opérations soit minimisée. Une formulation par programmation en nombre mixtes (MIP) a été présentée pour ce problème.
[Bierwirth et al., 2015] ont proposé une classification pour des publications récentes sur l'affectation des postes d'amarrage, l'attribution des grues de quai et les problèmes d'ordonnancement de grues de quai dans les terminaux à conteneurs des ports maritimes.
[Santini et al., 2015] ont étudié le QCSP avec des contraintes de croisement, en proposant un modèle de programmation linéaire en nombres entiers.
[Alnaqbi et al., 2016] ont étudié le problème d'affectation des emplacements d'amarrage avec l'ordonnancement des grues de quai. Ils ont cherché à minimiser la date de départ du navire porte-conteneurs et son temps de manutention. Pour cela, ils ont proposé un programme linéaire en nombres entiers et un algorithme hybride.
[Awar et al., 2016] ont proposé un algorithme basé sur un programme linéaire à nombres mixtes pour minimiser le temps de traitement complet pour tous les navires porte-conteneurs.
[Boysen et al., 2016] ont présenté un schéma de classification pour le QCSP sans tenir compte des contraintes d'interférence entre les grues de quai. Ce schéma a été appliqué pour classer la littérature existante, déterminer le statu quo des résultats de complexité et identifier les recherches futures.
[Liu et al., 2016a] ont étudié le QCSP avec deux grues de quai uniformes. Leur objectif était de minimiser la date de fin de manutention de tous les conteneurs dans le navire.
[Msakni et al., 2016] ont étudié le problème de l'affectation des grues aux navire porte-conteneurs et de l'ordonnancement de chaque grue. Ils ont développé une approche exacte basée sur une méthode d'optimisation combinatoire de type *branch and price*.
[Oliveira et al., 2016] se sont concentrés sur l'intégration des opérations à quai et le problème d'ordonnancement de grue de quai en tenant compte de la minimisation du temps passé et de la consommation de carburant.
[Haoyuan et al., 2017] ont proposé un modèle de simulation du terminal à conteneurs, afin de résoudre un QCSP, dans le but de minimiser le retard total pour tous les navires porte-conteneurs. Pour cette raison, ils ont développé un algorithme génétique et une méthode d'optimisation par essaims particulaires (PSO).
[Salhi et al., 2017] ont proposé un algorithme génétique pour résoudre trois problèmes distincts, à savoir l'attribution des postes d'amarrage, l'affectation des grues de quai et l'ordonnancement des grues.

Le tableau 2.2 présente une classification des travaux selon les méthodes de résolution adoptées (exactes ou approchées). On remarque que la plupart des chercheurs ont utilisé un MILP comme méthode exacte et un algorithme génétique (GA) comme méthode approchée pour résoudre le QCSP.

TABLE 2.2 – Classification des travaux sur le *Quay Crand Scheduling Problem* (QCSP)

Références	**Exacte**	**Approchée**
[Daganzo, 1989]	Principes d'optimalité	
[Peterkofsky et al., 1990]	ASE	
[Kim et al., 2004]	MILP, ASE	GRASP
[Lim et al., 2004]	DPA	Rech. Tabou
[Steeken et al., 2004]	Classification	
[Liang et al., 2008]		GA
[Moghaddam et al., 2008]	MIP	
[Bierwirth et al., 2009]	MILP et ASE	
[Wang et al., 2009]		ACF
[Chang et al., 2010]	Classification	
[Izquierdo et al., 2011]		EDA
[Chung et al., 2012]		GA
[Hakam et al., 2012]		GA
[Kaveshgar et al., 2012]		GA
[Yi et al., 2012]	Algorithme Polynomial-Temps	
[Chung et al., 2013]		GA
[Guo et al., 2013]		GEO et MGEO
[Azza et al., 2014]	MILP	ACF
[Diabat et al., 2014]	MIP	GA
[Al-Dhaheri et al., 2015]	MIP	
[Bierwirth et al., 2015]	Classification	
[Santini et al., 2015]	MILP	
[Liu et al., 2015]	ASE	
[Alnaqbi et al., 2016]	MILP	Algo. hybride
[Awar et al., 2016]	MILP	
[Boysen et al., 2016]	Classification	
[Liu et al., 2016a]	Algo. d'approximation intégré	
[Msakni et al., 2016]	ASE	
[Oliveira et al., 2016]	Optimisation multi-objectif	
[Haoyuan et al., 2017]		GA, PSO
[Salhi et al., 2017]	MILP	GA

ASE : Branch and bound / MILP : Mixed-integer linear programming / DPA : Dynamic programming algorithm / GRASP : Greedy randomized adaptive search procedure / ACF : ANT Colony Algorithm / EDA : Estimation of Distribution Algorithms / GA : Genetic algorithm / GEO : Generalized extremal optimization / MGEO : Modified generalized extremal optimization / PSO : Particle swarm optimization

2.4/ ORDONNANCEMENT DE GRUES DE QUAI AVEC LIVRAISON

Pour réduire le temps de rotation des navires, il est essentiel que les opérations des grues de quai, des camions de transport et des grues de triage soient coordonnées. La plupart des études ont cherché à optimiser chacun de ces processus indépendamment. Étant donné que les opérations des grues de quai et des camions de transport sont étroitement liées, il est nécessaire de développer et de résoudre ces opérations d'une

manière intégrée qui reflète les caractéristiques des terminaux à conteneurs maritimes.

Généralement dans les terminaux maritimes, plusieurs grues de quai et camions de transport travaillent ensemble (Figure 2.19), mais lorsque tous les conteneurs à décharger sont situés dans la même baie d'un navire, une seule grue de quai est autorisée à effectuer ces opérations, car plusieurs grues de quai ne peuvent pas fonctionner dans la même baie pour des raisons d'encombrement.

FIGURE 2.19 – Opérations de grues de quai et de camions de transport

La figure 2.20 détaille la démarche de manutention des conteneurs depuis le navire porte-conteneurs vers la zone de stockage ou vice versa. Les différentes opérations à réaliser sont numérotées comme suit :

1. Une grue de quai décharge un conteneur du navire porte-conteneurs.
2. La grue de quai charge le conteneur sur un camion.
3. Le camion transporte le conteneur vers la station où se trouvent les chariots frontaux/cavalier ou les grues portiques.
4. Un chariot/grue portique décharge le conteneur du camion.
5. Le chariot/grue portique place le conteneur dans la zone de stockage et le camion retourne chercher un autre conteneur sur le quai.

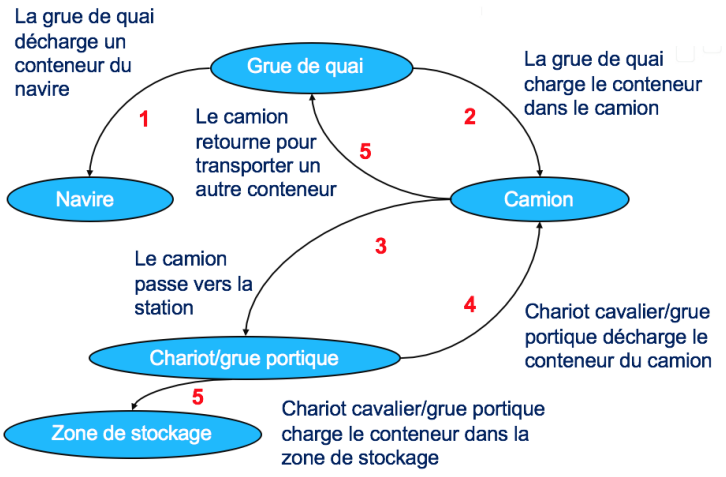

FIGURE 2.20 – Processus de manutention des conteneurs

En général, un ensemble fixe de camions est affecté à l'exécution des tâches de transport pour une grue de quai. Les camions suivent deux stratégies opérationnelles, à savoir le cycle unique et le cycle double. La figure 2.21 illustre les stratégies à cycle unique et à cycle double pour les opérations effectuées par des camions et deux grues de quai G1 et G2. Dans un cycle unique, les camions se déplacent entre une grue et une zone de

stockage données (partie gauche de la figure 2.21). Dans un cycle double, les camions chargent des conteneurs depuis plusieurs grues de quai et peuvent approvisionner plusieurs zone de stockage (partie droite de la figure 2.21). Même si le double cycle sert à augmenter l'efficacité du transport et réduire les mouvements à vide du *Yard Truck* (YT), ils ne sont adaptés qu'aux terminaux à conteneurs automatisés, car les stratégies de répartition complexes sont difficiles à suivre pour les conducteurs.

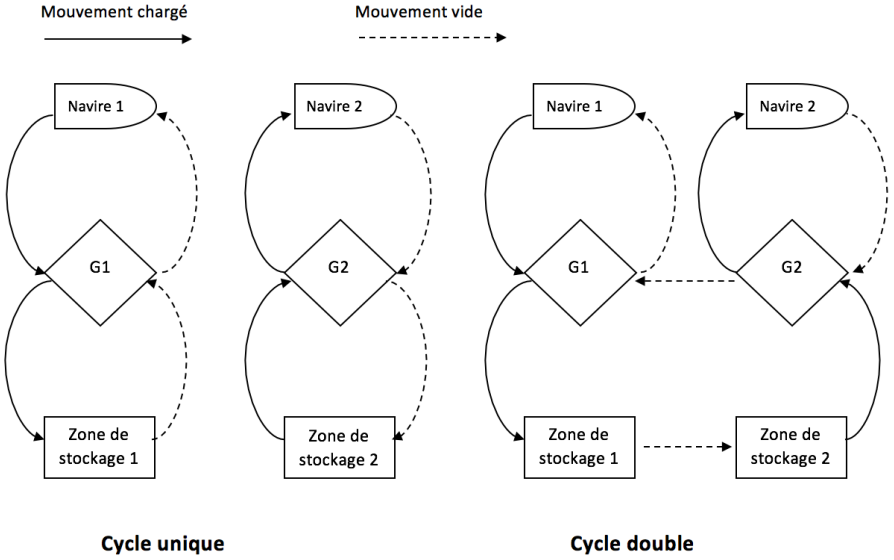

FIGURE 2.21 – Cycle unique et cycle double

L'ordonnancement des grues de quai et des camions, ou *Quay Crane and Yard Truck Scheduling Problem* (QCYTSP) dans sa version la plus complète, est un problème d'optimisation a déjà été étudié dans la littérature, comme nous le présentons ci-après.
[Jing, 2010] a étudié l'affectation des conteneurs aux camions, et a proposé un modèle de programmation mixte en nombres entiers et un algorithme à deux phases.
[Chen et al., 2013] ont proposé un MILP pour étudier les interactions entre la manutention par les grues de quai et le transport des camions dans un terminal à conteneurs maritime. Les camions sont partagés entre les différents navires, ce qui contribue à réduire les déplacements de camions vides dans la zone du terminal.
[Dkhil et al., 2013] visaient à minimiser le temps de déplacement de tous les conteneurs et le nombre requis de camions. Pour cela, ils ont utilisé la programmation linéaire en nombres entiers mixtes, la recherche tabou et un algorithme génétique hybride.
[Xue et al., 2013] ont étudié l'ordonnancement de grue de quai et des camions (QCYTSP), et l'attribution de la zone de stockage pour le déchargement des conteneurs. Ils ont proposé un modèle de type programmation en nombres mixtes, une optimisation par colonies de fourmis et un algorithme glouton (*greedy search algorithm*).
[Homayouni et al., 2014] ont développé un algorithme génétique (GA) pour résoudre l'ordonnancement des grues de quai avec l'intégration de véhicules guidés automatisés.
[Niu et al., 2014] ont étudié l'ordonnancement des camions de transport et l'affectation des conteneurs, et développé un algorithme d'optimisation par essaims particulaires (particle swarm optimization (PSO)).
[Tang et al., 2014] ont étudié un QCYTSP. Ils ont proposé un modèle de programmation linéaire à nombres mixtes (MILP) et développé une méthode par essaims particulaires

(PSO) pour obtenir le temps minimal de réalisation.
[Kaveshgar et al., 2015] ont proposé un programme en nombres entiers mixtes et un algorithme génétique (GA) hybridé avec un algorithme glouton pour un QCYTSP.
[Liu et al., 2016b] ont élaboré un nouveau modèle pour le QCYTSP. Ils ont proposé un algorithme d'optimisation par essaims particulaires (PSO) et un algorithme hybride.
[Zhang et al., 2016] ont conçu une nouvelle stratégie pour améliorer l'efficacité des opérations de stockage et le transport avec un double cycle de grue de quai. Ils ont proposé un MILP ainsi que des bornes inférieures.
[Zhen et al., 2016] ont également étudié le QCYTSP et ont proposé un programme en nombres mixtes (MIP) et un algorithme génétique (GA).
[Xiazhong et al., 2017] visait à minimiser le temps de manutention des conteneurs et les temps de déplacement des camions. Ils ont présenté un programme linéaire en nombres mixtes (MILP) et une recherche tabou pour résoudre le problème.
[Liang et al., 2018] ont proposé un premier modèle dit de répartition pour minimiser la durée d'exécution des tâches de la dernière grue de quai, et un second modèle dit de configuration pour minimiser le nombre de grues de quai. Ces modèles sont conçus pour optimiser le nombre et l'ordonnancement de grues de quai. Ils ont comparé leurs résultats aux résultats existants dans la littérature.
[Vahdani et al., 2019] ont proposé un modèle d'optimisation bi-objectif. Dans ce modèle, plusieurs phases d'affectation sont étudiées : les affectations du navire aux terminaux à conteneurs, des grues de quai aux terminaux, des grues aux navires et des camions aux grues. Le modèle vise également à améliorer l'efficience et l'efficacité des camions en les partageant entre différents terminaux.

Le tableau 2.3 résume les spécificités des articles cités précédemment. Les informations portées dans les colonnes du tableau sont les suivantes :

- **Référence** : référence bibliographique des articles cités, dont le détail est fourni en fin de ce manuscrit.
- **Problème** : nous reprenons les acronymes définis précédemment pour désigner les problèmes traités : QCSP pour *Quay Crane Scheduling Problem*, YTSP pour *Yard Truck Scheduling Problem*, QCYTSP pour *Quay Crane and Yard Truck Scheduling Problem*, et QCAVSP pour *Quay Crane and Autonomous Vehicle Scheduling Problem*.
- **Config. QC** : prise en compte du temps nécessaire pour qu'une grue se déplace d'une baie à une autre, intégrant le temps de déchargement du conteneur précédemment manipulé, y compris le temps de chargement sur le camion, plus le temps pour se repositionner en face de la baie pour la prochaine opération.
- **Config. YT** : prise en compte des mouvements à vide ou/et des temps d'attente des camions.
- **Position** : prise en compte des contraintes de position des conteneurs sur le navire pour la séquence de chargement/déchargement (autrement dit, l'emplacement des conteneurs est une donnée du problème traité).
- **Camions** : La case est cochée dans le cas de l'utilisation de camions qui peuvent transporter un conteneur à la fois (mono-remorque). Si les camions ne sont pas considérés, ou pour d'autres types de camions (multi-remorques, à guidage automatique...), la case n'est pas cochée.
- **Grue de zone** : prise en compte des ressources au niveau de la zone de stockage, pour charger/décharger des conteneurs vers/depuis le camion de transport (grue portique, chariots frontaux/cavalier...).
- **Méthodes de résolution** : méthodes exactes ou approchées uti-

lisées/développées dans l'article cité.

TABLE 2.3 – Synthèse de la littérature

Référence	**Problème**	**Contraintes**					**Méthodes de résolution**
		Config.		**Position**	**Camions**	**Grue de zone**	
		QC	**YT**				
[Daganzo, 1989]	QCSP			✓			MILP
[Jing, 2010]	QCYTSP		✓		✓	YC	MILP et C-W saving algorithm
[Chen et al., 2013]	QCYTSP		✓		✓	YC	MILP
[Dkhil et al., 2013]	QCAVSP	✓		✓		YC	MILP et GA
[Xue et al., 2013]	QCYTSP	✓			✓		MIP et ACF
[Diabat et al., 2014]	QCSP	✓		✓			MILP et GA
[Homayouni et al., 2014]	QCAVSP	✓	✓			YC	GA
[Niu et al., 2014]	YTSP		✓		✓	YC	MILP et PSO
[Tang et al., 2014]	QCYTSP	✓		✓	✓		MILP, PSO
[Al-Dhaheri et al., 2015]	QCSP	✓		✓			MIP
[Kaveshgar et al., 2015]	QCYTSP	✓		✓	✓	YC	MILP et GA
[Boysen et al., 2016]	QCSP						Survey-Classification
[Liu et al., 2016a]	QCSP	✓		✓			Approximation algorithm
[Liu et al., 2016b]	QCYTSP			✓	✓		MILP et PAO-AFSA
[Zhang et al., 2016]	QCYTSP		✓		✓	YC	MILP
[Zhen et al., 2016]	QCYTSP	✓	✓	✓	✓		MILP et PSO
[Haoyuan et al., 2017]	QCSP	✓		✓			PSO et SA
[Xiazhong et al., 2017]	QCSP	✓		✓		YC	MILP et PSO
[Liang et al., 2018]	QCSP	✓		✓		YC	Two MILP
[Vahdani et al., 2019]	QCYTSP			✓	✓	YC	MILP, MNSGA-II et MMOPSO

MILP : Mixed-integer linear programming / C-W : Two-phase algorithm / GA : Genetic algorithm / ACF : Ant colony optimization / PSO : Particle swarm optimization / PAO-AFSA : Hybridization of PSO / SA : Simulation-based optimization / MNSGA II et MMOPSO : Bi-objective optimization model

L'objectif de cette thèse est de résoudre des variantes du problème QCYTSP qui intègrent simultanément les diverses caractéristiques identifiées dans ce tableau. Pour ce faire, nous utiliserons des méthodes exactes et approchées déjà éprouvées pour ce type

de problèmes (programmation linéaire, méthode numérative et algorithmes génétiques). Avant de décrire nos travaux plus en détail, nous présentons dans la section suivante un aperçu des méthodes de recherche opérationnelle pour la résolution de problèmes d'optimisation combinatoire.

2.5/ MÉTHODES DE RÉSOLUTION

Selon la difficulté d'un problème en optimisation combinatoire [A Schwarz et al., 1994], sa résolution pourrait être possible en un temps raisonnable pour atteindre un optimum global. Dans ce cas, les méthodes de résolution dites exactes sont privilégiées. Sinon, les méthodes approchées permettent de chercher une solution de bonne qualité dans un délai acceptable. Pour des instances de grandes tailles, ces méthodes restent de toute évidence un moyen rapide pour l'obtention de solutions, même si elles ne contiennent pas forcément l'optimale. La figure 2.22 présente une classification de méthodes importantes de la littérature sous les deux branches : méthodes exactes et méthodes approchées.

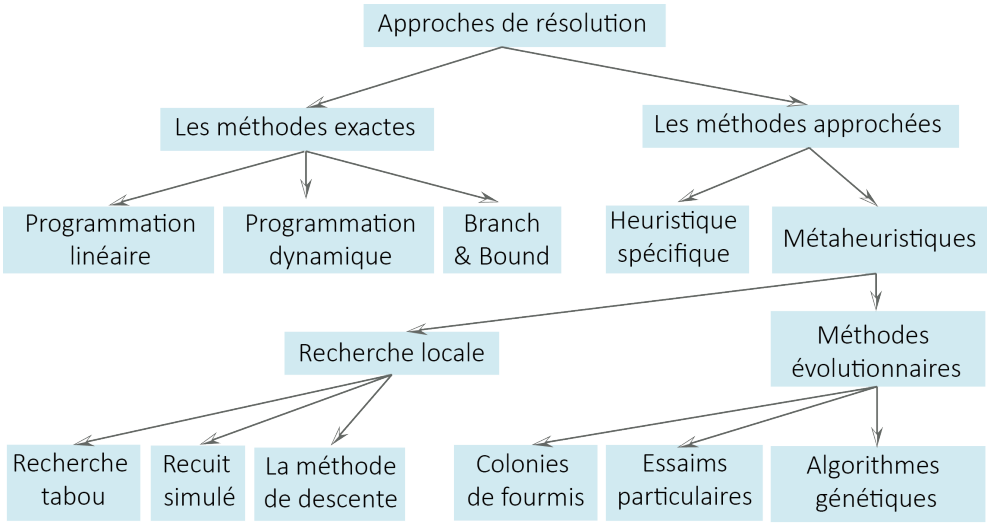

FIGURE 2.22 – Classification des méthodes de résolution

2.5.1/ MÉTHODES EXACTES

2.5.1.1/ PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS

Le modèle linéaire en nombres entiers communément appelé programmation linéaire en nombres entiers (PLNE) ou Integer Linear Programming (ILP) [Bradley et al., 1977] est une méthode de l'informatique théorique et des mathématiques qui décrit les problèmes d'optimisation sous forme de fonction de coût et de contraintes linéaires, avec des variables entières. Plus formellement, la programmation linéaire est une technique d'optimisation d'une fonction objectif linéaire, soumise à des contraintes d'égalité et d'inégalité linéaires. Sa région réalisable est un polytope convexe, qui est un ensemble défini comme l'intersection de plusieurs demi-espaces finis, chacun étant défini par une inégalité linéaire. Sa fonction objectif est une fonction linéaire à valeur réelle définie sur ce

polyèdre. Un algorithme de programmation linéaire trouve un point dans le polytope où cette fonction a la plus petite (ou la plus grande) valeur si un tel point existe.

Les programmes linéaires sont des modèles qui peuvent être exprimés sous forme canonique, comme :

$$\begin{gathered}\text{Maximiser } C_{max} \\ \text{Tel que :} \\ Ax \leq 0 \\ x \geq 0\end{gathered}$$

où A est une matrice ayant des valeurs entières. Le modèle mathématique peut être résolu à l'aide de plusieurs logiciels comme CPLEX, un outil informatique pour l'optimisation qui est commercialisé par IBM. Il est basé sur l'algorithme du simplexe [Dantzig, 1990] et le langage C. Le programme est composé d'un fichier exécutable et d'une bibliothèque de fonctions qui peuvent être liées avec différents langages de programmation : Java, Python, C, C-sharp et C++.

2.5.1.2/ ALGORITHME DE PROGRAMMATION DYNAMIQUE

La programmation dynamique est à la fois une méthode d'optimisation mathématique et une méthode de programmation informatique. Le concept de cette méthode a été introduit par Richard Bellman dans les années 1950 [Bellman, 1957] et il a trouvé des applications dans de nombreux domaines, de l'ingénierie aérospatiale à l'économie. Dans les deux contextes, il s'agit de simplifier un problème compliqué en le décomposant en sous-problèmes plus simples de manière récursive. Bien que certains problèmes de décision ne puissent pas être démontés de cette façon, les décisions qui s'étalent sur plusieurs points dans le temps se séparent souvent de manière récursive. De même, en informatique, si un problème peut être résolu de manière optimale en le décomposant en sous-problèmes, puis en trouvant récursivement les solutions optimales aux sous-problèmes, alors il est censé avoir une sous-structure optimale.

Si des sous-problèmes peuvent être imbriqués récursivement à l'intérieur de problèmes plus grands, de sorte que des méthodes de programmation dynamique sont applicables, alors il existe une relation entre la valeur du problème et les valeurs des sous-problèmes. Dans la littérature sur l'optimisation, cette relation est appelée l'équation de Bellman.

En termes d'optimisation mathématique, la programmation dynamique fait généralement référence à la simplification d'une décision en la décomposant en une séquence d'étapes de décision dans le temps. Cela se fait en définissant une séquence de fonctions de valeur V1, V2, ..., V_n en prenant y comme argument représentant l'état du système aux temps i de 1 à n. La définition de $V_n(y)$ est la valeur obtenue dans l'état y au dernier instant n. Les valeurs V_i à des moments antérieurs i = n - 1, n - 2, ..., 2, 1 peuvent être trouvées en travaillant en arrière, en utilisant une relation récursive appelée l'équation de Bellman. Pour i = 2, ..., n, V_i - 1 à n'importe quel état, y est calculé à partir de V_i en maximisant une fonction simple (généralement la somme) du gain d'une décision au temps i - 1 et la fonction V_i dans le nouvel état du système si cette décision est prise. Puisque V_i a déjà été calculé pour les états nécessaires, l'opération ci-dessus donne V_i - 1 pour ces états. Enfin, V_1 à l'état initial du système est la valeur de la solution optimale. Les valeurs optimales des variables de décision peuvent être récupérées, une par une, en retraçant les calculs déjà effectués.

L'algorithme de programmation dynamique suit les quatre étapes suivantes :

- Caractériser la structure d'une solution optimale,
- Définir récursivement la valeur d'une solution optimale,
- Calculer la valeur d'une solution optimale, généralement de façon ascendante,
- Construire une solution optimale à partir des informations calculées.

2.5.1.3/ ALGORITHME PAR SÉPARATION ET ÉVALUATION (BRANCH AND BOUND)

La méthode a été proposée pour la première fois par Ailsa Land et Alison Doig tout en effectuant des recherches dans "London School of Economics" organisée par British Petroleum en 1960 [Fukunaga et al., 1975] pour une programmation discrète, et est devenue l'outil le plus couramment utilisé pour résoudre des problèmes d'optimisation NP-difficiles [Arora et al., 2009].

Le *Branch and Bound* est un paradigme de conception d'algorithmes pour les problèmes d'optimisation discrets et combinatoires, ainsi que l'optimisation mathématique. Un algorithme par séparation et évaluation consiste en une énumération systématique des solutions candidates au moyen de la recherche dans l'espace d'état : l'ensemble des solutions candidates est considéré comme formant un arbre enraciné avec l'ensemble complet à la racine. L'algorithme explore les branches de cet arbre, qui représentent des sous-ensembles de l'ensemble de solutions. Avant d'énumérer les solutions candidates d'une branche, la branche est vérifiée par rapport aux bornes supérieures et inférieures estimées sur la solution optimale, et est ignorée si elle ne peut pas produire une meilleure solution que la meilleure trouvée jusqu'à présent (Figure 2.23). L'algorithme dépend d'une estimation efficace des limites inférieures et supérieures des régions/branches de l'espace de recherche. Si aucune limite n'est disponible, l'algorithme se transforme en une recherche complète.

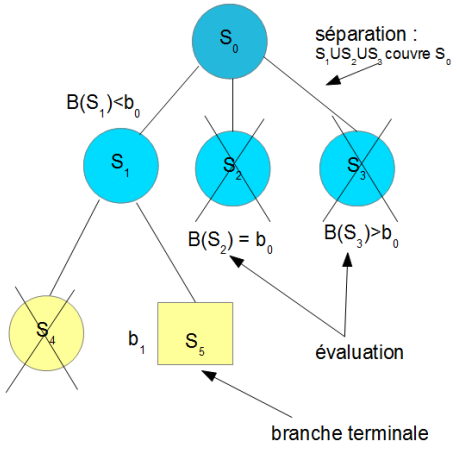

FIGURE 2.23 – Algorithme par séparation et évaluation

2.5.2/ MÉTHODES MÉTAHEURISTIQUES

Parmi les méthodes approchées, les métaheuristiques (figure 2.22) sont sans doute parmi les plus utilisées et souvent les plus efficaces. Elles ne fournissent pas toujours des

solutions optimales, mais des solutions proches de l'optimum avec un temps d'exécution raisonnable. Certaines méthodes sont toutefois plus populaires pour des applications dans la recherche ou applications industrielles notamment l'informatique décisionnelle. Parmi ces méthodes, on distingue celles basées sur l'évolution d'une solution unique et celles basées sur l'évolution d'une population de solutions. Dans la suite, nous décrivons quelques-unes de ces méthodes.

2.5.2.1/ RECHERCHE TABOU

A titre d'exemple de méthodes basées sur l'évolution d'une solution unique, nous présentons la méthode tabou. Cette méthode a été proposée par Fred Glover en 1997 [Glover, 1997], et s'appuie sur de la recherche locale.

Le principe de base de la recherche tabou par rapport à une méthode de descente simple est d'échapper aux minima locaux. Cette méthode consiste à poursuivre la recherche de solutions même lorsqu'un optimum local est rencontré. Il s'agit de permettre d'aller vers des solutions qui n'améliorent pas la solution courante. Ainsi on utilise le principe de mémoire pour éviter des mouvements cycliques. Une mémoire appelée liste taboue qui contient des mouvements ou des solutions temporairement interdits.

La figure 2.24 illustre le principe de cette méthode sous forme d'un organigramme. Dans cette figure, on considère :

- s0 : la solution initiale
- s* : la meilleure solution courante
- s : la nouvelle solution du voisinage de s*
- f(s) : la valeur de la fonction objectif à minimiser pour la solution s.

A chaque étape, un voisinage de la solution courante s* est généré. La meilleure solution s de ce voisinage, telle qu'elle n'appartient pas à la liste taboue, remplacera la solution courante à la prochaine itération, même si éventuellement elle dégrade la valeur de la fonction objectif (même si $f(s*) < f(s)$, dans un problème de minimisation). La liste taboue est la liste des derniers mouvements effectués qui est mise à jour à chaque itération pour interdire la sélection d'une solution récemment choisie.

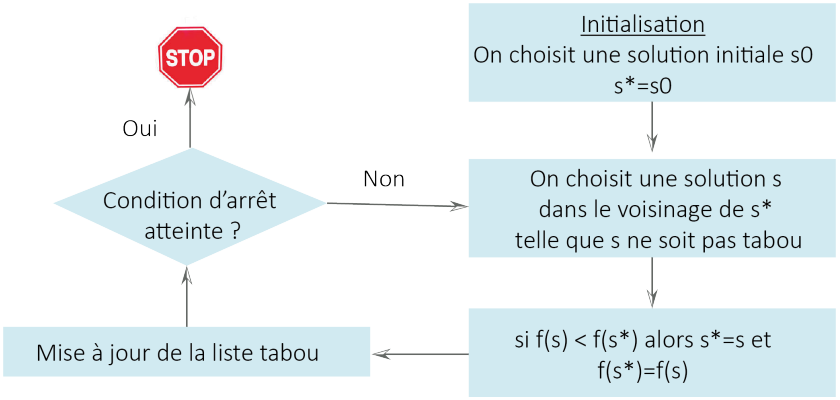

FIGURE 2.24 – Recherche tabou

Dans la suite nous présentons trois métaheuristiques à base de populations de solutions.

2.5.2.2/ COLONIES DE FOURMIS

Dans la nature, les fourmis ont une aptitude surprenante à trouver des chemins entre leur colonie et une source de nourriture. Elles échangent indirectement de l'information en déposant des phéromones. Une colonie de fourmis ayant le choix entre trois chemins de longueur menant à une source de nourriture aura tendance à utiliser le chemin le plus court (Figure 2.25).

1. Une fourmi (appelée « éclaireuse ») parcourt l'environnement autour de la colonie ;
2. Si celle-ci découvre une source de nourriture, elle rentre au nid en laissant sur son chemin une piste de phéromones ;
3. Ces phéromones étant attractives, les fourmis passant à proximité vont avoir tendance à suivre, de façon plus ou moins directe, cette piste ;
4. En revenant au nid, ces mêmes fourmis vont **«renforcer»** la piste ;
5. Si deux pistes sont possibles pour atteindre la même source de nourriture, celle étant la plus courte sera, dans le même temps, parcourue par plus de fourmis que la longue piste ;
6. La piste courte sera donc de plus en plus renforcée, et donc de plus en plus attractive ;
7. La longue piste, finira par disparaître, les phéromones étant volatiles ;
8. À terme, l'ensemble des fourmis a donc déterminé et **« choisi »** la piste la plus courte.

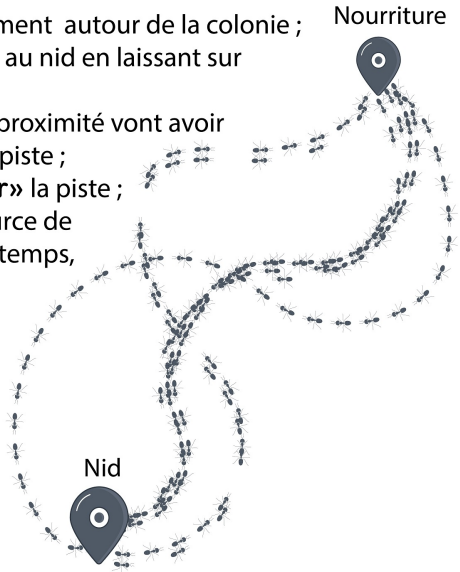

FIGURE 2.25 – Colonies de fourmis

Cette aptitude provient d'une intelligence collective et d'un ensemble d'interactions entre elles. Cette manière collective de résoudre les problèmes a inspiré la proposition d'une nouvelle métaheuristique en optimisation. Initialement proposé par Marco Dorigo en 1992 [Dorigo, 1992] dans sa thèse de doctorat, l'idée originale de ce nouveau type d'algorithmes est de résoudre une classe plus large de problèmes numériques (Figure 2.26).

```
Algorithme Colonie de fourmis
Etape 1 : initialisation
        Initialiser la quantité de phéromone de chaque arête à une valeur minimale
Etape 2 :
        Répéter
            Affecter, aléatoirement chaque fourmi à un nœud de départ
            Répéter
                Pour chaque fourmi faire
                    Début
                        Appliquer la règle de transition pour sélectionner le prochain nœud
                        Appliquer la règle de mise à jour locale
                    Fin
            Jusqu'à ce que chaque fourmi construise une solution
            Appliquer la règle de mise à jour globale
        Jusqu'à condition d'arrêt
```

FIGURE 2.26 – Algorithme de colonies de fourmis

Dorigo a appliqué le premier algorithme de ce type pour résoudre le problème du voyageur de commerce, le but étant de trouver le plus court circuit passant par un certain nombre de villes. Suite à ce travail, plusieurs problèmes d'optimisation combinatoires se sont vues appliquer une résolution par colonies de fourmis.

2.5.2.3/ ESSAIMS PARTICULAIRES (PARTICLE SWARM OPTIMIZATION)

Tout comme l'optimisation par des algorithmes de colonies de fourmis, l'intelligence collective provenant du monde du vivant est la source d'inspiration d'autres algorithmes d'optimisation comme les essaims particulaires (OEP ou PSO en anglais pour *Particle Swarm Optimization*). Sur ce principe, une nouvelle métaheuristique a été développée par l'ingénieur en électricité Russel Eberhart et le socio-psychologue James Kennedy en 1995 [Kennedy et al., 1995].

Cette méthode repose sur l'amélioration d'une solution candidate de manière itérative, par rapport à une mesure de qualité donnée. Elle démarre avec une population de solutions candidates, appelées particules. Chacune des particules est dotée d'une position au sens coordonnées dans l'ensemble de définition, et d'une vitesse de déplacement. Au cours des itérations, chaque particule change de position. Son mouvement évolue en fonction de son meilleur voisin, de sa meilleure position, et de sa position précédente. L'objectif est de faire converger l'essaim vers les meilleures positions qui correspondent aux meilleures solutions (Figure 2.27).

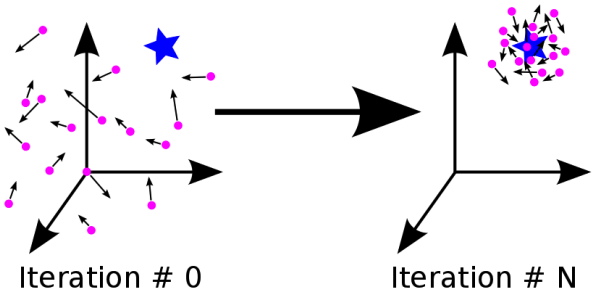

FIGURE 2.27 – Optimisation par essaims particulaires

2.5.2.4/ ALGORITHME GÉNÉTIQUE

En informatique et en recherche opérationnelle, un algorithme génétique (GA) est une métaheuristique inspirée du processus de sélection naturelle qui appartient à la classe des algorithmes évolutionnaires (EA). Les algorithmes génétiques sont couramment utilisés pour générer des solutions de haute qualité aux problèmes d'optimisation et de recherche en s'appuyant sur des opérateurs bio-inspirés tels que la mutation, le croisement et la sélection. John Holland a introduit des algorithmes génétiques en 1960 basés sur le concept de la théorie de l'évolution de Darwin ; par la suite, son élève David E. Goldberg a prolongé l'algorithme génétique en 1989 [Goldberg, 1994]. Dans un algorithme génétique, une population de solutions candidates (appelées individus ou phénotypes) à un problème d'optimisation évolue vers de meilleures solutions. Chaque solution candidate possède un ensemble de propriétés (ses chromosomes ou génotype) qui peuvent être mutées et modifiées. Traditionnellement, les solutions sont représentées en binaire

sous forme de chaînes de 0 et de 1, mais d'autres encodages sont également possibles. Une fois la représentation génétique et la fonction *fitness* définies, un algorithme génétique initialise une population de solutions puis l'améliore par l'application répétitive d'opérateurs de mutation, de croisement, d'inversion et de sélection (Figure 2.28).

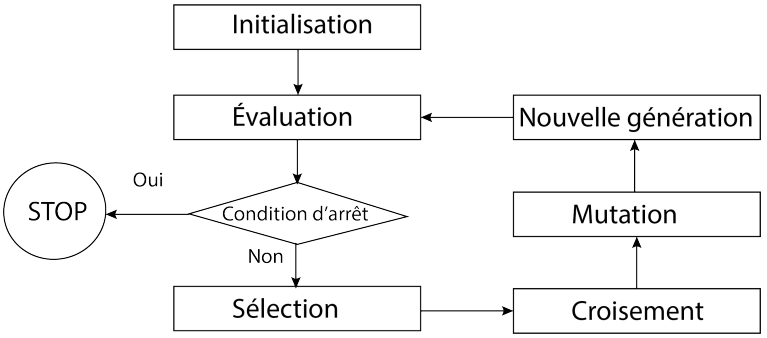

FIGURE 2.28 – Architecture de l'algorithme génétique

L'évolution commence généralement à partir d'une population d'individus générés aléatoirement, la population de chaque itération étant appelée génération. À chaque génération, la force (*fitness*) de chaque individu de la population est évaluée ; la *fitness* est généralement la valeur de la fonction objectif dans le problème d'optimisation à résoudre. Les individus les plus aptes sont sélectionnés aléatoirement dans la population actuelle, et le génome de chaque individu est modifié (recombiné et éventuellement muté au hasard) pour former une nouvelle génération. La nouvelle génération de solutions candidates est ensuite utilisée dans la prochaine itération de l'algorithme. Généralement, l'algorithme se termine lorsqu'un nombre maximal de générations a été produit ou qu'un niveau de forme physique satisfaisant a été atteint pour la population.

Notons toutefois que le succès de l'évolution de la résolution de tels algorithmes est tributaire de plusieurs essais pour chercher le meilleur paramétrage par rapport au problème. Il s'agit des paramètres comme la taille et le choix de la population initiale, les taux de croisement et de mutation de génération en génération, etc. De plus, l'expression de la fonction d'évaluation est déterminante. Elle doit être formulée avec attention pour éviter de tomber dans des problèmes comme celui de la convergence rapide vers des optima locaux souvent les moins intéressants.

1. **Principes de l'algorithme génétique**
   Les algorithmes génétiques sont un mélange de deux domaines, l'informatique et la biologie. Les mots techniques utilisés pour définir le fonctionnement de l'algorithme génétique sont les suivants :
   — Genèse : première étape qui produit une population initiale de taille fixe.
   — Chromosome : chaîne de gènes qui représente les caractéristiques de l'individu.
   — Phénotype : ensemble de caractéristiques observables décrivant un individu.
   — Évaluation : phase où la fonction *fitness* est calculée.
   — Sélection : choix des individus destinés à la reproduction.
   — Croisement : production de chromosomes fils (descendants) à partir de deux individus parents.
   — Mutation : modification d'un chromosome pour améliorer les caractéristiques de cet individu.

2. **Type du chromosome**
   - Numérique : réel ou binaire, il est utilisé si l'alphabet du problème est constitué de chiffres.
   - Symbolique : si l'alphabet du problème est constitué d'un ensemble de symboles ou de lettres alphabétiques.
   - Alpha-numérique : si l'alphabet du problème est constitué d'une combinaison de chiffres et de lettres.

3. **Méthodes de sélection**
   La sélection est le choix d'individus pour la reproduction et pour la mutation. Les méthodes les plus utilisées sont les suivantes :

   - **Sélection par rang**
     Cette méthode permet d'attribuer à chaque individu le classement en fonction de la valeur de la fonction objectif. En cas de la maximisation, les individus seront classés en ordre décroissant des valeurs de la fonction objectif. Ainsi l'individu qui possède la plus petite valeur de la fonction objectif prendra le rang 1. Ainsi, pour le cas de la minimisation, le classement sera l'inverse du cas de la maximisation. On prélève ensuite la nouvelle population à partir de l'ensemble des individus ordonnés en utilisant des probabilités indexées sur les rangs des individus.

   - **Sélection par roulette**
     Tout d'abord on calcule la somme de toutes les *fitness* appelée S, puis on génère un nombre aléatoire qui est compris entre 0 et S, puis on ajoute les valeurs de *fitness* à une somme partielle X à partir du sommet de la population, et enfin le chromosome choisi est le premier chromosome pour lequel X dépasse ce nombre aléatoire.

   - **Sélection par tournoi**
     Cette méthode de sélection consiste à choisir une sous-population, de taille N fixée aléatoirement. L'individu qui possède la meilleure qualité par rapport à la population choisie sera sélectionné pour lui appliquer les opérateurs de croisement. En réalité, c'est une compétition entre tous les individus d'une sous-population de taille N. Cette méthode peut donner une chance aux individus de qualité moindre pour participer à l'amélioration de la qualité de la population. Supposons que M soit la taille de la population, dans le cas où N=M le résultat par cette sélection donne un seul individu qui est le meilleur individu par rapport à la valeur de la fonction objectif. Dans ce cas, l'algorithme génétique sera réduit à un algorithme de recherche locale. Dans le cas où N=1, la sélection par tournoi correspond à la sélection aléatoire.

4. **Méthodes de croisement**
   L'opérateur de croisement est analogue à la reproduction et au croisement biologique. Dans ce cas, plus d'un parent est sélectionné et un ou plusieurs descendants sont produits en utilisant le matériel génétique des parents. Le croisement est généralement appliqué dans un algorithme génétique à forte probabilité. Ci-après quelques opérateurs de croisement :

— **Croisement à un point**
Un point de croisement aléatoire est sélectionné et les queues de ses deux parents sont permutées pour obtenir de nouveaux descendants.

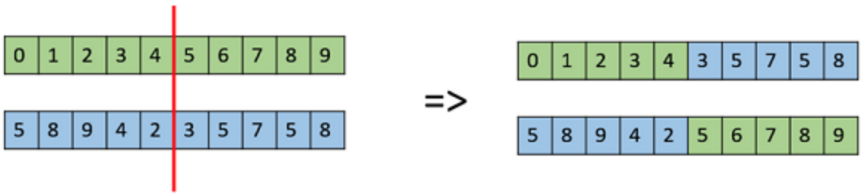

FIGURE 2.29 – Croisement à un point

— **Croisement à deux points**
C'est une généralisation du croisement à un point où des segments alternés sont échangés pour obtenir de nouveaux ressorts.

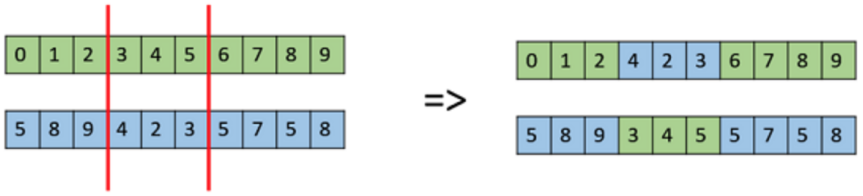

FIGURE 2.30 – Croisement à deux points

— **Croisement uniforme**
Ici, nous utilisons le principe du jet d'une pièce de monnaie. Soit deux parents, X et Y ; chacun des enfants est généré de la manière suivante : pour chaque gène de l'enfant on lance la pièce de monnaie ; si la pièce est face l'enfant prend le gène du parent X ; si la pièce est pile l'enfant prend le gène du parent Y.

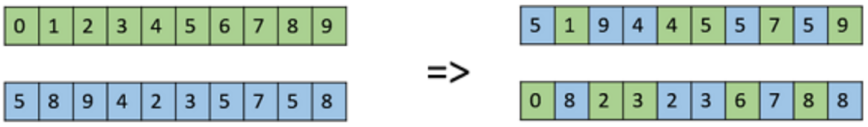

FIGURE 2.31 – Croisement uniforme

5. Mutation
En termes simples, la mutation peut être définie comme une modification aléatoire d'un chromosome, pour obtenir une nouvelle solution. Elle est utilisée pour maintenir et introduire de la diversité dans la population génétique et elle est généralement appliquée avec une faible probabilité. Si la probabilité est très élevée, l'algorithme génétique se réduit à une recherche aléatoire. La mutation est la partie de l'algorithme génétique qui est liée à "l'exploration" de l'espace de recherche. Il a été observé que la mutation est essentielle à la convergence de cet algorithme alors que le croisement ne l'est pas. Ci-après nous présentons trois opérateurs de mutation :

— **Bit Flip**
Dans cette mutation de retournement de bit, nous sélectionnons un bit aléatoirement et nous changeons sa valeur. Ceci est utilisé pour les GA codés en binaire.

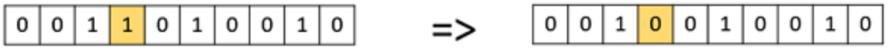

FIGURE 2.32 – Opérateur *Bit Flip*

— **Swap**
Nous sélectionnons deux positions sur le chromosome au hasard et nous échangeons les valeurs. Ceci est courant dans les codages basés sur la permutation.

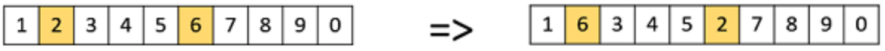

FIGURE 2.33 – Opérateur *Swap*

— **Inversion**
Nous sélectionnons un sous-ensemble de gènes comme dans la mutation de brouillage, mais au lieu de mélanger le sous-ensemble, nous inversons simplement la chaîne entière dans le sous-ensemble.

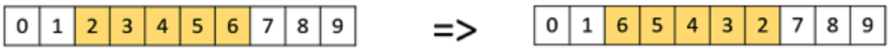

FIGURE 2.34 – Opérateur *Inversion*

En conclusion, les algorithmes génétiques représentent une approche originale et intéressante passée du stade de la recherche pure à celui de la recherche appliquée. Souvent les solutions locales de bonne qualité sont atteintes dans des temps raisonnables. Ces algorithmes ont bien fait leur preuve dans le milieu industriel. De plus, l'approche reste applicable aux problèmes multi-objectifs.

2.5.2.5/ ALGORITHME HYBRIDE

Les métaheuristiques sont souvent avantageusement hybridées avec d'autres méthodes. Ainsi on peut les combiner avec des méthodes exactes, par exemple dans les phases d'évaluation. On peut aussi le faire avec des méthodes approchées, telles que des heuristiques simples ou d'autres métaheuristiques, par exemple des méthodes par voisinage. Celles-ci permettent à des moments donnés de la résolution de faire de l'exploration en profondeur de certaines zones de l'espace de recherche. L'idée est d'exploiter au mieux les caractéristiques de chacune des méthodes, de sorte que l'algorithme global soit meilleur que les composants individuels.

2.6/ CONCLUSION

Cet état de l'art a présenté brièvement le contexte et les problèmes d'ordonnancement des grues de quai dans les terminaux maritimes, à partir de la représentation des problèmes de planification et d'ordonnancement, avec une vue d'ensemble des modèles à un et plusieurs niveaux. Nous avons également décrit l'ordonnancement intégré des grues de quai et des camions de transport, qui est un problème global et complexe rencontré dans tous les ports maritimes. Nous avons parcouru la littérature sur les modèles intégrés de base portant sur des problèmes d'ordonnancement des grues de quai, des camions de transport, sur le problème de croisement des grues de quai et le problème d'attente de chaque équipement dans le port. En outre, nous avons cité des extensions des modèles de livraison intégrés. Nous avons observé que la plupart des problèmes étudiés se limitent à un sous-ensemble des caractéristiques et contraintes que nous cherchions à modéliser conjointement.

Dans ce contexte, notre travail, rapporté dans la suite de ce mémoire, a consisté à modéliser et à résoudre trois variantes de problèmes intégrés d'ordonnancement de grues de quai (variantes de *Quay Crane Scheduling Problem)*. Dans une première étape, nous sommes tout d'abord partis de l'étude d'un QCSP de base incluant des grues de quai affectées à un seul navire porte-conteneurs, avec des contraintes de non collision. La seconde variante est un *Quay Crane and Yard Truck Scheduling Problem)* (QCYTSP), où un navire porte-conteneurs est desservi par une seule grue de quai couplée avec plusieurs camions ; le modèle développé considère le temps de déchargement/chargement des conteneurs, le temps de transport des conteneurs par les camions, le temps de déchargement des conteneurs des camions par les chariots frontaux et des contraintes de transport. La troisième variante étend enfin le modèle précédent au cas avec plusieurs grues de quai.

Pour ces trois variantes, nous nous sommes attachés à développer des algorithmes exactes et heuristiques efficaces, avec comme objectif la minimisation du temps total de manutention de tous les conteneurs. Pour chacun des problèmes étudiés, des expérimentations permettent de tester les performances de nos algorithmes et les résultats expérimentaux sont analysés et discutés.

3

ORDONNANCEMENT DE GRUES DE QUAI POUR UN NAVIRE PORTE-CONTENEURS

L'ordonnancement des grues de quai (QCSP) est un problème global que tous les ports du monde cherchent à résoudre, avec comme objectif principal de minimiser le temps de chargement et de déchargement des navires porte-conteneurs, et donc de réduire le temps d'accostage dans les terminaux maritimes. Dans ce chapitre, nous proposons trois méthodes de résolution pour le problème d'ordonnancement de grues de quai, dont deux méthodes exactes pour obtenir des solutions optimales, qui sont la programmation linéaire en nombre entiers et un algorithme numératif, et une méthode approchée qui est un algorithme génétique, pour obtenir une solution proche de l'optimale avec un temps d'exécution acceptable. Nous avons testé aléatoirement et validé nos méthodes sur des instances de différentes tailles. Enfin, nous avons comparé les résultats obtenus avec des résultats existant dans la littérature et avec des exemples réels du port de Tripoli au Liban.

3.1/ DÉFINITION DU PROBLÈME ET FORMULATION MATHÉMATIQUE

Dans cette section, nous définissons la première variante du problème étudié et nous en fournissons une formulation mathématique au moyen de notations et d'un modèle mathématique.

3.1.1/ HYPOTHÈSES

Le problème d'ordonnancement de grues de quai que nous étudions ici consiste à affecter ces grues au déchargement de conteneurs, et à trouver la séquence optimale des tâches de déchargement, afin de minimiser leur temps total d'exécution. Ce *Quay Crane Scheduling Problem* (QCSP) présente les caractéristiques suivantes :

— Un seul navire porte-conteneurs est considéré. Sa surface est divisée en plusieurs baies, et chaque baie contient un nombre spécifique de conteneurs placés dans une rangée ou les uns sur les autres. Le nombre de grues de quai est supposé strictement inférieur au nombre de baies.

— Les grues de quai se déplacent selon une voie unique. Cela impose qu'elles fonctionnent sans autoriser aucun croisement entre elles.
— Chaque grue décharge les conteneurs d'une seule baie à la fois, et ce n'est que lorsqu'elle en a fini avec cette baie qu'elle peut s'occuper des conteneurs d'une autre baie.
— Chaque grue peut décharger un conteneur à la fois.
— Chaque baie est gérée au plus par une grue à la fois.
— Nous avons négligé le temps de déplacement entre deux baies car il est court par rapport aux temps de déchargement.
— Aucune grue de quai n'est inactive.

Ces hypothèses se traduisent par des contraintes à respecter dans notre modèle.

3.1.2/ DONNÉES

— Q : ensemble des grues de quai
— $|Q|$: nombre de grues de quai
— i : index de grue de quai ($\forall i \in Q$)
— q_i : $i^{ème}$ grue de quai
— B : ensemble des baies
— $|B|$: nombre de baies
— j : index de baie ($\forall j \in B$) sur le navire
— C_j : nombre de conteneurs dans la baie j
— Tc : temps pour décharger un conteneur par la grue de quai et le placer dans la zone de stockage. On suppose qu'il est identique pour tous les conteneurs dans toutes les baies.
— M : très grand nombre (entier utilisé pour représenter l'infini

3.1.3/ VARIABLES DE DÉCISION

$\forall i \in Q, \forall j, j' \in B$,

— $x_{j,i} \begin{cases} = 1 & \text{si la baie } j \text{ est manipulée par la grue de quai } i \\ = 0 & \text{sinon} \end{cases}$
— $z_{j,j'} \begin{cases} = 1 & \text{si le déchargement de la baie } j \text{ termine avant le début} \\ & \text{du déchargement de la baie } j' \\ = 0 & \text{sinon} \end{cases}$
— t_j : date de fin du déchargement de la baie j
— C_{max} : makespan

3.1.4/ PROGRAMME LINÉAIRE EN NOMBRES ENTIERS (MILP)

Pour le QCSP que nous étudions, nous avons proposé le modèle de type MILP suivant :

Objectif

$$Minimize \quad C_{max} \tag{3.1}$$

Sous les contraintes

$$\sum_{i=1}^{|Q|} x_{j,i} = 1 \quad \forall j \in B \tag{3.2}$$

$$t_j \geq (Tc * C_j) \quad \forall j \in B \tag{3.3}$$

$$t_j - t_{j'} + (Tc * C_{j'}) + z_{j,j'} * M > 0 \quad \forall j, j' \in B \tag{3.4}$$

$$t_j - t_{j'} + (Tc * C_{j'}) - (1 - z_{j,j'}) * M \leq 0 \quad \forall j, j' \in B \tag{3.5}$$

$$\sum_{i=1}^{|Q|} i * x_{j,i} + 1 \leq \sum_{i'=1}^{|Q|} i' * x_{j',i'} + (z_{j,j'} + z_{j',j}) * M \quad \forall j, j' \in B, j < j' \tag{3.6}$$

$$C_{max} = \max_{j \in B} t_j \tag{3.7}$$

$$x_{j,i} \in [0,1] \quad \forall j \in B, \forall i \in Q \tag{3.8}$$

$$z_{j,j'} \in [0,1] \quad \forall j, j' \in B, j < j' \tag{3.9}$$

L'équation (3.1) représente la fonction objectif qui consiste à minimiser le *makespan* (date de fin des tâches de toutes les baies). La contrainte (3.2) impose que chaque baie ne doit être manipulée que par une seule grue de quai. La contrainte (3.3) garantit que la date de fin de travail dans chaque baie est supérieure au temps de travail associé à cette baie (temps de travail = nombre de conteneurs $\times$ temps nécessaire pour décharger un conteneur et le stocker). Les contraintes (3.4) et (3.5) permettent de définir la variable de décision $z_{j,j'}$. Ainsi pour $z_{j,j'} = 1$, l'équation (3.4) est toujours vérifiée, alors que l'équation (3.5) devient $t_j \leq t_{j'} - (Tc * C_{j'})$. Cela signifie que le travail dans la baie j se termine avant de commencer le travail dans la baie j'. A contrario, pour $z_{j,j'} = 0$, c'est l'équation (3.5) qui est toujours vérifiée, tandis que l'équation (3.4) exprime le fait que les tâches dans la baie j se terminent après le début du travail dans la baie j' ($t_j > t_{j'} - (Tc * C_{j'})$). La contrainte (3.6) interdit les interférences entre les grues de quai. Ainsi, supposons que les baies j et j' sont déchargées simultanément, respectivement par les grues de quai q_1 et q_2, avec j < j', ce qui signifie que $z_{j,j'} + z_{j',j} = 0$. Alors la contrainte (3.6) se traduit par $\sum_{i=1}^{Q} i * x_{j,i} + 1 \leq \sum_{i'=1}^{Q} i' * x_{j',i'}$, ce qui équivaut à $q_1 + 1 \leq q_2$, ce qui interdit le croisement entre ces grues. La contrainte (3.7) définit la valeur de l'objectif C_{max} en liant celui-ci aux variables de décision. Enfin les contraintes (3.8) et (3.9) définissent le domaine des variables de décision.

3.2/ MÉTHODES DE RÉSOLUTION

Dans cette section, nous proposons trois méthodes de résolution : la programmation linéaire en nombres entiers résolue par CPLEX qui est un outil informatique d'optimisation pour obtenir des solutions optimales, un algorithme numératif développé en JAVA, également pour obtenir des solutions optimales et un algorithme génétique qui est une méthode métaheuristique, aussi développé en JAVA pour obtenir des solutions proches des solutions optimales.

3.2.1/ PROGRAMME LINÉAIRE EN NOMBRES ENTIERS (MILP)

Notre première démarche a consisté à résoudre notre modèle linéaire à l'aide du solveur CPLEX qui est un solveur du commerce. Le langage utilisé dans CPLEX est OPL (Optimization Programming Language), c'est un langage de modélisation qui permet d'écrire facilement des programmes linéaires grâce à une syntaxe proche de la formulation mathématique. En outre, OPL offre une possibilité de séparer le modèle des données, de ce fait un même modèle peut être facilement testé avec différents données.

Les résultats obtenus avec CPLEX sont détaillés dans la section "Résultats expérimentaux" en fin de ce chapitre. Néanmoins, on sait que cette résolution "directe" est souvent limitée pour des problèmes d'optimisation complexes comme le nôtre, en tout cas à partir d'une certaine taille d'instances. C'est pourquoi nous avons eu recours aux deux autres méthodes expliquées ci-après.

3.2.2/ ALGORITHME NUMÉRATIF

3.2.2.1/ DESCRIPTION DE L'ALGORITHME

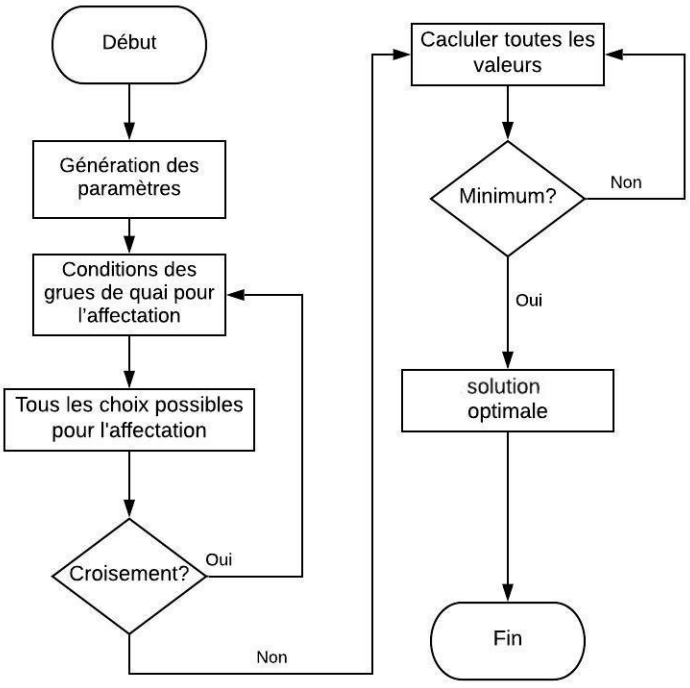

FIGURE 3.1 – Organigramme AN

Nous décrivons notre méthode sous forme de l'organigramme de la figure 3.1 et de l'algorithme associé (Algorithm 1). Comme le montre l'organigramme de la figure 3.1, nous commençons notre algorithme en définissant tous les paramètres nécessaires tels que le nombre de grues de quai, le nombre de baies, le nombre de conteneurs dans chaque baie et enfin le temps nécessaire à la grue de quai pour décharger un conteneur. Après cela, nous créons une méthode pour définir les conditions d'ordre de travail pour les grues de

quai pour éviter le croisement entre elles, puis obtenir toutes les affectations possibles. L'étape suivante est l'étape de test de affectation obtenue. S'il satisfait aux conditions précédentes, si oui nous continuons à la dernière étape, sinon nous allons tester une autre affectation. La dernière étape consiste à calculer le temps de réalisation de toutes les missions satisfaisant aux conditions de non-croisement. Ici, nous choisissons la solution avec un temps d'exécution minimal.

Cet organigramme nous a permis d'élaborer l'algorithme numératif *Algorithm* 1 que nous avons implémenté avec JAVA.

Algorithm 1 Algorithme numératif

```
1: Génération_variables() ;
2: Génération_affectations() ;
3: for chaque affectation do
4:    if croisement = 0 then
5:       calculer_temps_d'achèvement() ;
6:    else
7:       prochaine_affectation ;
8:    end if
9: end for
10: min←valeur_premier_affectation ;
11: for chaque affectation do
12:    if min > valeur_prochaine_affectation then
13:       min←valeur_prochaine_affectation ;
14:       Fin ;
15:    else
16:       prochaine_affectation ;
17:    end if
18: end for
```

Explication de *Algorithm* 1 :

- Ligne 1 : génération de toutes les données nécessaires (nombre de grues de quai, nombre de baies, nombre de conteneurs dans une baie spécifique et temps de déchargement d'un conteneur par la grue de quai).
- Ligne 2 : génération de tous les choix d'affectation des grues aux baies.
- Lignes 3 à 9 : pour chaque mission, test de l'existence d'un croisement entre les grues de quai. Si oui, test d'une autre affectation ; s'il n'y a pas d'interférence, calcul de la date de fin des opérations.
- Ligne 10 : création d'une nouvelle variable appelée min, à laquelle on donne la valeur du *makespan* calculée pour la première affectation.
- Lignes 11 à 18 : pour chaque affectation, test si la date d'achèvement est inférieure à la variable min ; si oui réglage de min sur la nouvelle date de fin, sinon passage à l'affectation suivante.

3.2.2.2/ EXEMPLE NUMÉRIQUE

Algorithm 1 nous permet de trouver tous les choix possibles pour l'affectation en tenant compte des contraintes de non-interférence. Cela signifie que toutes les affectations

doivent être effectuées sans aucun croisement entre les grues de quai. Par exemple, considérons 2 grues de quai et 4 conteneurs. Si la deuxième grue décharge le premier conteneur, et la première grue doit décharger un des 3 autres conteneurs. Dans ce cas, les grues seront amenées à se croiser, ce qui rend cette solution infaisable.

Supposons maintenant que nous avons 2 grues de quai et un navire porte-conteneurs avec 4 baies contenant respectivement 13, 16, 12 et 9 conteneurs à décharger. Les temps de déchargement des conteneurs de ces 4 baies sont respectivement 15.21, 18.72, 14.04 et 10.53 unités de temps.
Notre AN génère toutes les variables, puis toutes les affectations de grues aux baies. Dans ce cas, pour 4 baies et 2 grues, il y a initialement 2^4, soit 16 affectations possibles qui sont : 1-1-1-1, 1-1-1-2, 1-1-2-1, 1-1-2-2, 1-2-1-1, 1-2-1-2, 1-2-2-1, 1-2-2-2, 2-1-1-1, 2-1-1-2, 2-1-2-1, 2-1-2-2, 2-2-1-1, 2-2-1-2, 2-2-2-1, 2-2-2-2. Dans ces listes de type x-y-z-t, la valeur en $i^{ème}$ position représente le numéro de la grue affectée au déchargement de la baie i.
Dans un second temps, les 2 affectations (1-1-1-1 et 2-2-2-2) sont éliminées car toutes les grues réservées au navire sont censées contribuer à son déchargement. On s'interdit donc ici de ne faire travailler qu'une seule grue.
A l'étape suivante, un test de croisement est effectué, qui conduit à éliminer les affectations qui commencent par 2. En effet si la deuxième grue travaille dans la première baie il y aura un croisement entre les grues. Notons que l'inverse n'est pas vrai : la grue de quai 1 peut décharger les conteneurs de la baie 4, toutefois elle ne peut pas commencer par cette baie, ce qui sera limitatif au niveau de la phase d'ordonnancement. Pour cet exemple, on élimine donc encore 7 affectations. Il reste alors 7 affectations (figure 3.2), pour lesquelles la date de fin d'exécution est calculée, et que nous détaillons ci-après.
La solution optimale retenue sera celle de *makespan* minimal. Le calcul de la date de fin d'exécution d'une grue est effectué en ordonnançant les baies qui lui sont affectées par ordre d'indice croissant (travail de gauche à droite), et en tenant compte de la nécessité d'attendre la fin du travail des grues voisines pour pouvoir changer de baie, afin d'éviter des collisions.

Pour les 7 affectations restantes représentées dans la figure 3.2 :

- La première affectation est 1-2-1-2, ce qui signifie que la première grue QC1 décharge les conteneurs des baies 1 et 3 pour une durée de réalisation théorique de 29.25 (15.21+14.04), tandis que la deuxième grue QC2 décharge les baies 2 et 4 avec un temps de travail effectif de 29.25 (18.72+10.53). Le *makespan* devrait être en théorie 29.25. Néanmoins, une fois la baie 1 déchargée, la grue QC1 ne peut pas commencer à décharger la baie 3 car la grue QC2 ne termine son travail en baie 2 qu'à la date 18.72. QC1 doit donc attendre 3.51 unités de temps, et ensuite les deux grues peuvent se déplacer simultanément respectivement vers les baies 3 et 4. La date réelle de fin de travail de QC1 n'est donc pas 29.25 car il faut y ajouter un temps d'attente de 3.51 pour éviter une collision, ce qui donne 32.76. Le *makespan* est donc égal à 32.76.
- La deuxième affectation est 1-1-2-2, ce qui signifie que la grue QC1 décharge les conteneurs 1 et 2 avec un temps d'achèvement de 33.93, tandis que QC2 décharge les conteneurs des baies 3 et 4 avec un temps d'achèvement de 24.57. Le temps total pour effectuer ces déchargements est donc égal à 33.93.

	Baie 1	Baie 2	Baie 3	Baie 4		
Nombre de conteneurs	13	16	12	9		
Temps du déchargement	15.21	18.72	14.04	10.53		
1er affectation						Maximum
QC1	15.21		14.04		15.21 + 3.51 + 14.04 = 32.76	32.76
QC2		18.72		10.53	18.72 + 10.53 = 29.25	
2ème affectation						Maximum
QC1	15.21	18.72			15.21 + 18.72 = 33.93	33.93
QC2			14.04	10.53	14.04 + 10.53 = 24.57	
3ème affectation						Maximum
QC1	15.21				15.21	43.29
QC2		18.72	14.04	10.53	18.72 + 14.04 + 10.53 = 43.29	
4ème affectation						Maximum
QC1	15.21	18.72	14.04		15.21 + 18.72 + 14.04 = 47.97	47.97
QC2				10.53	10.53	
5ème affectation						Maximum
QC1	15.21	18.72		10.53	15.21 + 18.72 + 10.53 = 44.46	44.46
QC2			14.04		14.04	
6ème affectation						Maximum
QC1	15.21		14.04	10.53	15.21 + 3.51 + 14.04 + 10.53 = 43.29	43.29
QC2		18.72			18.72	
7ème affectation						Maximum
QC1	15.21			10.53	15.21 + 17.55 + 10.53 = 43.29	43.29
QC2		18.72	14.04		18.72 + 14.04 = 32.76	

FIGURE 3.2 – Scénarios d'affectation pour l'exemple numérique

— La troisième affectation est 1-2-2-2, ce qui signifie que QC1 décharge les 13 conteneurs de la baie 1 avec une date d'achèvement de 15.21, tandis que QC2 décharge les conteneurs des baies 2, 3 et 4 et termine son travail à la date 43.29. Le *makespan* est donc égal à 43.29.
— La quatrième affectation est 1-1-1-2 : QC1 décharge les baies 1, 2 et 3 jusqu'à la date 47.97, tandis que QC2 s'occupe de la baie 4 avec une date d'achèvement de 10.53. Le temps total de déchargement du navire est donc égal à 47.97.
— La cinquième affectation est 1-1-2-1 : QC1 décharge les conteneurs des baies 1, 2 et 4 et termine à 44.46, QC2 finit de décharger la baie 3 à 14.04. Le temps pour décharger les 50 conteneurs est donc égal à 44.46.
— La sixième affectation est 1-2-1-1 : QC1 décharge les conteneurs des baies 1, 3 et 4 et termine théoriquement à 39.78 (15.21+14.04+10.53), mais en réalité à 43.29. En effet, QC2 s'occupe de la baie 2 jusqu'à la date 18.72, et une fois la baie 1 déchargée à 15.21, QC1 doit attendre la fin du travail de QC2 pour pouvoir se déplacer vers les baies 3 et 4. QC1 doit donc subir une attente 3.51 (18.72-15.21). Le temps pour décharger les 50 conteneurs est donc égal à 43.29 (39.78+3.51).
— La septième affectation est 1-2-2-1 : la grue QC1 (respectivement QC2) décharge les conteneurs des baies 1 et 4 (respectivement 2 et 3) avec une date de fin de 43.29 (respectivement 32.76). Le *makespan* est donc égal à 43.29. Comme pour le scénario 6, cette date inclut un temps d'attente pour QC1, ici de 17.55 unités de temps entre le déchargement des baies 1 et 4, afin de laisser à QC2 le temps terminer son travail dans les baies 2 et 3 et éviter ainsi une collision.

3.2.3/ ALGORITHME GÉNÉTIQUE

Après avoir testé de grandes instances à l'aide du MILP et de notre algorithme numératif AN, nous avons observé que ces deux méthodes prenaient beaucoup de temps d'exécution (CPU time) pour résoudre le problème et obtenir la solution optimale. C'est pourquoi nous proposons une méthode métaheuristique, plus précisément un algorithme génétique (GA) pour obtenir une solution proche de l'optimum avec un temps d'exécution plus rapide. L'algorithme génétique a été proposé par [Holland, 1960] et développé et appliqué par [Goldberg, 1989]. Nous choisissons l'algorithme génétique notamment en raison de sa simplicité d'implémentation et de sa capacité à appréhender le paysage des solutions assez rapidement.

Les étapes de notre algorithme génétique sont les suivantes, illustrées par l'organigramme de la figure 3.3 :

- génération d'une population initiale de solutions
- pour chaque chromosome, affectation des grues aux baies
- test des contraintes de non-interférence pour les grues de quai pour éviter le croisement entre elles
- calcul de la valeur de fitness (égale à zéro si les grues se croisent)
- application des opérateurs de sélection, croisement et mutation décrits plus loin
- si la génération actuelle est la dernière, le programme est arrêté.

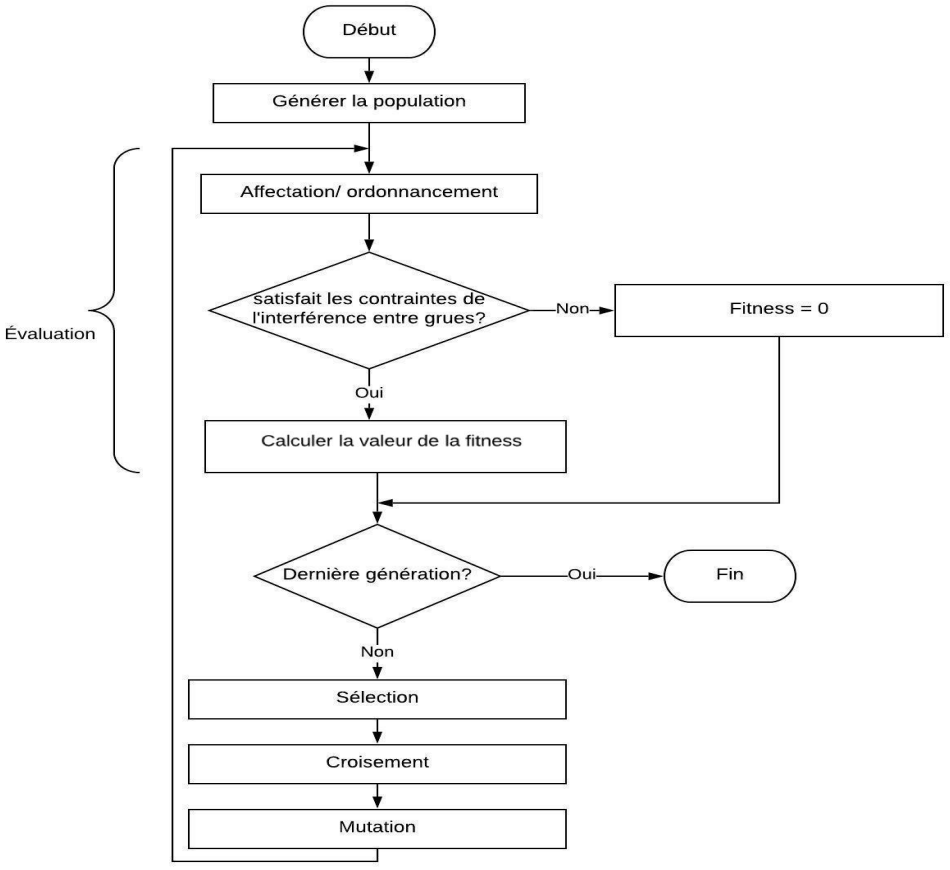

FIGURE 3.3 – Organigramme de l'algorithme génétique

3.2.3.1/ REPRÉSENTATION D'UNE SOLUTION

Une solution du problème est représentée par un codage appelé chromosome, qui est une liste ordonnée des baies à décharger, générée aléatoirement dans la population initiale. Pour illustrer la représentation adoptée dans notre étude, le tableau 3.1 montre un chromosome où chaque élément (gène) est un numéro de baie. Par exemple, le troisième élément de ce codage représente la baie numéro 4. Cette liste de baies est utilisée dans l'algorithme pour affecter les baies à chaque grue et ordonnancer leur déchargement.

TABLE 3.1 – Exemple de chromosome

7	9	4	10	3	8	6	1	2	5

3.2.3.2/ POSITION INITIALE DES GRUES DE QUAI

Tout d'abord nous commençons par fixer la position initiale post(QC_i) de chaque grue de quai QC_i au niveau de la position de la baie post(QC_i) = 1 + $(i-1) \times R$ ($i \in Q$), où R est un nombre aléatoire entre 1 et L, avec $L = |B| / |Q|$ (nombre de baies/nombre de grues de quai). Si L est un nombre rationnel, nous prenons sa partie entière. La formule adoptée permet une distribution plutôt homogène de la position initiale des grues de quai le long du navire porte-conteneurs. Notons que la position de la première grue de quai est toujours en face de la première baie (position la plus à gauche).

Exemple : Pour 10 baies et 3 grues, alors R devrait être un nombre entier aléatoire entre 1 et 10/3 (donc entre 1 et 3). Si $R = 2$, la position initiale de la grue 1 est au niveau de la baie 1, la grue 2 est au niveau de la baie 3 et la grue de quai 3 est au niveau de la baie 5.

3.2.3.3/ ORDONNANCEMENT DES GRUES DE QUAI

L'algorithme 2 explique comment est réalisé l'ordonnancement des activités de manutention par les grues de quai. Après avoir fixé les positions initiales des grues, la seconde étape consiste à affecter les baies aux grues. Pour cela, nous utilisons une procédure séquentielle qui considère les baies dans l'ordre où elles apparaissent dans le chromosome.

Algorithm 2 Ordonnancement des activités des grues de quai

if $q_{nb} = 1$ **then**
  $q_1 \leftarrow baie_j$;
  $baie_j \leftarrow -1$;
  j ← j+1 ;
  $comp_{q_1} \leftarrow comp_{q_1} + w_{baie_j}$;
end
else
  if $comp_{q_i}$ Not in $\{C_{max}\}/comp_{q_i}$ **then**
    $comp_{q_i}$ = Min$\{comp_q\}$;
    $baie_j \leftarrow -1$;
    j = Prochaine_baie() ;
    $comp_{q_i} \leftarrow comp_{q_i} + w_{baie_j}$;
  end
  else
    if $\Delta(baie) > 0$ **then**
      $q_i \leftarrow baie_j$ avec une distance plus courte ;
      $baie_j \leftarrow -1$;
      j = Prochaine_baie() ;
      $comp_{q_i} \leftarrow comp_{q_i} + w_{baie_j}$;
    end
    else
      $q_i \leftarrow baie_j$ avec un numéro d'indice plus petit ;
      $baie_j \leftarrow -1$;
      j = Min_baie() ;
      $comp_{q_i} \leftarrow comp_{q_i} + w_{baie_j}$;
    end
  end
end

Notation de l'algorithme :
q_{nb} : nombre de grues de quai,
q_i : la $i^{ème}$ grue,
$comp_{q_i}$: date de fin de travail de la grue de quai i (avec $C_{max} = \max_{q_i \in \{1 \ldots q_{nb}\}} comp_{q_i}$),
$baie_j$: baie numéro j,
w_{baie_j} : la durée de déchargement de la baie j par une grue de quai,
$\Delta(baie) = |baie_{k2} - baie_{k1}|$,
Prochaine_baie() : fonction pour déterminer l'indice de la baie suivante dans le chromosome,
Min_baie() : fonction pour obtenir le plus petit indice de baie entre les 2 baies considérées.

Les quatre étapes suivantes de cet algorithme sont présentées ci-après et illustrées avec un exemple numérique reprenant les paramètres (chromosome) de la section 3.2.3.1, et les positions initiales déterminées précédemment en considérant 3 grues de quai et 10 baies. Par ailleurs, dans l'exemple traité ici, nous supposons que le temps de traitement pour toutes les baies est de 10 u.t. (temps de déchargement des conteneurs d'une baie). Enfin le makespan (temps total de réalisation ou date d'achèvement) pour toutes les

grues de quai est initialisé à 0, car aucune opération de déchargement n'a été effectuée jusqu'à présent (itération *init* dans le tableau 3.2).
Le tableau 3.2 détaille le résultat de l'application de ces quatre étapes à chaque itération de l'algorithme 2. Dans ce tableau, la position des grues est exprimée en terme de numéro de baie, tandis que l'ordonnancement partiel fournit pour chaque colonne (donc pour chaque grue) la liste ordonnée des baies dont les conteneurs sont déchargés.
Dans le chromosome du tableau 3.1, nous considérons à chaque itération la baie courante (gène courant) qui correspond à une baie non encore affectée, et nous l'affectons à une grue.

Interférences (étape 1) : La première étape est un test d'interférence qui identifie les deux grues qui sont candidates pour décharger la baie dans le gène courant. Les grues candidates sont celles dont la position actuelle est immédiatement à gauche ou à droite de cette baie. Par exemple, prenons le premier gène du chromosome (baie 7). Au départ, les positions initiales des 3 grues sont respectivement devant les baies 1, 3 et 5. Donc à l'itération 1 du tableau 3.2, seule la grue 3 est candidate pour décharger la baie 7. Par contre à l'itération 3, les grues 2 et 3, respectivement positionnées à gauche et à droite de la baie 4 courante sont candidates pour décharger cette baie, sans interférence avec la grue de quai 1.

Sélection sur critère temporel (étape 2) : Si une seule grue de quai est candidate, les étapes 2 et 3 ne s'appliquent pas et on passe à l'étape 4 directement. Si deux grues sont candidates, nous choisissons celle qui a la plus petite date de disponibilité. Par exemple à l'itération 3 du tableau 3.2, la grue 2 termine son travail à la date 10, contre 20 pour la grue 3. C'est donc la grue 2 qui est choisie pour décharger la baie 4. En cas d'égalité, donc si les deux grues ont la même date de fin de déchargement dans l'ordonnancement partiel, alors on applique un critère de distance à l'étape 3 qui permet de départager les 2 grues candidates. Dans notre exemple, les dates de disponibilité pour la grue de quai 2 et pour la grue de quai 3 sont égales à 30 à l'itération 10.

Sélection sur critère distance (étape 3) : Si aucune grue de quai n'a pu être choisie à l'étape 2 sur la base du critère de date de disponibilité, nous affectons la baie courante à la grue qui en est la plus proche. En cas d'égalité (si les deux grues de quai sont à égale distance de la baie), c'est la grue à gauche (avec le plus petit indice) qui sera sélectionnée. Ainsi à l'itération 10, la grue 2 se trouve en face de la baie 2 donc à une distance de 3 baies de la baie courante numéro 5, contre une distance de 5 baies pour la grue 3 (en position courante 10). La grue 2, plus proche que la grue 3, est donc choisie.

Affectation (étape 4) : Une fois la grue sélectionnée à une itération donnée, on lui affecte la baie courante et on passe à la baie suivante après mise à jour des paramètres temporels (nouvelle date de réalisation pour la grue) et spatiaux (nouvelle position de la grue). Dans notre exemple, à l'itération 1, nous affectons la baie 7 à la grue de quai 3. Nous mettons à jour la position de la grue de quai 3 (qui passe de 5 à 7) et la date d'achèvement associée (qui passe de 0 à 10). Ensuite, le nouveau gène courant devient le gène suivant dans le chromosome (gène 2, baie 9). Les quatre étapes sont répétées jusqu'à ce que toutes les baies soient affectées à une seule grue de quai.

TABLE 3.2 – Exemple détaillé pour 10 baies et 3 grues de quai

Itération		QC1	QC2	QC3
init	position des grues	1	3	5
	ordonnancement partiel	∅	∅	∅
	date d'achèvement	0	0	0
1	position des grues	1	3	7
	ordonnancement partiel	∅	∅	7
	date d'achèvement	0	0	10
2	position des grues	1	3	9
	ordonnancement partiel	∅	∅	7, 9
	completion time	0	0	20
3	position des grues	1	4	9
	ordonnancement partiel	∅	4	7, 9
	date d'achèvement	0	10	20
4	position des grues	1	4	10
	ordonnancement partiel	∅	4	7, 9, 10
	date d'achèvement	0	10	30
5	position des grues	3	4	10
	ordonnancement partiel	3	4	7, 9, 10
	date d'achèvement	10	10	30
6	position des grues	3	8	10
	ordonnancement partiel	3	4, 8	7, 9, 10
	date d'achèvement	10	20	30
7	position des grues	6	8	10
	ordonnancement partiel	3,6	4, 8	7, 9, 10
	date d'achèvement	20	20	30
8	position des grues	1	8	10
	ordonnancement partiel	3,6,1	4, 8	7, 9, 10
	date d'achèvement	30	20	30
9	position des grues	1	2	10
	ordonnancement partiel	3,6,1	4, 8, 2	7, 9, 10
	date d'achèvement	30	30	30
10	position des grues	2	5	10
	ordonnancement partiel	3,6,1	4, 8, 2, 5	7, 9, 10
	date d'achèvement	30	40	30

3.2.3.4/ CALCUL DE LA FITNESS

Dans la section précédente, nous avons proposé une procédure pour éviter les interférences entre les grues de quai. Cependant, nous devons vérifier que toutes les grues de quai satisfont l'ensemble des contraintes. Nous effectuons cette vérification en utilisant les deux équations (3.4) et (3.5) puis nous appliquons la contrainte d'interférence (3.6) pour nous assurer qu'il n'y a pas de collision entre les grues. La phase d'évaluation d'un individu se termine par le calcul de sa *fitness*, réalisée à partir de l'équation (3.10). Si les contraintes ne sont pas satisfaites, la valeur de la *fitness* sera mise à zéro.

$$Fitness = 1/C_{max} \tag{3.10}$$

3.2.3.5/ SÉLECTION

Le processus de sélection permet de choisir des individus dans une population en vue de leur reproduction, en général en fonction de leur *fitness*, pour former une nouvelle population. Les individus évoluent au cours des itérations successives de sélection, appelées générations.

Nous appliquons ici une sélection par roulette, où les individus d'une population sont sélectionnés proportionnellement à la valeur de leur *fitness* : chaque individu occupe une surface de la roue proportionnelle à sa *fitness*, et pour chaque sélection d'un individu, une simple rotation de la roue donne le candidat sélectionné. Donc un individu avec une *fitness* plus élevée aura une plus grande probabilité d'être choisi qu'un individu avec une valeur de *fitness* plus faible. Cette fonction peut être considérée comme une mesure de profit ou de qualité que l'on souhaite maximiser.

De manière plus précise, la sélection fonctionne comme suit : nous calculons tout d'abord la somme de toutes les *fitness*, puis nous générons un nombre aléatoire compris entre 0 et cette somme ; nous ajoutons les valeurs de *fitness* à une somme partielle X à partir du sommet de la population, et enfin le chromosome choisi est le premier chromosome pour lequel X dépasse le nombre aléatoire. La procédure est illustrée dans la figure 3.4.

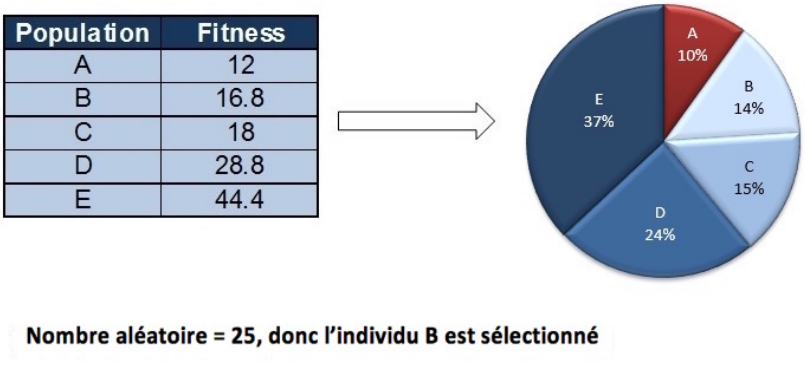

FIGURE 3.4 – Sélection par roulette

3.2.3.6/ CROISEMENT ET MUTATION

Les individus sélectionnés évoluent au sein de la population par application d'opérateurs de croisement et de mutation. Le croisement permet de combiner deux individus parents pour donner naissance à deux nouveaux individus enfants censés hériter des meilleurs caractéristiques de leurs parents. Le croisement que nous appliquons dans notre algorithme génétique est un croisement en 2 points qui consiste à sélectionner au hasard une sous-chaîne d'un des 2 parents choisi aléatoirement. Cette sous-chaîne est copiée dans la même position dans un premier enfant. Ensuite, les gènes du second parent servent à remplir les positions vides de l'enfant, de gauche à droite, en ayant préalablement supprimé du deuxième parent les gènes présents dans la sous-chaîne. Le deuxième enfant est créé en inversant les deux parents dans cette procédure. Ce croisement est illustré par la figure 3.5.

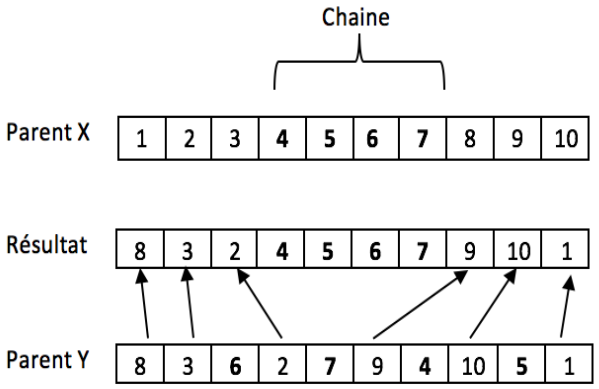

FIGURE 3.5 – Croisement

La procédure de mutation adoptée est une heuristique de type 2-échange qui consiste à choisir au hasard deux positions (deux gènes) dans le chromosome sélectionné, puis à échanger les valeurs de ces deux positions. La procédure est illustrée dans la figure 3.6.

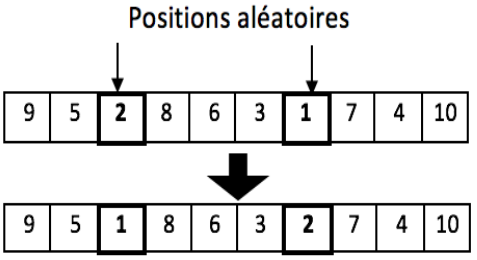

FIGURE 3.6 – Mutation

Pour créer la population à l'itération suivante, nous appliquons une stratégie de remplacement qui consiste à conserver les meilleurs individus, pour constituer 55% de la nouvelle population. Celle-ci est complétée pour 25% par le résultat des croisements et pour 20% par le résultat des mutations.

3.3/ RÉSULTATS EXPÉRIMENTAUX

3.3.1/ IMPLÉMENTATION ET INSTANCES

Dans cette section, nous évaluons les performances de nos méthodes (MILP, algorithme numératif et algorithme génétique). Le MILP a été résolu par CPLEX 12.7.1, tandis que l'algorithme numératif (AN) et l'algorithme génétique (GA) sont codés avec JAVA J2EE. Nos programmes ont été exécutés sur un MacBook Pro Intel Core i5 2,7 GHz avec 8 Go de RAM DDR3 1867 MHz fonctionnant sous OSX 10.11.6.

En termes de paramétrage de notre algorithme génétique, des tests préliminaires nous ont conduits à fixer la taille de la population, la probabilité de la mutation, la probabilité du croisement et le nombre maximum de générations respectivement à 300, 0.2, 0.25

et 1000. Pour chaque instance, entre 5 et 10 exécutions sont effectuées et, dans les tableaux rapportant les résultats d'expérimentation, soit la meilleure valeur du *makespan* est prise en compte, soit on trouve les valeurs pour chaque exécution.

Nous avons testé nos méthodes sur plusieurs jeux d'instances comprenant au total 51 configurations.

- Le premier jeu est constitué de 38 instances générées de manière aléatoire et nommées A1 à A38. Elles comprennent 2 ou 3 grues de quai, 4 à 22 baies, et entre 5 et 30 conteneurs générés aléatoirement pour chaque baie. La répartition des conteneurs dans chacune des 22 baies est la suivante : [13, 16, 12, 9, 13, 23, 11, 28, 29, 16, 7, 7, 11, 12, 23, 13, 16, 12, 9, 13, 23, 11]. Par exemple, pour 4 baies, le nombre de conteneurs dans les baies 1 à 4 est respectivement 13, 16, 12, 9 ; pour une instance à 5 baies, le nombre de conteneurs dans les baies 1 à 5 est respectivement 13, 16, 12, 9 et 13 ;
- Le second jeu correspond à un cas réel lié au port de Tripoli-Liban, à partir duquel nous avons pu générer 6 instances, identifiées TL1 à TL6, comportant 2 grues de quai, 3 à 6 baies et 25 à 600 conteneurs à décharger au total.
- Le troisième jeu est issu de la littérature ([MEISEL et al., 2011]) et comprend 7 instances avec 2 grues et de 10 à 40 tâches, que nous avons appelées MB1 à MB7. Ici le nombre de tâches est égal au nombre de conteneurs pour les auteurs, et au nombre de baies dans notre problème.

3.3.2/ ALGORITHME NUMÉRATIF (AN) VERSUS MILP

Nous avons tout d'abord appliqué nos méthodes exactes sur le premier jeu d'instances. Comme le montre le tableau 3.3, les résultats CPLEX et les résultats AN sont logiquement les mêmes (même durée pour le *makespan*) pour les petites et grandes instances, mais AN est meilleur en termes de temps d'exécution (CPU time). Si pour les 12 premières instances le temps d'exécution peut être considéré comme équivalent pour les deux méthodes, pour les suivantes l'écart se creuse de manière importante en faveur de AN, et ce dès 10 baies, avec des gains en moyenne de 78.65% et variant entre 72.12% et 98.11%, pour les cas où la mesure a été possible.

En effet, pour les plus grandes instances du tableau, notre MILP montre ses limites. En effet, à partir de 14 baies, CPLEX ne parvient pas à résoudre le problème dans un temps donnée. Ainsi, pour l'instance A21 avec 14 baies et 2 grues de quai, CPLEX n'a pas fourni de résultat après 2 heures d'exécution, tandis que le AN a résolu cette instance et fourni une solution optimale. De même, pour les instances A23 et A24, nous avons attendu plus de 3 heures avec CPLEX sans obtenir de résultat. Par contre, ce n'est pas vrai pour l'instance A22 avec 14 baies et 3 grues à quai, pour laquelle nous obtenons la solution optimale avec nos deux méthodes exactes. Pour les autres instances de ce jeu de test (A25 à A38), CPLEX n'a pas non plus fourni de résultat en temps raisonnable, ce qui explique que nous n'avons pas ajouté les lignes correspondantes dans le tableau 3.3.

Ces tests ont permis de valider notre modèle, mais ils montrent aussi de manière prévisible l'efficacité de AN par rapport au MILP. Donc dans la section suivante, nous comparons les performances de notre méthode métaheuristique avec celles du AN.

TABLE 3.3 – Résolution exacte pour les instances générées aléatoirement

No.	$\|Q\|$ x $\|B\|$	Makespan		Temps d'exécution		
		CPLEX (min)	AN (min)	CPLEX (s)	AN (s)	Ecart (%)
A1	2 x 4	32.76	32.76	< 1	< 1	-
A2	3 x 4	24.57	24.57	< 1	< 1	-
A3	2 x 5	39.78	39.78	< 1	< 1	-
A4	3 x 5	29.25	29.25	< 1	< 1	-
A5	2 x 6	51.48	51.48	< 1	< 1	-
A6	3 x 6	37.44	37.44	< 1	< 1	-
A7	2 x 7	58.5	58.5	< 1	< 1	-
A8	3 x 7	39.78	39.78	< 1	< 1	-
A9	2 x 8	73.71	73.71	< 1	< 1	-
A10	3 x 8	51.48	51.48	< 1	< 1	-
A11	2 x 9	90.09	90.09	< 1	< 1	-
A12	3 x 9	60.84	60.84	< 1	< 1	-
A13	2 x 10	99.45	99.45	4.77	1.33	-72.12
A14	3 x 10	70.2	70.2	1.16	0.35	-69.83
A15	2 x 11	104.13	104.13	24.43	3.89	-84.08
A16	3 x 11	73.71	73.71	5.57	1.13	-79.71
A17	2 x 12	107.64	107.64	115.27	13.34	-88.43
A18	3 x 12	73.71	73.71	7.39	3.28	-55.62
A19	2 x 13	114.66	114.66	1723.85	32.59	-98.11
A20	3 x 13	76.05	76.05	20.38	4.57	-77.58
A21	2 x 14	N.A.	119.34	N.A.	82.44	-
A22	3 x 14	81.9	81.9	197.83	34.86	-82.38
A23	2 x 15	N.A.	127.53	N.A.	131.77	-
A24	3 x 15	N.A.	91.26	N.A.	104.82	-

$|Q|$ x $|B|$: nbre de grues x nbre des baies//N.A. : pas de résultat dans le délai imparti
Pour le temps d'exécution, Ecart = $((temps_{AN} - temps_{CPLEX})/temps_{CPLEX}) * 100$

3.3.3/ ALGORITHME GÉNÉTIQUE VERSUS ALGORITHME NUMÉRATIF

3.3.3.1/ COMPARAISON SUR DES INSTANCES ALÉATOIRES

Le tableau 3.4 montre les résultats obtenus en résolvant les instances de notre jeu de test aléatoire, avec AN et GA. Ces résultats sont présentés sous forme du *makespan* obtenu et du temps de résolution associé pour chaque méthode, ces deux facteurs étant comparés en termes d'écart relatif (Ecart1 et Ecart2) pour chaque instance.

Dans le tableau 3.4, nous pouvons observer que pour les instances de petite taille (au moins pour A1 à A6), nous retrouvons les mêmes valeurs de *makespan* avec GA et AN, avec un temps de résolution (CPU time) très rapide, inférieur à la seconde. Les deux méthodes sont donc équivalentes ici et ces premiers tests valident les capacités de notre méthode approchée qui fournit des solutions optimales pour ces instances. A partir de l'instance A7, GA nous donne des solutions proches des solutions optimales obtenues avec AN, avec un écart variant entre 0% et 4.13%. Si nous raisonnons sur l'ensemble des 38 instances, l'écart moyen entre les deux méthodes est seulement de 1.52%.

TABLE 3.4 – Comparaison GA vs AN pour le premier jeu d'instances

No.	$\|Q\|$ x $\|B\|$	Makespan			Temps d'exécution		
		AN (min)	GA (min)	Ecart1 (%)	AN (s)	GA (s)	Ecart2 (%)
A1	2 x 4	32.76	32.76	0	< 1	< 1	-
A2	3 x 4	24.57	24.57	0	< 1	< 1	-
A3	2 x 5	39.78	39.78	0	< 1	< 1	-
A4	3 x 5	29.25	29.25	0	< 1	< 1	-
A5	2 x 6	51.48	51.48	0	< 1	< 1	-
A6	3 x 6	37.44	37.44	0	< 1	< 1	-
A7	2 x 7	58.5	59.67	2	< 1	< 1	-
A8	3 x 7	39.78	39.78	0	< 1	< 1	-
A9	2 x 8	73.71	75.71	2.71	< 1	< 1	-
A10	3 x 8	51.48	52.8	2.72	< 1	< 1	-
A11	2 x 9	90.09	91.26	1.3	< 1	< 1	-
A12	3 x 9	60.84	63.35	4.13	< 1	< 1	-
A13	2 x 10	99.45	100.62	1.18	1.33	< 1	$>$ - 24.81
A14	3 x 10	70.2	72.71	3.58	< 1	< 1	-
A15	2 x 11	104.13	107.6	3.33	3.89	< 1	$>$ - 74.29
A16	3 x 11	73.71	76.02	3.13	1.13	< 1	$>$ - 11.50
A17	2 x 12	107.64	109.98	2.17	13.34	< 1	$>$ - 92.50
A18	3 x 12	73.71	74.88	1.58	3.28	< 1	$>$ - 69.51
A19	2 x 13	114.66	115.83	1.02	32.59	< 1	$>$ - 96.93
A20	3 x 13	76.05	76.05	0	4.57	< 1	$>$ - 78.12
A21	2 x 14	119.34	122.85	2.94	112.44	< 1	$>$ - 99.11
A22	3 x 14	81.9	83.07	1.42	54.86	< 1	$>$ - 98.18
A23	2 x 15	127.53	129.87	1.83	231.77	< 1	$>$ - 99.57
A24	3 x 15	91.26	92.43	1.28	104.82	< 1	$>$ - 99.05
A25	2 x 16	132.21	132.21	0	303.04	2.31	- 99.24
A26	3 x 16	95.94	98.28	2.44	112.94	1.59	- 98.59
A27	2 x 17	139.23	142.74	2.52	509.96	4.36	- 99.15
A28	3 x 17	101.79	105.3	3.45	203.34	3.12	- 98.47
A29	2 x 18	146.25	147.42	0.9	1144.12	9.67	- 99.15
A30	3 x 18	105.3	106.47	1.11	708.32	7.39	- 98.96
A31	2 x 19	153.27	155.61	1.53	1392.84	13.34	- 99.04
A32	3 x 19	113.49	113.49	0	912.48	10.62	- 98.84
A33	2 x 20	161.46	164.97	2.17	1601.13	32.99	- 97.94
A34	3 x 20	118.17	119.34	0.99	1044.19	28.64	- 97.26
A35	2 x 21	170.82	173.16	1.37	3563.7	58.28	- 98.36
A36	3 x 21	125.19	127.53	1.87	2390.13	34.72	- 98.55
A37	2 x 22	N.A.	180.18	-	N.A.	77.14	-
A38	3 x 22	N.A.	134.69	-	N.A.	56.18	-

$|Q|$ x $|B|$: nbre de grues x nbre des baies//N.A. : pas de résultat dans le délai imparti
Ecart1 et Ecart2 calculés sur la base : $((valeur_{GA} - valeur_{AN})/valeur_{AN}) * 100$

Mais l'avantage de GA est un temps de résolution bien plus faible, en moyenne supérieur à 88%, et dans la grande majorité des grandes instances supérieur à 97%. Plus précisément, ce temps de résolution avec GA reste inférieur à la seconde jusqu'à A24

inclus, puis reste inférieur à la minute pour des instances plus grandes, tandis qu'avec AN, il augmente progressivement avec la taille du problème entre A13 et A36, pour atteindre 40 minutes. À partir de l'instance A37, AN n'a pu fournir aucune solution après plus de 4 heures d'exécution de l'algorithme, tandis que GA fournit des solutions dans un temps très réduit. Nous pouvons donc dire que l'algorithme génétique proposé est réalisable et pratique pour notre problème, mais il reste à tester sa stabilité.

Stabilité de l'algorithme génétique

Pour évaluer si notre algorithme génétique est stable, nous avons exécuté la même instance 5 fois. Pour les petites instances, nous avons obtenu le même résultat, comme indiqué dans le tableau 3.5, c'est à dire que toutes les exécutions trouvent la solution optimale à chaque exécution.

TABLE 3.5 – Analyse de stabilité pour les petites instances

	A1	A2	A3	A4
Exécution	2 x 4	3 x 4	2 x 5	3 x 5
1	32.76	24.57	39.78	29.25
2	32.76	24.57	39.78	29.25
3	32.76	24.57	39.78	29.25
4	32.76	24.57	39.78	29.25
5	32.76	24.57	39.78	29.25

Pour de plus grandes instances (tableau 3.6), au moins 40% des exécutions (au moins 2 exécutions sur les 5) donnent le meilleur résultat parmi ceux trouvés, proche de l'optimum. Les autres exécutions donnent des résultats éloignés de ce meilleur résultat d'au plus 1 à 3% suivant les instances.

TABLE 3.6 – Analyse de stabilité pour les grandes instances

	A17	A18	A35	A36
Exécution	2 x 12	3 x 12	2 x 21	3 x 21
1	109.98*	74.88*	173.16*	127.53*
2	113.49	76.39	173.16*	127.53*
3	109.98*	77.22	175.5	128.7
4	109.98*	77.22	173.16*	128.7
5	112.32	74.88*	175.5	127.53*
moyenne	111.15	76.12	174.096	127.998
écart type	1.48	1.06	1.15	0.57
makespan optimal	107.64	73.71	170.82	125.19

*meilleur *makespan* obtenu

Par exemple, pour 2 grues de quai et 12 baies (instance A17), les première, troisième et quatrième exécutions donnent la solution proche de l'optimale obtenue par l'algorithme génétique (109,98), tandis que les deuxième et cinquième exécutions donnent des résultats à plus ou moins 3% de la meilleure solution trouvée par GA.

D'après le tableau 3.6, la moyenne des 5 exécutions est à environ 3% (respectivement 2%) de la solution optimale pour les instances A17 et A18 (respectivement A35 et A36), et

l'écart type est très faible, ce qui est plutôt positif et nous conduit à penser que l'algorithme génétique proposé est efficace et stable.

3.3.3.2/ COMPARAISON SUR DES INSTANCES RÉELLES

Nous avons à nouveau comparé les performances de nos deux méthodes, exacte AN et approchée GA, cette fois sur des instances issues du port de Tripoli au Liban. Nous avons donc exécuté nos algorithmes sur 6 instances décrites précédemment, et dont nous retrouvons les caractéristiques dans les premières colonnes du tableau 3.7. Notons que pour ces exemples, le temps de déchargement d'un conteneur par une grue de quai est d'environ 1,17 minutes.

Dans le tableau 3.7, les résultats que nous avons obtenus auprès de la direction du port sont comparés à ceux de l'algorithme numératif (AN), tandis que dans le tableau 3.8, ils sont comparés aux résultats de l'algorithme génétique (GA). La dernière colonne fournit pour chaque instance l'écart relatif entre le *makespan* du port et celui de notre algorithme.

TABLE 3.7 – Résultats AN vs port

No.	Grues	Baies	Conteneurs	Makespan port	Makespan AN	Temps d'exécution	Ecart %
TL1	2	6	25	40	19.5	< 1	-51.25
TL2	2	3	40	70	32.76	< 1	-53.20
TL3	2	4	62	95	43.29	< 1	-54.43
TL4	2	3	90	103	70.2	< 1	-31.84
TL5	2	4	84	96	49.14	< 1	-48.81
TL6	2	5	600	480	351	< 1	-26.88

$Ecart = ((makespan_{AN} - makespan_{port})/makespan_{port}) * 100$

TABLE 3.8 – Résultats GA vs port

No.	Grues	Baies	Conteneurs	Makespan port	Makespan GA	Temps d'exécution	Ecart %
TL1	2	6	25	40	21.84	< 1	-45.40
TL2	2	3	40	70	34.89	< 1	-50.16
TL3	2	4	62	95	45.95	< 1	-51.63
TL4	2	3	90	103	72.54	< 1	-29.57
TL5	2	4	84	96	52.01	< 1	-45.82
TL6	2	5	600	480	355.68	< 1	-25.90

$Ecart = ((makespan_{GA} - makespan_{port})/makespan_{port}) * 100$

Pour les instances du port de Tripoli, nous observons que nos deux méthodes améliorent de manière substantielle les résultats du port, en moyenne de 44.4% pour AN et de 41.41% pour GA (tableaux 3.7 et 3.8). L'écart relatif moyen entre AN et GA est de 5.86% en faveur de AN, ce qui est 2 fois plus important que pour le premier jeu d'instances testé, mais cela provient de la première instance du port qui a le plus fort écart relatif à 12% et entraîne donc une augmentation de cette moyenne. Si on retire cet exemple, donc pour les 5 autres instances, l'écart relatif moyen redescend à 4.63%, pour des variations entre

1.33 et 6.50%. Notons que l'écart le plus faible à 1.33% correspond à la plus grande instance testée TL6 comprenant 600 conteneurs à décharger, ce qui montre que notre métaheuristique GA peut être très efficace pour des problèmes réalistes.

Le graphe situé dans la figure 3.7 illustre également les résultats présentés dans les tableaux 3.7 et 3.8). Les barres bleues correspondent aux résultats du port de Tripoli, tandis que les rouges et les vertes concernent respectivement les résultats de AN et de GA. Nous pouvons observer ici que le *makespan* du port est bien supérieur à celui issu de nos deux méthodes. Par ailleurs, dans les 3 cas, le *makespan* augmente avec le nombre de conteneurs (par exemple, le *makespan* de l'instance TL6 est bien plus élevé en raison du grand nombre de conteneurs).

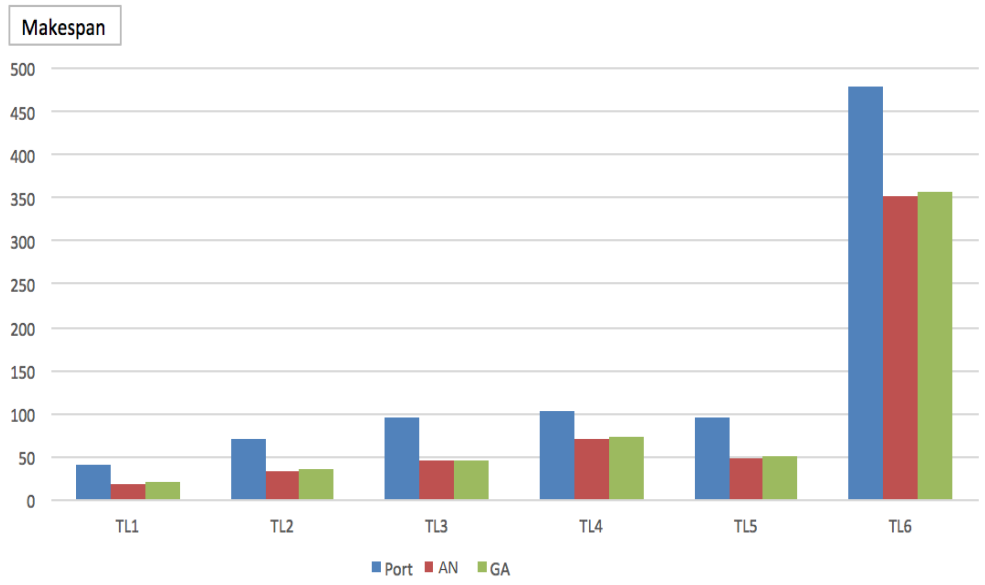

FIGURE 3.7 – *Makespan* par méthode et par instance

A titre d'exemple, nous présentons dans le tableau 3.9 l'ordonnancement obtenu pour un exemple à 4 baies et 2 grues de quai, et 62 conteneurs (avec respectivement 25, 13, 14 et 10 conteneurs dans les baies 1 à 4). L'affectation des grues est 1-2-2-2, ce qui signifie que la grue QC1 travaille au niveau de la baie 1, et que la seconde grue QC2 décharge les conteneurs des 3 autres baies. La date de fin de manutention de QC1 est 29.25 minutes et pour QC2 elle est égale à 43.29 minutes (15.21+16.38+11.7). Le *makespan* est la plus grande de ces deux valeurs, donc 43.29 minutes.

TABLE 3.9 – Ordonnancement pour l'instance TL3 avec 4 baies et 2 grues de quai

baies	B1	B2	B3	B4
nombre de conteneurs	25	13	14	10
durée de déchargement par conteneur	1.17			
affectation des grues	QC_1	QC_2	QC_2	QC_2
date de fin par baie	29.25	15.21	16.38	11.7
date de fin par grue	29.25	43.29		

3.3.4/ ALGORITHME GÉNÉTIQUE VERSUS LITTÉRATURE

Dans cette sous-section, nous comparons les résultats obtenus par notre algorithme génétique avec ceux de Frank Meisel et Christian Bierwirth ([MEISEL et al., 2011]). Ces derniers ont présenté une approche dite unifiée pour évaluer les performances de différentes classes de modèles et procédures de résolution. Ils ont implémenté en Java un schéma de génération d'instances nommé QCSPgen (site http ://prodlog.wiwi.uni-halle.de/qcspgen consulté en 2020), qui s'exécute sur n'importe quel système d'exploitation avec un environnement JavaRuntime. Une interface graphique est utilisée pour définir les paramètres pour le schéma de génération des instances (Figure 3.8). Dans cette interface, le nombre de tâches correspond au nombre de tâches effectuées par les grues (nombre des conteneurs), la capacité des baies est exprimée en nombre de conteneurs, le*handling rate* représente le pourcentage de conteneurs manutentionnés dans un service par rapport à la capacité totale du navire, le paramètre de position correspond à la distribution des groupes de conteneurs sur l'ensemble des baies, (*uni* signifie que les conteneurs seront distribués de manière uniforme), la densité de précédence traduit la génération des relations de priorité entre les groupes de conteneurs, la densité de simultanéité représente la génération de relations non simultanées entre les groupes de conteneurs, le nombre aléatoire *seed* sert à initialiser le générateur de nombres pseudo-aléatoires utilisé dans le schéma de génération, et enfin la marge de sécurité est exprimée en nombre de baies minimal à maintenir entre deux grues voisines. Cependant, [MEISEL et al., 2011] n'ont pas limité le paramétrage de QCSPgen, pour fournir la plus grande flexibilité possible pour la génération d'instances. Leur interface graphique permet ainsi de "court-circuiter" les paramètres non utilisés. Par exemple si on ne veut pas utiliser la marge de sécurité, il suffit de fixer sa valeur à zéro.

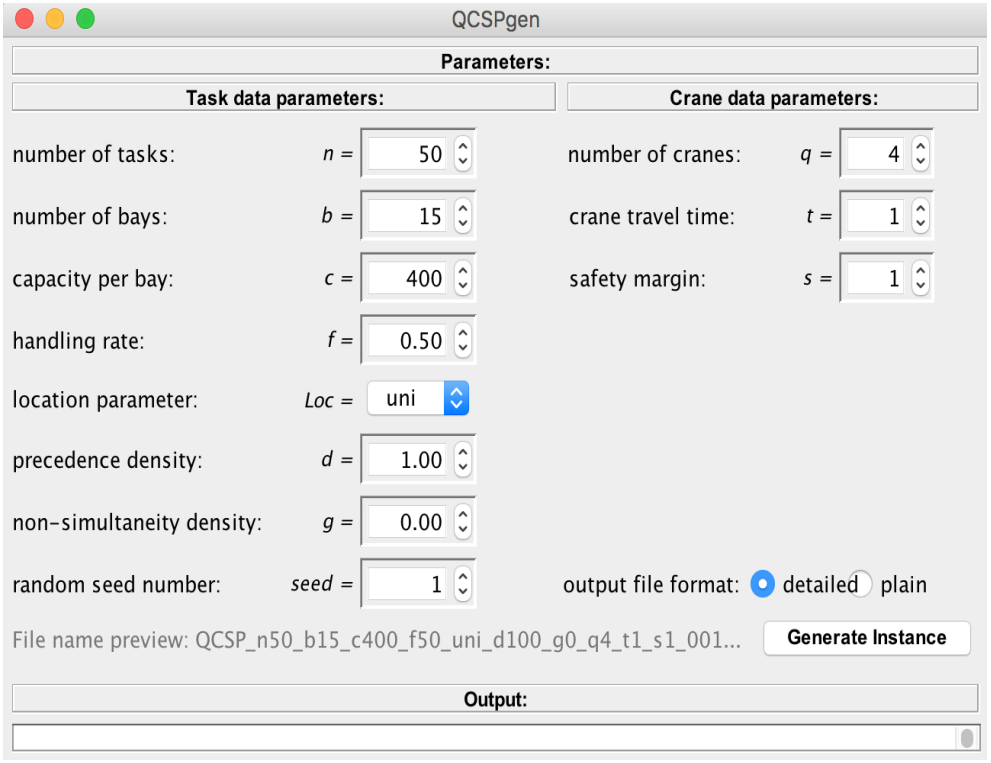

FIGURE 3.8 – QCSPgen

Le tableau 3.10 compare les paramètres que nous considérons et ceux utilisés par [MEISEL et al., 2011]. Si les paramètres semblent de prime abord identiques, deux différences sont néanmoins à souligner. La première est que dans le problème que nous étudions, chaque grue de quai doit terminer le déchargement de tous les conteneurs dans la baie actuelle avant de passer à une autre baie. Donc pour nous, le nombre de tâches à réaliser peut être assimilé au nombre de baies du navire à décharger. La seconde différence est que nous avons choisi de négliger les temps de déplacement d'une grue entre deux baies, car nous considérons que leur valeur est faible par rapport aux autres temps. [MEISEL et al., 2011] n'ont pas fait ces hypothèses dans leur travail. Le choix de ces deux hypothèses provient en partie des observations que nous avons faites au niveau des opérations sur le port de Tripoli, qui nous ont servi de point de départ pour définir notre variante de base du problème QCSP. Ceci dit, nous sommes aussi conscients que ces observations ne sont pas forcément systématiquement généralisables.

TABLE 3.10 – Comparaison avec [MEISEL et al., 2011]

Paramètres	[MEISEL et al., 2011]	GA proposé
Nombre de tâches	x	[a]
Nombre de baies	x	x
Nombre de grues de quai	x	x
Nombre de conteneurs	x	x
Temps de déplacement de grues	x	[b]
Capacité de chaque baie	x	x
Marge de sécurité	x	x

[a] : Puisque chaque grue de quai doit terminer le déchargement de tous les conteneurs avant de passer à une autre baie, le nombre de baies est égal au nombre de tâches.
[b] : Le temps de déplacement d'une grue de quai entre deux baies est ignoré car il est petit par rapport aux autres variables (10 secondes en moyenne).

Le tableau 3.11 compare les *makespan* obtenus par l'algorithme développé par [MEISEL et al., 2011] avec notre algorithme GA. L'approche de [MEISEL et al., 2011] consiste en une hybridation d'une méthode approchée (type GRASP) et d'une méthode exacte (*Branch and Bound*) inspirées de [Kim et al., 2004]. Le tableau précise pour chaque instance le nombre de grues et de tâches, sachant que dans notre cas le nombre de tâches est égal au nombre de baies du navire. Enfin, l'écart relatif entre les *makespan* moyens est calculé, où le *makespan* moyen est obtenu pour 10 exécutions réalisées.

Le tableau 3.11 montre que nos résultats sont acceptables par rapport à ceux de [MEISEL et al., 2011], l'écart relatif moyen variant entre 4.46% et 6.04%, et ce bien qu'il y ait une différence dans le fonctionnement des grues de quai. En effet, nous avons utilisé les mêmes variables que [MEISEL et al., 2011], afin de les intégrer dans notre algorithme génétique. Néanmoins, les deux différences relevées précédemment peuvent expliquer les écarts observés. En effet, l'application de notre méthode exacte AN montre que le *makespan* optimal que nous obtenons est supérieur à la solution de [MEISEL et al., 2011]. Par exemple pour l'instance MB1 (respectivement MB2), la solution optimale de notre problème est égale à 528 (respectivement 523), soit 2.25% (respectivement 2.57%) au-dessus de [MEISEL et al., 2011] qui traite un problème un peu différent. Cela est dû au fait que nos hypothèses de départ limitent les possibilités d'affectation des grues, donc de solutions, puisque nous imposons qu'une seule grue soit affectée à une baie donnée. Notons également ici que cela ramène l'écart relatif de notre algorithme GA à la solu-

tion optimale à respectivement 2.46% et 3%, ce qui est cohérent avec les performances obtenues avec le premier jeu de tests.

Au vu des résultats, et bien que notre algorithme génétique donne des résultats un peu moins bons que l'algorithme de [MEISEL et al., 2011], nous remarquons qu'il semble relativement plus stable. En effet, les écarts types calculés par instance varient entre 2.4 et 5.85, contre une variation entre 0.63 et 11.28 pour [MEISEL et al., 2011]. Néanmoins ces écarts types restent faibles par rapport aux moyennes calculées pour le *makespan*.

TABLE 3.11 – Comparaison des résultats avec [MEISEL et al., 2011]

No.	**Ex.**	**Nombre de grues x nombre de tâches**	**[MEISEL et al., 2011]**	**GA proposé**	**Écart**
MB1	1	2 x 10	520	539	
	2		508	541	
	3		513	539	
	4		510	539	
	5		515	539	
	6		513	545	
	7		511	539	
	8		513	545	
	9		512	539	
	10		549	545	
Moyenne			516.4	541	4.76
Écart type			11.28	2.68	
MB2	1	2 x 15	514	536	
	2		507	536	
	3		515	536	
	4		513	536	
	5		507	545	
	6		508	536	
	7		507	536	
	8		508	545	
	9		507	536	
	10		513	545	
Moyenne			509.9	538.7	5.65
Écart type			3.21	4.12	
MB3	1	2 x 20	508	534	
	2		509	542	
	3		509	534	
	4		509	534	
	5		506	542	
	6		508	534	
	7		507	546	
	8		510	542	
	9		508	546	
	10		507	534	
Moyenne			508.1	538.8	6.04
Écart type			1.14	5	

Ins.	Ex.	Nombre de grues x nombre de tâches	[MEISEL et al., 2011]	GA proposé	Écart
MB4	1	2 x 25	508	532	
	2		507	532	
	3		507	546	
	4		507	532	
	5		507	532	
	6		507	532	
	7		508	541	
	8		507	546	
	9		506	532	
	10		506	541	
Moyenne			507	536.6	5.84
Écart type			0.63	5.85	
MB5	1	2 x 30	506	529	
	2		508	529	
	3		507	529	
	4		507	529	
	5		506	529	
	6		506	529	
	7		508	535	
	8		508	535	
	9		506	529	
	10		506	529	
Moyenne			506.8	530.2	4.62
Écart type			0.87	2.4	
MB6	1	2 x 35	506	527	
	2		507	536	
	3		506	527	
	4		507	536	
	5		507	527	
	6		511	527	
	7		507	527	
	8		506	527	
	9		506	536	
	10		508	527	
Moyenne			507.1	529.7	4.46
Écart type			1.45	4.12	
MB7	1	2 x 40	506	531	
	2		506	531	
	3		505	531	
	4		507	539	
	5		506	531	
	6		507	531	
	7		507	531	
	8		506	531	
	9		506	539	
	10		507	542	
Moyenne			506.3	533.7	5.41
Écart type			0.64	4.2	

3.4/ CONCLUSION

Dans ce chapitre, nous avons étudié le problème d'ordonnancement des activités de déchargement d'un navire porte-conteneurs par plusieurs grues de quai. Après avoir posé les hypothèses de notre étude, nous avons modélisé ce problème par un programme linéaire en nombres mixtes (MILP). Nous avons proposé trois méthodes de résolution, dont deux exactes en résolvant notre MILP par CPLEX, et en développant un algorithme numératif. Ensuite, nous avons développé un algorithme génétique pour résoudre le problème de manière approchée. Pour valider notre modèle et nos algorithmes, nous avons réalisé des expérimentations sur 3 jeux d'instances incluant des cas réels et des benchmarks de la littérature. Ces tests montrent que notre algorithme génétique présente un bon compromis entre qualité des solutions et temps de résolution, en particulier quand la taille des instances augmente. De manière générale, les résultats obtenus, notamment pour des instances issues d'un cas d'étude réel, nous semblent encourageants. Dans les chapitres suivants, nous étudions deux extensions du problème QCSP, pour considérer notamment ce qui se passe après déchargement sur les quais du terminal maritime.

4

ORDONNANCEMENT D'UNE GRUE DE QUAI ET DE PLUSIEURS CAMIONS DE TRANSPORT

Dans ce chapitre, nous étudions le problème d'ordonnancement des activités de manutention des conteneurs pour une grue à quai unique et plusieurs camions de transport, avec intégration des chariots frontaux en entrée de la zone de stockage. Il s'agit d'une variante de *Quay Crane and Yard Truck Scheduling Problem* (QCYTSP). Pour la résoudre, nous proposons deux méthodes exactes (programme linéaire et algorithme numératif) pour obtenir une solution optimale, ainsi qu'une méthode métaheuristique (algorithme génétique) pour obtenir une solution proche de l'optimale avec un temps d'exécution acceptable. L'objectif considéré est de minimiser la durée de manutention de tous les conteneurs du porte-conteneurs entre le navire et la zone de stockage. Enfin, nous comparons nos résultats sur divers jeux d'instances dont ceux de la littérature et des cas réels issus du port de Tripoli au Liban.

4.1/ DÉFINITION DU PROBLÈME ET FORMULATION MATHÉMATIQUE

4.1.1/ INTRODUCTION

Le chapitre précédent nous a permis de résoudre une variante de base des problèmes d'ordonnancement des grues de quai (*Quay Crane Scheduling Problem*). Néanmoins, une fois les conteneurs déchargés par les grues sur le quai, ils doivent encore être acheminés vers des zones de stockage, comme défini au chapitre d'état de l'art. Pour cela, en plus des grues (*quay cranes*), deux autres modes de transport sont nécessaires : les camions (*yard trucks*) et les chariots frontaux (*reach stackers*) (Figure 4.1). Ce nouveau chapitre étudie donc une seconde variante qui intègre ce cheminement, en cherchant à minimiser le temps total des opérations subies successivement par les conteneurs. Nous nous intéressons donc ici à un *Quay Crane and Yard Truck Scheduling Problem* (QCYTSP).

Dans cette section, nous fournissons une description du problème traité, que nous modélisons sous forme mathématique.

FIGURE 4.1 – Transbordement de conteneurs entre le navire et la zone de stockage

4.1.2/ HYPOTHÈSES

Le QCYTSP que nous étudions consiste à ordonnancer le déchargement de conteneurs depuis un navire vers les zones de stockage du port, les conteneurs étant successivement pris en charge par une grue de quai, des camions et des chariots frontaux. Il présente les caractéristiques suivantes, qui se traduiront par des contraintes dans notre modèle :

- Une seule grue de quai est considérée ainsi que plusieurs camions de transport.
- Tous les conteneurs sont similaires et ont la même largeur et la même hauteur.
- La grue de quai peut décharger au plus un conteneur à la fois.
- Chaque camion ne peut transporter qu'un seul conteneur à la fois.
- Chaque conteneur peut être transporté par au plus un camion.
- Nous ne prenons pas en compte le nombre des chariots frontaux, nous supposons qu'il y a toujours un chariot frontal disponible pour transférer un conteneur du camion au stock.

4.1.3/ DONNÉES

- C : ensemble de conteneurs
- $|C|$: nombre de conteneurs, auquel nous ajoutons 2 conteneurs fictifs représentant respectivement l'état d'origine (indexé 0) et le second l'état final (indexé $C+1$). Ces ajouts fictifs permettent d'avoir pour chaque conteneur réel un conteneur qui le précède et un qui le suit, y compris pour le premier et le dernier de la séquence.
- i : index de conteneur ($\forall i \in C$)
- j : index de conteneur ($\forall j \in C$)
- T : ensemble de camions
- $|T|$: nombre de camions
- t : index de camion ($\forall t \in T$)
- w_i : temps pour décharger le conteneur i ($\forall i \in C$) par la grue de quai et le poser sur le camion
- t_{ij} : temps de repositionnement (*setup time*) de la grue de quai entre la dépose du conteneur i dans un camion et la prise du conteneur suivant j sur le navire ($\forall i \in \{0...|C|\}$, $\forall j \in \{1...|C|\}$), lorsque le conteneur j est déchargé après le conteneur i. Notons qu'ici, nous avons relaxé la contrainte où une grue doit décharger tous les conteneurs d'une baie avant de passer à la baie suivante.
- λ_i : temps de transport du conteneur i ($\forall i \in \{1...|C|\}$) par un camion jusqu'à la zone

de stockage (identique au temps de retour du camion depuis la zone de stockage pour aller chercher un nouveau conteneur déchargé par la grue de quai)
— r : temps nécessaire à un chariot frontal pour décharger un conteneur du camion de transport
— M : très grand nombre (entier utilisé pour représenter l'infini)

4.1.4/ VARIABLES DE DÉCISIONS

— $A_{ij} \begin{cases} = 1 & \text{si la grue de quai décharge le conteneur } j \text{ directement après} \\ & \text{le déchargement du conteneur } i, \forall i \in \{0...|C|\}, \forall j \in \{1...|C|+1\} \\ = 0 & \text{sinon, } (A_{ii} = 0) \end{cases}$

— $B_{it} \begin{cases} = 1 & \text{si le conteneur } i \text{ est affecté au camion } t \text{ , } \forall i \in \{1...|C|\}, \forall t \in \{1...|T|\} \\ = 0 & \text{sinon} \end{cases}$

— $X_{ijt} \begin{cases} = 1 & \text{si le camion } t \text{ transporte le conteneur } j \text{ directement après} \\ & \text{avoir transporté le conteneur } i, \forall i \in \{0...|C|\}, \forall j \in \{1...|C|+1\}, \forall t \in \{1...|T|\} \\ = 0 & \text{sinon, } (X_{iit} = 0) \end{cases}$

— s_i : date à laquelle la grue de quai commence à décharger le conteneur i, $\forall i \in \{1...|C|\}$
— s'_i : date de début de transport du conteneur i par un camion, $\forall i \in \{0...|C|\}$ $(s'_0 = 0)$
— cp_i : date de fin de manutention du conteneur i, $\forall i \in \{1...|C|\}$
— C_{max} : date de fin de manutention de tous les conteneurs (*makespan*)

4.1.5/ MODÈLE LINÉAIRE POUR LE QCYTSP

Dans cette section, nous proposons un modèle sous forme de programme linéaire mixte en nombres entiers, pour l'ordonnancement de la grue de quai et des camions de transport :

Objectif

$$minimize \quad C_{max} \tag{4.1}$$

L'objectif est de minimiser le temps total de manutention de l'ensemble des conteneurs affectés à la grue de quai, depuis le navire jusqu'à la zone de stockage.

Sous les contraintes :

$$\sum_{i=0}^{|C|} A_{ij} = 1 \quad \forall j \in \{1...|C|+1\} \tag{4.2}$$

$$\sum_{j=1}^{|C|+1} A_{ij} = 1 \quad \forall i \in \{0...|C|\} \tag{4.3}$$

Les contraintes (4.2) et (4.3) sont utilisées pour affecter tous les conteneurs à la grue de quai et déterminer la séquence de déchargement du navire.

$$\sum_{t=1}^{|T|} B_{it} = 1 \quad \forall i \in \{1...|C|\} \tag{4.4}$$

La contrainte (4.4) impose qu'un conteneur i est transporté par exactement un et un seul camion.

$$\sum_{j=1}^{|C|+1} X_{0jt} = 1 \quad \forall t \in \{1...|T|\} \tag{4.5}$$

$$\sum_{i=0}^{|C|} X_{i,|C|+1,t} = 1 \quad \forall t \in \{1...|T|\} \tag{4.6}$$

Les contraintes (4.5) et (4.6) sont utilisées pour trouver le premier conteneur et le dernier conteneur qui seront transportés par chaque camion.

$$\sum_{j=1}^{|C|+1} X_{ijt} = B_{it} \quad \forall i \in \{1...|C|\}, \forall t \in \{1...|T|\} \tag{4.7}$$

$$\sum_{i=0}^{|C|} X_{ijt} = B_{jt} \quad \forall j \in \{1...|C|\}, \forall t \in \{1...|T|\} \tag{4.8}$$

Les contraintes (4.7) et (4.8) assurent que pour chaque conteneur transporté par le camion, il y a un conteneur qui le précède et un autre qui le suit.

$$X_{ijt} + X_{jit} \leq 1 \quad \forall i, j \in \{1...|C|\}, \forall t \in \{1...|T|\} \tag{4.9}$$

$$X_{ijt} + X_{jit} \leq B_{it} + B_{jt} + 1 \quad \forall i, j \in \{1...|C|\}, \forall t \in \{1...|T|\} \tag{4.10}$$

Les contraintes (4.9) et (4.10) déterminent les conteneurs qui doivent être déplacés par le même camion.

$$s_j + (1 - A_{ij}) * M \geq t_{ij} + s'_i \quad \forall i \in \{0...|C|\}, \forall j \in \{1...|C|\} \tag{4.11}$$

La contrainte (4.11) garantit qu'il y a un temps de déplacement à vide t_{ij} (repositionnement) pour la grue de quai avant de commencer à décharger le conteneur j après le déchargement du conteneur i.

$$s'_i \geq s_i + w_i \quad \forall i \in \{1...|C|\} \tag{4.12}$$

La contrainte (4.12) garantit que chaque conteneur ne peut être transporté par un camion qu'après avoir été déchargé du navire par la grue de quai.

$$s'_j + (1 - X_{ijt}) * M \geq cp_i + \lambda_i \quad \forall i \in \{1...|C|\}, \forall j \in \{1...|C|\}, \forall t \in \{1...|T|\}, \tag{4.13}$$

$$s'_j + (1 - X_{0jt}) * M \geq 0 \quad \forall j \in \{1...|C|\}, \forall t \in \{1...|T|\} \tag{4.14}$$

Les contraintes (4.13) et (4.14) établissent la relation entre les dates de fin de transport (incluant le retour à vide) et de début de transport de deux conteneurs déplacés successivement par un même camion.

$$s'_i + \lambda_i + r \leq cp_i \quad \forall i \in \{1...|C|\} \tag{4.15}$$

Contrainte (4.15) vérifie que chaque conteneur subit des temps de transport par un camion puis de gerbage par un chariot frontal après son déchargement par la grue.

$$C_{max} \geq cp_i \quad \forall i \in \{1...|C|\} \tag{4.16}$$

La contrainte (4.16) calcule la date de fin de manutention du dernier conteneur.

$$A_{ij} = [0,1] \quad \forall i \in \{0...|C|\}, \quad \forall j \in \{1...|C|+1\} \tag{4.17}$$

$$B_{it} = [0,1] \quad \forall i \in \{1...|C|\}, \quad \forall t \in \{1...|T|\} \tag{4.18}$$

$$X_{ijt} = [0,1] \quad \forall i \in \{0...|C|\}, \quad \forall j \in \{1...|C|+1\}, \forall t \in \{1...|T|\} \tag{4.19}$$

$$s_i, s'_i, cp_i \in \mathbb{R} \tag{4.20}$$

Les contraintes (4.17), (4.18), (4.19) et (4.20) définissent le domaine des variables de décision.

4.2/ MÉTHODES DE RÉSOLUTION DU PROBLÈME

Dans cette section, nous proposons à nouveau plusieurs méthodes pour résoudre le problème d'ordonnancement des grues de quai et des camions de transport : la résolution de notre programme linéaire en nombres entiers (MILP3) se fait avec CPLEX, nous ne développerons donc pas cette partie ici, mais fournirons plus tard le résultat des tests menés. Par contre nous détaillons l'algorithme numératif et l'algorithme génétique que nous avons élaborés. Au préalable, nous proposons des bornes inférieures et supérieures pour notre problème.

4.2.1/ BORNES INFÉRIEURES ET BORNES SUPÉRIEURES

Dans cette section, nous proposons une analyse des limites inférieures et supérieures pour le problème d'ordonnancement de grue de quai et des camions de transport. En optimisation, la détermination de bornes inférieure et supérieure de la fonction objectif permet d'encadrer la solution optimale et d'accélérer la résolution du problème en convergeant plus rapidement vers cette solution (Figure 4.2).

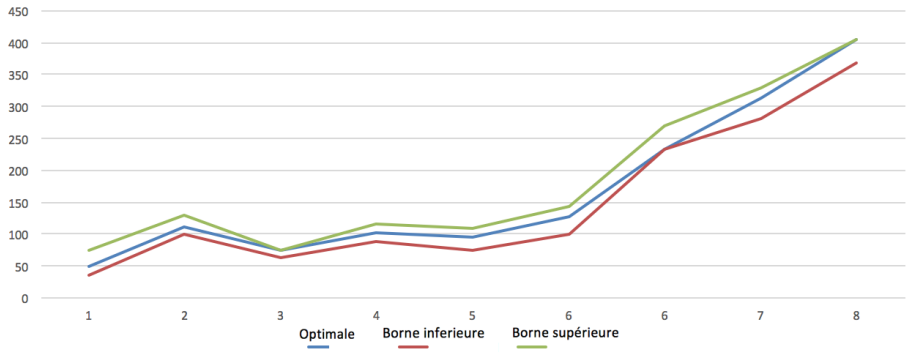

FIGURE 4.2 – Solution optimale, bornes inférieures et bornes supérieures

Les limites inférieures que nous proposons sont explicitées par les deux équations (4.21) et (4.22). La première borne est basée sur les conteneurs et la seconde est basée sur les ressources de transport. La limite supérieure est donnée par l'équation (4.24).

4.2.1.1/ BORNE INFÉRIEURE

Les deux équations suivantes nous donnent la solution réalisable de limite inférieure pour le problème d'ordonnancement de grue de quai et de camion de transport.

L'équation (4.21) montre la limite inférieure basée sur les conteneurs. En effet, pour chaque conteneur, le cycle de déchargement comprend le repositionnement de la grue depuis sa position courante vers la baie contenant le conteneur, le temps de déchargement par la grue et de dépose sur un camion, le temps de transport vers la zone de stockage affectée, et le temps de gerbage du conteneur du camion par un chariot frontal.

$$lb_1 = \max_{i=1...|C|} (\min_{j=0...|C|} t_{ji} + w_i + \lambda_i + r) \tag{4.21}$$

L'équation (4.22) traduit une borne inférieure basée sur les ressources de manutention.

$$\begin{aligned} lb_2 = & \underbrace{\min_{i=1..|C|} (w_i + \min_{j=0..|C|} t_{ji})}_{première-partie} + \underbrace{\frac{\sum_{i=1}^{|C|}(2\lambda_i + r)}{|T|}}_{deuxième-partie} \\ & + \underbrace{\frac{1}{|T|} \sum_{t=1}^{|T|-1} [M^{[t]}_{i=1..|C|}(w_i + \min_{j=0..|C|} t_{ji}) - \min_{i=1..|C|} (w_i + \min_{j=0..|C|} t_{ji})] - \min_{i=1..|C|} \lambda_i}_{troisième-partie} \end{aligned} \tag{4.22}$$

où $M^{[t]}_{i=1..|C|}(w_i + \min_{j=0..|C|} t_{ji})$ est le $(t+1)^{ème}$ plus petit élément d'une liste R dont le $i^{ème}$ élément prend la valeur $w_i + \min_{j=0..|C|} t_{ji}$. Prenons l'exemple du problème dans lequel $C = 3$, $w_1 = 4$, $w_2 = 3$, $w_3 = 5$, $t_{01} = t_{02} = t_{03} = 2$, $t_{12} = t_{21} = 2$, $t_{13} = t_{31} = 4$ et $t_{23} = t_{32} = 5$. Dans ce cas : $R = ((w_1 + min(t_{01}\ ,\ t_{11}\ ,\ t_{21}\ ,\ t_{31}))\ ;\ (w_2 + min(t_{02}\ ,\ t_{12}\ ,\ t_{22}\ ,\ t_{32}))\ ;\ (w_3 + min(t_{03}\ ,\ t_{13}\ ,\ t_{23}\ ,\ t_{33})))$. Cela donne R=(6, 5, 9), et par conséquent $M_R^{[0]}$=min(6,5,9)=5, $M_R^{[1]}$=min(6,9) et $M_R^{[2]}$=9.

La première partie de l'équation (4.22) représente le temps minimal requis par la grue pour qu'un premier conteneur soit prêt à être transporté par un camion. Ce temps comprend les temps de travail et de repositionnement. La deuxième partie de cette équation indique que tous les conteneurs sont transportés sans préemption par les camions de transport. Enfin la troisième partie de lb_2 découle de l'observation que le deuxième camion ne travaille que lorsque le deuxième conteneur arrive et ainsi de suite pour tous les camions jusqu'au conteneur numéro $|T|$. Le minimum est alors la somme de tous les temps nécessaires pour les deuxième, troisième et autres conteneurs qui seront prêts pour le transport et affectés aux camions. Les deux premiers termes correspondent à la borne inférieure du moment où le dernier conteneur est terminé et où le camion retourne à la grue de quai. Dans le dernier terme, le temps de retour du camion est soustrait pour trouver le temps de parcours minimum du conteneur afin de calculer la borne inférieure du *makespan*.

Enfin, l'équation (4.23) est la borne inférieure déduite des deux valeurs lb_1 et lb_2.

$$lb = \max(lb_1, lb_2) \tag{4.23}$$

4.2.1.2/ BORNE SUPÉRIEURE

L'équation (4.24) nous donne une solution réalisable de limite supérieure pour le problème d'ordonnancement.

$$ub = \sum_{i=1}^{|C|}(w_i + r) + \frac{(2|C|-1) \times \max\limits_{i=1..|C|} \lambda i}{|T|} + [|C| \times \max_{j=0..|C|}^{i=1..C} t_{ji}] \tag{4.24}$$

L'équation (4.24) vise à nous donner la date maximale de fin de manutention de tous les conteneurs du navire porte-conteneurs vers la zone de stockage, en calculant tous les temps de traitement pour tous les conteneurs et le temps de transport total avec la somme du temps de repositionnement maximal de la grue de quai entre les conteneurs.

Exemple numérique pour l'équation (4.24) :

Supposons que nous avons 1 conteneur et 1 camion de transport, par suite la date d'achèvement est $C_{max} = t_{01} + w_1 + \lambda_1 + r$, avec un seul repositionnement. Appliquons l'équation (4.24) : $ub = w_1 + r + ((2-1) \times \lambda_1)/1 + t_{01}$, alors $C_{max} = ub$, donc nous avons bien $C_{max} \leq ub$.

Supposons maintenant que nous avons 2 conteneurs et 1 camion de transport. La date de fin de manutention est alors : $C_{max} = t_{01} + w_1 + max(t_{12} + w_2 \,;\, \lambda_1 + r + \lambda_1^{'}) + \lambda_2 + r$, avec deux repositionnements.

Appliquons l'équation (4.24) : $ub = w_1 + r + w_2 + r + ((4-1) \times \lambda)/1 + 2t$, avec $t = \max\limits_{j=0..|C|}^{i=1..C} t_{ji}$ et $\lambda = \max\limits_{i=1..|C|} \lambda i$. Donc $ub = w_1 + w_2 + 2r + 3\lambda + 2t$.

Si la grue de quai est la ressource critique pour le transport du second conteneur, alors $C_{max} = t_{01} + w_1 + t_{12} + w_2 + \lambda_2 + r$. Donc, comme $\forall i, j \in \{1...|C|\}$, $\lambda \geq \lambda_i$ et $t \geq t_{ji}$, alors $C_{max} \leq 2t + w_1 + w_2 + \lambda + r \leq ub$. ub est donc bien une borne supérieure ici.

Sinon, le camion est la ressource arrivant au point de rendez-vous le plus tard, et donc $C_{max} = t_{01} + w_1 + \lambda_1 + r + \lambda_1^{'} + \lambda_2 + r$. Par conséquent $C_{max} \leq t + w_1 + 2r + 3\lambda$. Comme $w_2 + t \geq 0$, alors $C_{max} \leq ub$. ub est donc bien ici une borne supérieure du *makespan* dans tous les cas.

4.2.2/ ALGORITHME NUMÉRATIF

Nous proposons dans cette section un algorithme numératif, que nous appelons AN2, pour résoudre notre problème. L'organigramme de la figure 4.3 et l'algorithme 3 associé fournissent une description du fonctionnement de notre algorithme numératif à partir de la génération des paramètres, terminant par l'obtention de la solution optimale.

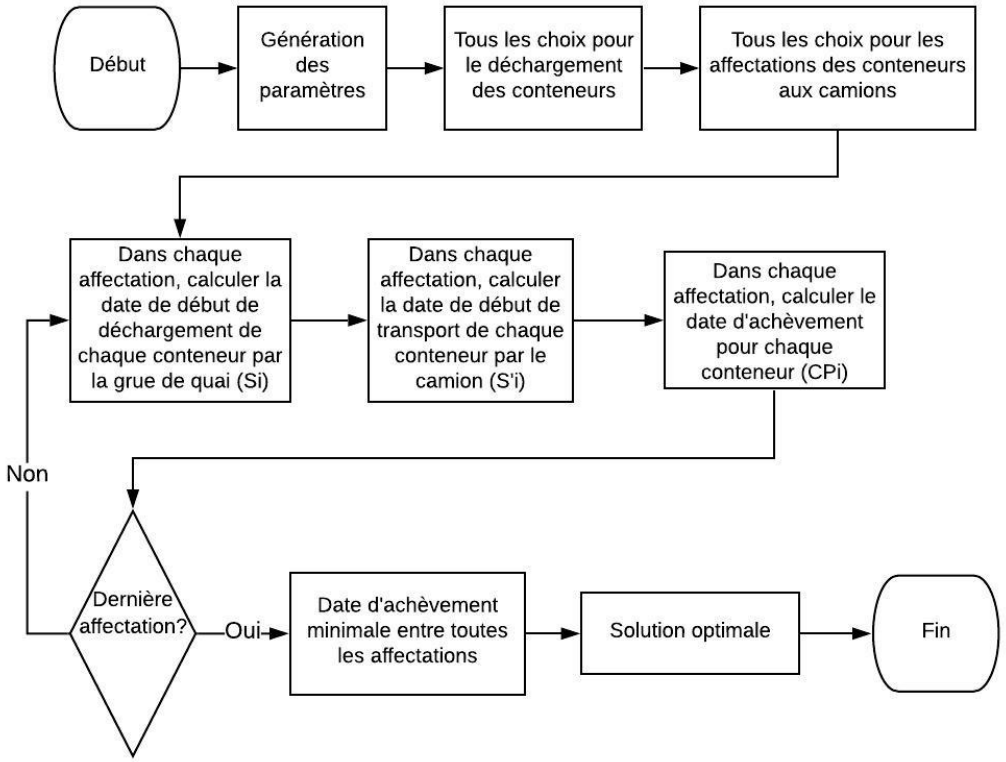

FIGURE 4.3 – Organigramme pour QCYTSP

Algorithm 3 Algorithme numératif AN2

```
1:  générer_tous_variables() ;
2:  générer_ordre_conteneurs() ;
3:  générer_paires_conteneurs() ;
4:  for chaque affectation do
5:     date_début_QC() ;
6:     date_début transport_YT() ;
7:     for chaque conteneur do
8:        calculer_date_d'achèvement() ;
9:     end for
10:    calculer_date_d'achèvement_totale() ;
11:    if dernière_affectation = vrai then
12:       date_d'achèvement_minimale() ;
13:       Fin ;
14:    else
15:       prochaine_affectation() ;
16:    end if
17: end for
```

Explications :

— Ligne 1 : nous générons d'abord l'ensemble des paramètres tels que le nombre de conteneurs et de camions de transport, le temps nécessaire pour décharger

le conteneur par la grue de quai, le temps de repositionnement pour la grue de quai pour décharger le conteneur après un précédent, le temps nécessaire au camion pour transporter le conteneur jusqu'à la zone de stockage, et enfin le temps nécessaire au chariot frontal pour décharger un conteneur du camion de transport.

— Ligne 2 : tous les choix possibles d'ordre de déchargement par la grue des conteneurs sont générés. Par exemple si nous avons 4 conteneurs et si le premier choix de séquence de déchargement est 1324, cela signifie que le conteneur 1 est le premier à être déchargé, puis le conteneur 3, puis le 2 et enfin le conteneur 4.
— Ligne 3 : tous les choix possibles sont générés pour l'affectation et l'ordonnancement du transport de paires de conteneurs par les camions. Par exemple si le premier choix est 321, les deux premiers chiffres désignent des indices de conteneurs et le troisième est l'indice d'un camion. Ainsi dans ce cas, les conteneurs 2 et 3 sont affectés au camion 1 qui les transporte dans l'ordre 3-2.
— Ligne 4 à ligne 17 : pour chaque affectation, nous calculons pour la grue, la date de début de déchargement de chaque conteneur (ligne 5), puis nous calculons la date de début de transport par camion de chaque conteneur (ligne 6) ; nous déterminons pour chaque conteneur la date de fin de manutention (lignes 7, 8 et 9) et pour chaque affectation, la date de fin de manutention de tous les conteneurs (ligne 10). Une fois toutes les affectations évaluées, nous en déduisons la solution optimale, celle de plus petite date d'achèvement (lignes 11 à 13), sinon (ligne 15) la procédure est répétée à partir de la ligne 4 pour l'affectation suivante.

4.2.3/ EXEMPLES NUMÉRIQUES

Nous présentons ici deux exemples d'ordonnancement, le premier est pour 3 conteneurs et 1 camion de transport YT (Figure 4.4), et le second est pour 3 conteneurs et 2 camions YT1 et YT2 (Figure 4.5). Les temps de déchargement w_i des conteneurs par la grue de quai sont w_1=w_2=3 et w_3=4. Les temps de repositionnement de la grue entre les déchargements des conteneurs 1 à 2, et 2 à 3, sont t_{12}=t_{23}=3. Le temps de transport λ_i par un camion depuis le quai jusqu'à la zone de stockage égale 4 pour tous les conteneurs, et il est le même pour le retour à vide du camion noté λ'. Le temps de déchargement du camion par le chariot frontal (r) est de 2 pour chaque conteneur. Enfin, nous supposons que l'ordre de déchargement des conteneurs par la grue de quai est 123.

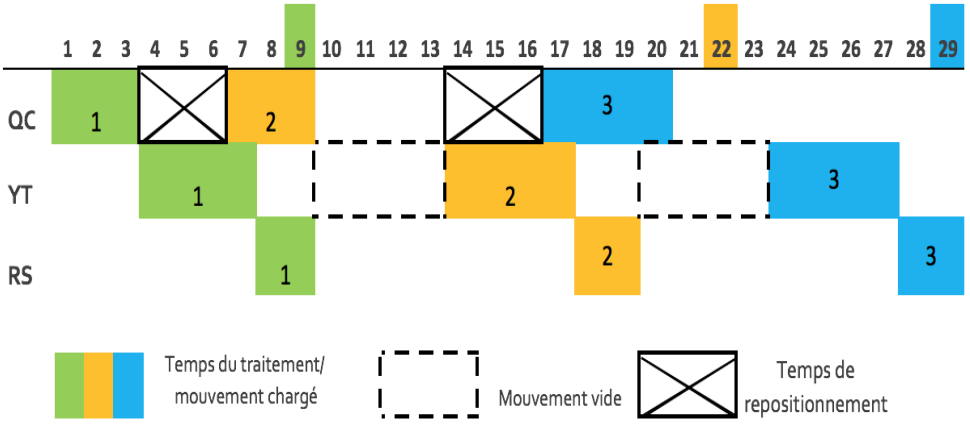

FIGURE 4.4 – Exemple avec un camion

Pour l'exemple à un camion, l'évaluation de l'ordonnancement illustré par la Figure 4.4 donne le résultat suivant :

* Conteneur 1 : durée de manutention $dm_1 = w_1 + \lambda_1 + r = 9$
* Conteneur 2 : durée de manutention $dm_2 = w_2 + \lambda_2 + r = 9$
* Conteneur 3 : durée de manutention $dm_3 = w_3 + \lambda_3 + r = 10$
* Temps total de manutention de tous les conteneurs :

$C_{max} = w_1 + max(t_{12} + w_2 \,;\, \lambda_1 + r + \lambda_1^{'}) + max(t_{23} + w_3 \,;\, \lambda_2 + r + \lambda_2^{'}) + \lambda_3 + r = 29$

Dans la figure, nous pouvons voir que le temps de repositionnement de la grue n'impacte pas sur l'évaluation globale, dans la mesure où elle est inférieure aux temps de transport du camion. Par ailleurs, en pratique sur les quais, des contraintes de sécurité plus strictes sont parfois appliquées liées à la sécurité et interdisant qu'un conteneur reste en suspens au-dessus du quai sans camion pour le réceptionner. Dans ce cas, une grue ne commencera un nouveau déchargement que si un camion est déjà présent. Bien sûr cela implique des délais supplémentaires. Cette contrainte dite de sécurité n'est pas prise en compte dans nos modèles.

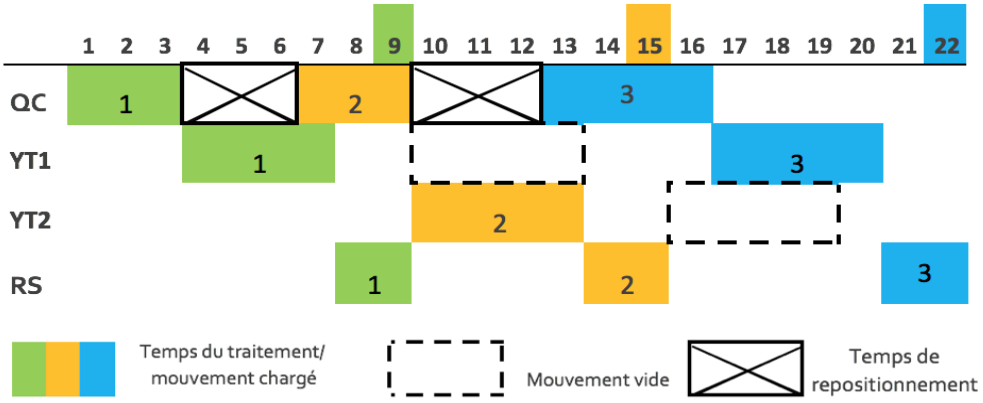

FIGURE 4.5 – Exemple avec deux camions

Dans le second exemple illustré par la Figure 4.5, le camion YT1 transporte les conteneurs 1 et 3, tandis que YT2 s'occupe du conteneur 2. L'évaluation de la solution présentée donne comme résultat :

* Manutention du conteneur 1 :
  - durée $dm_1 = w_1 + \lambda_1 + r = 9$
  - date de fin $cp_1 = s_1 + dm_1 = 0 + 6 = 9$
* Manutention du conteneur 2 :
  - durée $dm_2 = w_2 + \lambda_2 + r = 9$
  - date de fin $cp_2 = s_2 + dm_2 = 6 + 9 = 15$
* Manutention du conteneur 3 :
  - durée $dm_3 = w_3 + \lambda_3 + r = 10$
  - date de fin $cp_3 = s_3 + dm_3$, avec $s_3 = s_2 + w_2 + t_{23} = 6+3+3 = 12$, donc $cp_3 = 12+10 = 22$
* Temps total de manutention (*makespan*) $C_{max} = Max(cp_1, cp_2, cp_3) = 22$

4.2.4/ ALGORITHME GÉNÉTIQUE

Comme au chapitre précédent, nous avons choisi de développer un algorithme génétique comme méthode approchée à comparer à notre méthode exacte présentée précédemment. Le principe et les principaux éléments ont été repris de la méthode développée au chapitre 3 (algorithme 4).

Algorithm 4 Algorithme génétique GA2

```
Population ;
foreach chromosome do
    Évaluation() ;
    if dernière génération then
        Solution ;
    end
    else
        selection() ;
        croisement() ;
        mutation() ;
    end
end
```

La population initiale de l'algorithme génétique est générée aléatoirement. Le nombre de générations est défini a priori. L'évaluation des solutions consiste à affecter/ordonnancer les opérations des camions et déterminer la valeur de la fonction objectif. Les opérateurs de l'algorithme génétique (sélection, croisement, mutation) seront appliqués de génération en génération. La solution optimale est la meilleure obtenue sur la dernière génération.

Dans notre métaheuristique, un chromosome représente une séquence de déchargement des conteneurs par la grue de quai. Chaque gène correspond à l'indice d'un conteneur (tableau 4.1). Le chromosome est décodé par la base de transmission du camion de transport qui attribue la tâche de transfert au premier camion de transport disponible.

TABLE 4.1 – Représentation du chromosome

5	1	4	2	3

Pour initialiser la population de solutions et former la première génération, certaines solutions réalisables sont générées aléatoirement. Supposons que S soit la taille de la population. Par conséquent, S chromosomes sont générés et chaque chromosome est une chaîne décrivant une séquence de déchargement de la grue de quai.

Dans cette séquence, un conteneur ne peut pas être déchargé tant que tous les conteneurs qui le précèdent ne sont pas déchargés ; en d'autres termes, la grue de quai ne peut pas commencer la tâche de déchargement d'un conteneur spécifique tant que tous les conteneurs de son prédécesseur dans le chromosome n'ont pas été déchargés du navire. Chaque conteneur nécessite un mouvement de grue qui le décharge du navire et le dépose sur un camion. Celui-ci doit donc être disponible en position de réception à l'arrivée de la grue. Les $|T|$ premiers conteneurs ($|T|$ étant le nombre de camions) sont

chacun affectés à un camion différent. Nous affectons ensuite le conteneur suivant au premier camion disponible. Autrement dit, si un camion est occupé par un autre transport, sa date de disponibilité pour la prise en charge du prochain conteneur sera le moment où il termine son transport en cours, auquel il faut ajouter le temps de trajet à vide de la destination du conteneur déjà chargé à la position du prochain conteneur ; si un camion est libre, sa prochaine date de disponibilité sera simplement le temps de trajet entre son emplacement actuel et l'emplacement du prochain conteneur. Sur la base de ces horaires, nous sélectionnons ensuite le camion qui peut arriver au plus tôt à l'emplacement de réception du conteneur suivant.

A chaque génération, les individus d'une population sont également évalués en terme de fitness déterminée pour chaque chromosome. La fitness est représentée dans l'équation 4.25, qui est à nouveau l'inverse du temps total de manutention des conteneurs.

$$Fitness = 1/C_{max} \tag{4.25}$$

Les opérateurs de sélection, croisement et mutation sont les mêmes que pour le problème de base étudié QCSP, à savoir une sélection par roulette, où les étapes sont les suivantes :

1. Calculez la somme de tous les *fitness* et appelez-la S_1.
2. Générez un nombre au hasard entre 0 et S_1 et appelez-le *R*.
3. Ajoutez tous les *fitness* du haut de la population à une somme appelée S_2, avec une condition que $S_2 < S_1$.
4. Nous choisissons l'individu pour lequel S_2 dépasse *R*.

Un croisement en deux points est ensuite réalisé pour chaque couple de parents sélectionnés d'une manière aléatoire. Enfin une procédure de mutation est appliquée aux enfants résultant du croisement, consistant à interchanger aléatoirement la position de 2 gènes d'un individu. Pour créer la population à l'itération suivante, nous appliquons une stratégie de remplacement qui consiste à conserver les meilleurs individus, pour constituer 25% de la nouvelle population. Celle-ci est complétée pour 45% par le résultat des croisements et pour 30% par le résultat des mutations.

4.3/ RÉSULTATS EXPÉRIMENTAUX

4.3.1/ IMPLÉMENTATION ET INSTANCES

Notre programme linéaire en nombres mixtes est résolu par CPLEX 12.8 (MILP2) tandis que l'algorithme numératif (AN2), les bornes inférieures (LB), les bornes supérieures (UB) et l'algorithme génétique (GA2) sont implémentés en Java. Tous les programmes sont exécutés sur un Macbook professionnel avec 8 Go de RAM et Intel Core i5.

Nous avons testé nos méthodes sur 3 jeux d'instances totalisant 43 configurations :

— Le premier jeu est constitué de 20 instances générées aléatoirement, et nommées B1 à B20. Elles comprennent 1, 2 ou 3 camions et entre 5 et 15 conteneurs. Les autres paramètres nécessaires sont générés aléatoirement : w_i est généré aléatoirement entre 250 et 300 unités de temps, t_{ij} entre 20 et 30 unités de temps, λ_i entre 150 et 200 fois unités et r entre 30 et 50 unités de temps.

— Le second jeu correspond à un cas réel lié au port de Tripoli-Liban, à partir duquel nous avons pu générer 6 nouvelles instances, identifiées par TL7 à TL12, comportant entre 1 et 3 camions et 5 à 8 conteneurs à décharger au total.
— Le troisième jeu est issu de la littérature ([Zhen et al., 2016]) et comprend 17 instances appelées Z1 à Z17, avec 1 à 3 camions et 5 à 11 conteneurs.

4.3.2/ ALGORITHME NUMÉRATIF (AN2) VERSUS MILP

Nous avons tout d'abord appliqué nos méthodes exactes sur le premier jeu d'instances. Comme le montre le tableau 4.2, les résultats CPLEX et les valeurs du *makespan* issues de AN2 sont identiques pour les instances B1 à B15. Mais AN2 est meilleur en termes de temps d'exécution (un Ecart négatif dans le tableau traduit une amélioration du temps CPU), avec des gains en moyenne de 69.86% et variant entre 4.76% et 97.29%, pour les 12 instances B4 à B15. Ces gains dépassent 65% dans 10 de ces cas. Pour les instances B1à B3, les temps de résolution étant inférieurs à la seconde, nous n'avons pas calculé les écarts qui ne nous semblent pas significatifs. Pour les plus grandes instances du tableau (B16 à B19), CPLEX ne parvient pas à résoudre notre MILP2 après 9 heures d'exécution. De même, pour l'instance B19, nous avons attendu 5 heures avec AN2 sans obtenir de résultat. Ici, AN2 révèle également ses limites.

TABLE 4.2 – MILP2 vs AN2

No.	$\|C\|$	$\|T\|$	Makespan		Temps d'exécution		
			CPLEX (u.t)	AN2 (u.t)	CPLEX (s)	AN2 (s)	Ecart (%)
B1	5	1	3069	3069	< 1	< 1	-
B2	5	2	2294	2294	< 1	< 1	-
B3	5	3	1815	1815	< 1	< 1	-
B4	6	1	3924	3924	1.05	< 1	>-4.76
B5	6	2	2902	2902	1.32	< 1	>-24.24
B6	6	3	1967	1967	5.47	1.34	-75.5
B7	7	1	6992	6992	6.44	1.53	-76.24
B8	7	2	3469	3469	22.75	1.62	-92.88
B9	7	3	2432	2432	39.18	4.41	-88.74
B10	8	1	8634	8634	42.57	5.61	-86.82
B11	8	2	4305	4305	242.63	6.57	-97.29
B12	8	3	3262	3262	102.37	29.32	-71.36
B13	9	1	11199	11199	2678.94	531.4	-80.16
B14	9	2	6512	6512	3984.12	997.34	-74.97
B15	9	3	5060	5060	5816.85	2013.81	-65.38
B16	10	3	N.A.	9335	N.A	2581.95	-
B17	10	4	N.A.	9123	N.A	3797.16	-
B18	10	5	N.A.	6647	N.A	6914.45	-
B19	15	3	N.A.	N.A	N.A	N.A	-

$|C|$: nbre de conteneurs // $|T|$: nbre de camions // N.A. : pas de résultat dans le délai imparti
Pour le temps d'exécution, Ecart = $((temps_{AN2} - temps_{CPLEX})/temps_{CPLEX}) * 100$

Ces tests ont permis de valider notre modèle, et de mettre en évidence l'efficacité de AN2

qui résout à l'optimalité plus d'instances que MILP2 et plus rapidement. Donc dans les sections suivantes, nous comparons les performances de notre méthode métaheuristique avec celles du AN2.

4.3.3/ BORNES INFÉRIEURE ET SUPÉRIEURE

Dans le tableau 4.3, nous comparons le *makespan* donné par AN2 avec les bornes inférieure et supérieure proposées. Nous remarquons que le temps total de manutention obtenu par AN2 est toujours compris entre les valeurs calculées pour ces limites inférieure et supérieure.

Dans tous les cas résolus, l'écart de la solution optimale avec la borne inférieure est inférieur à 8.13% et est en moyenne de 5.65%. Cet écart est en moyenne de 12.27% par rapport à la borne supérieure.

TABLE 4.3 – Comparaison entre LB, UB et AN2

No.	$\|C\|$	$\|T\|$	Makespan			Ecart	
			LB (u.t)	AN2 (u.t)	UB (u.t)	(LB/AN2) (%)	(UB/AN2) (%)
B1	5	1	2868	3069	3166	-6.55	3.16
B2	5	2	2122	2294	2398	-7.5	4.53
B3	5	3	1718	1815	2204	-5.34	21.43
B4	6	1	3634	3924	4248	-7.39	8.26
B5	6	2	2666	2902	3240	-8.13	11.65
B6	6	3	1809	1967	2424	-8.03	23.23
B7	7	1	6798	6992	8065	-2.77	15.35
B8	7	2	3257	3469	3573	-6.11	2.99
B9	7	3	2291	2432	2555	-5.8	5.06
B10	8	1	8365	8634	9022	-3.12	4.49
B11	8	2	4176	4305	4459	-2.99	3.58
B12	8	3	3121	3262	4192	-4.32	28.51
B13	9	1	10305	11199	13509	-7.98	20.63
B14	9	2	6158	6512	6647	-5.43	2.07
B15	9	3	4892	5060	6514	-3.32	28.74
B16	10	3	9082	9335	10141	-2.71	8.63
B17	10	4	8419	9123	9464	-7.71	3.74
B18	10	5	6212	6647	8296	-6.54	24.8
B19	15	3	12223	N.A	15157	-	-
B20	15	4	10747	N.A	13861	-	-

$|C|$: nbre de conteneurs // $|T|$: nbre de camions // N.A. : pas de résultat dans le délai imparti
Ecart (LB/AN2) = $((temps_{LB} - temps_{AN2})/temps_{AN2}) * 100$
Ecart (UB/AN2) = $((temps_{UB} - temps_{AN2})/temps_{AN2}) * 100$

4.3.4/ ALGORITHME GÉNÉTIQUE VERSUS ALGORITHME NUMÉRATIF

Le tableau 4.4 compare AN2 et GA2 sur les instances de notre jeu de test aléatoire. Il fournit le *makespan* obtenu et le temps de résolution associé pour chaque méthode, ces

deux facteurs étant comparés en termes d'écart relatif (Ecart1 et Ecart2) par instance.

Dans ce tableau, nous pouvons observer que GA2 nous donne des solutions proches des solutions optimales obtenues avec AN2, avec un écart variant entre 0% et 2.86%. Si nous raisonnons sur l'ensemble des 18 premières instances, B1 à B18, l'écart moyen entre les deux méthodes est seulement de 1.92%, ce qui montre la qualité des solutions obtenues avec notre algorithme génétique.

TABLE 4.4 – GA2 vs AN2

No.	$\|C\|$	$\|T\|$	Makespan			Temps d'exécution		
			AN2 (u.t)	GA2 (u.t)	Ecart1 (%)	AN2 (s)	GA2 (s)	Ecart2 (%)
B1	5	1	3069	3136	2.18	< 1	< 1	-
B2	5	2	2294	2359	2.83	< 1	< 1	-
B3	5	3	1815	1815	0	< 1	< 1	-
B4	6	1	3924	4023	2.52	< 1	< 1	-
B5	6	2	2902	2963	2.10	< 1	< 1	-
B6	6	3	1967	2011	2.24	1.34	< 1	>-25.37
B7	7	1	6992	7166	2.49	1.53	< 1	>-34.64
B8	7	2	3469	3544	2.16	1.62	1.03	-36.42
B9	7	3	2432	2463	1.27	4.41	3.12	-29.25
B10	8	1	8634	8839	2.37	5.61	2.56	-54.37
B11	8	2	4305	4402	2.25	6.57	3.97	-39.57
B12	8	3	3262	3328	2.02	29.32	7.19	-75.48
B13	9	1	11199	11199	0	531.4	13.43	–97.47
B14	9	2	6512	6642	1.99	997.34	21.47	-97.85
B15	9	3	5060	5164	2.06	2013.81	27.38	-98.64
B16	10	3	9335	9602	2.86	2581.95	34.22	-98.67
B17	10	4	9123	9236	1.24	3797.16	43.69	-98.85
B18	10	5	6647	6779	1.99	6914.45	59.77	-99.14
B19	15	3	N.A	12926	-	N.A	102.35	-
B20	15	4	N.A	11471	-	N.A	119.83	-

$|C|$: nbre de conteneurs // $|T|$: nbre de camions // N.A. : pas de résultat dans le délai imparti
Ecart1 et Ecart2 calculés sur la base : $((valeur_{GA} - valeur_{AN2})/valeur_{AN2}) * 100$

À partir de l'instance B19, AN2 n'a pu fournir aucune solution après plus de 5 heures d'exécution de l'algorithme, tandis que GA2 fournit des solutions dans un temps très réduit. Ce temps est de l'ordre de 2 minutes pour ces instances à 15 conteneurs, mais nous voyons qu'il est plus de 2 fois supérieur au temps CPU des instances à 10 conteneurs qui reste inférieur à la minute.

Comme AN2 n'a pu résoudre les instances B19 et B20 dans le temps imparti, nous ne pouvons évaluer la qualité du *makespan* donné par GA2 par rapport à l'optimum. Mais en nous basant sur les tableaux 4.3 et 4.4, nous déterminons que l'écart de ce *makespan* aux bornes inférieure et supérieure est respectivement de 5.44% et 17.26% pour B19, et respectivement de 6.31% et 20.84% pour B20. Donc ces écarts à la borne inférieure pour B19 et B20 restent du même ordre de grandeur que l'écart moyen obtenu avec AN2 pour les autres instances. Ces résultats montrent l'intérêt et l'efficacité de notre algorithme génétique GA2.

4.3.5/ COMPARAISON SUR DES INSTANCES RÉELLES

Nous avons comparé les performances de nos deux méthodes, exacte AN2 et approchée GA2, cette fois sur des instances issues du port de Tripoli au Liban. Nous avons exécuté nos algorithmes sur les 6 instances du second jeu de test dont nous retrouvons les caractéristiques dans les premières colonnes du tableau 4.5. Nous avons utilisé les mêmes paramètres utilisés dans le port pour faire une comparaison réaliste des résultats.

TABLE 4.5 – (AN2,GA2) vs port

No.	$\|C\|$	$\|T\|$	Makespan			Ecart	
			AN2 (s)	GA2 (s)	Port (s)	(AN2/port) (%)	(AN2/port) (%)
TL7	5	2	472	503	590	-20	-14.75
TL8	6	2	574	613	737	-22.12	-16.82
TL9	6	3	422	422	518	-18.53	-18.53
TL10	8	1	1051	1131	1360	-22.72	-16.84
TL11	8	2	913	989	1154	-20.88	-14.3
TL12	8	3	723	781	901	-19.76	-13.32

$$Ecart(AN2/port) = ((makespan_{AN2} - makespan_{port})/makespan_{port}) * 100$$
$$Ecart(GA2/port) = ((makespan_{GA2} - makespan_{port})/makespan_{port}) * 100$$

Dans le tableau 4.5, les solutions appliquées par le port sont comparées à celles des algorithmes numératif (AN2) et génétique (GA2). Les deux dernières colonnes fournissent pour chaque instance les écarts relatifs entre le *makespan* du port et celui de AN2 et GA2.

Par exemple pour l'instance TL8 pour 6 conteneurs et 2 camions de transport, le *makespan* du port est 737 secondes tandis que le *makespan* du AN2 est de 574 secondes, donc AN2 nous permet de gagner du temps pour 22,12 %. Pour le même exemple, *makespan* du GA2 est également de 613 et permet de gagner du temps pour 16.82%.

Plus généralement, pour les instances considérées, nos deux méthodes améliorent de manière significative les résultats du port de Tripoli, en moyenne de 20.67% pour AN2 et de 15.76% pour GA2.

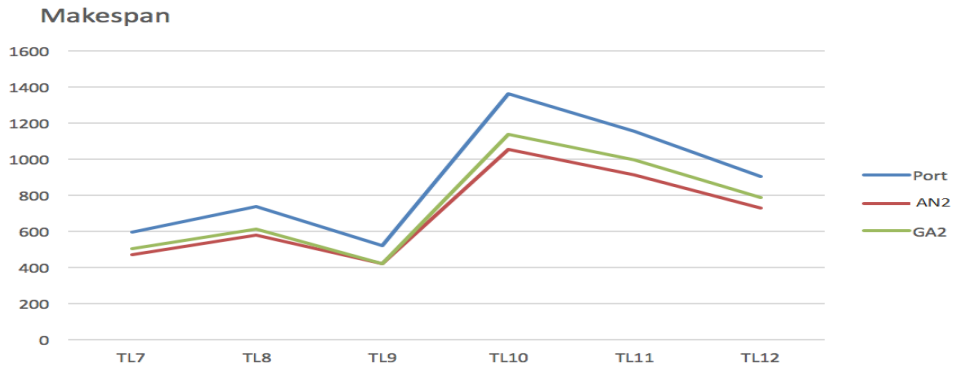

FIGURE 4.6 – *Makespan* par méthode et par instance

Le graphe de la figure 4.6 illustre également les résultats présentés dans le tableau 4.5.

Les barres bleues correspondent aux résultats du port de Tripoli, tandis que les rouges et les vertes concernent respectivement les résultats de AN2 et de GA2. Nous pouvons observer ici que le *makespan* du port est bien supérieur à celui issu de nos deux méthodes et que la courbe de GA2 est proche de celle de l'optimum AN2.

4.3.6/ MÉTHODE EXACTE VERSUS LITTÉRATURE

[Zhen et al., 2016] ont étudié le problème d'optimisation intégré sur l'ordonnancement de grues de quai et de camions de transport dans les terminaux à conteneurs. Un modèle de programmation linéaire en nombres entiers est formulé. Pour ce modèle, [Zhen et al., 2016] ont montré que le problème d'ordonnancement intégré est fortement NP-difficile. Pour résoudre le modèle proposé dans un temps raisonnable, une solution basée sur l'optimisation par essaims particulaires (PSO) a été développée. Des expériences numériques ont été menées pour comparer la méthode proposée avec le solveur CPLEX. Leurs résultats valident l'efficacité du modèle proposé et de leur méthode de résolution.

Le tableau 4.6 fournit un exemple avec deux grues de quai et quatre camions déchargeant six conteneurs pour illustrer le problème d'ordonnancement traité par [Zhen et al., 2016]. La première ligne du tableau est associée aux indices des conteneurs, la deuxième ligne fournit les temps de déchargement de chaque conteneur par une grue de quai, la troisième ligne donne les temps de transport pour chaque conteneur par un camion, et enfin les deux dernières lignes indiquent les résultats de [Zhen et al., 2016] en termes d'affectation et d'ordre de déchargement des conteneurs par chaque grue de quai.

TABLE 4.6 – Exemple issu de [Zhen et al., 2016]

Conteneur	**1**	**2**	**3**	**4**	**5**	**6**
temps QC	35	26	44	24	65	70
temps YT	50	40	30	40	50	45
Ordre des conteneurs pour la 1ère grue : 2→1→4→3						
Ordre des conteneurs pour la 2ème grue : 5→6						

Sur la figure 4.7, nous observons que dans la solution optimale fournie par le modèle linéaire de [Zhen et al., 2016], les conteneurs 1, 2, 3, 4 sont affectés à la grue de quai 1 (QC1) avec la séquence de manipulation $2 \rightarrow 1 \rightarrow 4 \rightarrow 3$; Les conteneurs 5 et 6 sont affectés à la grue de quai QC2, avec la séquence de manipulation $5 \rightarrow 6$. Nous pouvons également voir l'ordonnancement des camions suivant l'axe des ordonnées. Par exemple, à la date 100, 3 camions sont occupés sur 4, ce qui est donc inférieur au nombre de camions disponibles. Si nous répartissions uniformément la charge de travail (c.-à-d. en affectant les 3 premières tâches et 2 camions YT à QC1, les 3 tâches restantes et 2 YT à QC2), le temps de traitement total serait de 206, ce qui est plus long que le résultat optimal de 181 unités de temps.

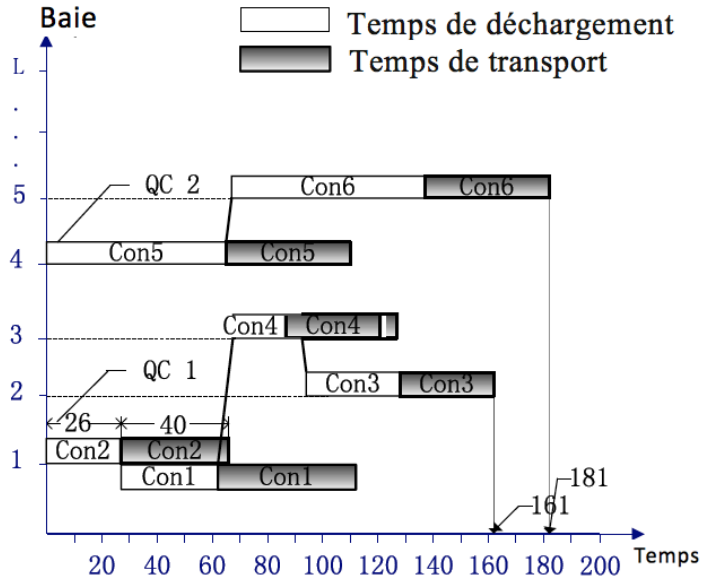

FIGURE 4.7 – Illustration de l'ordonnancement d'un exemple issu de [Zhen et al., 2016]

Afin de situer notre contribution dans la littérature, les tableaux 4.7 et 4.8 résument les similitudes et les différences entre notre étude et celle de [Zhen et al., 2016]. Les figures 4.8 et 4.9 illustrent aussi en partie ces différences.
En résumé, notre modèle adresse l'ordonnancement d'une seule grue de quai avec plusieurs camions de transport avec intégration des chariots frontaux dans le modèle, tandis que [Zhen et al., 2016] ont proposé un modèle pour plusieurs grues de quai et plusieurs camions de transport mais sans considération des chariots frontaux.

TABLE 4.7 – Comparaison avec [Zhen et al., 2016]

Paramètres	**[Zhen et al., 2016]**	**Notre modèle**
Nombre de grues de quai	plusieurs	1 seule
Nombre de camions de transport	plusieurs	plusieurs
Nombre de conteneurs	x	x
Nombre de baies	x	[a]
Temps de repositionnement des grues	x	x
Temps de déchargement	x	x
Temps de transport des camions	x	x
Priorité des conteneurs	x	x
Marge de sécurité	x	[b]
Grues de zone (chariots frontaux...)	[c]	x
Objectif : minimisation du *makespan*	x	x

[a] : nous avons utilisé une seule grue de quai, donc nous n'avons pas besoin de fixer le nombre des baies et nous avons utilisé les positions à la place des baies.
[b] : Pas de marge de sécurité car nous considérons une seule grue de quai.
[c] : [Zhen et al., 2016] ont considéré un temps de transport total pour les camions sans considération des ressources dans la zone de stockage.

TABLE 4.8 – Similitudes et différences avec [Zhen et al., 2016]

Référence	Problème	Contraintes					Méthodes de résolution
		Config.		Position	Camions	Grue de zone	
		QC	YT				
[Zhen et al., 2016]	QCYTSP	✓	✓	✓	✓		MILP et PSO
notre étude	QCYTSP	✓	✓	✓	✓	RS	MILP, AN, LB, UB et GA

MILP : Mixed-integer linear programming / GA : Genetic algorithm / PSO : Particle swarm optimization / AN : Enumerative algorithm / LB, UB : Lower and Upper bounds
Pour l'explication des contraintes, se référer au tableau 2.3

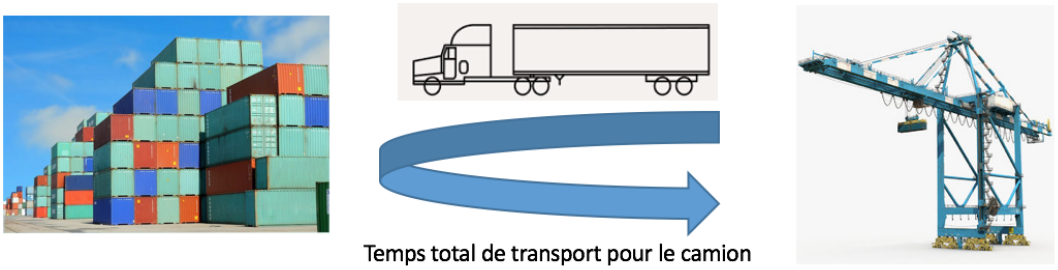

FIGURE 4.8 – Le cas de [Zhen et al., 2016]

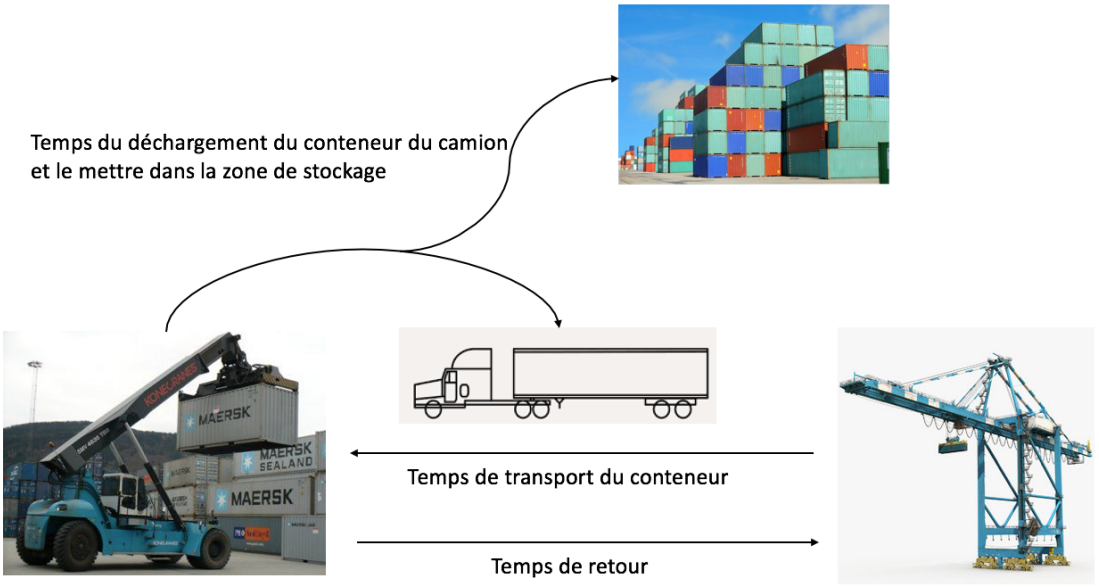

FIGURE 4.9 – Notre configuration

Malgré nos différences, nous pouvons comparer nos résultats à ceux de [Zhen et al., 2016]. Pour cela, il suffit de :

- programmer le modèle du [Zhen et al., 2016] et le paramétrer pour une seule grue ;
- supposer que le temps total du transport d'un camion, chargé d'un conteneur, de la zone des grues de quai vers la zone de stockage, avec le temps de retour (figure 4.8, est égal au trois temps de notre modèle qui sont : le temps de transport du conteneur entre ces deux zones, le temps du déchargement de conteneur du camion, et le temps du retour du camion.

Le Tableau 4.9 montre que les résultats fournis par notre meilleure méthode exacte (AN2) et les résultats fournis par le modèle linéaire de [Zhen et al., 2016] sont les mêmes pour les instances testées. Pour les instances Z17, comportant 11 conteneurs et 3 camions, [Zhen et al., 2016] n'obtient pas de résultat après 3 heures d'exécution. En revanche, notre algorithme numératif AN2 permet de résoudre les 17 instances à l'optimalité. Les temps d'exécution sont également largement à l'avantage de notre méthode exacte puisque les gains (Ecarts) observés sont compris entre 58.48% et 97.51% et sont en moyenne de 80.59%.

TABLE 4.9 – Comparaison des résultats avec [Zhen et al., 2016]

No.	$\|C\|$	$\|T\|$	Makespan		Temps d'exécution		
			AN2 (u.t)	[Zhen et al., 2016] (u.t)	AN2 (s)	[Zhen et al., 2016] (s)	Écart (%)
Z1	5	1	3069	3069	0.12	1.28	-90.63
Z2	5	2	2294	2294	0.24	1.54	-84.42
Z3	5	3	1815	1815	0.22	1.77	-87.57
Z4	6	1	3924	3924	0.31	1.28	-75.78
Z5	6	2	2902	2902	0.44	2.93	-84.98
Z6	6	3	1967	1967	1.34	3.21	-58.26
Z7	7	1	6992	6992	1.53	8.54	-82.08
Z8	7	2	3469	3469	1.62	30.32	-94.66
Z9	7	3	2432	2432	4.41	44.78	-90.15
Z10	8	1	8634	8634	5.61	53.47	-89.51
Z11	8	2	4305	4305	6.57	263.67	-97.51
Z12	8	3	3262	3262	29.32	121.22	-75.81
Z13	9	1	11199	11199	531.4	3012.32	-82.36
Z14	9	2	6512	6512	997.34	4081.12	-75.56
Z15	9	3	5060	5060	2013.81	5924.13	-66.01
Z16	10	3	9335	9335	2581.95	6217.97	-58.48
Z17	11	3	11741	N.A	5819.31	-	-

N.A. : pas de résultat dans le délai imparti

$Ecart(AN2/Zhen) = ((temps_{AN2} - temps_{Zhen}/temps_{Zhen}) * 100$

4.4/ CONCLUSION

Dans cette étude, nous avons proposé plusieurs contributions pour résoudre la variante étudiée de *Quay Crane and Yard Truck Scheduling Problem* (QCYTSP) : un modèle mathématique sous forme de MILP, des bornes inférieures et supérieures pour le *ma-*

kespan, un algorithme numératif (AN) et un algorithme génétique (GA). Les solutions optimales obtenues avec AN se situent toujours entre les bornes inférieure et supérieure, et assez proches de la borne inférieure. Les solutions du GA sont également proches des solutions optimales. Les tests réalisés sur des instances générées aléatoirement et des instances de la littérature nous ont ainsi permis de valider notre modèle et de vérifier l'efficacité de nos algorithmes. Nos méthodes peuvent donc en particulier être appliquées efficacement dans un port tel que celui de Tripoli, en améliorant les ordonnancements qui y sont mis en oeuvre.

La suite de l'étude menée dans ce chapitre sera d'étendre notre modèle à des variantes de QCYTSP plus complexes, en considérant par exemple plusieurs grues de quai, et/ou en considérant un nombre limité de chariots frontaux. Dans le chapitre suivant, nous avons décidé de nous focaliser sur une extension intégrant plusieurs grues de quai, mais dédiées à deux types d'opérations : le chargement et le déchargement de navires porte-conteneurs, chaque navire n'étant associé qu'à un seul type d'opérations.

5

ORDONNANCEMENT CONJOINT DE GRUES DE QUAI ET CAMIONS DE TRANSPORT AVEC FLUX ENTRANTS ET SORTANTS

L'ordonnancement des grues de quai et celui des camions de transport sont deux sous-problèmes importants étroitement liés dans les opérations des terminaux à conteneurs. Pour augmenter l'efficacité du système de manutention, il est nécessaire de résoudre ces deux sous-problèmes de manière intégrée. En particulier, lorsque le problème inclut des flux opposés, c'est à dire à la fois des conteneurs à décharger des navires et à charger dans les porte-conteneurs, on cherche à améliorer le taux d'utilisation des ressources et à réduire le taux de fonctionnement à vide des camions de transport. Pour appréhender cette variante de *Quay Crane and Yard Truck Scheduling Problem* (QCYTSP), ce chapitre propose un modèle exact de type programmation linéaire à nombres entiers mixtes utilisant une stratégie dite à double cycle afin de résoudre conjointement l'ordonnancement des grues de quai et celle des camions de transport. Dans ce modèle, les conteneurs sortants et entrants, les interférences et la marge de sécurité entre les grues de quai sont notamment pris en compte.

5.1/ DÉFINITION DU PROBLÈME

Dans ce chapitre, nous considérons une troisième variante de *Quay Crane Scheduling Problem*, dans laquelle deux navires porte-conteneurs simultanément à quai doivent subir respectivement des opérations de déchargement et de chargement de conteneurs. Plusieurs grues de quai (QC pour *Quay Crane*) peuvent être affectées à chaque navire, mais une grue ne peut s'occuper que d'un navire exclusivement. Les camions (YT pour *Yard Truck*) peuvent indifféremment amener les conteneurs depuis ou vers la zone de stockage, et les chariots frontaux (RS pour *Reach Stackers*) prennent le relais des camions au niveau de la zone de stockage pour les activités de récupération/empilement des conteneurs (figure 5.1). Les deux catégories de conteneurs manipulés par ces ressources de manutention sont appelées U-conteneurs et L-conteneurs. Les U-conteneurs sont les conteneurs à décharger d'un navire vers la zone de stockage dédiée, tandis que les L-conteneurs sont les conteneurs à transporter de la zone de stockage par les

camions de transport vers les grues de quai pour être chargés dans le second navire.

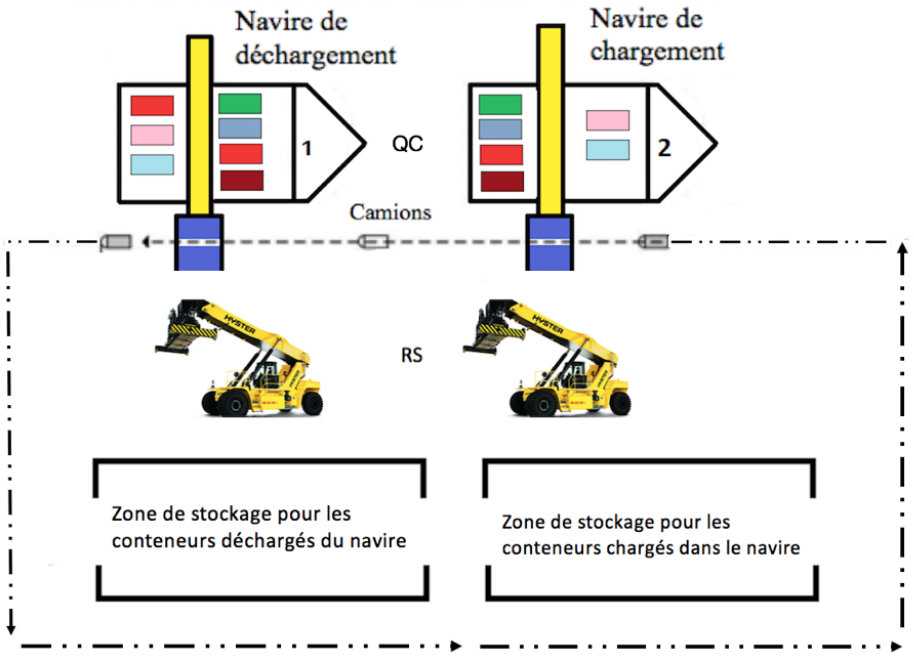

FIGURE 5.1 – Coopération entre ressources de manutention dans un terminal

Plus précisément, le processus de déchargement comprend les trois étapes suivantes (figure 5.2) :

- Les grues de quai déchargent les conteneurs du navire dit de déchargement et les chargent sur les camions de transport,
- Les camions transportent les conteneurs vers les chariots frontaux,
- Les chariots frontaux empilent les conteneurs dans la zone de stockage.

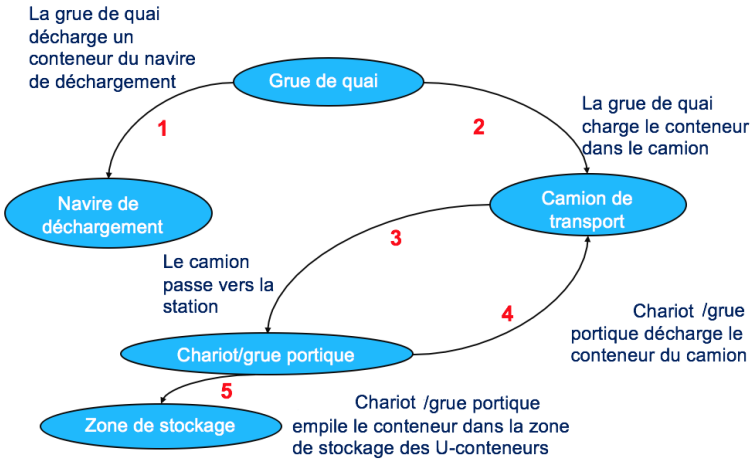

FIGURE 5.2 – Processus de déchargement des U-conteneurs

Le processus de chargement comprend également trois étapes, avec un flux inverse du précédent (figure 5.3) :

— Les chariots frontaux récupèrent les conteneurs de la zone de stockage et les chargent sur les camions de transport,
— Les camions acheminent les conteneurs vers la grue de quai,
— Les grues de quai déchargent les conteneurs des camions et les chargent sur le navire dit de chargement.

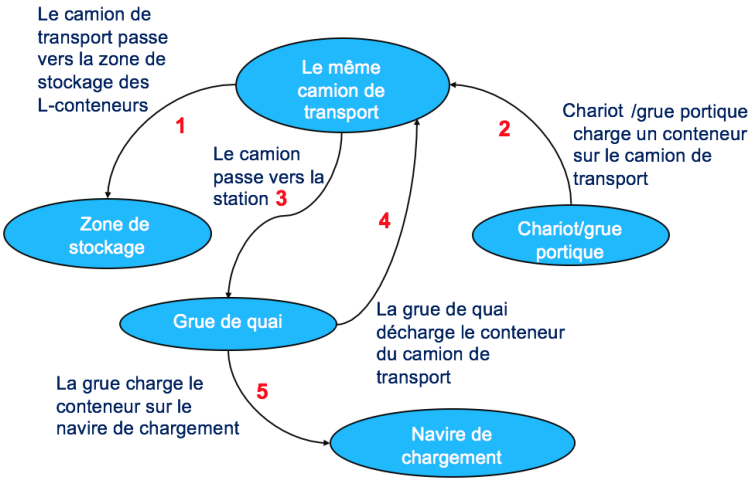

FIGURE 5.3 – Processus de chargement des L-conteneurs

En raison des processus de chargement et de déchargement particuliers, il existe deux stratégies opérationnelles de fonctionnement des camions de transport, appelées stratégie à cycle unique (ou simple) et stratégie à double cycle. La figure 5.4 donne la différence entre ces deux stratégies (avec G1 et G2 désignant respectivement l'ensemble des grues affectées à chaque navire).

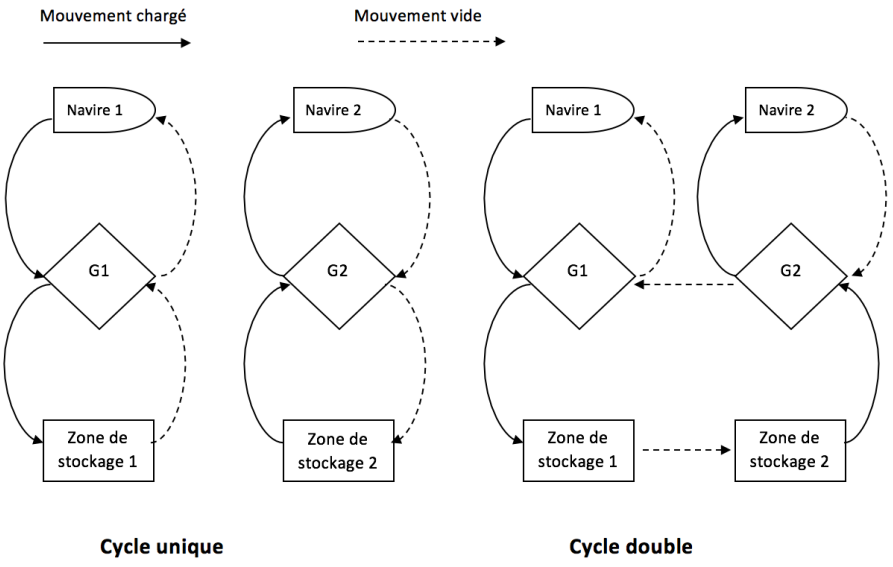

FIGURE 5.4 – Les cycles simple et double

Dans le cas d'un cycle simple, les deux navires sont gérés indépendamment, les ressources de manutention mobilisées sont dédiées à chaque type de conteneur et aux opérations associés : soit les opérations de déchargement du porte-conteneurs, soit les opérations de chargement. Dans un double cycle, les conteneurs partagent a minima les mêmes camions de transport. Un camion qui apporte un U-conteneur en zone de stockage peut en profiter pour prendre dans cette même zone, ou dans une zone voisine, un L-conteneur à amener sur le quai de chargement. Et une fois ce conteneur livré, il peut se déplacer vers le navire "voisin" pour récupérer un nouveau U-conteneur déchargé par une grue. Cette stratégie génère des déplacements à vide qui sont normalement de longueurs plus faibles que pour le cycle simple. Si nous comparons les avantages des deux stratégies, la stratégie à cycle unique est plus facile à suivre par les conducteurs, tandis que la stratégie à cycle double augmente évidemment l'efficacité du transport et réduit le nombre de mouvements à vide des camions.

Pour illustrer le mode de fonctionnement considéré dans le port, nous présentons deux exemples simples où il n'y a qu'un seul camion de transport. Le premier exemple consiste à décharger un U-conteneur du navire de déchargement vers la zone de stockage et à charger un L-conteneur de la zone de stockage vers le navire de chargement (figure 5.5) ; le second exemple consiste à décharger 2 U-conteneurs du navire de déchargement vers la zone de stockage (figure 5.6). Nous supposons que les temps de chargement et déchargement par les grues de quai sont tous égaux à 6 unités de temps ; le temps pour transporter un conteneur par le camion est de 6 unités de temps et enfin les temps chargement et déchargement par les chariots frontaux sont tous de 5 unités de temps.

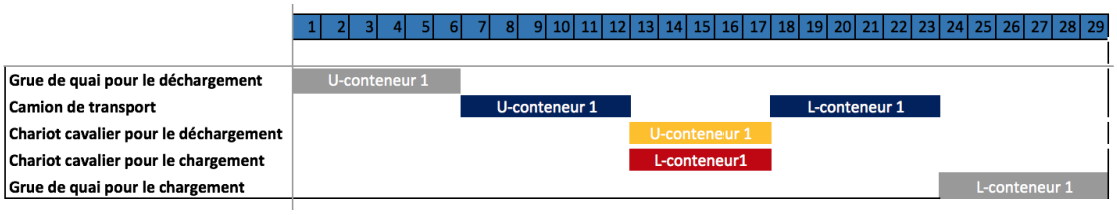

FIGURE 5.5 – Exemple pour un U-conteneur et un L-conteneur

Dans la figure 5.5, le camion transporte les 2 conteneurs (U-conteneur et L-conteneur) qui sont donc associés dans un même cycle (cycle double). Ici nous supposons qu'il n'y a pas d'intersection temporelle entre les deux chariots frontaux au niveau du camion (le U-conteneur est retiré du camion et est acheminé dans le stock avant que le L-conteneur soit enlevé d'une pile et arrive au camion). Dans cet exemple, le *makespan* est égal à 29. Notons enfin que si nous utilisions deux camions, le *makespan* pourrait être réduit à 17 en parallélisant complètement les deux opérations de chargement et déchargement.

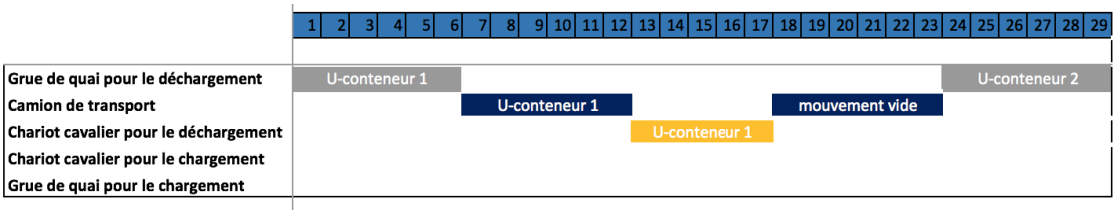

FIGURE 5.6 – Exemple pour 2 U-conteneurs

La figure 5.6 illustre un autre exemple avec les mêmes variables précédentes, mais cette fois pour 2 U-conteneurs. Nous observons que le camion transporte un premier U-conteneur, puis il revient à vide de la zone de stockage vers le navire de déchargement pour transporter le second U-conteneur. Ici le makespan sera de 40 unités de temps.

Notre étude se concentre sur le problème d'ordonnancement des grues de quai et des camions de transport (QC et YT) dans les terminaux à conteneurs. Le problème à résoudre est une seconde variante de *Quay Crane and Yard Truck Scheduling Problem* (QCYTSP), plus complexe que celle considérée au chapitre précédent, puisqu'elle considère plusieurs grues de quai et deux navires à charger et décharger. Pour augmenter le taux d'utilisation des ressources et diminuer le taux de fonctionnement à vide des camions, une stratégie à double cycle est appliquée. Pour modéliser ce problème intégré, un programme linéaire en nombres entiers (MILP3) est proposé, qui prend en compte les opérations de chargement et de déchargement, la capacité limitée des ressources, les interférences des grues de quai et la marge de sécurité des grues de quai.

5.2/ FORMULATION MATHÉMATIQUE

En général, les conteneurs existant dans le même groupe de conteneurs sont tous des conteneurs à décharger (U-conteneurs) ou des conteneurs à charger (L-conteneurs) et sont tous de même taille. Même si les U-conteneurs et L-conteneurs se trouvent sur le même navire, ils sont situés dans des baies différentes. Ainsi, les U-conteneurs et les L-conteneurs sont supposés initialement indépendants, c'est-à-dire qu'un navire de déchargement et un navire de chargement sont pris en compte dans ce modèle.

En outre, notre problème doit satisfaire des contraintes de capacité, de non interférence des grues de quai et de sécurité. La contrainte de non interférence doit empêcher les grues de quai de se croiser puisqu'elles se déplacent sur un même rail le long d'un navire. La contrainte de sécurité renforce la précédente, et exige que les grues soient maintenues à une distance de sécurité les unes des autres lorsqu'elles fonctionnent simultanément. Pour augmenter le taux d'utilisation des ressources et diminuer le taux de mouvements à vide des camions de transport, la stratégie à double cycle est utilisée. Pour exécuter la stratégie de double cycle, nous associons chaque U-conteneur à un unique L-conteneur. Dans le cas où le nombre de U-conteneurs est égal à celui des L-conteneurs, il est facile de satisfaire cette condition dans le sens où il existe toujours au moins un regroupement des conteneurs par paires. Dans le problème traité, nous considérons les hypothèses listées dans la sous-section suivante.

5.2.1/ HYPOTHÈSES

— Tous les conteneurs sont identiques.
— Chaque conteneur peut être transporté par un seul camion à la fois.
— Chaque conteneur peut être déchargé/chargé par une seule grue de quai à la fois.
— Chaque ressource de manutention (grue, camion, chariot frontal) est de capacité unitaire et ne peut manipuler/transporter qu'un conteneur à la fois.
— Le nombre de chariots frontaux est supposé suffisamment grand pour qu'un chariot soit toujours disponible.
— Chaque grue de quai est dédiée à un seul navire porte-conteneurs à la fois, et

chaque navire peut être traité par une ou plusieurs grues de quai.
— Les grues de quai sont numérotées par ordre croissant de gauche à droite par rapport au navire.
— Les positions des U-conteneurs/L-conteneurs dans le navire de déchargement/la zone de stockage sont connues.
— Les temps de manutention sont tous connus : les temps requis pour le chargement et le déchargement des conteneurs par les grues de quai et les chariots frontaux, les temps requis pour le transport des conteneurs, le placement des conteneurs déchargés et la recherche des conteneurs chargés dans les zones de stockage.
— En cas d'absence de grue de quai ou de camion de transport au point de rendez-vous, il peut exister un temps d'attente pour les camions ou les grues de quai (ils s'attendent les uns les autres pour le transfert des conteneurs).

Pour résoudre le problème posé, nous avons développé un modèle linéaire appelé MILP3, en utilisant la stratégie à double cycle. Au préalable, nous définissons ci-après Les notations pour les données et variables de décision liées à cette étude.

5.2.2/ DONNÉES

— Q : ensemble des grues de quai utilisées pour décharger les U-conteneurs du navire de déchargement
— $|Q|$: nombre de grues de quai utilisées pour décharger les U-conteneurs du navire de déchargement
— $Q^{'}$: ensemble des grues de quai utilisées pour décharger les L-conteneurs de la zone de stockage
— $|Q^{'}|$: nombre de grues de quai utilisées pour décharger les L-conteneurs de la zone de stockage
— T : ensemble des camions de transport
— $|T|$: nombre de camions de transport
— C : ensemble des U-conteneurs
— $|C|$: nombre de U-conteneurs
— $C^{'}$: ensemble des L-conteneurs
— $|C'|$: nombre de L-conteneurs
— p_i Position du U-conteneur i sur le navire de déchargement, $\forall i \in C$
— p'_i Position du L-conteneur i sur le navire de chargement, $\forall i \in C^{'}$
— v Temps de déplacement d'un camion du navire de chargement au navire de déchargement
— v'_i Temps de déplacement en charge d'un camion du navire de déchargement à la zone de stockage du U-conteneur i, $\forall i \in C$
— v''_i Temps de déplacement à vide d'un camion de la zone de stockage du U-conteneur i au navire de déchargement, $\forall i \in C$
— v'''_i Temps de déplacement en charge d'un camion de la zone de stockage du L-conteneur i au navire de chargement, $\forall i \in C^{'}$
— λ_{ij} Temps de déplacement d'un camion de la zone de stockage du U-conteneur i à la zone de stockage du L-conteneur j, $\forall i \in C$ and $\forall j \in C^{'}$
— d_i Temps de déchargement du U-conteneur i par une grue de quai, $\forall i \in C$
— d'_i Temps de chargement du L-conteneur i par une grue de quai, $\forall i \in C^{'}$
— rs Temps de déchargement d'un U-conteneur par un chariot frontal
— rs' Temps de chargement d'un L-conteneur par un chariot frontal

— s_0 Ddistance de sécurité requise entre deux grues adjacentes
— $\Omega 1$ Ensemble des paires successives des U-conteneurs
— $\Omega 2$ Ensemble des paires successives des L-conteneurs
— u Temps requis par la grue de quai pour déplacer une unité de marchandises, soit un U-conteneur soit un L-conteneur
— M Grand entier (représente l'infini)

5.2.3/ VARIABLES DE DÉCISIONS

— $X_{ijq} \begin{cases} = 1 & \text{Si la grue de quai } q \text{ décharge le U-conteneur } j \\ & \text{directement après le U-conteneur } i, \forall i \in \{0, ..., |C|\}, \forall j \in \{1, ..., |C| + 1\}, \forall q \in Q \\ = 0 & \text{sinon} \end{cases}$

— $Y_{ijq} \begin{cases} = 1 & \text{Si la grue de quai } q \text{ charge le L-conteneur } j \\ & \text{directement après le L-conteneur } i, \forall i \in \{0, ..., c'\}, \forall j \in \{1, ..., c' + 1\}, \forall q \in Q' \\ = 0 & \text{sinon} \end{cases}$

— $H_{ij} \begin{cases} = 1 & \text{Si le U-conteneur } i \text{ correspond au L-conteneur } j, \forall i \in C, \forall j \in C' \\ = 0 & \text{sinon} \end{cases}$

— $Z_{ijt} \begin{cases} = 1 & \text{Si le camion } t \text{ transporte le U-conteneur } j \text{ directement après le U-container } i \\ & \forall i \in \{0, ..., c\}, \forall j \in \{1, ..., c + 1\}, \forall t \in T \\ = 0 & \text{sinon} \end{cases}$

— $W_{ij} \begin{cases} = 1 & \text{Si la date de début de déchargement du U-conteneur } j \text{ par une grue de quai} \\ & \text{est supérieure à la date de fin du déchargement du U-conteneur } i \\ & \text{par cette grue, } \forall i \in C, \forall j \in C \\ = 0 & \text{sinon} \end{cases}$

— $W'_{ij} \begin{cases} = 1 & \text{Si la date de début de chargement du L-conteneur } j \text{ par une grue de quai} \\ & \text{est supérieure à la date de fin du chargement du L-conteneur } i \\ & \text{par cette grue, } \forall i \in C', \forall j \in C' \\ = 0 & \text{sinon} \end{cases}$

— E_i La date de fin du déchargement du U-conteneur i par une grue de quai, $\forall i \in C$
— E'_i La date de fin du chargement du L-conteneur i par une grue de quai, $\forall i \in C'$
— HA'_i La date de début de transport du U-conteneur i par un camion, $\forall i \in C$
— HA''_i La date de début de transport du L-conteneur i par un camion, $\forall i \in C'$
— HA'''_i La date de début de transport par camion du L-conteneur correspondant au U-conteneur i, $\forall i \in C$
— C_{max} La date d'achèvement de tous les navires

5.2.4/ MODÈLE LINÉAIRE

Le programme linéaire en nombres entiers que nous proposons pour l'ordonnancement des grues de quai et des camions de transport s'exprime de la manière suivante :

Objectif

$$minimize \quad C_{max} \tag{5.1}$$

(5.1) est la fonction objectif qui vise à minimiser la date d'achèvement des opérations associées aux navires de déchargement et de chargement.

Sous les contraintes

$$\sum_{j=1}^{c+1} X_{0jq} = 1 \quad \forall q \in Q \tag{5.2}$$

$$\sum_{j=1}^{c'+1} Y_{0jq} = 1 \quad \forall q \in Q' \tag{5.3}$$

$$\sum_{i=0}^{c} X_{i(c+1)q} = 1 \quad \forall q \in Q \tag{5.4}$$

$$\sum_{i=0}^{c'} Y_{i(c'+1)q} = 1 \quad \forall q \in Q' \tag{5.5}$$

$$\sum_{q \in Q} \sum_{j=1}^{c+1} X_{ijq} = 1 \quad \forall i \in C \tag{5.6}$$

$$\sum_{q \in Q'} \sum_{j=1}^{c'+1} Y_{ijq} = 1 \quad \forall i \in C' \tag{5.7}$$

$$\sum_{j=1}^{c+1} X_{ijq} = \sum_{j=0}^{c} X_{jiq} \quad \forall i \in C, \forall q \in Q \tag{5.8}$$

$$\sum_{j=1}^{c'+1} Y_{ijq} = \sum_{j=0}^{c'} Y_{jiq} \quad \forall i \in C', \forall q \in Q' \tag{5.9}$$

Les contraintes (5.2), (5.4), (5.6) et (5.8) définissent la séquence de déchargement des U-conteneurs par les grues de quai : elles assurent que le 1^{er} U-conteneur doit être déchargé du navire de déchargement ainsi que le dernier U-conteneur, et que toutes les affectations grue-conteneur pour le déchargement sont faites.
Les contraintes (5.3), (5.5), (5.7) et (5.9) définissent la séquence de chargement des L-conteneurs par les grues de quai : elles garantissent que le 1^{er} L-conteneur doit être chargé sur le navire de chargement ainsi que le dernier L-conteneur, et que toutes les affectations grue-conteneur pour le chargement sont faites).

$$\sum_{i=1}^{c+1} Z_{0it} = 1 \quad \forall t \in T \tag{5.10}$$

$$\sum_{i=0}^{c} Z_{i(c+1)t} = 1 \quad \forall t \in T \tag{5.11}$$

$$\sum_{t \in T} \sum_{j=1}^{c+1} Z_{ijt} = 1 \quad \forall i \in C \tag{5.12}$$

$$\sum_{j=1}^{c+1} Z_{ijt} = \sum_{j=0}^{c} Z_{jit} \quad \forall i \in C, \forall t \in T \tag{5.13}$$

Les contraintes (5.10), (5.11), (5.12) et (5.13) fournissent la séquence de transport des U-conteneurs par les camions, depuis le navire de déchargement.

$$\sum_{i' \in C'} H_{ii'} = 1 \quad \forall i \in C \tag{5.14}$$

$$\sum_{i \in C} H_{ii'} = 1 \quad \forall i^{'} \in C' \tag{5.15}$$

Les contraintes (5.14), (5.15) donnent les affectations uniques pour les paires des U-conteneurs et des L-conteneurs associés.

$$E_i \geq d_i - M * (1 - \sum_{q \in Q} X_{0iq}) \quad \forall i \in C \tag{5.16}$$

$$E_i \geq E_j - M * (1 - \sum_{q \in Q} X_{jiq}) + (p_i - p_j) * u + d_i \quad \forall i \in C, \forall j \in C \tag{5.17}$$

Les contraintes (5.16), (5.17) fournissent la date de fin de déchargement des U-conteneurs par les grues de quai.

$$E'_i \geq E'_j - M * (1 - \sum_{q \in Q'} X_{jiq}) + (p'_i - p'_j) * u + d'_i \quad \forall i \in C', \forall j \in C' \tag{5.18}$$

$$E'_i \geq HA''_i + d'_i \quad \forall i \in C' \tag{5.19}$$

Les contraintes (5.18), (5.19) fournissent la date de fin de chargement des L-conteneurs par les grues de quai sur le navire de chargement.

$$HA'_i \geq E_i + v'_i + rs \quad \forall i \in C \tag{5.20}$$

$$HA'_i \geq HA'''_j - M * (1 - \sum_{t \in T} Z_{jit}) + v'_i + rs \quad \forall i \in C, \forall j \in C \tag{5.21}$$

Les contraintes (5.20), (5.21) fournissent la date de fin de transport des U-conteneurs par les camions.

$$HA''_{i'} \geq HA'_i - M * (1 - H_{ii'}) + \lambda_{ii'} + rs' + v'''_{i'} + v \quad \forall i \in C, \forall i' \in C' \tag{5.22}$$

Les contraintes (5.22) déterminent la date de fin de transport des L-conteneurs par les camions.

$$HA'''_i \geq HA''_{i'} - M * (1 - H_{ii'}) \quad \forall i \in C, \forall i' \in C' \tag{5.23}$$

Les contraintes (5.23) déterminent la date de fin de transport du L-conteneur associé à un U-conteneur, par les camions.

$$E_j - d_j \geq E_i \quad \forall (i, j) \in \Omega 1 \tag{5.24}$$

Les contraintes (5.24) assurent que le déchargement de chaque U-conteneur doit être terminé avant celui du U-conteneur qui le suit, s'ils appartiennent à $\Omega 1$.

$$E'_j - d'_j \geq E'_i \quad \forall (i, j) \in \Omega 2 \tag{5.25}$$

Les contraintes (5.25) garantissent que le chargement de chaque L-conteneur doit être terminé avant celui du L-conteneur qui le suit, s'ils appartiennent à $\Omega 2$.

$$M * (1 - W_{ij}) \geq E_i - (E_j - d_j) \quad \forall i \in C, \forall j \in C \tag{5.26}$$

$$M * (1 - W'_{ij}) \geq E'_i - (E'_j - d'_j) \quad \forall i \in C', \forall j \in C' \tag{5.27}$$

$$E_j - d_j - E_i \leq M * W_{ij} \quad \forall i \in C, \forall j \in C \tag{5.28}$$

$$E'_j - d'_j - E'_i \leq M * W'_{ij} \quad \forall i \in C', \forall j \in C' \tag{5.29}$$

$$(p_i - p_j) * (i - j) + M * (W_{ij} + W_{ji}) \geq (i - j) * s_0 \quad \forall i \in C, \forall j \in C, i \neq j \tag{5.30}$$

$$(p'_i - p'_j) * (i - j) + M * (W'_{ij} + W'_{ji}) \geq (i - j) * s_0 \quad \forall i \in C', \forall j \in C', i \neq j \tag{5.31}$$

Les contraintes (5.26), (5.27), (5.28), (5.29), (5.30) et (5.31) garantissent le non-croisement et la marge de sécurité entre les grues de quai.

$$C_{max} \geq E'_i \quad \forall i \in C' \tag{5.32}$$

La contrainte (5.32) détermine la date d'achèvement des opérations associées au navire de chargement.

En pratique, il est courant que le nombre de conteneurs de chaque navire soit différent. Sans perte de généralité, nous supposons que le nombre de U-conteneurs (c) est supérieur à celui des L-conteneurs (c'), donc $c > c'$. Dans ce cas, il existe $c - c'$ U-conteneurs n'ayant pas de L-conteneurs correspondants, de sorte que les camions retournent à vide de la zone de stockage au navire de déchargement. Pour obtenir une correspondance mutuelle, $c - c'$ L-conteneurs fictifs sont ajoutés. Le modèle exprimé ci-après est une extension de notre modèle au cas où $c > c'$:

Objectif

$$minimize \quad C_{max} \tag{5.33}$$

Sous les contraintes

contraintes (5.2) → (5.14), (5.16) → (5.21), (5.23) → (5.31)

Et sous les contraintes

$$\sum_{i \in C} H_{ii'} = 1 \quad \forall i' \in \{1, ..., c', ..., c\} \tag{5.34}$$

$$HA''_{i'} \geq HA'_i - M * (1 - H_{ii'}) + v''_i \quad \forall i \in C, \forall i' \in \{c' + 1, ..., c'\} \tag{5.35}$$

$$C_{max} \geq HA'''_i \quad \forall i \in C \tag{5.36}$$

Exprimé autrement, dans notre premier modèle, nous avons remplacé la contrainte (5.15) par (5.34), la contrainte (5.22) par (5.35) et la contrainte (5.32) par (5.36).

La contrainte (5.34) fournit les affectations pour les paires incluant les L-conteneurs fictifs. La contrainte (5.35) détermine la date d'achèvement des U-conteneurs avec les mouvements à vide. Enfin, la contrainte (5.36) définit le *makespan* de tous les navires considérés.

5.3/ RÉSULTATS EXPÉRIMENTAUX

5.3.1/ IMPLÉMENTATION ET INSTANCES

Pour valider le modèle proposé, deux jeux d'instances sont évalués, de tailles variables selon les cas. Toutes les données numériques sont générées de manière aléatoire. Le modèle est résolu à l'aide du solveur CPLEX 12.6, et les tests sont exécutés sur MacBook Pro 2,7 GHz Intel Core i5 avec 8GB RAM 1867 MHz DDR3 sous OSX 10.11.6.

Les deux jeux testés comprennent au total 45 instances, caractérisées comme suit :

- Le premier jeu est constitué de 40 instances nommées C1 à C40. Elles comprennent 1 à 9 grues de quai (au plus 5 par navire), 1 à 10 camions et entre 4 et 30 conteneurs par type (au total entre 10 et 60 conteneurs).
- Le second jeu est associé à notre application réelle liée au port de Tripoli-Liban, à partir de laquelle nous avons pu générer 5 nouvelles instances, nommées TL13 à TL17. Elles comportent 2 grues de quai (une affectée à chaque navire), 1 ou 2 camions et de 2 à 8 conteneurs par type à charger ou décharger (donc au total de 4 à 15 conteneurs).

5.3.2/ IMPACT DU NOMBRE DE CAMIONS SUR LE MAKESPAN

Nous nous sommes tout d'abord intéressés à l'effet du nombre de camions de transport sur le *makespan* obtenu par le modèle proposé. Le tableau 5.1 illustre cet impact.
Les valeurs de Makespan1 sont obtenues pour 20 U-conteneurs et 20 L-conteneurs, les valeurs de Makespan2 sont obtenues pour 25 U-conteneurs et L-25 conteneurs, les valeurs de Makespan3 sont obtenues pour 30 U-conteneurs et 30 L-conteneurs et les valeurs de Makespan4 sont obtenues pour 35 U-conteneurs et 35 L-conteneurs.

TABLE 5.1 – Relation entre *makespan* et nombre de camions de transport

Nombre de camions	Makespan1	Makespan2	Makespan3	Makespan4
2	790	1190	1600	2000
5	600	930	1240	1540
7	550	745	1130	1340
9	510	630	850	1050
12	380	540	730	990
15	240	380	510	730

La figure 5.7 schématise aussi la relation entre le nombre de camions de transport et la valeur de la fonction objectif. Comme le montre cette figure, le nombre de camions de transport a un effet évident sur la valeur de la fonction objectif pour un problème à grande échelle, cependant, il fait peu d'effet sur un problème à petite échelle.
Les quatre histogrammes démontrent la variation du *makespan* par rapport au nombre de camions de transport. On remarque que, lorsque le nombre de camions de transport dépasse 12, l'effet devient plus faible.

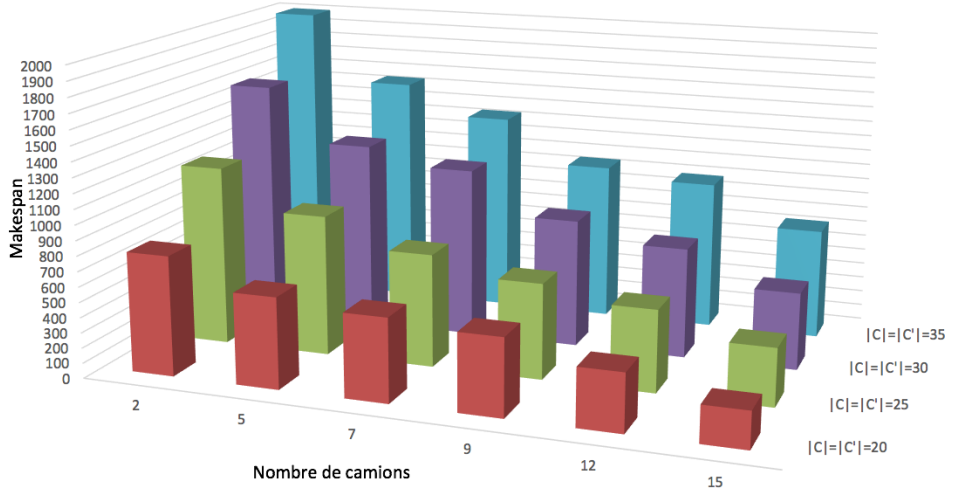

FIGURE 5.7 – Effet du nombre de camions sur le makespan

5.3.3/ RÉSULTATS DE MILP3

Le tableau 5.2 montre les résultats des tests dans le cas où le nombre des U-conteneurs est le même que celui des L-conteneurs sortants. Le tableau 5.3 montre les résultats dans le cas où le nombre de U-conteneurs est supérieur à celui des L-conteneurs.

TABLE 5.2 – Résultats du MILP3 dans le cas où $|C|= |C'|$

No.	$\|C\|= \|C'\|$	$\|Q\|$	$\|Q'\|$	$\|T\|$	MILP3
C1	6	1	1	1	1115
C2	8	1	1	2	748
C3	10	2	1	3	686
C4	10	2	2	4	467
C5	12	2	1	3	402
C6	12	2	2	5	452
C7	12	1	2	4	502
C8	15	2	2	5	558
C9	15	2	2	4	562
C10	15	3	2	5	545
C11	20	2	2	4	348
C12	20	2	2	6	476
C13	20	2	3	6	476
C14	25	3	3	8	268
C15	25	2	3	6	268
C16	25	2	2	6	270
C17	30	2	2	8	168
C18	30	2	2	6	N.A
C19	30	2	2	4	N.A
C20	30	3	2	6	N.A

N.A. : pas de résultat dans le délai imparti

A partir de l'instance C18 dans le tableau 5.2, pour 60 conteneurs, 4 ou 5 grues de quai et 4 ou 6 camions de transport, CPLEX n'a pas fourni de résultat après 3 heures d'exécution, ainsi qu'à partir de l'instance C38 du tableau 5.3 pour 53 conteneurs, 7 à 9 grues et 10 camions. Ainsi, notre MILP3 fonctionne pour les petites et moyennes instances et est moins efficace pour les grandes instances.

TABLE 5.3 – Résultats du MILP3 dans le cas où $|C| \neq |C'|$

No.	$\|C\|$	$\|C'\|$	$\|Q\|$	$\|Q'\|$	$\|T\|$	MILP3
C21	6	4	1	1	2	492
C22	8	6	2	2	3	433
C23	10	6	2	2	3	527
C24	10	8	2	2	4	450
C25	12	8	2	2	4	445
C26	12	10	2	2	4	545
C27	12	10	2	2	5	351
C28	14	12	2	2	5	382
C29	14	12	3	2	6	270
C30	16	12	3	2	6	312
C31	16	14	3	2	6	433
C32	16	14	3	2	7	327
C33	18	14	4	3	8	340
C34	18	16	4	3	10	337
C35	18	16	4	3	8	229
C36	25	20	4	3	10	229
C37	26	24	4	3	10	225
C38	28	25	4	3	10	N.A
C39	28	25	5	4	10	N.A
C40	30	25	5	4	10	N.A

N.A. : pas de résultat dans le délai imparti

5.3.4/ COMPARAISON AVEC DES RÉSULTATS DU PORT DE TRIPOLI

Nous avons à nouveau comparé les performances de notre MILP3, avec des instances issues du port de Tripoli au Liban. Nous avons donc exécuté nos algorithmes sur les 5 instances décrites précédemment, et dont nous retrouvons les caractéristiques dans les premières colonnes du tableau 5.4.

TABLE 5.4 – Comparaison avec des résultats réels du port

No.	$\|C\|$	$\|C'\|$	$\|T\|$	Makespan		Écart
				Port (s)	MILP3 (s)	(%)
TL13	2	2	1	783	629	-19.67
TL14	5	4	2	965	782	-18.96
TL15	6	6	2	1129	912	-19.22
TL16	7	4	2	1046	835	-20.17
TL17	8	7	2	1393	1076	-22.76

$Ecart = ((makespan_{CPLEX} - makespan_{port})/makespan_{port}) * 100$

Dans le tableau 5.4, la première colonne donne le nom de l'instance testée, les trois colonnes suivantes donnent les caractéristiques de chaque instance, les colonnes 5 et 6 fournissent les *makespan* issus du port de Tripoli et de notre MILP. La dernière colonne fournit l'écart relatif entre le *makespan* du port et celui du MILP3.

Par exemple pour l'instance TL17 pour 8 U-conteneurs, 7 L-conteneurs et 2 camions, le *makespan* du port est 1393 secondes tandis que le *makespan* du MILP3 est de 1076 secondes, donc notre modèle nous permet d'améliorer notre temps total de 22,76%. Globalement, pour les instances du port de Tripoli, nous observons que le MILP3 pourrait améliorer de manière substantielle les performances du port, en moyenne de 20.16%.

5.4/ CONCLUSION

Dans ce chapitre, nous avons étudié une seconde variante de *Quay Crane and Yard Truck Scheduling Problem* (QCYTSP) avec plusieurs grues et camions de transport, en considérant des flux entrants et sortants des navires. Une stratégie dite à cycle double a été utilisée pour optimiser le taux d'utilisation des ressources et diminuer les temps de fonctionnement à vide des camions de transport. Nous avons proposé un modèle de type MILP dont la fonction objectif doit tendre à minimiser la durée de stationnement des navires à quai. Les tests réalisés nous ont permis de valider notre modèle exact. Néanmoins, celui-ci présente des performances limitées pour des problèmes de grande taille. De prochaines études devront notamment inclure le recours à des méthodes approchées plus efficaces pour un passage à l'échelle, avec l'élaboration de nouveaux algorithmes pour comparer les performances opérationnelles dans un terminal à conteneurs.

6

CONCLUSION ET PERSPECTIVES

6.1/ CONCLUSION

Cette thèse présente de nouveaux modèles et approches de résolution pour plusieurs variantes de problème d'ordonnancement de type *Quay Crane Scheduling Problem*. L'objectif principal de ce travail était de développer une méthodologie de planification générale et intégrée pour les cas complexes étudiés.
Un cas d'étude réel a été exploité comme "test-bed" dans cette recherche. L'accent a été mis sur les spécificités de ce secteur maritime, impliquant un travail préalable de détermination du fonctionnement des fonctions d'ordonnancement. Les principales contributions de cette thèse et recommandations pour de futures recherches sont résumées dans ce qui suit.

Résumé des contributions

Les problèmes d'ordonnancement ont fait l'objet d'une grande attention au cours des décennies passées, en raison de leur importance pour l'efficacité de la gestion opérationnelle des systèmes. Une grande variété d'approches de modélisation est apparue dans la littérature, introduisant différents types de formulation et impliquant de multiples décisions et objectifs. Cependant, la modélisation, les performances de calcul et l'intégration de méthodes d'optimisation dans les processus décisionnels réels des entreprises restent des questions ouvertes que nous avons essayé de prendre en compte dans cette étude. Ainsi, les principales contributions de nos travaux peuvent être résumées comme suit :

Pour le premier problème étudié avec un seul navire porte-conteneurs et plusieurs grues à quai, nous avons proposé un algorithme génétique efficace qui est comparé à un algorithme numératif que nous avons développé et à un modèle de programmation mixte en nombres entiers comme solutions exactes. Bien que ce problème corresponde à une variante plutôt basique de QCSP, nos méthodes nous ont permis d'apporter quelques améliorations à l'ordonnancement des grues à quai mis en oeuvre en pratique dans le port de Tripoli-Liban.

Dans le deuxième problème étudié, de type QCYTSP, avec un seul navire, une seule grue de quai et plusieurs camions de transport, nous avons considéré les temps de repositionnement des grues de quai ainsi que d'autres contraintes afin de prendre en compte les temps de chargement et de déchargement des conteneurs. A notre connaissance, cette configuration n'a pratiquement pas été étudiée jusqu'à maintenant. Pour ce problème,

nous avons développé un algorithme numératif basé sur un modèle de programmation linéaire en nombres entiers résolu par CPLEX. Ensuite, un algorithme génétique a été développé et comparé aux méthodes exactes proposées.

Le troisième problème étudié est le plus complexe. C'est une variante de QCYTSP intégrant deux flux de marchandises "opposés", avec deux navires, plusieurs grues de quai et plusieurs camions de transport. Pour le résoudre, nous avons proposé un modèle général de type programme linéaire en nombres entiers résolu par CPLEX. Ce modèle, quoi que exact, nous a permis à nouveau de nous confronter à des résultats réels et d'optimiser le temps total nécessaire pour servir les clients du port de Tripoli.

Ces travaux présentent deux avantages substantiels :
Premièrement, l'étude de ces problèmes d'ordonnancement a permis de concentrer notre recherche sur le développement de modèles d'optimisation pouvant être efficacement résolus. Ainsi, les exigences de programmation pratiques ont été discutées avec des personnels des ports maritimes et ont été prises en compte dans nos modèles chaque fois qu'il a été possible. Deuxièmement, bien que les modèles d'ordonnancement et les approches de résolution développés aient été initialement motivés par une étude de cas, celle du port de Tripoli, ils peuvent être appliqués à d'autres types de terminaux, tant que le problème d'ordonnancement associé présente une structure et un mode de fonctionnement similaires.

Les résultats de cette thèse contribuent à la modélisation des problèmes d'ordonnancement dans les ports, et à la résolution de vrais problèmes maritimes. Ils pourraient également être considérés comme une base solide pour le développement d'outils d'aide à la décision.

6.2/ PERSPECTIVES

Les modèles intégrés présentés ici sont soumis à certaines hypothèses restrictives, qui pourront faire l'objet de recherches plus poussées. En relaxant certaines de ces restrictions, le problème deviendra plus complexe mais aussi plus réaliste. Cependant, afin d'améliorer l'applicabilité et les performances du modèle, les extensions suivantes sont suggérées :

La première opportunité de recherche concerne le problème d'ordonnancement de plusieurs grues de quai (QCSP) devant décharger tous les conteneurs d'un seul navire porte-conteneurs. Nous avons supposé que chaque grue de quai doit terminer le déchargement de tous les conteneurs d'une baie avant de décharger les conteneurs d'autres baies. Relaxer cette hypothèse reviendrait à permettre qu'une grue puisse décharger n'importe quel conteneur dans n'importe quelle baie. Cela augmente la combinatoire du problème mais pourrait conduire à de meilleures performances.

La deuxième opportunité de recherche concerne le problème d'ordonnancement d'une seule grue de quai et plusieurs camions de transport, pour décharger tous les conteneurs d'un seul navire (variante de QCYTSP). Dans ce travail, nous avons considéré qu'il existe à tout moment des chariots frontaux disponibles pour décharger les conteneurs des camions de transport. Cette hypothèse n'est pas toujours vraie et nous pourrions intégrer les chariots de frontaux de manière plus réaliste dans le problème d'ordonnancement. Nous aurions alors à traiter un problème de synchronisation entre camions et chariots de

même type que celui entre grues et camions.

La troisième opportunité de recherche concerne le problème d'ordonnancement de plusieurs grues de quai et plusieurs camions de transport (seconde variante de QCYTSP), dans lequel la limitation du nombre de chariots frontaux et la congestion des camions de transport ne sont pas prises en considération. Là encore, nous pourrions étendre notre modèle pour intégrer ces cas.

Outre les extensions de QCSP suggérées, d'autres perspectives sont envisagées, relatives cette fois aux méthodes de résolution associées aux problèmes traités. Notamment, d'autres méthodes approchées pourraient être développées et comparées à nos algorithmes génétiques, telles que des ALNS particulièrement adaptées à de nombreux problèmes d'optimisation. Des études comparatives pourraient être menées sur la base des jeux d'instances déjà utilisés, voire sur d'autres jeux de taille plus importante, pour évaluer les avantages et les limites de chaque méthode.

BIBLIOGRAPHIE

[A Schwarz et al., 1994] A Schwarz, D., Berry, M., et Shaviv, E. (1994). **Representing andsolving the automated building design problem**. *Computer-aided design*, 26, no.9 :689–698.

[AAPA, 2018] AAPA (2018). **Port industry statistics**. *American Association of Port Authorities, http ://www.aapa-ports.org/*.

[Al-Dhaheri et al., 2015] Al-Dhaheri, N., et Diabat, A. (2015). **The quay crane scheduling problem**. *Journal of Manufacturing Systems*, 36 :87–94.

[Alnaqbi et al., 2016] Alnaqbi, B., Alrubaiai, H., et Alawi, S. A. (2016). **Combination of a dynamic-hybrid berth allocation problem with a quay crane scheduling problem**. *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*.

[Arora et al., 2009] Arora, S., et Barak, B. (2009). **Computational complexity : A modern approach**. *Cambridge University Press, ISBN 0-521-42426-7*.

[Avriel et al., 1993] Avriel, M., et Penn, M. (1993). **Exact and approximate solutions of the container ship stowage problem**. *Computers and Industrial Engineering*, page 271–274.

[Avriel et al., 1998] Avriel, M., Penn, M., shiprer, N., et Witteboon, S. (1998). **Stowage planning for container ships to reduce the number of shifts**. *Annals of Operations Research*, 76 :55–71.

[Awar et al., 2016] Awar, K. A., Alawani, M., et Jaberi, S. A. (2016). **A multi-vessel quay crane scheduling problem**. *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*.

[Azza et al., 2014] Azza, L., El-merouani, M., et Medouri, A. (2014). **Ant colony system for solving quay crane scheduling problem in container terminal**. *2014 International Conference on Logistics Operations Management*.

[Bazzazi et al., 2009] Bazzazi, M., Safaei, N., et Javadian, N. (2009). **A genetic algorithm to solve the storage space allocation problem in a container terminal**. *Computers and Industrial Engineering*, 36 :1711–1725.

[Bellman, 1957] Bellman, R. (1957). **Dynamic programming**. *Princeton, Princeton University Press, Réimpression 2003, Dover Publication, Mineola, New-York, (ISBN 0-486-42809-5)*.

[Bierwirth et al., 2015] Bierwirth, C., et F, M. (2015). **A follow-up survey of berth allocation and quay crane scheduling problems in container terminals**. *European Journal of Operational Research*, 244 :675–689.

[Bierwirth et al., 2009] Bierwirth, C., et Meisel, F. (2009). **A fast heuristic for quay crane scheduling with interference constraints**. *Journal of Scheduling*, 12 :345–360.

[Bish et al., 2001] Bish, E., Leong, T., Li, C., Ng, J., et Simchi-Levi, D. (2001). **Analysis of a new vehicle scheduling and location problem**. *Naval research logistics*.

[Boysen et al., 2016] Boysen, A. N., Briskorn, B. D., et Meisel, C. F. (2016). **A generalized classification scheme for crane scheduling with interference**. *European Journal of Operational Research*, 258 :343–357.

[Bradley et al., 1977] Bradley, Hax, et Magnanti (1977). **Integer programming**. *Applied Mathematical Programming, Addison-Wesley*.

[Castilho et al., 1993] Castilho, B. D., et Daganzo, C. (1993). **Handling strategies for import containers at marine terminals**. *Transportation Research B*, 27 :151–166.

[Chang et al., 2010] Chang, D., liang He, J., et Bi, Z. (2010). **An investigation into berth and quay crane scheduling for container terminals based on knowledge**. *International Conference on Future Information Technology and Management Engineering*.

[Chen et al., 2013] Chen, L., Langevin, A., et Lu, Z. (2013). **Integrated scheduling of crane handling and truck transportation in a maritime container terminal**. *European Journal of Operational Research*, 225 :142–152.

[Chen et al., 2004] Chen, P., Fu, Z., Lim, A., et Rodrigues, B. (2004). **Port yard storage optimization**. *Automation Science and Engineering*, pages 26–37.

[Chung et al., 2012] Chung, S., et Choy, K. (2012). **A modified genetic algorithm for quay crane scheduling operations**. *Expert Systems with Applications*.

[Chung et al., 2013] Chung, S., et T, C. F. (2013). **A workload balancing genetic algorithm for the quay crane scheduling problem**. *International Journal of Production Research*, 51 :4820–4834.

[Daganzo, 1989] Daganzo, C. (1989). **The quay crane scheduling problem**. *Transportation research*, 23 :159–175.

[Dantzig, 1990] Dantzig, G. (1990). **Origins of the simplex method**. *A History of Scientific Computing, ACM Press, Reading, MA, USA*, pages 141–151.

[Diabat et al., 2014] Diabat, A., et Theodorou, E. (2014). **An integrated quay crane assignment and scheduling problem**. *Computers and industrial engineering*, 73 :115–123.

[Dkhil et al., 2020] Dkhil, H., Diarrassouba, I., Benmansour, S., et Yassine, A. (2020). **Modelling and solving a berth allocation problem in an automotive transshipment terminal**. *Journal of the Operational Research Society*, pages 1–14.

[Dkhil et al., 2013] Dkhil, H., Yassine, A., et Chabchoub, H. (2013). **Optimization of container handling systems in automated maritime terminal**. *Springer-Verlag Berlin Heidelberg*.

[Dorigo, 1992] Dorigo, M. (1992). **Optimization, learning and natural algorithms**. *PhD Thesis, Politecnico di Milano*.

[Dubreuil, 2008] Dubreuil, J. (2008). **La logistique des terminaux portuaires de conteneurs**. *CIRRELT-2008-38*.

[Fukunaga et al., 1975] Fukunaga, K., et Narendra, P. (1975). **A branch and bound algorithm for computing k-nearest neighbors**. *IEEE Transactions on Computers*, pages 750–753, doi :10.1109/t–c.1975.224297.

[Glover, 1997] Glover, M. (1997). **F., laguna, tabu search**. *Kluwer Academic Publishers, Dordrecht*.

[Goldberg, 1989] Goldberg, D. (1989). **Genetic algorithms in search**. *Reading, MA : Addison-Wesley Professional*, ISBN 978-0201157673.

[Goldberg, 1994] Goldberg, D. E. (1994). **Algorithmes génétiques**. *Addison-Wesley France*.

[GOTHA, 1993] GOTHA (1993). **Les problèmes d'ordonnancement**. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, tome 27, no 1 :77–150.

[Guo et al., 2013] Guo, P., W, C., et Y, W. (2013). **A modified generalized extremal optimization algorithm for the quay crane scheduling problem with interference constraints**. *Engineering Optimization*, 46 :1411–1429.

[Hakam et al., 2012] Hakam, M. H., D, S. W., et T, H. (2012). **A genetic algorithm approach for quay crane scheduling with non-interference constraints at narvik container terminal**. *International Journal of Logistics Research and Applications*, 15 :269–281.

[Haoyuan et al., 2017] Haoyuan, L., et Qi, S. (2017). **Simulation-based optimization on quay crane scheduling of container terminals**. *2017 29th Chinese Control And Decision Conference (CCDC)*.

[Holland, 1960] Holland, J. (1960). **Genetic algorithms, computer programs that evolve in ways that resemble natural selection can solve complex problems even their creators do not fully understand**. *http ://www2.econ.iastate.edu/tesfatsi/holland.GAIntro.htm*.

[Homayouni et al., 2014] Homayouni, S., Tang, S., et Motlagh, O. (2014). **A genetic algorithm for optimization of integrated scheduling of cranes, vehicles, and storage platforms at automated container terminals**. *Journal of Computational and Applied Mathematics*, 270 :545–556.

[Imai et al., 2008] Imai, A., Chen, H., Nishimura, E., et Papadimitriu, S. (2008). **The simultaneous berth and quay crane allocation problem**. *Transportation Research Part E*, 44 :900–920.

[Imai et al., 1997] Imai, A., Nagaiwa, K., et Tat, C. (1997). **Efficient planning of berth allocation for container terminals in asia**. *Journal of Advanced Transportation*, 31 :75–94.

[Imai et al., 2006] Imai, A., Sasaki, K., Nishimura, E., et 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*.

[ISEMAR, 2002] ISEMAR (2002). **La révolution du conteneur, note de synthèse n°49, novembre 2002**. *www.isemar.fr/wp-content/uploads/2016/11/note-de-synthese-isemar-49.pdf*.

[Izquierdo et al., 2011] Izquierdo, C. E., Velarde, J. L. G., Batista, B. M., et Moreno-Vega, J. M. (2011). **Estimation of distribution algorithm for the quay crane scheduling problem**. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2011)*, pages 183–194.

[Jing, 2010] Jing, S. (2010). **A heuristic algorithm for the integrated yard truck scheduling in container terminal with twin 40-foot quay crane**. *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE)*.

[Kaveshgar et al., 2015] Kaveshgar, N., et Huynh, N. (2015). **Integrated quay crane and yard truck scheduling for unloading inbound containers**. *Int. Production Economics*, 159 :168–177.

[Kaveshgar et al., 2012] Kaveshgar, N., N, H., et K, R. S. (2012). **An efficient genetic algorithm for solving the quay crane scheduling problem**. *Expert Systems with Applications*, 39 :13108–13117.

[Kennedy et al., 1995] Kennedy, J., et Eberhart, R. (1995). **Particle swarm optimization**. *IEEE International Conference on Neural Networks*, 4 :1942–1948.

[Kim, 2008] Kim, K. (2008). **Operational issues in modern container terminals**. *Intelligent Freight Transportation*, pages 51–69.

[Kim et al., 2003] Kim, K., et Moon, K. (2003). **Berth scheduling by simulated annealing**. *Transportation Research Part B*, 37 :541–560.

[Kim et al., 2004] Kim, K., et Park, Y. (2004). **A crane scheduling method for port container terminals**. *European Journal of Operation Research*, 156 :752–768.

[Kumar et al., 2008] Kumar, S., et Vlacic, L. (2008). **Performance analysis of container unloading operations at the port of suva using a simplified analytical model (sam)**. *J. Adv. Comput. Intell. Intell. Inform.*, 12 :355–360.

[Le-Anh et al., 2010] Le-Anh, T., de Koster, R., et Yu, T. (2010). **Performance evaluation of dynamic scheduling approaches in vehicle-based internal transport systems**. *International Journal of Production Research*, 48 :7219–7242.

[Liang et al., 2018] Liang, C., Fan, L., Xu, D., Ding, Y., et Gen, M. (2018). **Research on coupling scheduling of quay crane dispatch and configuration in the container terminal**. *Computers and Industrial Engineering*, 125 :649–657.

[Liang et al., 2008] Liang, C., Huang, Y., et Yang, Y. (2008). **A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning**. *Computers and industrial engineering*, 56 :1021–1028.

[Liang et al., 2009] Liang, C., Huang, Y., et Yang, Y. (2009). **A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning**. *Computers and Industrial Engineering*, 56 :1021–1028.

[Lim et al., 2004] Lim, A., Rodrigues, B., Xiao, F., et Zhu, Y. (2004). **Quay crane scheduling with spatial constraints**. *Naval Research Logistics*, 51 :386–406.

[Liu et al., 2015] Liu, M., Wang, S., et Chu, C. (2015). **A branch-and-price framework for the general double-cycling problem with internal-reshuffles**. *2015 IEEE 12th International Conference on Networking, Sensing and Control*.

[Liu et al., 2016a] Liu, M., Zheng, F., Xu, Y., et Chu, C. (2016a). **Approximation algorithm for uniform quay crane scheduling at container ports**. *Discrete Mathematics, Algorithms and Applications*.

[Liu et al., 2016b] Liu, Y., et Liu, T. (2016b). **The hybrid intelligence swarm algorithm for berth-quay cranes and trucks scheduling optimization problem**. *15th Int'l Conf. on Cognitive and Cognitive Computing*.

[Lloyd's-List, 2018] Lloyd's-List (2018). **Top 100 container ports**. *http ://www.lloydslist.com/*.

[Meer, 2000] Meer, R. V. (2000). **Operational control of internal transport**. *ERIM Ph.D. series Research in Management*.

[MEISEL et al., 2011] MEISEL, F., et BIERWIRTH, C. (2011). **A unified approach for the evaluation of quay crane scheduling models and algorithms**. *Computers and Operations Research*, 38 :683–693.

[Moghaddam et al., 2008] Moghaddam, R. T., Makui, A., Salahi, S., Bazzazi, M., et Taheri, F. (2008). **An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports**. *Computers & Industrial Engineering*.

[Msakni et al., 2016] Msakni, M., Diabat, A., Rabadi, G., et Kotachi, M. (2016). **An integrated quay crane assignment and scheduling problem using branch-and-price**. *International Conference on Computational Science and Computational Intelligence*.

[Niu et al., 2014] Niu, B., Xie, T., Duan, Q., et Tan, L. (2014). **Particle swarm optimization for integrated yard truck scheduling and storage allocation problem**. *IEEE Congress on Evolutionary Computation (CEC)*.

[Oliveira et al., 2016] Oliveira, J. D., Barbosa, J., et Lamprou, M. (2016). **Multi-objective optimization of the quay crane assignment and scheduling problem : Time and movement optimization**. *7th International Conference on Information, Intelligence, Systems and Applications (IISA)*.

[Park et al., 2003] Park, Y., et Kim, K. (2003). **A scheduling method for berth and quay cranes**. *OR Spectrum*, 25 :1–23.

[Peterkofsky et al., 1990] Peterkofsky, R., et Daganzo, C. (1990). **A branch and bound solution method for the quay crane scheduling problem**. *Transportation research*, 24 :159–172.

[Salhi et al., 2017] Salhi, A., Alsoufi, G., et Yang, X. (2017). **An evolutionary approach to a combined mixed integer programming model of seaside operations as arise in container ports**. *ADVANCES IN THEORETICAL AND APPLIED COMBINATORIAL OPTIMIZATION*.

[Santini et al., 2015] Santini, A., A, F. H., et S, R. (2015). **A note on a model for quay crane scheduling with non-crossing constraints**. *Engineering Optimization*, 47 :860–865.

[Sciomachen et al., 2007] Sciomachen, A., et Tanfani, E. (2007). **A 3d-bpp approach for optimising stowage plans and terminal productivity**. *European Journal of Operational Research*, 183 :1433–1446.

[Shields, 1984] Shields, J. (1984). **Container stowage : A computer aided preplanning system**. *Marine Technology*, 21.

[Steeken et al., 2004] Steeken, D., et Stahlbock, R. (2004). **Container terminal operation and operations research - classification and literature review**. *OR Spectrum*, 26 :3–49.

[Tang et al., 2014] Tang, L., Zhao, J., et Liu, J. (2014). **Modeling and solution of the joint quay crane and truck scheduling problem**. *European Journal of Operational Research*, 236 :978–990.

[Vahdani et al., 2019] Vahdani, B., Mansour, F., Soltani, M., et Veysmoradi, D. (2019). **Bi-objective optimization for integrating quay crane and internal truck assignment with challenges of trucks sharing**. *Knowledge-Based Systems*, 163 :675–692.

[Vis et al., 2003] Vis, I. F., et Koster, R. D. (2003). **Transshipment of containers at a container terminal : an overview**. *European journal of operational research*, 147.1 :1–16.

[Wang et al., 2009] Wang, S., et Hu, W. (2009). **Multi quay crane scheduling problem based on aco in container terminals**. *2009 International Conference on Management and Service Science*.

[Wang et al., 2015] Wang, Z. X., Chan, F. T. S., Chung, S. H., et Niu, B. (2015). **Minimization of delay and travel time of yard trucks in container terminals using an improved ga with guidance search**. *Mathematical Problems in Engineering, doi.org/10.1155/2015/710565*, 2015.

[Wilson et al., 2000] Wilson, I., et Roach, P. (2000). **Container stowage planning : a methodology for generating computerised solutions**. *Journal of the Operational Research Society*, 51 :1248–1255.

[Xiazhong et al., 2017] Xiazhong, C., Ye, Z., et Hongtao, H. (2017). **Optimization research of joint quay crane scheduling and block selection in container terminals**. *International Conference on Service Systems and Service Management*.

[Xue et al., 2013] Xue, Z., Zhang, C., Miao, L., et Lin, W. (2013). **An ant colony algorithm for yard truck scheduling and yard location assignment problems with precedence constraints**. *Journal of Systems Science and Systems Engineering*, pages 021–037.

[Yi et al., 2012] Yi, D., GuoLong, L., et ChengJi, L. (2012). **Model and heuristic algorithm for quay crane scheduling at container terminal**. *9th International Conference on Fuzzy Systems and Knowledge Discovery*.

[Zhang et al., 2003] Zhang, C., Liu, J., Wan, Y., Murty, K., et Linn, R. (2003). **Storage space allocation in container terminals**. *Transportation Research Part B*, 37 :883–903.

[Zhang et al., 2016] Zhang, X., Zeng, Q., et Yang, Z. (2016). **Modeling the mixed storage strategy for quay crane double cycling in container terminals**. *Transportation Research Part E*, 94 :171–187.

[Zhen et al., 2016] Zhen, L., Yu, S., Wang, S., et Sun, Z. (2016). **Scheduling quay cranes and yard trucks for unloading operations in container ports**. *Springer Science+Business Media New York*.

TABLE DES FIGURES

LISTE DES TABLES

I

ANNEXES

7

ANNEXE DU TROISIÈME CHAPITRE

Dans ce chapitre, nous proposons des exemples numériques qui illustrent la démarche du modèle linéaire, l'algorithme numératif et l'algorithme génétique, proposés au troisième chapitre.

Commençons par l'exemple ci-dessous illustrant le modèle linéaire codé par CPLEX.

Q = 2

B = 4

C_j = (13,10,12,11)

T_c = 1.17

t_j = (15.21,26.91,26.91,12.87)

$$X = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

$$Y = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

C_{max} = 26.91

Ci-dessous un exemple de l'affichage de l'algorithme numératif codé par JAVA, pour 2 grues de quai et 4 baies avec les même valeurs précédentes dans le modèle linéaire. La première figure montre comment l'algorithme génère tous les choix possibles pour l'affectation des grues aux baies.

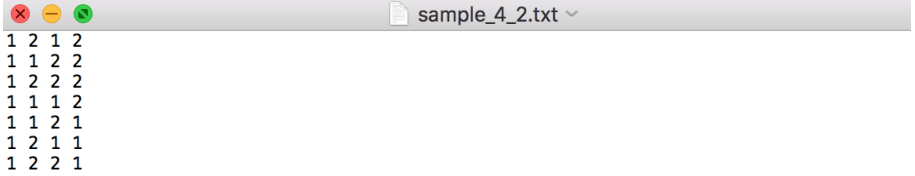

FIGURE 7.1 – Toutes les affectations possibles

La deuxième figure montre les résultats obtenus pour les affectations précédentes (ce n'est pas le même exemple présenté dans le chapitre 3), et enfin la sélection de la solution optimale avec le temps d'exécution.

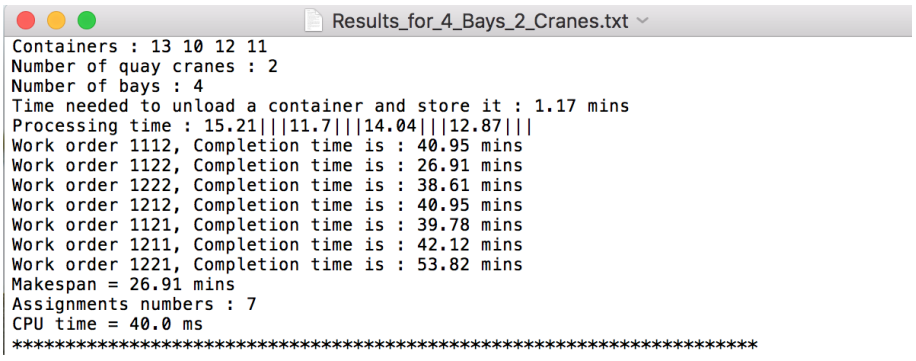

FIGURE 7.2 – Résultat de l'algorithme numératif

Enfin, la figure ci-dessous montre le résultat obtenu par l'algorithme génétique.

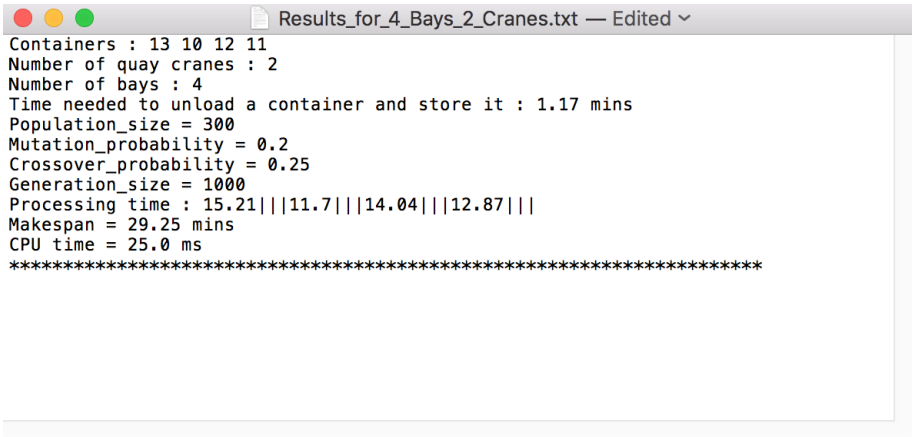

FIGURE 7.3 – Résultat de l'algorithme génétique

Ci-dessous les paramètres et les résultats de calcul pour les ensembles de référence QCSP qui ont été introduits dans le document de [MEISEL et al., 2011]. Les tableaux 1 à 7 indiquent les temps de manutention des navires pour toutes les instances. Les navires de petite taille avec 10 baies et une capacité correspondante de 200 conteneurs sont desservis par 2 grues ; les navires de taille moyenne avec 15 baies et une capacité correspondante de 400 conteneurs sont desservis par 4 grues ; les navires de grande taille avec 20 baies et une capacité correspondante de 600 conteneurs sont desservis par 6 grues. Les solutions fournies ont été générées par l'heuristique proposée.

Files created using the ‚plain' file format of *QCSPgen* have the following structure:

- The first line contains the following seven comma-separated integer values:
  - number of tasks n
  - number of bays b
  - number of precedence relations in set Φ
  - number of non-simultaneous relations in set Ψ
  - number of cranes q
  - travel time t
  - safety margin s
- The second line contains n integer values indicating the processing times p_i of tasks $i=1,2,...n$.
- The third line contains n integer values indicating the bay locations l_i of tasks $i=1,2,...n$.
- The fourth line contains q integer values indicating the ready times r^k of cranes $k=1,2,...q$.
- The fifth line contains q integer values indicating the initial bay positions l_0^k of cranes $k=1,2,...q$.
- The sixth line contains the text 'Phi:'
- Each of the subsequent $|\Phi|$ lines contains one task pair i,j with a precedence relation in set Φ.
- The next line contains the text 'Psi:'
- Each of the subsequent $|\Psi|$ lines contains one task pair i,j with a non-simultaneity relation in set Ψ.
- The file ends with text 'end.' within the last line.

Benchmark results for the Quay Crane Scheduling Problem (QCSP)

This document provides computational results for the QCSP benchmark sets that have been introduced in the paper 'A unified approach for the evaluation of quay crane scheduling models and algorithms' by Frank Meisel and Christian Bierwirth (submitted for publication).

Tables 1 to 7 report vessel handling times (VHTs) for the instances in sets A to G. These solutions have been generated by the heuristic proposed in Bierwirth and Meisel: *A fast heuristic for quay crane scheduling with interference constraints*, J. Sched 12, 2009, p. 345-360.

Table 1: VHTs observed for small sized vessels (set A).

no.	$n = 10$	$n = 15$	$n = 20$	$n = 25$	$n = 30$	$n = 35$	$n = 40$
1	520	514	508	508	506	506	506
2	508	507	509	507	508	507	506
3	513	515	509	507	507	506	505
4	510	513	509	507	507	507	507
5	515	507	506	507	506	507	506
6	513	508	508	507	506	511	507
7	511	507	507	508	508	507	507
8	513	508	510	507	508	506	506
9	512	507	508	506	506	506	506
10	549	513	507	506	506	508	507
$CVHT$	5 164	5 099	5 081	5 070	5 068	5 071	5 063

Table 2: VHTs observed for medium sized vessels (set B).

no.	$n = 45$	$n = 50$	$n = 55$	$n = 60$	$n = 65$	$n = 70$
1	758	774	758	781	758	766
2	759	771	783	756	799	764
3	759	772	779	758	803	760
4	789	765	759	765	758	760
5	758	762	758	760	758	757
6	789	765	789	758	757	761
7	798	782	768	786	757	759
8	759	761	767	757	756	758
9	797	798	801	785	758	757
10	792	759	757	805	786	779
$CVHT$	7 758	7 709	7 719	7 711	7 690	7 621
ACT	713	1 251	1 078	1 317	2 118	2 231

Table 3: VHTs observed for large sized vessels (set C).

no.	$n = 75$	$n = 80$	$n = 85$	$n = 90$	$n = 95$	$n = 100$
1	1 178	1 173	1 049	1 014	1 174	1 014
2	1 011	1 023	1 017	1 020	1 090	1 104
3	1 182	1 013	1 027	1 011	1 014	1 107
4	1 107	1 202	1 186	1 063	1 138	1 202
5	1 192	1 036	1 082	1 062	1 144	1 015
6	1 123	1 117	1 010	1 193	1 055	1 136
7	1 200	1 201	1 195	1 108	1 173	1 098
8	1 174	1 040	1 105	1 094	1 015	1 151
9	1 074	1 192	1 010	1 075	1 019	1 023
10	1 188	1 207	1 166	1 049	1 011	1 015
$CVHT$	11 429	11 204	10 847	10 689	10 833	10 865
ACT	$\top$	3 248	$\top$	$\top$	$\top$	$\top$

Table 4: VHTs observed under various spatial distributions of tasks (set D).

no.	$f = 0.2$			$f = 0.8$		
	$Loc = cl1$	$Loc = cl2$	$Loc = uni$	$Loc = cl1$	$Loc = cl2$	$Loc = uni$
1	544	453	415	1 214	1 207	1 207
2	556	430	307	1 206	1 208	1 209
3	680	439	426	1 222	1 211	1 216
4	578	312	324	1 221	1 209	1 210
5	356	349	309	1 210	1 210	1 207
6	414	307	307	1 213	1 212	1 207
7	439	373	325	1 217	1 211	1 208
8	383	308	349	1 213	1 208	1 208
9	420	308	387	1 209	1 207	1 208
10	380	397	346	1 212	1 208	1 210
$CVHT$	4 750	3 676	3 495	12 137	12 091	12 090
ACT	3 240	2 212	2 100	7	508	430

Table 5: VHTs observed under various precedence densities (set E).

no.	$d = 0.80$	$d = 0.85$	$d = 0.90$	$d = 0.95$	$d = 1.00$
1	774	774	774	774	774
2	771	771	771	771	771
3	772	772	772	772	772
4	758	761	762	762	765
5	761	761	761	761	762
6	757	757	757	757	765
7	782	782	782	782	782
8	758	758	761	761	761
9	798	798	798	798	798
10	759	759	759	759	759
$CVHT$	7 690	7 693	7 697	7 697	7 709
ACT	3 306	1 958	870	481	41

Table 6: VHTs observed under various numbers of serving cranes (set F).

no.	$q = 2$	$q = 3$	$q = 4$	$q = 5$	$q = 6$
1	1 509	1 007	774	730	730
2	1 510	1 008	771	768	768
3	1 510	1 008	772	770	769
4	1 510	1 009	765	748	746
5	1 509	1 007	762	732	732
6	1 509	1 008	765	714	714
7	1 511	1 009	782	780	779
8	1 509	1 008	761	650	643
9	1 510	1 012	798	797	797
10	1 510	1 008	759	684	683
$CVHT$	15 097	10 084	7 709	7 373	7 361
ACT	11	199	1248	2892	$\top$

Table 7: VHTs observed under various safety requirements (set G).

no.	$s = 0$	$s = 1$	$s = 2$	$s = 3$	$s = 4$
1	757	774	1 059	1 381	1 501
2	759	771	950	1 288	1 328
3	759	772	976	1 098	1 275
4	759	765	1 104	1 365	1 443
5	758	762	833	1 040	1 327
6	758	765	1 031	1 262	1 622
7	760	782	1 042	1 431	1 448
8	757	761	954	1 092	1 345
9	758	798	1 075	1 252	1 328
10	759	759	930	1 190	1 437
$CVHT$	7 584	7 709	9 954	12 399	14 054
ACT	2 440	1 248	3 252	$\top$	$\top$

Tables 8 to 10 report lower bounds for the minimum vessel handling times of instances in sets A to C. These lower bounds have been delivered by ILOG CPLEX 11 within a limited runtime of two hours per instance. A '*' attached to a value indicates that an optimal solution with corresponding vessel handling time has been found. CPLEX struggles if applied to the larger instances leading to similar lower bounds for most instances within set B and within set C.

Table 8: Lower bounds on VHTs for instances in set A.

no.	$n = 10$	$n = 15$	$n = 20$	$n = 25$	$n = 30$	$n = 35$	$n = 40$
1	*520	*513	506	506	505	*506	505
2	*508	*507	507	*507	507	506	*506
3	*513	*513	507	506	506	*506	*505
4	*510	*513	507	*507	506	506	506
5	*514	*507	*506	506	*506	506	*506
6	*513	*508	507	506	505	505	505
7	*511	*507	506	506	507	506	*507
8	*513	*508	508	*507	506	505	*506
9	*512	*507	507	*506	*506	505	*506
10	*549	*513	*507	*506	*506	506	506

Table 9: Lower bounds on VHTs for instances in set B.

no.	$n = 45$	$n = 50$	$n = 55$	$n = 60$	$n = 65$	$n = 70$
1	755	753	754	753	753	753
2	754	754	754	753	753	753
3	754	754	753	753	753	753
4	755	753	753	754	753	753
5	754	753	754	753	753	753
6	754	753	754	753	754	753
7	754	754	754	753	754	754
8	755	753	753	753	754	753
9	754	754	753	753	754	753
10	754	753	754	753	754	753

Table 10: Lower bounds on VHTs for instances in set C.

no.	$n = 75$	$n = 80$	$n = 85$	$n = 90$	$n = 95$	$n = 100$
1	1003	1003	1004	1003	1003	1003
2	1003	1003	1003	1003	1003	1003
3	1003	1003	1003	1003	1003	1003
4	1003	1003	1003	1003	1003	1003
5	1003	1002	1003	1003	1003	1003
6	1003	1004	1003	1003	1003	1003
7	1003	1002	1003	1003	1003	1003
8	1003	1003	1003	1003	1003	1003
9	1003	1004	1003	1003	1003	1003
10	1003	1003	1003	1003	1003	1003

8

ANNEXE DU QUATRIÈME CHAPITRE

Dans ce chapitre, nous proposons des exemples numériques qui illustrent la démarche du modèle linéaire, de l'algorithme numératif et de l'algorithme génétique, ainsi que les bornes inférieures et les bornes supérieures proposées dans le quatrième chapitre.

Commençons par l'exemple ci-dessous illustrant le modèle linéaire codé par CPLEX.

C = 4

T = 1

w = (5,4,5,6)

$$t = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 5 \\ 3 & 0 & 4 & 5 \\ 3 & 3 & 0 & 5 \\ 3 & 3 & 4 & 0 \end{pmatrix}$$

λ = (7,7,7,7)

r = 5

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

B = (1,1,1,1)

c = (20,39,58,77)

cp = (32,51,70,89)

s = (15,35,53,71)

s' = (15,20,39,58,77)

$$X = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & 5 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 3 & 1 & 0 \\ 1 & 4 & 1 & 0 \\ 1 & 5 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 2 & 2 & 1 & 0 \\ 2 & 3 & 1 & 1 \\ 2 & 4 & 1 & 0 \\ 2 & 5 & 1 & 0 \\ 3 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \\ 3 & 3 & 1 & 0 \\ 3 & 4 & 1 & 1 \\ 3 & 5 & 1 & 0 \\ 4 & 1 & 1 & 1 \\ 4 & 2 & 1 & 1 \\ 4 & 3 & 1 & 0 \\ 4 & 4 & 1 & 0 \\ 4 & 5 & 1 & 1 \end{pmatrix}$$

C_{max} = 89

Ci-dessous un exemple de l'affichage de l'algorithme numératif codé pas JAVA, pour 1 U-conteneur, 1 L-conteneur et 1 camion de transport avec les même valeurs suovan :

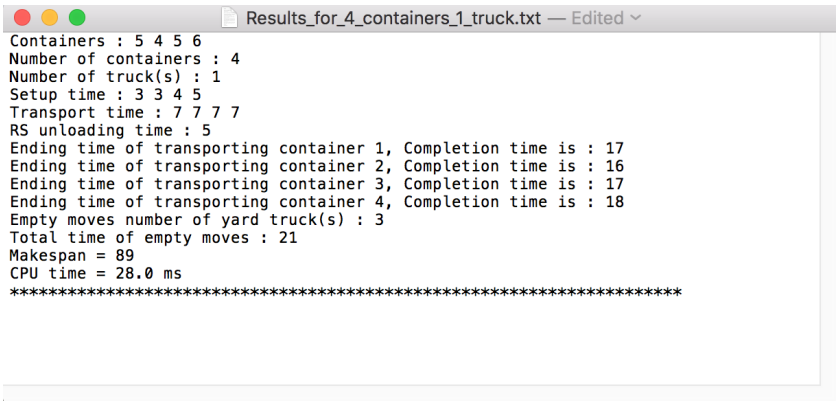

FIGURE 8.1 – Résultat de l'algorithme numératif

La deuxième figure démontre le résultat obtenu pour les bornes inférieures et supérieures.

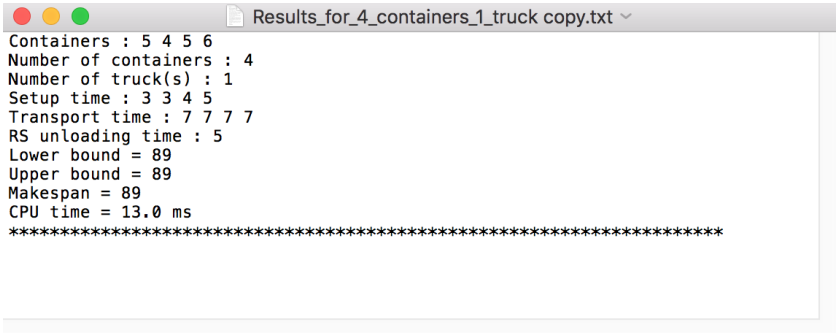

FIGURE 8.2 – Résultat des bornes inférieures et supérieures

Enfin, la figure ci-dessous démontre le résultat obtenu par l'algorithme génétique.

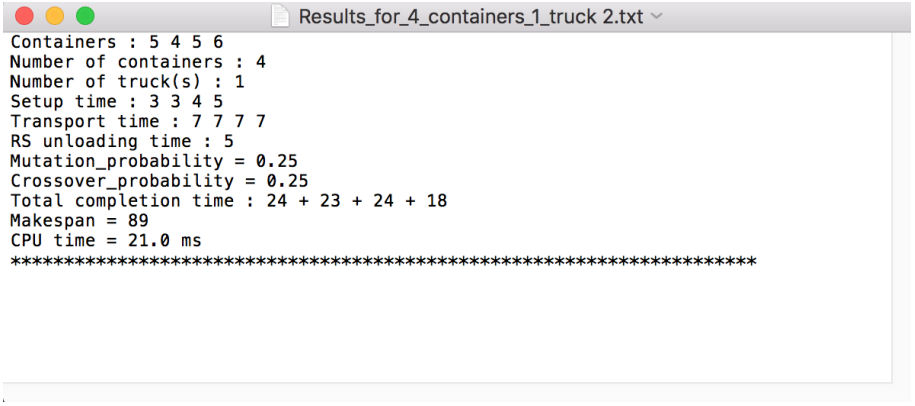

FIGURE 8.3 – Résultat de l'algorithme génétique

9

ANNEXE DU CINQUIÈME CHAPITRE

Dans ce chapitre, nous proposons un exemple numérique qui illustre la démarche du modèle linéaire codé par CPLEX et proposé dans le quatrième chapitre, pour 2 U-conteneurs, 2 L-conteneurs et 1 camion de transport.

C = C' = 2

Q = Q' = 1

T = 1

p = p' = (1,2)

v = 12

v' = (10,11)

v''' = (12,13)

$\lambda_{ij} = \begin{pmatrix} 5 & 0 \\ 0 & 6 \end{pmatrix}$

d_i = (10,11)

d'_i = (13,14)

rs = 10

rs' = 10

s_0 = 2

u = 5

E = (10,26)

E' = (126,145)

HA''' = (69,131)

HA' = (30,90)

HA'' = (69,131)

$$X = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 3 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 3 & 0 \\ 2 & 1 & 1 & 0 \\ 2 & 1 & 2 & 0 \\ 2 & 1 & 3 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$Y = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 3 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 3 & 0 \\ 2 & 1 & 1 & 0 \\ 2 & 1 & 2 & 0 \\ 2 & 1 & 3 & 1 \end{pmatrix}$$

$$Z = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 3 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 3 & 0 \\ 2 & 1 & 1 & 0 \\ 2 & 1 & 2 & 0 \\ 2 & 1 & 3 & 1 \end{pmatrix}$$

$$W = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$W' = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$C_{max} = 145$

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — http://www.multiagent.fr/ThesisStyle
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/