Laboratoire d'Informatique de l'X - École Polytechnique
École doctorale de l'X

THÈSE

présentée par Antoine JEANJEAN

pour l'obtention du grade de

Docteur de l'École Polytechnique
Spécialité : Informatique

Recherche locale pour l'optimisation en variables mixtes : méthodologie et applications industrielles

---

Soutenue publiquement le 10 octobre 2011 devant le jury composé de Messieurs :

Philippe BAPTISTE	Directeur de recherche, CNRS LIX (Directeur de Thèse)
Thierry BENOIST	Ingénieur, Bouygues e-lab (Co-directeur de Thèse)
Jacques CARLIER	Professeur, Université Technologique de Compiègne (Rapporteur)
Frédéric GARDI	Ingénieur, Bouygues e-lab (Co-directeur de Thèse)
Léo LIBERTI	Professeur chargé de cours, École Polytechnique (Examinateur)
Alain QUILLIOT	Professeur, ISIMA, Université Clermont II (Examinateur)
Francis SOURD	Ingénieur (HDR), SNCF Innovation & Recherche (Rapporteur)
Michel VASQUEZ	Enseignant-chercheur (HDR), École des Mines d'Alès (Rapporteur)

Quo semel est inbuta recens servabit odorem / testa diu [1]

À la mémoire de Marjep.

L'élève dit à ses maîtres que le temps s'écoulera toujours trop vite pour tout ce qui lui reste à apprendre.

Denis Diderot (1713-1784)

1. *Lorsqu'une amphore neuve est imprégnée d'une odeur, elle la conserve très longtemps.* Horace (*Epodes*).

Remerciements

Je tiens à remercier tout d'abord Monsieur Jacques Carlier, Professeur de Recherche Opérationnelle à l'Université Technologique de Compiègne, Monsieur Francis Sourd, ingénieur de recherche et directeur de l'équipe de Recherche Opérationnelle de la SNCF et Monsieur Michel Vasquez, Enseignant Chercheur au LGI2, qui m'ont fait l'honneur d'être les rapporteurs de cette thèse. Ensuite, je remercie Monsieur Léo Liberti, Professeur Chargé de Cours et Directeur du laboratoire d'Informatique de l'X et Monsieur Alain Quilliot, Professeur à l'ISIMA, école d'ingénieur dont je suis diplômé, d'avoir accepté d'assister en qualité d'examinateurs à la soutenance de cette thèse. J'ai eu la chance de rencontrer Philippe Baptiste qui a toujours gardé un lien amical avec l'équipe du Bouygues e-lab où il avait lui même fait son doctorat. Ma thèse a débuté alors qu'il était encore directeur du laboratoire d'informatique de l'X et j'ai beaucoup apprécié les échanges que nous avons pu avoir au cours de ce doctorat, malgré l'emploi du temps densifié par ses nouvelles fonctions de directeur de l'Institut français d'Informatique au CNRS. Je le remercie d'avoir accepté d'être le directeur de cette thèse.

C'est Thierry Benoist qui m'a proposé de réfléchir quelques jours avant les vacances de Noël 2007 au scénario d'un doctorat avec pour objectif premier de structurer mon activité de R&D au sein du Bouygues e-lab. Nous savions que la tâche n'allait pas être aisée, car mener un travail d'ingénieur et une thèse en parallèle allait nécessiter beaucoup d'efforts, mais je tiens à remercier Thierry de m'avoir motivé pour m'entraîner dans cette formidable expérience de travail et de recherche, qui m'a transformé à jamais. J'ai beaucoup appris au cours de ce doctorat, ainsi que pendant les six années passées au sein de l'équipe du e-lab. Je tiens à remercier d'ailleurs Benoît Rottembourg et Etienne Gaudin d'avoir cru en ma candidature envoyée depuis les Etats-Unis alors que j'étais en master, ainsi que pour les différentes expériences de travail que j'ai été amené à mener avec eux. J'ai trouvé au sein de l'équipe optimisation et aide à la décision du Bouygues e-lab non seulement une méthodologie, une rigueur et une passion pour le travail accompli, mais aussi une ambiance chaleureuse propice à un épanouissement personnel. Je remercie sincèrement Thierry d'avoir cru en mes capacités, de m'avoir confié des projets de conseil et de développement passionnants et variés, toujours accompagnés de défis, tout en ayant réussi à me transmettre sa passion pour les mathématiques et la RO. Un clin d'oeil aux parties de squash qui auront permis d'éliminer nos (trop) nombreuses tartes au citron ingérées !

En 2007, l'arrivée de Frédéric Gardi au sein de l'équipe du e-lab a contribué au dynamisme de l'équipe. Je tiens à remercier chaleureusement Frédéric pour les nombreuses discussions que nous avons eues concernant la vision de la RO (80-20 !), les sciences en général, la gestion de projet, l'engagement dans le travail, la passion ainsi que sur l'enseignement. Je le remercie d'avoir accompagné Thierry dans cette difficile tâche d'encadrement d'un thésard, surtout dans ce contexte. Merci pour ces conseils avisés, pour cet apprentissage de la rigueur dans le développement de logiciel de production, merci pour les nombreuses relectures d'articles, de présentations et autres résumés divers et variés toujours ponctués de remarques très pertinentes.

Parmi l'équipe du e-lab, je tiens à remercier particulièrement Guillaume Rochart pour son aide précieuse en développement, pour l'ambiance de travail dans les divers projets menés conjointement ainsi que pour les nombreuses heures passées à discuter au labo ou en soirée. J'adresse bien plus qu'un merci pour notre assistante Catherine Bernez qui aura pendant toute ces années été beaucoup plus qu'une assistante d'équipe, et qui sera devenue au fil des ans une amie sincère. Merci Catherine pour ces milles services que tu m'as rendu et profite bien de cette retraite plus que méritée ! Je remercie aussi Philippe Caillol pour ces belles discussions sur la créativité et l'innovation ainsi que pour sa passion communicative pour les voyages et les découvertes diverses et variées. Plus généralement, je remercie toute l'équipe du e-lab et les nombreux stagiaires pour cette belle ambiance de travail. Le e-lab est devenu au fil des ans bien plus qu'une équipe à mes yeux ! Je remercie Alain Pouyat et plus généralement l'entreprise Bouygues d'avoir accepté de soutenir ce projet de thèse auquel j'aurais au final consacré un tiers de mon temps de travail ces quarante derniers mois. J'en profite pour remercier les formidables personnes que j'ai croisées dans le groupe Bouygues, aussi bien à la maison mère, que chez Bouygues Construction et TF1, filiales avec lesquelles j'ai été amené à travailler. J'y ai été en contact avec des collaborateurs partageant une grande rigueur et une impressionnante force de travail.

Merci à l'équipe du L.I.X. et celle de l'école doctorale de l'X de m'avoir accueilli. Je remercie le binôme Vincent Jost et David Savourey pour leur bonne humeur, leur état d'esprit et pour nos échanges de qualité. Un grand merci à David qui a mené une relecture détaillée de mon manuscrit.

Enfin, une thèse est aussi une aventure humaine menée en dehors de heures de travail. Je tiens donc à remercier ma famille ainsi que tous les amis qui ont su être là dans les moments de doutes. Un grand merci tout d'abord à mon père qui a soutenu ce projet (et bien d'autres !) depuis le début et qui a trouvé les mots pour toujours me motiver, comme tout au long de ma scolarité. Il a excellé ici dans le rôle du coach de cette épreuve de fond. Je remercie aussi ma chère maman pour ces échanges toujours personnels et sincères qui m'ont permis de ne jamais perdre la direction à suivre. Merci à mes deux adorables soeurs (et à leur cher conjoint !) pour leur bonne humeur, leur conseil et leur amour qui font de cette belle famille un univers propice à un épanouissement personnel. Merci à la petite Louise pour cette collection de sourires distribués tout au long de cette dernière année, qui auront

été autant de bouffées d'oxygène. Une chère pensée pour mon oncle Michel disparu trop tôt. Merci à mon ami Loïc d'avoir été toujours là, en toute circonstance, en tout lieu et à toute heure, longue vie à notre amitié ! Merci à ma chouchou pour ces belles discussions sur la vie. Merci à petit Masm pour nos nombreux échanges électroniques depuis Tahiti, avec des ondes positives qui ont su traverser la planète. Merci à Romain pour les beaux débats sur la recherche française, à Jean-Noël pour nos déjeuners 100 000 volts, à mes petites chéries Neigeou & Isa, merci à Julio, à Aurel, à Tahit' et à la troupe de l'ISIMA avec laquelle je suis toujours aussi content de passer de très bons moments et merci à mes vieilles (mais toujours aussi chères) amitiés bretonnes ! Merci aussi à toute la MonExTeam et notamment à mes 3 associés Pierre-Etienne, Cédric et Benoît, qui auront su faire preuve de patience et de compréhension et que je rejoins avec une formidable impatience et motivation dans cette nouvelle aventure du développement durable.

Et enfin, un remerciement spécial revient à Manon qui m'a donné envie de mordre la vie à pleines dents et qui a décuplé ma force de travail. Merci pour tes nombreuses relectures d'un texte qui t'a paru parfois un peu dense, merci pour tous ces moments magiques et merci d'avoir cru en moi. A cette belle vie sans thèse qui nous attend ! Et viva Elvis !

À Paris, le 31 Octobre 2011

A. J.

Résumé

De nombreux problèmes industriels comportent à la fois des dimensions combinatoire et continue. Lorsqu'ils sont de grande taille, ces problèmes mixtes sont souvent résolus par décomposition en sous-problèmes. Les décompositions les plus fréquentes reposent sur les spécificités "métier" du problème. La programmation mathématique fournit également des méthodes formelles de décomposition. Toutefois, ces approches ont des inconvénients en pratique : difficulté de garantir la qualité voire l'admissibilité des solutions, complexité technique de mise en œuvre et complexité logicielle en cas d'utilisation de solveurs de programmation mathématique. Dans cette thèse, nous proposons une approche directe, par recherche locale, des problèmes d'optimisation en variables mixtes. Notre méthodologie se focalise sur deux points : la définition d'une large variété de mouvements assurant une diversification combinatoire et une évaluation incrémentale fondée sur des algorithmes approchés mais très efficaces en temps, traitant simultanément les dimensions combinatoire et continue du problème.

Après une introduction détaillant cette méthodologie, nous en présentons une première application dans un cadre purement combinatoire : l'optimisation de la gestion de stock de banches sur les chantiers de construction (Bouygues Habitat Résidentiel). Puis, nous l'appliquons sur un problème d'ordonnancement de mouvements de terre pour le terrassement d'autoroutes et de voies ferrées (DTP Terrassement). Il s'agit de planifier sur un horizon de plusieurs années des tâches avec précédences à l'aide de ressources contraintes par des fenêtres de temps. Ce problème est résolu par recherche locale, une de ses caractéristiques étant que les quantités de terre déplacées peuvent passer d'une tâche à une autre tandis que les dates d'affectation de celles-ci sont simultanément modifiées. Le logiciel, aujourd'hui en exploitation, permet la planification de grands chantiers linéaires en quelques minutes. Nous étudions également le cas à une ressource correspondant à un problème de voyageur de commerce linéaire avec précédences. Ce dernier est prouvé NP-complet ; nous proposons alors un algorithme par programmation dynamique, exponentiel mais efficace en pratique. Les solutions obtenues nous ont permis d'apprécier l'effectivité de la recherche locale.

Enfin, nous traitons un problème de tournées de véhicules avec gestion des stocks, posé par un client industriel du Groupe Bouygues. Ce problème consiste en l'optimisation des coûts de distribution d'un produit fluide par camion, sur des zones géographiques comptant plusieurs centaines de clients dont la gestion des stocks est confiée au fournisseur. Une heuristique de recherche locale est proposée pour résoudre le problème à court terme (15 jours) en

quelques minutes. Afin d'assurer une optimisation sur le long terme, nous introduisons une fonction objectif de remplacement fondée sur un calcul de bornes inférieures. L'originalité de notre approche réside dans le fait que les quantités livrées au cours des tournées peuvent varier alors même que ces dernières sont modifiées géographiquement ou temporellement. Nous nous appuyons également sur un algorithme approché de réaffectation des volumes, un millier de fois plus rapide en pratique qu'un algorithme exact de flot maximum. L'exploitation de ce logiciel, quotidienne et à l'échelle mondiale, a permis une réduction des coûts logistiques de l'ordre de 15 %.

Titre : Recherche locale pour l'optimisation en variables mixtes : méthodologie et applications industrielles.

Mots clés : Recherche locale, optimisation en variables mixtes, ordonnancement de tâches, tournées de véhicules avec gestion des stocks.

Abstract

Many industrial problems today combine continuous and combinatorial dimensions. These mixed variable optimization problems are often solved by decomposing them into sub-problems, especially when they are large. Most of the common decomposition techniques are based on practical specificities of the problem. Mathematical programming also provides formal decomposition methods. However, these decomposition approaches have practical drawbacks : difficulties in guaranteeing the quality or even feasible solutions, high technical complexities of development projects, and high software complexities while using mathematical programming solvers. In this thesis, we propose a direct approach for solving these mixed variable optimization problems using a local search method. Our methodology focuses on two points : the definition of a large pool of varied moves to ensure combinatorial diversification and an incremental evaluation based on approximate, yet time efficient, algorithms. The combinatorial and continuous dimensions are addressed simultaneously.

Following the introduction of our methodology, Chapter 2 presents the application in a pure combinatorial context, the minimization of formwork stocks on construction sites (Bouygues Habitat Residentiel). In Chapter 3, we rely on this methodology to optimize earthwork scheduling for highway and railway projects (DTP Terrassement). The schedules count hundreds of earthmovings planned over a horizon of several years, using resources constrained by time windows. This problem is solved by a local search heuristic with a special feature : the amounts of soil displaced can move from one task to another, while the assignment dates of these tasks are also changed simultaneously. Within minutes, the algorithm computes an earthwork plan in such real-life projects. We also investigate a single-resource case corresponding to a line traveling salesman problem with precedences, proved here as NP-hard. Then, we introduce a dynamic programming algorithm which is exponential, but effective in practice. The analysis of this single-resource case helps to assess the effectiveness of the local search.

Finally, we consider a inventory routing problem, posed by the industrial customer group, Bouygues. Inventory routing refers to the optimization of transportation costs for the replenishment of customers' inventories : based on consumption forecasts, the vendor organizes delivery routes. A local search heuristic is able to provide a proposed solution the problem for the short term (15 days) in minutes. In order to ensure savings in the long term, we introduce a new surrogate objective function based on long-term lower bounds. The approach is

original in that the quantities delivered during tours can vary even when they are geographically or temporally altered. We also use a volume allocation algorithm which is a thousand times faster in practice than an exact maximum flow algorithm. Confirming the promised gains in operations (15 % on average), the resulting decision support system is progressively deployed worldwide.

Title : Local search for mixed-integer optimization : methodology and industrial applications.

Key words : Local search, mixed-variable optimization, task scheduling, inventory routing.

Table des matières

Table des figures

Avant-propos

Cette thèse s'inscrit dans la continuité des travaux de recherche opérationnelle débutés en Master à l'Université d'Oklahoma (USA) avec Suleyman Karabuk [63] et poursuivis ces dernières années au sein du Bouygues e-lab dans l'équipe de Thierry Benoist. Mes travaux de Master concernait l'optimisation de tournées de camions à partir de jeux de données réelles provenant d'une industrie textile de Caroline du Nord. Diverses heuristiques à base de décomposition du type *Branch and Price*, c'est-à-dire des algorithmes de génération de colonnes intégrées à un algorithme de *Branch and Bound*, ont été testées afin de traiter des instances de grandes tailles et de trouver des coupes améliorant la qualité des résultats (graphes avec plusieurs centaines de points à visiter).

De par leur échelle et les enjeux économiques sous-jacents, les problèmes d'optimisation que j'ai abordés au sein du groupe Bouygues [2] dans l'équipe du e-lab [3] ont offert un terrain propice à la recherche autour des techniques d'optimisation. Depuis 2005, mon travail d'ingénieur au sein du Bouygues e-lab a essentiellement été focalisé sur la conception d'algorithmes, leur implémentation au sein de systèmes d'information et la réalisation d'études statistiques. Une des principales missions menées ces dernières années au e-lab a concerné l'optimisation de la planification de la publicité sur les chaînes du groupe TF1. Il s'agissait non seulement de mener des études statistiques afin de comprendre par exemple le comportement d'annulation des annonceurs sur la chaîne TF1 [17] mais aussi de faire évoluer le moteur de planification des publicités pour l'antenne : évolutions commerciales (vente en package, offre promotionnelle), thématiques (coupe du monde de foot, de rugby, émission) ou fonctionnelles. Ce logiciel est utilisé à chaque ouverture de planning (tous les deux mois) et d'autres logiciels servent à l'optimisation du planning entre cette première planification et la diffusion à l'antenne du programme et de la publicité. Sur internet, les missions réalisées ont concerné l'écriture d'un algorithme de bandits manchots pour l'affichage de publicités [64] et la prévision des audiences web à l'aide d'algorithmes fondés sur des indicatrices, de

2. Bouygues est un groupe industriel diversifié, structuré par une forte culture d'entreprise et dont les métiers s'organisent autour de deux pôles : la Construction avec Bouygues Construction (BTP et Energies & Services), Bouygues Immobilier et Colas (Routes), et les Télécoms-Médias avec TF1 et Bouygues Telecom.

3. En support de l'innovation dans le Groupe, Bouygues SA s'appuie sur une équipe de recherche et développement, le e-lab, pour développer différents services aux métiers. Les prestations du e-lab proposées aux métiers s'exercent dans deux domaines privilégiés : l'aide à la décision afin d'améliorer l'efficacité des processus complexes et les nouvelles technologies afin de concevoir des services et produits innovants.

l'apprentissage et des historiques d'événements influençant les audiences [69, 66]. Une étude sur la pression des plannings radio (Réseau "Les Indépendants" en régie par TF1 Publicité) a permis de créer un outil analysant les congestions de planning et d'anticiper les évolutions de remplissage, tout en maintenant une politique tarifaire cohérente. Ces projets d'analyse de données ont donné lieu à une autre collaboration avec le groupe hôtelier Louvre Hôtels. L'idée était d'appliquer les techniques de prévisions pour les fréquentations et le chiffre d'affaires des hôtels de ce groupe. L'outil permet désormais de prévoir les fréquentations de plus de 500 hôtels et d'aider les gestionnaires dans la définition des prix et objectifs communiqués aux hôtels [70].

Une autre thématique de recherche sur laquelle j'ai été amené à travailler au cours de ces six années passées au Bouygues e-lab, a été l'optimisation dans le domaine de la construction. Par exemple, j'ai été amené à travailler pour ETDE, filiale de Bouygues Construction, sur un outil d'aide à la décision pour la réponse à appel d'offre dans le domaine de l'éclairage public. Le simulateur optimise le planning des interventions sur chaque site et en déduit un bilan financier global sur la durée du contrat [85]. Un autre exemple de projet dans le domaine de la construction est le moteur pour l'optimisation des plannings de sous-traitants sur des chantiers résidentiels [14]. Ce moteur de calcul a pour objectif d'améliorer la replanification hebdomadaire de chantiers de construction suite aux relevés d'avancement. Des contraintes de précédences entre la cinquantaine d'activités réalisées sur des chantiers d'une centaine de zones faisaient naître un problème de planification d'activité quotidienne sur des horizons de temps de plusieurs mois. Une approche fondée sur la programmation par contraintes a permis de construire un logiciel, testé sur chantiers, pour optimiser les interventions de sous-traitance.

Cette thèse se focalise sur 3 problèmes traités pour le groupe Bouygues ou pour un client externe. Il s'agit de la minimisation du stock de coffrages (Bouygues Habitat Résidentiel), que nous détaillerons dans le chapitre 2, de la planification des mouvement de terre sur des chantiers linéaires (DTP Terrassement) que nous présenterons dans le chapitre 3, et enfin de l'optimisation de tournées de véhicules avec gestion de stock, présenté en chapitre 4. Nous invitons le lecteur intéressé par les autres missions évoquées dans cet avant propos à se reporter aux différentes publications dont elles ont fait l'objet.

Chapitre 1

Introduction

1.1 Principales contributions

De nombreux problèmes industriels d'optimisation comportent à la fois des dimensions combinatoire et continue. Les aspects combinatoires peuvent concerner par exemple l'affectation de ressources à des activités ou l'affectation d'une activité à une tournée. Les aspects continus peuvent être quant à eux relatifs à des volumes distribués, à des quantités de matériaux déplacés ou encore à des dates de visites fixées sur un horizon temporel continu. Pour aborder ces problèmes en variables mixtes, une approche de résolution couramment employée est la décomposition en sous-problèmes, notamment lorsqu'ils sont de grande taille. Par exemple, dans le cas d'un problème de tournées de véhicules avec gestion de stock, la décomposition usuelle consiste à d'abord évaluer les quantités à livrer à chaque client (dimension continue) puis à construire des tournées pour satisfaire celles-ci (dimension combinatoire). Dans le problème de planification des arrêts de centrales nucléaires [98], une décomposition naturelle consiste d'abord à positionner les arrêts (dimension combinatoire) puis à déterminer un plan de production satisfaisant la demande (dimension continue). La programmation mathématique fournit également des méthodes formelles de décomposition (Dantzig-Wolfe, Benders, *Branch-Cut-Price*). Toutefois, ces approches peuvent avoir des inconvénients en pratique : difficulté de garantir la qualité voire l'admissibilité des solutions, complexité technique de mise en œuvre, complexité logicielle en cas d'utilisation de solveurs de programmation mathématique.

Dans cette thèse, nous proposons une *approche directe*, par recherche locale, pour l'optimisation en variables mixtes. Notre méthodologie se focalise sur deux points. Tout d'abord, nous définissons une *large variété de mouvements randomisés* travaillant simultanément sur les dimensions combinatoire et continue du problème, afin d'assurer une exploration diversifiée de l'espace de recherche. Ensuite, nous accélérons l'évaluation de ces mouvements en nous appuyant sur une *algorithmique incrémentale et approchée*. Cette méthodologie est éprouvée sur deux problèmes industriels rencontrés au Bouygues e-lab : l'ordonnancement de

mouvements de terre pour le terrassement d'autoroutes et de voies ferrées et l'optimisation de tournées de véhicules avec gestion des stocks. Les logiciels résultants de ces travaux sont aujourd'hui en exploitation, permettant des réductions de coûts significatives. Après un bref rappel de la définition d'un problème d'optimisation en variables mixtes, nous présentons les méthodes de décomposition généralement utilisées pour leur résolution ainsi que leurs limites en pratique. Nous explicitons ensuite notre méthodologie pour une approche par recherche locale de ces problèmes mixtes.

1.2 Problèmes d'optimisation en variables mixtes

Les problèmes d'optimisation en variables mixtes, aussi appelés problèmes d'optimisation mixte, conjuguent les attributs des problèmes d'optimisation combinatoire et continue. Après un rappel des principales caractéristiques de ces deux types de problèmes d'optimisation, nous définirons formellement ce que nous entendons par problème d'optimisation mixte et présenterons quelques exemples de problèmes rencontrés dans l'industrie.

Les problèmes d'optimisation continue consistent à minimiser (ou maximiser) une fonction sur un domaine continu. Des algorithmes polynomiaux existent pour de nombreux problèmes de cette famille, comme par exemple pour la minimisation d'une fonction convexe sur un domaine convexe [92]. En particulier, si le domaine est défini par un ensemble d'équations linéaires et que la fonction à minimiser est elle-même linéaire, on parle de programmation linéaire [32]. Le calcul des mouvements de terre sur un chantier de terrassement, introduit par Monge en 1781 [93], est un problème classique d'optimisation continue. Il consiste à déterminer les quantités de terre à déplacer entre les déblais et les remblais de façon à minimiser la somme des moments de transport (produit de la quantité de terre déplacée par la longueur du déplacement). Nous détaillerons sa modélisation sous la forme d'un programme linéaire en préambule du Chapitre 3 consacré à l'optimisation des mouvements de terre sur chantiers linéaires.

Les problèmes d'optimisation combinatoire consistent à minimiser une fonction sur un domaine discret [104]. Bien que décrit de façon concise, ce domaine (aussi appelé ensemble des solutions) est souvent de très grande taille. Le Chapitre 2 est consacré à un problème purement combinatoire consistant à choisir parmi tous les jeux de coffrages permettant de réaliser un chantier de construction, celui dont le coût de location est le plus faible. Le problème de remplissage des écrans publicitaires de TF1 [24] est un autre problème d'optimisation combinatoire : affecter des spots au sein d'écrans tout en respectant des contraintes de capacité et d'exclusion mutuelle, de façon à maximiser le chiffre d'affaires. Nous invitons le lecteur intéressé à consulter l'ouvrage de Guéret *et al.* [57] présentant un panorama de problèmes combinatoires rencontrés dans l'industrie (transport, télécommunications, énergie). Des algorithmes efficaces (polynomiaux) existent pour trouver l'optimum de certains problèmes d'optimisation combinatoire, mais la plupart sont NP-difficiles [50]. Les deux problèmes précités sont d'ailleurs NP-difficiles ; la complexité du premier sera établie au

Chapitre 2.

Les problèmes d'optimisation mixte comportent des *variables de décision discrètes et continues*. Si les contraintes et l'objectif du problème peuvent être exprimés de manière linéaire en fonction de ses variables de décision, alors on parle de programmation linéaire mixte. Dans le Chapitre 3, nous abordons un problème d'optimisation de mouvements de terre sur chantiers linéaires [67]. La dimension combinatoire du problème est induite par l'affectation des mouvements aux machines de terrassement, tandis que la dimension continue est relative aux quantités de terre déplacées et au placement de l'activité sur un horizon de temps continu. Dans le Chapitre 4, nous traitons un problème d'optimisation de tournées de camions avec gestion des stocks [20]. Ce problème, déjà abordé dans la littérature [27, 29, 34], nécessite la construction de tournées de véhicules (dimension combinatoire). En outre, il possède une dimension continue : déterminer le volume de produit (fluide) à livrer ou charger à chaque site visité.

Ces problèmes d'optimisation en variables mixtes apparaissent dans divers domaines industriels. La planification des arrêts de centrales nucléaires [49, 54, 98] en est un exemple dans le secteur de l'énergie. Ce problème consiste à ordonnancer les arrêts de maintenance des centrales nucléaires et à déterminer un plan de production de toutes les centrales thermiques du parc, sur un horizon donné. Comme autres exemples de problèmes d'optimisation mixte à forts enjeux économiques, nous pouvons citer : planification prévisionnelle de maintenance des raffineries [80], optimisation de réseaux de conduites pétrolières [62], optimisation du placement des puits en ingénierie de réservoir [62], optimisation du chargement de porte-containers [100], construction des rotations d'avions avec affectation des passagers [2]. La plupart des approches proposées pour résoudre ces problèmes de grande taille se basent sur des décompositions, qui font l'objet de la section suivante.

1.3 Approches par décomposition

Décompositions empiriques. Lorsqu'ils sont de grande taille, une approche couramment utilisée pour résoudre ces problèmes mixtes est leur décomposition en sous-problèmes (combinatoire et continu). Ces décompositions exploitent les propriétés intrinsèques du problème (généralement qualifiées de "propriétés métier") ; elles sont souvent issues des pratiques historiques des ingénieurs en charge des opérations. Par exemple, pour le problème d'optimisation des mouvements de terre, cette approche consiste à séparer l'affectation des mouvements de terre aux ressources et la détermination des dates de leur réalisation (sur un horizon temporel continu) [96]. Pour ce qui est de l'optimisation des tournées de véhicules avec gestion des stocks, il s'agit en premier lieu d'évaluer les quantités à livrer à chaque client, puis à construire des tournées pour satisfaire celles-ci [27]. Concernant la planification des arrêts des centrales nucléaires, une décomposition séquentielle amène à positionner les dates d'arrêts des centrales puis à calculer un plan de production global permettant de satisfaire la demande [54].

Décompositions mathématiques. La programmation mathématique fournit également des méthodes formelles de décomposition. Une technique de résolution désormais célèbre en programmation linéaire en nombres entiers (PLNE) est le *Branch and Price*, couplant *Branch and Bound* et génération de colonnes [9, 38, 71, 101]. Cette technique est notamment employée pour traiter les programmes induits par décomposition de Dantzig-Wolfe [36], comportant généralement un très grand nombre (exponentiel) de colonnes. Ce type de décomposition consiste en une reformulation du programme original sous une forme étendue ; une des plus fréquentes en pratique est la reformulation par couverture d'ensembles (*set covering* en anglais). D'un autre côté, Benders [11] a introduit une méthode de décomposition adaptée à la résolution de problèmes mixtes. Cette technique consiste à décomposer un programme mixte sous la forme de deux programmes maître-esclave, l'un discret et l'autre continu. On parle ici de décomposition par partitionnement de variables. Pour des exemples d'applications industrielles s'appuyant sur ce type de décomposition, nous invitons le lecteur à consulter [48, 52, 89, 90, 91, 97]. Les décompositions de Benders sont intéressantes pour attaquer les problèmes dont la partie continue est difficile à traiter (programme linéaire de très grande taille). Toutefois, elles ne permettent pas de réduire directement la complexité de résolution relative à la partie combinatoire. Or, la difficulté des problèmes mixtes que nous abordons ici est en premier lieu induite par leur caractère fortement combinatoire.

Limites pratiques. Les décompositions empiriques ont un inconvénient majeur : elles sont par nature sous-optimales. En effet, les objectifs des sous-problèmes résolus s'avèrent généralement contradictoires. Plus il y a de contraintes couplantes entre variables combinatoires et continues, plus la décomposition sera délicate ; l'obtention d'une solution admissible peut même s'avérer difficile. Bien que garantissant en théorie l'obtention d'une solution optimale, les décompositions mathématiques ont elles aussi des limitations en pratique. Tout d'abord, la relaxation linéaire des problèmes riches rencontrés dans l'industrie est généralement faible. C'est notamment le cas des problèmes comportant une forte dimension combinatoire (comme le routage ou l'ordonnancement). Cela impose des reformulations du problème voire la génération de coupes pendant l'exploration de l'arbre de recherche ; cette génération d'inégalités valides est bien souvent coûteuse en temps [46]. Dans le cadre du *Branch and Price* [83], une difficulté supplémentaire apparaît : le sous-problème de génération de colonnes, rencontré lors du calcul de la relaxation linéaire à chaque nœud de l'arbre de recherche, est rarement polynomial. Il est déjà NP-complet dans le cas du problème classique de tournées de véhicules avec fenêtres de temps [40]. De plus, la convergence de la méthode de génération de colonnes est en pratique perturbée par des problèmes de stabilisation [46]. Pour plus de détails, nous renvoyons le lecteur au tutoriel de Feillet [46] sur les difficultés rencontrées dans les approches de type *Branch-Price-Cut* pour la résolution des problèmes de tournées de véhicules. Dans le cas des problèmes mixtes, une difficulté supplémentaire peut être posée par la taille du sous-problème continu à résoudre.

En conclusion, nous observons qu'en dépit d'une littérature académique abondante sur le sujet, les approches par décomposition implémentées dans des contextes industriels sont

très rarement "exactes", au sens où elles intègrent des composantes heuristiques dans leur schéma de résolution empêchant de garantir la qualité des solutions obtenues.

1.4 Recherche locale pour l'optimisation mixte

1.4.1 Introduction

La recherche locale en optimisation combinatoire est une technique de résolution mathématique consistant à partir d'une solution initiale appartenant à l'espace de recherche, puis à améliorer la solution courante de façon itérative en explorant un voisinage. Le principe fondamental de l'algorithme est d'explorer l'espace de recherche en progressant d'une solution à une solution voisine par application de mouvements (aussi appelées "transformations") sur la solution courante (voir Figure 1.1). Lin et Kernighan [82] ont utilisé la recherche locale en l'appliquant au problème du voyageur de commerce et en formalisant les concepts sous-jacents. Ceci a contribué à la démocratisation de la notion de recherche locale. La technique a connu un regain d'intérêt dans les années 90 [1] avec l'avènement de la notion de métaheuristique, introduite par Glover [53]. Les métaheuristiques font appel à des algorithmes sous-jacents de recherche locale, c'est-à-dire qu'elles décrivent les stratégies utilisées pour trouver rapidement et efficacement une solution de qualité en un temps raisonnable. Ce sont généralement des algorithmes stochastiques itératifs. On peut citer notamment l'algorithme de *Recuit Simulé* [77], les algorithmes génétiques (Koza [78], Goldberg [55]), la recherche tabou (Glover [51], Gendreau [53]) et les algorithmes de colonies de fourmis (Dorigo [39]). L'ouvrage de Gonzales et al. [56] présente en détails quelques unes de ces techniques et compare leurs résultats. Nous invitons également le lecteur à consulter l'ouvrage de Talbi sur l'implémentation des méta-heuristiques [106] et celui de Aarts et Lenstra [1] pour un panorama d'applications des métaheuristiques en optimisation combinatoire.

1.4.2 Une méthodologie pour l'optimisation mixte

La méthodologie présentée ici étend les travaux introduits par Estellon *et al.* [45] pour l'optimisation combinatoire. La première caractéristique de notre approche est que la recherche locale est *directe*. En effet, aucune décomposition du problème n'est ici utilisée, le problème étant attaqué frontalement. L'espace des solutions exploré par l'algorithme est proche de l'espace original des solutions. En particulier, les dimensions combinatoire et continue du problèmes sont traitées simultanément : les décisions combinatoire et continue sont prises par un même mouvement au cours de la recherche locale. Dans le Chapitre 3 concernant les mouvements de terre, la transformation consistant à insérer un mouvement de terre au sein du planning d'une des ressource modifie à la fois l'affectation du mouvement à la ressource (aspect combinatoire) et la quantité de terre déplacée par la ressource (aspect continu). Dans le cas de l'*Inventory Routing Problem* (IRP) (voir Chapitre 4), quand on

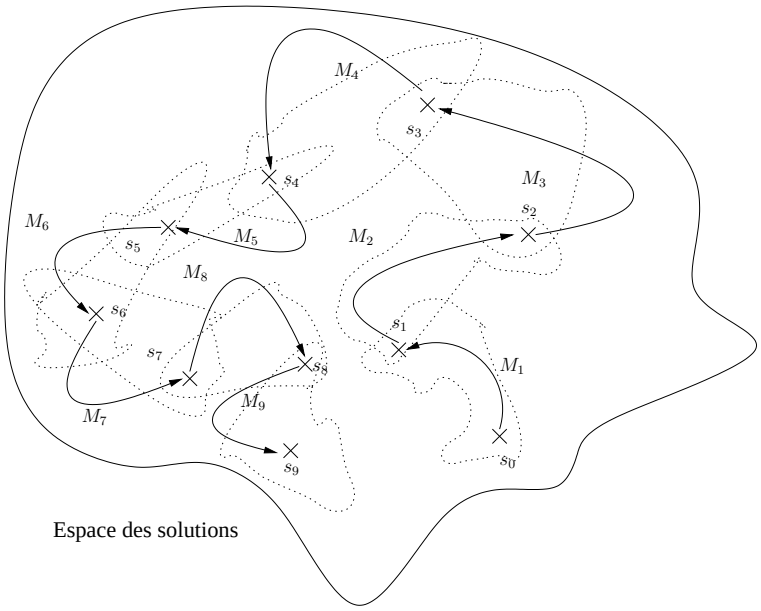


FIGURE 1.1 – Recherche locale dans le voisinage des solutions.

déplace une livraison d'un client dans le temps, l'évaluation de l'impact de la transformation nécessite le calcul des dates de la tournée concernée et des autres tournées du camion et de la remorque (aspect combinatoire) ainsi que la réaffectation des volumes livrés à tous les clients de la tournée et ceux de tous les clients livrés par cette remorque (aspect continu). Les décompositions sont généralement source de sous-optimalité (à moins qu'une structure singulière se cache derrière le problème). En évitant dans notre approche les décompositions et les réductions, aucune solution n'est perdue et la probabilité de trouver une solution de bonne qualité est augmentée. De plus, la recherche locale est *pure*. Elle ne fait appel à aucune hybridation : ni métaheuristique, ni technique de recherche arborescente ne sont utilisées ici. Les algorithmes présentés dans cette thèse utilisent la stratégie par "descente standard" (connue en anglais sous le nom de *First Improvement descent*) qui permet d'éviter un paramétrage complexe, comme cela est fréquemment le cas lors de l'emploi d'une métaheuristique. On recherche ici une première solution améliorante dans le voisinage exploré. Cette stratégie est utilisée dans les travaux d'Helsgaun [60]. Hansen *et al.* [58] démontre son efficacité par rapport à une approche par recherche de la meilleure solution améliorante (appelée *Best Improvment Descent*).

Une autre caractéristique de cette recherche locale est qu'elle visite un voisinage fortement *randomisé*. Les décisions sont prises de façon (pseudo-)aléatoire. La taille de notre voisinage est généralement quadratique en la taille de l'entrée du problème (soit $O(n^2)$ avec n la taille de l'entrée), mais la constante cachée par le grand O est grande. Par exemple, dans le Chapitre 4 concernant l'IRP, nous faisons appel à un ensemble de plus de 50 mouvements, chacun induisant un voisinage de taille quadratique en le nombre d'opérations.

L'union de ces voisinages randomisés de petite taille induit un voisinage très large en pratique. En pratique, ceci permet ainsi une convergence vers des optima locaux de grande qualité, bien que l'espace de recherche soit fortement contraint. Le fait que la recherche soit randomisée rend la recherche non déterministe. C'est la raison pour laquelle notre recherche locale est aussi *aggressive* : des millions de solutions faisables sont visitées au cours du temps imparti. Pour cela, nous utilisons une machinerie d'évaluation évoluée couplée à un vaste ensemble de mouvements. L'évaluation se fait de manière incrémentale en se basant sur des algorithmes approchés mais très efficaces en temps. Les performances de la recherche locale sont dépendantes de l'optimisation de ces méthodes d'évaluation et de rétablissement rapide d'une solution courante modifiée par les mouvements finalement non acceptés. L'algorithmie incrémentale repose notamment sur l'exploitation d'invariants entre la solution courante et la solution modifiée [76].

Nos heuristiques de recherche locale sont construites en trois niveaux : l'heuristique générale, les mouvements et la machinerie d'évaluation. Cette dernière constitue le coeur de la recherche locale et calcule l'impact de chaque mouvement sur les contraintes et les objectifs au cours de la recherche. Le temps passé à développer chaque couche se repartit ainsi : 10 % sur l'heuristique générale, 20 % sur les mouvements et 70 % sur la machinerie d'évaluation. Le développement d'une heuristique de recherche locale se résume donc essentiellement dans la définition du large ensemble de mouvements randomisés permettant l'exploration effective de larges espaces de recherche et dans l'accélération du processus d'évaluation de ces mouvements. Dans le contexte particulier des problèmes en variables mixtes, les mouvements adressent les dimensions combinatoire et continue en même temps. Ainsi, les mouvements modifient les variables combinatoires et continues à chaque itération. Une fois la partie combinatoire rétablie, une solution admissible doit donc être retrouvée pour le sous-problème continu, induit par ces variables combinatoires fixées. Et lorsque le sous-problème continu est de grande taille, cela peut devenir très coûteux en temps, même s'il est polynomial. La recherche locale pour l'optimisation mixte nécessite donc de résoudre la partie continue de manière incrémentale, approchée, et même éventuellement randomisée. Par exemple, dans le cas de l'IRP, le sous-problème continu induit est un problème de flot maximum. Nous avons concentré notre attention dans l'accélération de la résolution de ce sous-problème continu, comme détaillé en Chapitre 4.

Afin d'améliorer l'*efficacité* de l'implémentation de ces heuristiques, nous appliquons les principes d'ingénierie logicielle énoncés par Moret [95] notamment le profilage de la consommation en temps et en espace (mémoire vive et mémoire cache). La programmation par assertions, couplée à une vérification des données incrémentales, permet de contrôler la *fiabilité* de chacun de nos algorithmes. L'objectif de ces vérificateurs est de contrôler la validité des structures après chaque mouvement (en mode debug). En moyenne, 10 à 20 % du code développé est dédié à cette vérification. Enfin, nous utilisons dans nos procédures de tests des instances pathologiques afin de s'assurer de la convergence de la recherche locale même sur des cas critiques. Nous fournirons des détails sur ces différents aspects pour chacun des algorithmes présentés dans cette thèse.

1.5 Organisation de la thèse

Nous présentons tout d'abord un premier problème industriel purement combinatoire sur lequel nous avons appliqué un algorithme de recherche locale. L'objectif est d'optimiser les commandes de banches (les coffrages servant à couler les murs d'un chantier de construction). Nous verrons qu'une heuristique à base de recherche locale a permis de trouver rapidement des solutions de qualité pour toutes les instances de ce problème et que le moteur est désormais utilisé en production pour tous les chantiers de Bouygues Bâtiment Résidentiel.

Ensuite, nous nous focaliserons sur le problème d'ordonnancement des mouvements de terre sur chantiers linéaires résolu par une heuristique de recherche locale. Une de ses caractéristiques est que les quantités de terre déplacées peuvent passer d'une tâche à l'autre, alors que les dates d'affectation de ces tâches sont aussi modifiées. Le logiciel en exploitation chez Bouygues DTP Terrassement permet la planification de grands chantiers en quelques minutes. Nous nous intéresserons aussi à l'étude fondamentale du cas à une ressource, correspondant à un problème de voyageur de commerce linéaire avec précédences, prouvé ici comme étant NP-difficile. Nous obtenons des bornes inférieures permettant d'apprécier la qualité de la recherche locale.

Le problème d'optimisation des tournées de véhicules avec gestion de stock a pour objectif de réduire les coûts de distribution d'un produit par camion, sur des zones géographiques comptant plusieurs centaines de clients, dont la gestion des stocks est confiée au fournisseur. Une heuristique de recherche locale est proposée pour résoudre le problème à court terme (15 jours) en quelques minutes. Afin d'assurer une optimisation sur le long terme, nous introduisons une fonction objectif de remplacement fondée sur un calcul de bornes inférieures. L'originalité de notre approche réside dans le fait que les quantités livrées au cours des tournées peuvent varier alors même que ces dernières sont modifiées géographiquement ou temporellement. L'exploitation de ce logiciel, quotidienne et à l'échelle mondiale, a permis des réductions de coûts logistiques de l'ordre de 15 % chez le client industriel du Groupe Bouygues.

Chapitre 2

Optimisation du stock de banches sur des chantiers de construction

Bouygues Habitat Résidentiel est une filiale de Bouygues Construction spécialisée dans la construction de résidences privées. Leur bureau des méthodes est spécialisé dans l'organisation, la planification et l'optimisation de ces chantiers s'étalant sur des périodes allant de 6 à 12 mois. Ils ont aussi bien la responsabilité de la gestion des ressources humaines que matérielles. La construction doit être livrée en temps et en heure sous peine de pénalité de retard. Or une construction résidentielle est un projet complexe qui implique de nombreux interlocuteurs. Le problème décrit ici concerne l'optimisation du stock de banches utilisées sur les chantiers résidentiels [1]. Comme nous l'avons vu en introduction de cette thèse, ce problème est purement combinatoire et nous présenterons ici une heuristique à base de recherche locale pour le résoudre de manière efficace et robuste.

2.1 Contexte

Le problème traité ici concerne la phase de gros œuvre du chantier qui est une des tâches principales de la construction. En général, un chantier moyen d'Habitat Résidentiel compte une dizaine d'étages et une centaine de murs, que l'on nommera ici des voiles. Chaque étage doit être terminé en une dizaine de jours environ, en utilisant un nombre limité de banches verticales. Ces banches sont des coffrages métalliques dans lesquelles on vient couler du béton formant les voiles. La commande, la rotation et l'assemblage de ces coffrages sont trois sous-problèmes dont le bureau des méthodes a la responsabilité. Les commandes doivent en effet être passées en avance et sont fixes pour toute la durée du chantier (aucune banche ne peut être commandée ou retournée pendant la durée du chantier, sauf cas exceptionnel).

1. Ces travaux de recherche ont été menés en 2007 en collaboration avec Thierry Benoist (Bouygues e-lab) et Pascal Molin (Institut de Mathématiques de Bordeaux) [15, 18]

Minimiser le coût de location de ces matériaux de coffrage requiert de calculer le plus petit stock possible de banches et de les utiliser judicieusement au fil des jours. En effet, une quantité trop importante de banches sur le chantier entraîne des problèmes de stockage et donc de congestion sur le site.

Une banche est une paire de panneaux métalliques verticaux utilisée pour construire les murs : du béton encore liquide y est coulé entre les panneaux. Ce processus nécessite plusieurs jours de séchage avant que le coffrage ne puisse être ouvert et utilisé ailleurs sur le site. On dispose de plusieurs familles de banches caractérisées notamment par leur taille allant de 60 à 480 cm et leur stabilité. Les voiles à couler sont généralement plus longs que la longueur maximale des banches : pour couvrir un mur, on assemble plusieurs coffrages ensemble. Par exemple, la Figure 2.1 montre qu'un train de banches de 10,8 mètres peut être construit en utilisant différentes combinaisons de banches. La première combinaison est un assemblage avec des banches de 4 mètres, 2 mètres et 0,8 mètre alors que la deuxième combinaison utilise deux types de banches : celles de 3 mètres et de 1 mètre. Si on dépasse la longueur, on utilise un bouchon pour l'ajuster. Pour chaque voile, la contrainte principale (contrainte de couverture) est que la somme des longueurs des banches allouées doit dépasser la longueur du mur. L'ensemble de banches de chaque type affecté à un mur est appelé un vecteur de couverture.

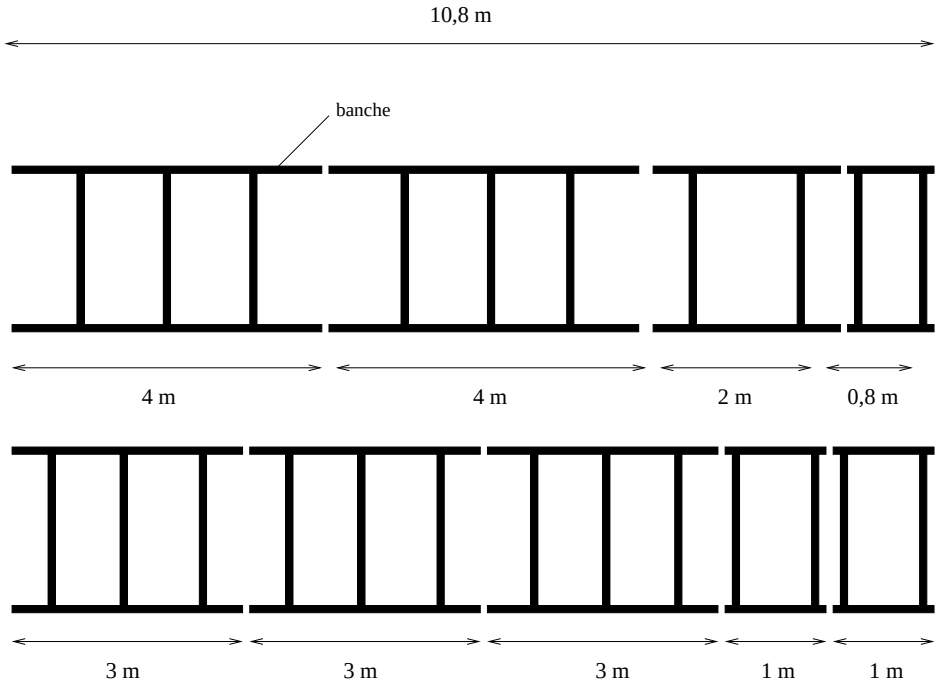


FIGURE 2.1 – Deux trains de banches de longueur 10,8 m.

Nous définissons le problème de minimisation du stock de banches dans la Section 2.2. Puis en Section 2.3, nous introduisons une première approche par programmation mixte

en nombres entiers et puis par programmation par contraintes. Ensuite, en Section 2.4 un deuxième modèle utilisant la recherche locale est présenté. Des algorithmes exacts et approchés sont aussi présentés. La NP-complétude du problème maître est prouvée en Section 2.5. Les résultats pour chaque approche sont fournis en Section 2.6.

2.2 Présentation du problème

La modélisation de ce problème a été introduite en 1999 par Le Pape et Baptiste [81]. Ce problème consiste à minimiser la quantité de matériels de coffrage requis pour le site, en faisant un choix approprié des banches affectées à chaque voile. Le planning exact d'ordonnancement des voiles est calculé au préalable, c'est-à-dire qu'on connaît pour chaque voile le jour où il est érigé. On note D l'ensemble des jours où la construction doit être effectuée, B l'ensemble des banches disponibles et V l'ensemble des voiles à couler. Pour chaque jour $d \in D$, on note $V(d) \subseteq V$ l'ensemble des voiles qui doivent être coulés.

Voiles. Pour chaque voile $v \in V$, la longueur totale couverte par les banches allouées doit correspondre à la longueur attendue : elle doit appartenir à l'intervalle $[\ L_{min}(v),\ L_{max}(v)\ ]$. La longueur $L_{min}(v)$ est généralement plus grande que la largeur du voile car des extrémités doivent être positionnées au bout du train de banches pour le fermer à l'aide de bouchons, assurant l'étanchéité du coffrage, mais grignotant une partie de la longueur couverte. $L_{max}(v)$ est bornée par la longueur maximale de ces bouchons ou par la position des voiles voisins. Quand un mur est positionné entre deux autres, la longueur du voile à couler est de longueur fixe et on a donc $L_{min}(v) = L_{max}(v)$. De tels voiles sont appelés "murs bloqués". La Figure 2.2 illustre ces différentes notations pour un voile à trois banches. Dans cet exemple, pour couler le voile v, on fait appel à trois banches et un unique bouchon, puisqu'une des extrémités est bloquée par un voile déjà existant.

Banches. Les trains de banches sont construits par assemblage de plusieurs banches de différentes longueurs. L'épaisseur d'un voile n'est pas prise en compte. Le problème d'optimisation des stocks ne se focalise que sur la recherche des vecteurs de couvertures à affecter à chaque voile. L'optimisation des séquences de banches est un autre problème traité indépendamment [16].

Chaque banche $b \in B$ est définie par sa stabilité qui peut être de trois types. Les *stables* pour les modèles avec des béquilles, résistants à des vents d'environ 80 km/h. Les *neutres* qui sont les banches ajustables et les petites. Les *instables* qui sont de tailles intermédiaires mais dans des matériaux plus légers. Pour chaque voile, le train de banches doit être stable, chaque banche stable étant capable de stabiliser deux banches instables. En d'autres termes, le nombre de banches instables ne peut pas excéder deux fois le nombre de stables (voir inégalités (2.4)). On note $S \subseteq B$ (resp. $I \subseteq B$) l'ensemble des banches stables (resp.

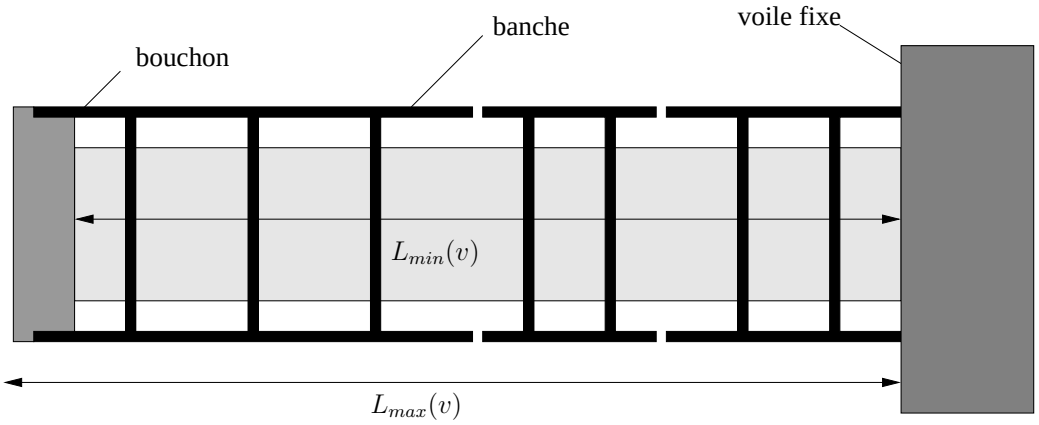


FIGURE 2.2 – Un voile avec trois banches.

instables). Chaque banche $b \in B$ est aussi décrite par sa longueur qui peut être ajustée entre deux longueurs $[L_{min}(b); L_{max}(b)]$. Seuls les plus petits modèles ont des longueurs ajustables (on note $A \subseteq B$ l'ensemble des banches ajustables). Ces modèles ne peuvent être utilisés que dans les extrémités : on définit $M(v)$ comme le nombre maximum de banches de petite taille qui peuvent être utilisées par train pour le voile v (voir inégalités (2.3)).

De plus, on sait que certaines banches sont dites "spéciales" : leur utilisation doit être limitée le plus possible car elles compliquent les tâches d'assemblages. Soit $S \subseteq I$ l'ensemble des banches spéciales. Leur longueur cumulée sur un voile est bornée par une constante Z spécifiée par les ingénieurs méthodes (voir inégalités (2.5)).

Contraintes. Sur chaque voile v, l'utilisation d'un ensemble de banches "lourdes" $H(v)$ (voir inégalités (2.6)) est à éviter. Les capacités limitées de la grue font que des contraintes de poids doivent être imposées pour chaque voile (la capacité diminue en fonction de la distance au pied de la grue). On note $x(v, b)$ le nombre de banches de type $b \in B$ utilisées pour couvrir le voile $v \in V$ (on dit encore que $x(v, b)$ est le vecteur couverture du voile v). On définit $n(b, d)$ la quantité de banches de modèle b utilisées un jour $d \in D$ et $m(b)$ la quantité de banches de type $b \in B$ nécessaires sur le site, qui est déterminée par la quantité du jour le plus demandeur pour le modèle $b \in B$. Notons que $n(b, d)$ et $m(b)$ sont totalement déterminés par les vecteurs couvrants. Le problème est donc décrit par les contraintes suivantes.

$$\forall\ v \in V,\ \sum_{b \in B} L_{min}(b) \times x(v, b) \leq L_{max}(v) \tag{2.1}$$

$$\forall\ v \in V,\ \sum_{b \in B} L_{max}(b) \times x(v, b) \geq L_{min}(v) \tag{2.2}$$

$$\forall\ v\ \in\ V,\ \sum_{b\ \in\ A} x(v,b)\ \leq\ M(v) \tag{2.3}$$

$$\forall\ v\ \in\ V,\ 2\ \times\ \sum_{b\ \in\ I} x(v,b)\ \leq\ \sum_{b\ \in\ S} x(v,b) \tag{2.4}$$

$$\forall\ v\ \in\ V,\ L_{min}(v)\ -\ \sum_{b\ \in\ B\ \setminus\ S} L_{max}(b)\ \times\ x(v,b)\ \leq\ Z \tag{2.5}$$

$$\forall\ b\ \in\ H(v),\ x(v,b)\ =\ 0 \tag{2.6}$$

$$\forall\ d\ \in\ D,\ b\ \in\ B,\ \sum_{v\ \in\ V(d)} x(v,b)\ =\ n(b,d) \tag{2.7}$$

$$\forall\ b\ \in\ B,\ \max_{d\ \in\ D} n(b,d)\ =\ m(b) \tag{2.8}$$

Objectif. L'objectif de l'ingénieur méthode est de minimiser le coût de location du stock. Avec $R(b)$ le coût de location fixe d'une banche du modèle $b\ \in\ B$, cet objectif s'écrit sous la forme suivante :

$$min\ \sum_{b\ \in\ B} R(b)\ \times\ m(b) \tag{2.9}$$

Cependant, ce n'est pas le seul objectif poursuivi. En effet, on cherche à minimiser aussi le nombre total de banches inutilisées, car elles posent des problèmes de congestion d'espace sur le site pendant la construction. En effet, quand une banche ne sert pas à coffrer, le responsable de chantier doit trouver une place pour la stocker, ce qui va nécessiter du temps de grue pour la déplacer à cet endroit, voire d'autres engins de chantier. Il est donc primordial de minimiser ce type d'intervention et de maximiser l'utilisation quotidienne de tout le parc de banches. Ainsi, on introduit une pénalité $P(b)$ pour chaque nouvelle banche afin de modéliser cet objectif. Ce coefficient est généralement plus grand pour les banches plus grandes qui sont encore plus difficiles à déplacer et qui prennent beaucoup de place à stocker. De même, les coûts de location augmentent aussi avec la taille de la banche. On obtient ainsi la fonction objectif modifiée 2.10.

Modèle 1 :

$$min\ \sum_{b\ \in\ B} R(b)\ \times\ m(b)\ +\ \sum_{b\ \in\ B,\ d\ \in\ D} P(b)\ \times\ (\ m(b)\ -\ n(b,d)\ ) \tag{2.10}$$

Comme l'objectif est de laisser la main à l'ingénieur méthode en lui proposant un outil d'aide à la décision, il peut arriver que l'utilisateur lui-même souhaite limiter la commande d'un des modèles en contrôlant le maximum pour chaque modèle. Par exemple, limiter le nombre de banches de petite taille peut permettre de diminuer les temps d'assemblage ou peut aussi modéliser une contrainte de stock disponible. On introduit $Q(b)$ qui est le maximum de banches du type b commandées.

$$\forall\ b\ \in\ B,\ m(b)\ \leq\ Q(b) \tag{2.11}$$

Nous considérons ici un ensemble de 36 instances issues de données réelles comportant jusqu'à 11 types de banches, 124 voiles et 14 jours. Ces données proviennent de 12 sites de construction d'Habitat Résidentiel. Nous présentons dans les sections suivantes les différentes approches pour résoudre ce problème d'optimisation.

2.3 Heuristique constructive

Le modèle présenté dans la section précédente peut être interprété par un solveur de programmation entière si on remplace la contrainte 2.8 par :

$$\forall\ b\ \in\ B,\ d\ \in\ D,\ m(f)\ \geq\ n(b,d) \tag{2.12}$$

Cependant, même si une solution continue peut être trouvée en quelques secondes, les solveurs de l'état de l'art nécessitent plusieurs minutes (voir plusieurs heures) pour trouver une solution entière faisable comme le montre le tableau des résultats en Section 2.6. De même, la stratégie de recherche par défaut d'un solveur de programmation par contraintes comme Choco [79] ne produit pas de solutions faisables en un temps raisonnable (voir Section 2.6).

Ainsi, une heuristique constructive est proposée dans l'Algorithme 1. L'idée est de se focaliser tout d'abord sur les banches qui peuvent être utilisées les D jours. Puis, on s'intéresse à celles pouvant être utilisées $D-1$ jours, en diminuant petit à petit le nombre de jours un à un. Pour chaque valeur de ce paramètre, on parcourt les banches ordonnées par longueur décroissante (la liste ordonnée B_o) : les coûts marginaux de location sont donc ordonnés du plus économique (pour les modèles de grande taille) au plus cher (pour les plus petites). Pour chaque banche, tant que la contrainte 2.10 n'est pas saturée, on essaie de sélectionner pour chaque jour, un voile pour lequel une telle banche peut être affectée. Dans le cas où aucune contrainte 2.10 n'est posée, on vérifie qu'il existe toujours au moins un support utilisant les modèles de type b pour le voile v grâce à la fonction $\mathit{EstAffectable}(b,\ v)$. Si aucune solution ne peut être trouvée et que le minimum de jours ne peut être satisfait, on passe au modèle de banche suivant. Autrement, cette banche est affectée à chaque voile sélectionné. La fonction $\mathit{EstAffectable}(b,\ v)$ a pour but de détecter qu'un voile v a reçu suffisamment de banches, c'est-à-dire que les contraintes de 2.1 à 2.6 sont bien satisfaites, auquel cas on marque les

voiles comme “fermés” et la fonction retourne toujours faux pour ce voile dans ce cas. À chaque étape de l'algorithme, les décisions sur les voiles (comme l'affectation d'une banche à un voile) et les décisions sur les banches (comme décider du nombre de banches commandées pour un modèle spécifique) propagent des réductions de domaines pour les autres voiles ou les autre banches. Par exemple, la décision de commander une banche de 1,4 mètres pour un voile de 2,4 mètres va exclure d'utiliser des banches de plus d'un mètre. De même, si le stock maximal d'une banche est fixé à 3, dès qu'on atteint ce niveau, elle devient inutilisable et cette décision est propagée en réduisant son domaine à 0 pour ces voiles. L'algorithme renvoie la liste de banches affectées.

Algorithme 1: HEURISTIQUE CONSTRUCTIVE

input : Les banches ordonnées par taille décroissante
output : Une affectation des banches à un voile

```
begin
    for minUsage = |D|a1 do
        for b ∈ B_o do
            stop = faux
            while !stop and !stocksLimitesAtteints(b) do
                V = 0
                for d ∈ D do
                    while v ∈ V_d do if EstAffectable(b, v) then Choisir V
                    if |V| < minUsage then stop = vrai
                    else for v ∈ V do Affecter(b, v)
    Return la liste de banches affectées.
```

Cet algorithme à base de propagation et de filtrage tourne en moins d'une seconde sur toutes les instances du benchmark et a été utilisé pendant plusieurs années en exploitation pour résoudre ce problème sur des dizaines de chantiers d'Habitat Résidentiel. En effet, chaque jour, des dizaines d'ingénieurs méthodes commandaient leur stock de banches sur les sites de construction à l'aide d'un logiciel utilisant cet algorithme, qui a été remplacé en 2009 par la nouvelle approche décrite dans la section suivante. Nous verrons dans la Section 2.6 que le nouvel algorithme obtient des résultats numériques améliorant les solutions de 7 % à 14 %.

2.4 Approche par recherche locale

Modélisation par couverture d'ensembles. Une étude sur les cas réels de notre benchmark révèle une caractéristique-clef du problème à un voile. Dans toutes les instances, les

contraintes sur les murs rendent le nombre d'affectations faisables sur chaque voile suffisamment petit pour être calculé de manière préliminaire en une fraction de seconde. La Figure 2.3 montre que pour la plupart des murs d'une des instances, le nombre de vecteurs de couverture possibles, dont on nomme l'ensemble C, dépasse rarement 30. Sur des voiles plus grands, le nombre peut être plus grand (jusqu'à 1 800 ici), à cause de la largeur de l'intervalle de longueur. En moyenne, la durée d'un cycle est de moins de 10 jours, avec une douzaine de voiles par jour : le nombre total de vecteurs de couverture est donc toujours traitable.

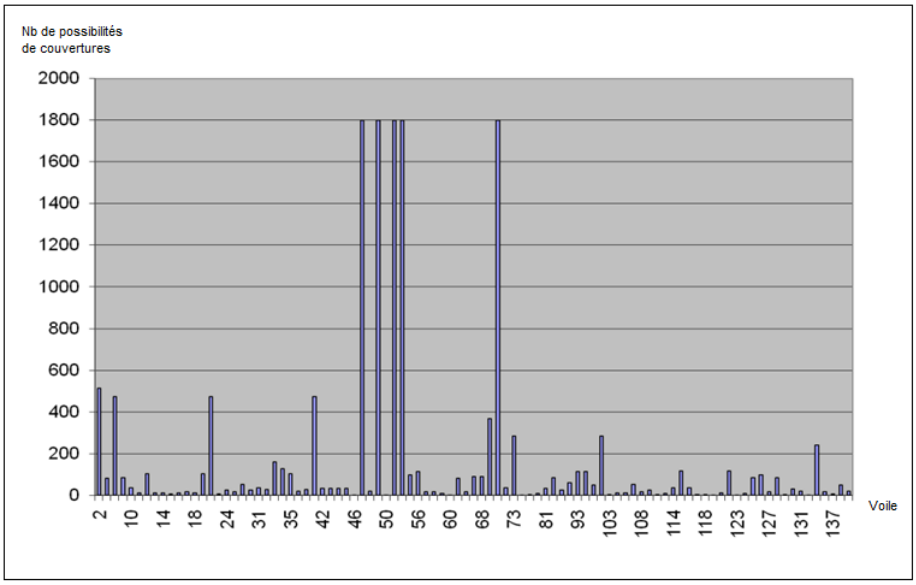


FIGURE 2.3 – Ensemble des couvertures admissibles pour chaque voile.

On est donc en mesure de proposer un modèle alternatif avec une résolution en deux étapes : La phase dite "géométrique" qui consiste à calculer tous les vecteurs de couverture possibles pour chaque mur. La phase d'optimisation qui calcule à nouveau le problème de couverture d'ensembles induit.

L'énumération des vecteurs couvrants possibles $C(v)$ pour chaque voile $v \in V$ peut être réalisée soit par un solveur de programmation par contraintes qui s'appuie sur les contraintes 2.1 à 2.5, soit par un algorithme dédié réalisant une recherche arborescente dédiée. On notera $X(v)$ l'ensemble des vecteurs couvrants candidats pour un voile v. Soit $X(b,\ i)$ la quantité de banches du modèle $b \in B$ dans une solution $i \in X(v)$. Le modèle d'optimisation résultant se lit ainsi, avec $z(v,\ i)$ le booléen validant la sélection d'une solution i pour le voile v :

Modèle 2 :

$$min \sum_{b \in B} R(b) \times m(b) + \sum_{b \in B,\ d \in D} P(b) \times (m(b) - n(b,d)) \tag{2.13}$$

$$\forall\ b \in B,\ \max_{d \in D} n(b,d) = m(b) \tag{2.14}$$

$$\forall\ b \in B,\ m(b) \leq Q(b) \tag{2.15}$$

$$\forall\ v \in V,\ \forall\ i \in X(v),\ z(v,\ i) \in 0,1 \tag{2.16}$$

$$\forall\ v \in V,\ \sum_{i \in X(v)} z(v,\ i) = 1 \tag{2.17}$$

$$\forall\ d \in D,\ b \in B,\ \sum_{v \in V(d)} \sum_{i \in X(v)} X(b,\ i) \times z(v,\ i) = n(b,d) \tag{2.18}$$

Résolution par programmation linéaire en nombres entiers. La résolution du problème introduit dans le Modèle 2 permet d'obtenir des solutions optimales pour toutes les instances . Selon les cas, le parcours de l'arbre de *Branch and Bound* prend entre 0,2 secondes et 2 heures, tandis que la première solution entière est obtenue en un temps variant de 0,1 à 17 secondes (résultats avec Xpress MP). Nous allons voir qu'une approche par recherche locale permet de trouver des solutions de qualité comparable, avec des temps de résolution plus faible.

Résolution par recherche locale. Une recherche locale très simple permet d'atteindre des résultats de bonne qualité en des temps très courts. Dans nos expérimentations nous comparerons ces valeurs aux solutions optimales calculées par PLNE (en un peu moins de 2 heures pour l'instance la plus difficile). Construisant une solution initiale à partir de l'optimum fractionnaire du modèle linéaire, cette recherche locale utilise un mouvement très simple consistant à choisir un nouveau train de banches pour un mur (parmi la liste des trains possibles pour ce mur). Une simple descente dans ce voisinage permet d'obtenir les résultats présenté dans le Tableau 2.4.

2.5 Complexité

La preuve de NP-complétude pour ce problème de minimisation du stock de banches dérive des contraintes 2.1 et 2.2 dont la conjonction définit un problème de somme de sous-ensembles. Cependant, les observations effectuées dans la section précédente nous conduisent

à définir un deuxième modèle où les contraintes 2.1 à 2.6 sont supprimées et représentées par une liste de vecteurs couvrants pour chaque voile. Comme mentionné précédemment, le nombre total de tels vecteurs couvrants est borné (en pratique) par une constante K. Une question naît naturellement : est-ce que ce problème de minimisation du stock de banches est polynomial en K.

Si on ne considère pas l'objectif (voir (2.10)), le problème est linéaire pour $|D| = 1$ puisque dans ce cas $m(b) = n(b, d)$ et le problème est séparable en $|V|$ sous-problèmes. Dans cette section, on va prouver que la faisabilité du problème de minimisation du stock de banches est NP-complet même si les listes de vecteurs couvrants pour chaque voile sont considérées comme une partie des données d'entrée. Il est important de montrer qu'un corollaire de cette preuve est que le problème d'optimisation avec $|D| > 1$ est NP-Dur, même sans la contrainte 2.10. Cependant, la question de la faisabilité d'une instance du problème pour un jour unique peut être réglée rapidement en ajoutant un jour avec les voiles nécessitant exactement $Q(b)$ banches de type b (pour chaque type). Avec $R(b) = 1$ quelque soit b, la faisabilité des contraintes est équivalente à un stock minimum de $\sum_{b \in B} Q_b$.

Un certificat de faisabilité pour une instance peut être représenté par la matrice $x(v, b)$ de taille $|V|\ |B|$, sur laquelle les $5\ |V| + (|D| + 2)\ |F|$ contraintes linéaires peut être vérifiées dans un temps polynomial. On conclut que le problème de minimisation du stock de banches est dans NP.

Avant de commencer notre preuve, il doit être noté que même si les contraintes (2.1) à (2.6) sont maintenant exprimées *in extenso* comme une liste de vecteurs-couvertures sur chaque voile, ces listes doivent représenter les solutions d'un tel ensemble de contraintes. Soit $S_{a,\omega}$ un problème de somme de sous-ensembles défini par une constante A et par les poids ω_i pour $i < n$, avec $\sum_i \omega_i = \Omega$. Maintenant, considérons une instance du problème de minimisation du stock de banches avec 2 banches et n voiles, chacune ayant 2 vecteurs couverture $(\omega_i, \Omega - \omega_i)$ et $(0, \Omega)$. Fixer $Q_0 = A$ et $Q_1 = n\ \Omega - A$ nous assurera l'équivalence des deux problèmes, mais l'existence de caractéristiques sur les voiles et les banches conduisant à cette liste exacte de vecteurs couvrants n'est pas prouvée donc cette transformation ne prouve rien.

Ayant mentionné cette difficulté, nous pouvons considérer le problème de matching à 3 dimensions (3DM), qui prouve la NP-complétude [74] : soient 3 ensembles X, Y et Z avec la même cardinalité q et M un sous-ensemble de $X \times Y \times Z$. Existe-t-il q éléments de M qui n'ont pas de composante commune ? Dans cette section, nous prouvons que toutes les instances du $3DM$ peuvent être transformées en une instance du problème de minimisation du stock de banches.

Avec $l = 2^{3\ q+1}$, on définit $3\ q + 1$ types de banches :

- F_X : q banches stables de longueur $4\ l + 2^i$ avec $i \in [0, q-1]$
- F_Y : q banches instables de longueur $2\ l + 2^i$ avec $i \in [q, 2\ q-1]$
- F_Z : q banches instables de longueur $l + 2^i$ avec $i \in [2\ q, 3\ q-1]$
- Une banche **ajustable** de longueur $7\ l$ à $8\ l$

En indexant les éléments de X, Y, Z de 0 à $q-1$, on définit $|V|$ voiles d'une longueur exacte $7\,l + 2^i + 2^{q+j} + 2^{2q+k}$ avec $(i,\ j,\ k) \in V$. Tous ces voiles sont rattachés au même jour unique. Et les contraintes de stocks sont définies à $Q_b = 1$ pour toutes les banches non ajustables et à $Q_b = |V| - q$ pour celles ajustables. Comme l'instance ici est faite de $3q+1$ banches et $|V|$ voiles, et que toutes les tailles sont des entiers plus petits que $2^{3\ q\ +\ 5}$, la taille du problème est polynomiale dans le respect de l'instance initiale du problème de 3DM.

Soit une solution $V^* \subseteq V$ du 3DM, trouver une solution au problème de minimisation du stock de banches est évident. Pour chaque voile $(i,\ j,\ k) \in M^*$, on utilise une banche stable de longueur $4\ l + 2^i$ et deux banches instables de longueur respectives $2\ l + 2^{q\ +\ j}$ et $l + 2^{2\ q\ +\ k}$. La banche ajustable est utilisée pour les $|M| - q$ voiles restants. Par définition de M^*, toutes les contraintes de stock sont satisfaites.

Inversement, nous prouvons maintenant que la faisabilité du problème de minimisation du stock de banches implique la faisabilité d'une instance originale du problème 3DM.

Proposition : Toutes les solutions du problème de minimisation du stock de banches $(i,\ j,\ k)$ qui n'utilisent pas la banche ajustable sont obligatoirement couverte par les 3 types de banches : une de longueur 2^i (de l'ensemble F_X), une de longueur 2^{q+j} (de l'ensemble F_Y) et une de longueur 2^{2q+k} (de l'ensemble F_Z).

Preuve : Au moins une banche de F_X doit être utilisée puisqu'au moins une stable est requise. En utiliser plus d'une est impossible puisqu'aucun mur n'a une longueur plus grande que $8\ l$. Ainsi, au plus deux banches instables peuvent être choisies dans $F_Y \cup F_Z$. Puisque ces 2 banches doivent avoir une longueur cumulée entre $[3\ l, 4\ l[$, la seule solution possible est d'en sélectionner une dans chaque ensemble. La longueur du voile est en effet de $(7\ l + 2^i + 2^{q+j} + 2^{2q\ +\ k})$, or seul le i^e élément de l'ensemble F_X peut être candidat pour la position i, seul le j^e élément de l'ensemble F_Y peut prendre la position $q + j$ et seul le k^e élément de l'ensemble F_Z peut prendre la position $2\ q + k$.

En conclusion, au plus $|M| - q$ voiles peuvent utiliser des banches ajustables et on est sûr qu'au moins q voiles sont couverts par le triplet $(i,\ j,\ k)$. En plus de cela, la contrainte de stock empêche chaque élément de $F_X \cup F_Y \cup F_Z$ d'être utilisé deux fois. Par conséquent, ces q voiles définissent un sous-ensemble de M avec les propriétés requises, ce qui prouve la faisabilité de l'instance originale du problème 3DM. ■

En conclusion, le problème de minimisation du stock de banches avec une liste explicite de vecteurs couvrants est NP-Difficile.

2.6 Résultats numériques

Cet algorithme a été testé sur un ensemble d'instances provenant de Bouygues Construction. Ces instances n'imposaient pas de limite sur les quantités à commander (en d'autres

termes, la contrainte 2.10 est inactive). Le Tableau 2.4 fournit les résultats obtenus pour 12 instances du problème de minimisation du stock de banches avec 3 combinaisons du coût de location R_b et du coût de pénalité P_b. Les résultats du MIP sont obtenus en utilisant Xpress MP.

R_b est égale à la longueur maximale L_b^{max} plus une constante entre 20 cm et 100 cm. Plus la constante est grande, plus l'objectif est focalisé sur la minimisation du nombre de petites banches. Par conséquent, cela réduit le nombre de tâches d'assemblages et de désassemblages.

P_b est égale à la longueur maximale L_b^{max} divisée par une constante entre 10 et 1 000. Plus la constante est grande, plus l'objectif est focalisé sur la minimisation des banches non-utilisées. Par conséquent, cela tend à réduire la gêne sur le site, donc il est important d'utiliser au maximum toutes les banches tous les jours.

Les 5 premières colonnes du Tableau 2.4 décrivent chaque instance. Les trains de banches sont ici précalculés et même sur les instances avec un nombre très grand de voiles et de banches, le nombre d'ensemble de couverture ne dépassent pas les 16 000 (voir la colonne 6).

Dans le tableau récapitulatif 2.5, on compare ici les 3 approches décrites précédemment : L'algorithme glouton à base de programmation par contraintes (Colonne PPC), la résolution exacte par programmation linéaire en nombres entiers (Colonne $PLNE$) et enfin l'approche fondée sur une solution fractionnaire optimale rendue entière par recherche locale (Colonne RL). Pour la programmation linéaire en nombre entier, on fournit le temps nécessaire pour calculer l'optimum (Colonne Solution Optimale $S.O.$) et celui pour atteindre la première solution entière (Colonne Première Solution Entière $P.S.E.$) : 3,5 secondes en moyenne, 17 secondes au maximum.

	Param.					PLNE					PPC			RL		
id	$R_b - L$	L/P_b	|B|	|V|	|C|	S.O.	T (s)	P.S.E.	Gap (%)	TFS	S.O.	Gap (%)	T (s)	S.O.	Gap (%)	T (s)
s1	100	1 000	10	51	4 547	34 274	93,3	39 614	16	3,0	34 866	1,8	0,3	35 173	2,7	0,6
s2	100	1 000	8	59	965	19 637	0,3	19 637	0	0,1	21 579	9,9	0,5	20 119	2,5	0,2
s3	100	1 000	10	63	1 492	46 155	3,4	46 281	0	0,1	47 162	2,2	0,4	46 354	0,5	0,2
s4	100	1 000	10	25	1 135	22 564	0,5	22 564	0	0,1	24 070	6,7	0,2	25 254	12,0	0,2
s5	100	1 000	10	52	4 570	34 275	21,2	34 277	0	2,0	35 819	4,6	0,4	35 175	2,7	0,6
s6	100	1 000	10	119	5 430	21 539	34,8	24 348	13	17,0	27 571	28,1	0,9	23 952	11,3	1,3
s7	100	1 000	10	52	4 571	34 274	25,8	34 276	0	3,0	35 578	3,9	0,4	34 669	1,2	0,6
s8	100	1 000	10	73	9 482	55 999	23,5	56 299	1	3,0	56 325	0,6	0,6	57 409	2,6	1,4
s9	100	1 000	10	64	1 492	46 156	2,6	46 158	0	1,0	47 135	2,2	0,4	46 353	0,5	0,3
s10	100	1 000	8	73	1 213	21 155	0,7	21 155	0	0,1	24 760	17,1	0,5	22 062	4,3	0,2
s11	100	1 000	11	124	15 384	51 159	628,2	53 585	5	12,0	52 400	2,5	1,0	52 773	3,2	2,4
s12	100	1 000	11	89	13 750	39 308	76,4	39 309	0	12,0	46 253	17,7	1,0	39 724	1,1	1,7
s1	50	500	10	51	4 547	33 200	64,9	34 229	3	2,0	33 555	1,1	0,3	35 696	7,6	0,6
s2	50	500	8	59	965	19 158	0,4	19 158	0	0,1	20 844	8,9	0,5	19 280	0,7	0,2
s3	50	500	10	63	1 492	44 681	2,4	44 756	0	0,1	45 548	2,0	0,4	44 797	0,3	0,2
s4	50	500	10	25	1 135	21 823	0,8	21 823	0	0,1	23 189	6,3	0,2	22 562	3,4	0,2
s5	50	500	10	52	4 570	33 201	20,4	33 323	0	3,0	34 356	3,5	0,4	34 528	4,0	0,6
s6	50	500	10	119	5 430	20 937	8,3	20 939	0	6,0	26 652	27,3	0,9	21 993	5,1	1,2
s7	50	10	10	52	4 571	33 199	43,3	34 057	3	4,0	34 214	3,1	0,4	34 478	3,9	0,6
s8	50	500	10	73	9 482	54 483	1367,0	55 740	2	4,0	54 614	0,3	0,6	54 590	0,2	1,3
s9	50	500	10	64	1 492	44 682	1,0	44 682	0	0,1	45 374	1,6	0,4	45 961	2,9	0,2
s10	50	500	8	73	1 213	20 377	1,0	20 378	0	0,1	23 937	17,5	0,5	20 445	0,4	0,2
s11	50	500	11	124	15 384	50 165	546,4	50 361	0	17,0	51 141	2,0	1,0	50 273	0,3	0,1
s12	50	500	11	89	13 750	38 280	539,0	38 387	0	10,0	44 874	17,3	1,0	38 363	0,3	2,0
s1	20	10	10	51	4 547	35 718	53,1	35 818	0	3,0	36 258	1,6	0,3	38 228	7,1	0,6
s2	20	10	8	59	965	26 276	2,2	26 296	0	0,1	31 160	18,6	0,5	27 896	6,2	0,2
s3	20	10	10	63	1 492	52 034	6667,0	52 274	0	1,0	53 974	3,8	0,4	52 808	1,5	0,2
s4	20	10	10	25	1 135	22 536	1,3	22 596	0	1,0	24 696	9,6	0,2	22 858	1,5	0,1
s5	20	10	10	52	4 570	35 740	9,3	39 480	10	4,0	37 660	5,4	0,4	38 430	7,6	0,5
s6	20	10	10	119	5 430	31 202	35,1	36 430	17	8,0	54 164	73,6	0,9	35 186	12,8	1,2
s7	20	10	10	52	4 571	35 658	25,5	37 104	4	2,0	37 478	5,2	0,4	39 566	11,0	0,6
s8	20	10	10	73	9 482	57 696	43,5	59 136	2	7,0	58 270	1,0	0,6	61 942	7,4	1,2
s9	20	10	10	64	1 492	52 056	4,5	54 466	5	1,0	53 636	3,1	0,4	52 972	1,8	0,2
s10	20	10	8	73	1 213	23 170	3,7	25 430	10	0,1	31 944	37,9	0,5	24 946	7,7	0,2
s11	20	10	11	124	15 384	60 917	257,8	61 398	1	16,0	60 712	0,4	1,0	64 426	5,8	0,1
s12	20	10	11	89	13 750	46 330	20,4	46 510	0	13,0	61 996	33,9	1,0	52 710	13,8	1,5

FIGURE 2.4 – Résultats pour la PLNE, PPC et la Recherche Locale (RL).

		Solution PPC	Solution optimale PLNE	Première Solution entière PNLE	Solution RL
Rf= Lfmax + 100 Pf=L/1000	Tps min	0,1	0,2	0,1	0,1
	Tps max	0,9	628	17	2,3
	Tps moy	0,5	75	4,5	0,8
	Gap min	0,58 %	-	0 %	0,43 %
	Gap max	28,01 %	-	15,58 %	11,92 %
	Gap moy	8,05 %	-	2,85 %	3,65 %
Rf= Lfmax + 50 Pf=L/500	Tps min	0,1	0,3	0,1	0,1
	Tps max	0,9	1367	17	1,9
	Tps moy	0,6	216	3,9	0,57
	Gap min	0,24 %	-	0 %	0,20 %
	Gap max	27,3 %	-	3,1 %	7,52 %
	Gap moy	7,53 %	-	0,77 %	2,38 %
Rf= Lfmax + 20 Pf=L/10	Tps min	0,1	1.3	0,1	0,1
	Tps max	0,9	6667	16	1,5
	Tps moy	0,5	593	4,7	0,5
	Gap min	0 %	-	0,08 %	1,43 %
	Gap max	73,59 %	-	16,76 %	13,77 %
	Gap moy	16,07 %	-	4,2 %	6,97 %

FIGURE 2.5 – Résumé des résultats.

2.7 Synthèse

Le problème de minimisation du stock de banches est résolu à l'aide de trois méthodes que nous comparons sur un jeu de 36 instances réelles. Le deuxième modèle nous permet de faire des améliorations significatives en termes de qualité de la solution obtenue par rapport à l'approche historique utilisant une heuristique gloutonne à base de programmation par contraintes. On note un gain moyen de 9 % par rapport à l'approche par programmation par contraintes, ce qui représente une réduction significative des coûts de location pour un site. L'approche par recherche locale initialisée à l'aide de la relaxation du problème maître permet de maintenir un temps d'exécution inférieur à 3 secondes tout en produisant des solutions en moyenne à 5 % de l'optimum. En pratique, les ingénieurs en charge de la commande du parc de banches utilisent de manière interactive le logiciel intégrant cet algorithme, en résolvant plusieurs fois le problème tout en ajoutant des contraintes additionnelles.

Il est important de noter que le problème de minimisation du stock de banches introduit dans ce chapitre est une sous-partie d'une suite logicielle complète de recherche opérationnelle pour la construction. Le planning des voiles décidant le jour de coulage de chaque voile est établi dans une étape préliminaire. Ce premier sous-problème doit respecter les fenêtres de temps, les précédences, les exclusions mutuelles et une capacité de travail maximale journalière (ce maximum est un nombre de voiles maximum et une longueur maximale). Ce problème est résolu par un autre logiciel à base de programmation par contraintes, en quelques secondes. Le résultat de ce premier sous-problème forme l'entrée du problème de minimisation du stock de banches, qui alimente lui-même le 3^e problème, consistant à déterminer l'ordonnancement des trains de banches afin de minimiser les assemblages / désassemblages de trains. Ce dernier problème, connu sous le nom de problème de paire de banches, est résolu par un algorithme de recherche locale, mis en place par Benoist [16].

Chapitre 3

Optimisation de mouvements de terre sur des chantiers linéaires

Le problème d'optimisation de mouvements de terre sur des chantiers linéaires prend comme données d'entrée les résultats d'un des premiers problèmes de "recherche opérationnelle" connu sous le nom de problème de transport optimal de masse, encore appelé *problème de Monge-Kantorovich*. Le problème décrit ici concerne la planification des mouvements de terre par des engins de terrassement [1] : il s'agit de planifier la réalisation de ces mouvements de terre sur un horizon de plusieurs mois. Les tâches (les mouvements de terre) doivent donc être ordonnancées sur un certain nombre de machines parallèles (les engins de terrassements). À cette dimension combinatoire s'ajoute une problématique continue, relative à la répartition de la quantité de terre d'une tâche entre les différentes machines. Ce problème est donc un problème d'optimisation en variables mixtes. Un chantier peut contenir jusqu'à 400 ouvrages, 2000 mouvements de terre, 30 échelons sur une échelle temporelle pouvant dépasser les 2 ans. Comme expliqué en introduction de cette thèse, nous proposerons une approche de résolution par recherche locale traitant simultanément les dimensions combinatoire et continue de ce problème.

3.1 Contexte industriel

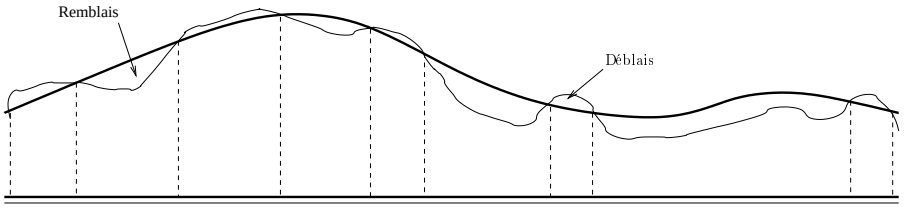
Également présente à l'international, Bouygues DTP Terrassement est spécialiste du terrassement en France. Cette filiale de Bouygues Construction conduit des projets de terrassement avec une flotte de 900 engins dont 600 à l'étranger. Ses activités vont du chantier de proximité au terrassement de routes, d'autoroutes, de lignes ferroviaires à grande vitesse

1. Les travaux de recherche présentés dans ce chapitre ont été menés de 2008 à 2011. Ils ont donné lieu à une publication [67] et plusieurs communications [68, 65, 25]. Des travaux complémentaires avec Thierry Benoist (Bouygues e-lab) et Vincent Jost (du LIX) ont permis d'étudier le cas à une ressource.

ou encore de mines à ciel ouvert. Sur les chantiers linéaires, DTP Terrassement a pour mission de générer la liste optimale des mouvements de terre, d'affecter ces mouvements à des ressources et de proposer un planning détaillé des plusieurs mois d'activités.

Les chantiers linéaires sont des autoroutes, des routes, des voies ferrées ou des canaux. Les travaux de terrassement incluent le déplacement de la terre bien évidemment, mais aussi le drainage et l'évacuation des eaux pluviales, la gestion des excédents et des carences en matériaux. L'activité dans sa globalité représente environ 25 % des coûts de construction d'une autoroute et environ 35 % des coûts de construction d'une ligne de chemin de fer. Le marché du terrassement représentait en 2008 en France 17 % du chiffre d'affaire des Travaux Publics. La somme des volumes déplacés cette même année était de plus de 76 millions de m^3, soit une quantité de terre suffisante pour recouvrir la surface de l'agglomération parisienne d'une couche de 30 cm de terre [3].

Un ouvrage linéaire est décrit par une ligne de référence (appelée aussi "Ligne Rouge") qui détermine un profil altimétrique cible (voir la Figure 3.1). Les ouvrages peuvent être projetés sur trois plans perpendiculaires à cette ligne de référence : un plan horizontal, un profil en long, un profil en travers. Dans les trois plans, les tracés doivent répondre à des caractéristiques de rayons et de pentes suivant la vitesse de référence des engins circulant à terme sur le tracé (par exemple, pour les voies ferrées, la vitesse de référence utilisée est 350 km/h). Les pentes maximales autorisées sont définies pour chaque type d'ouvrage ainsi que les rayons minima et maxima des creux et des collines. La différence d'altitude entre le profil initial et la cible permet de déduire le volume, les types de matériaux et la localisation de chaque ouvrage. Ceux-ci sont de deux types : les déblais d'où on doit extraire des matériaux et les remblais où on doit en importer. Le profil altimétrique cible est conçu en respectant les contraintes de vitesse de référence tout en limitant le volume total de terre déplacée, afin que le chantier s'auto-satisfasse. On parle ici du "calage de la ligne du projet" en trouvant la meilleure adéquation entre ces contraintes. L'étude des sols en terrassement influence aussi la définition de ce profil cible. Elle aide à déterminer la composition et le taux de compactage du terrain et permet de calculer la hauteur des couches de matériaux qui composent les ouvrages.



Les déblais et les remblais sont définis en fonction du profil initial et du profil cible.

FIGURE 3.1 – Profil altimétrique d'un chantier.

D'autres activités complémentaires doivent être réalisées au cours d'un chantier de terrassement, comme par exemple le drainage et l'évacuation des eaux pluviales qui peuvent

parfois représenter une part non négligeable des coûts du chantier total. La bonne gestion des excédents et des carences en matériaux est aussi un enjeu important du chantier. Les excédents de terrassement sont considérés comme des déchets traités à la charge du terrassier. Cette contrainte complexifie l'étude car elle peut engendrer des surcoûts importants à l'échelle du chantier. Ces exports de matériaux seront donc à minimiser, quitte à les traiter sur place pour les réutiliser. De même, les carences en matériaux doivent être comblées par l'import de matières premières provenant de carrières extérieures. Ces matériaux sont achetés à des fournisseurs et l'objectif est de limiter ces emprunts, surtout si les matériaux peuvent être extraits du chantier.

Dans la Section 3.2, nous présentons un modèle du problème de transport optimal de masse permettant de générer la liste des mouvements de terre servant de données d'entrée au problème de planification. En Section 3.3, nous introduisons une modélisation de ce problème de planification de mouvements de terre, puis nous nous focalisons en Section 3.4 au cas à une ressource, pour lequel nous détaillons l'étude de complexité, un algorithme exact à base de programmation dynamique et enfin un calcul de bornes inférieures. Enfin, en section 3.5, nous présentons la recherche locale que nous avons implémentée pour traiter ce problème de planification de manière pure et directe, sans décomposition. Nous fournissons des détails concernant les mouvements utilisés ainsi que l'algorithmie d'évaluation sous jacente. La Section 3.6 présentent les résultats obtenus sur des jeux de données provenant de 12 chantiers de DTP Terrassement.

3.2 Transport de masse

Le transport optimal de masse est initialement introduit par Monge en 1781 dans son *mémoire sur la théorie des déblais et des remblais* [93], revisité par Kantorovich en 1942 [72] ainsi que par Dantzig en 1949 [35]. Les articles de la littérature sur les mouvements de terre ne concernent pas uniquement les autoroutes ou les voies ferrées [7, 73, 41, 75, 86, 96]. Certains traitent aussi de problématiques rencontrées lors de la construction d'aéroports [26], de routes de forêt [4], de bases militaires [42] ou plus généralement de gestion d'espaces urbains [42].

Les ouvrages $o \in O$ sont composés de déblais $d \in D \subset O$ et de remblais $r \in R \subset O$. Chaque ouvrage est défini par son abscisse notée $A(o)$ et sa liste de couches notée $C(o)$. La taille de cet ensemble peut varier suivant les ouvrages et certains sont composés d'une couche unique. Chaque couche c est définie par une quantité notée $Q(c)$ à déplacer si c'est une couche source (appartenant à un déblai) ou à recevoir si c'est une couche destination (appartenant à un remblai). Chaque couche est composée d'un unique matériau. Un mouvement de terre consiste à déplacer une certaine quantité du matériau d'une couche origine vers une couche destination. Il faut noter que le problème est séparable par matériau. Dans toute la suite de la description du problème, nous considérerons qu'il y a un unique matériau. La Figure 3.2 modélise un chantier avec 5 blocs, composés chacun d'une ou plusieurs couches, entre lesquels

différents mouvements de terre sont réalisés.

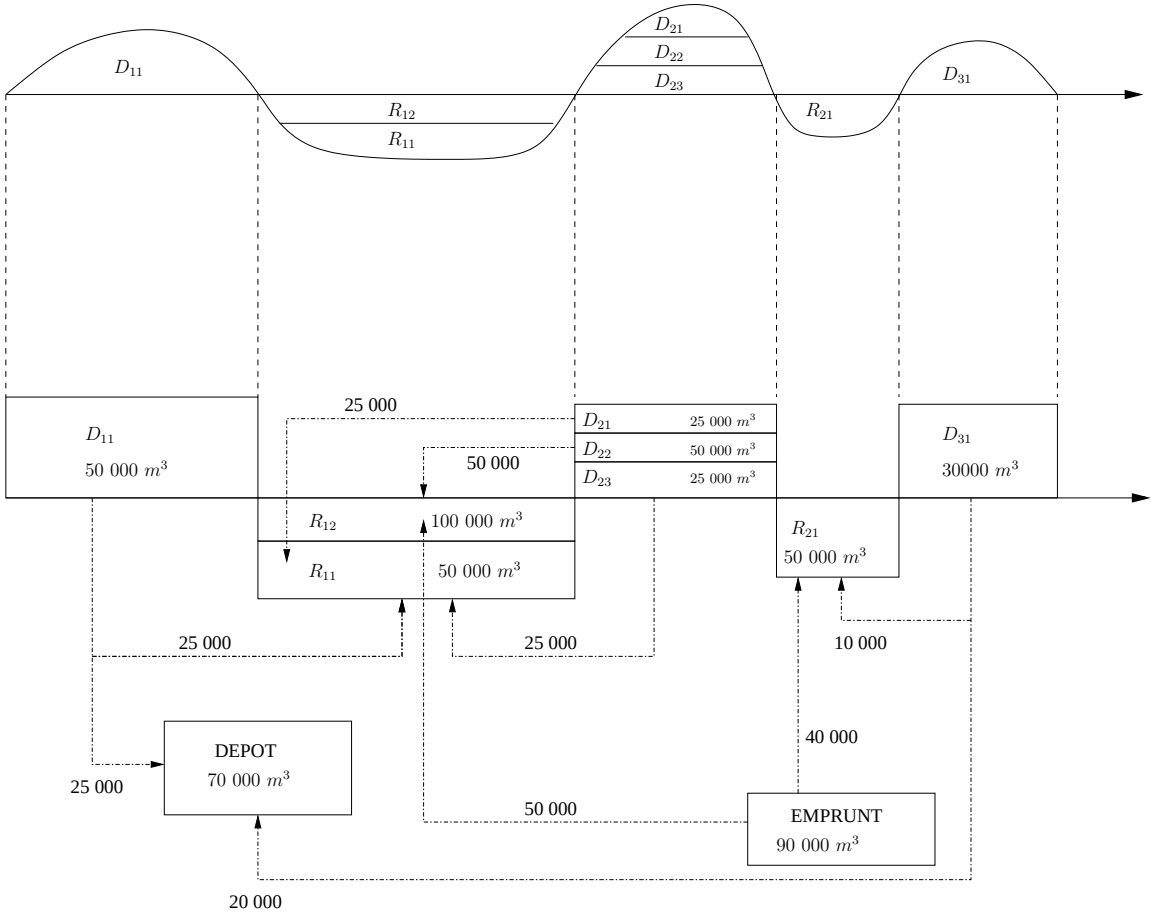
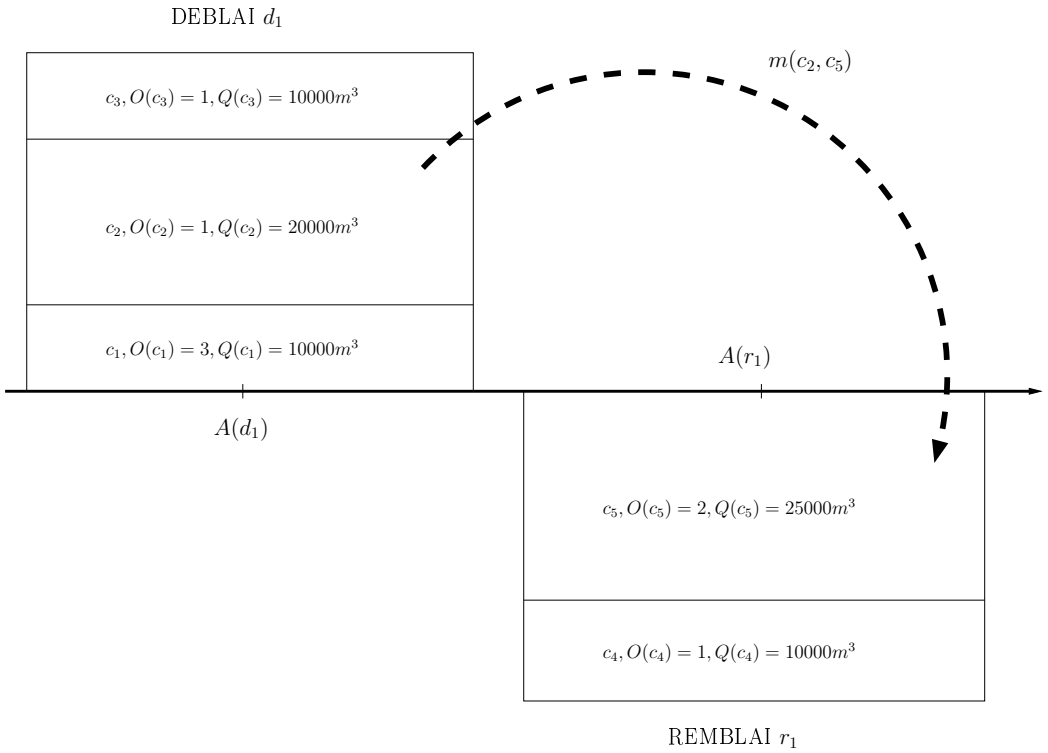


FIGURE 3.2 – Mouvements de terre entre blocs.

Un mouvement de terre $m(i,\ j)\ \in\ M$ consiste à déplacer une quantité notée $q(m)$ depuis la couche source $i\ \in\ c(u)$ d'un ouvrage u vers une couche destination $j\ \in\ c(v)$ d'un ouvrage v ayant besoin d'une certaine quantité de ce type de matériau. On notera $S_{sortant}(c)$ l'ensemble des mouvements de terre qui quittent la couche c et $S_{entrant}(c)$ les mouvements qui apportent des matériaux à la couche c. La Figure 3.3 résume ces notations.

L'objectif du problème de génération des mouvements de terre est de minimiser les moments de transport, c'est-à-dire le produit de la quantité transportée à chaque mouvement par la distance parcourue entre les blocs sources et destinations. Tous les déblais doivent être vidés et tous les remblais doivent être remplis avec la quantité exacte demandée, en utilisant la quantité disponible dans ces déblais. Des emprunts extérieurs notés $x\ \in\ X$ peuvent être sollicités. Ces carrières jouent le rôle de sources de capacité infinie et sont définies comme les ouvrages mais avec un coût, ce qui permet de modéliser le surcoût associé à leur utilisation. Les emprunts ne sont pas utilisés si le problème peut être résolu uniquement

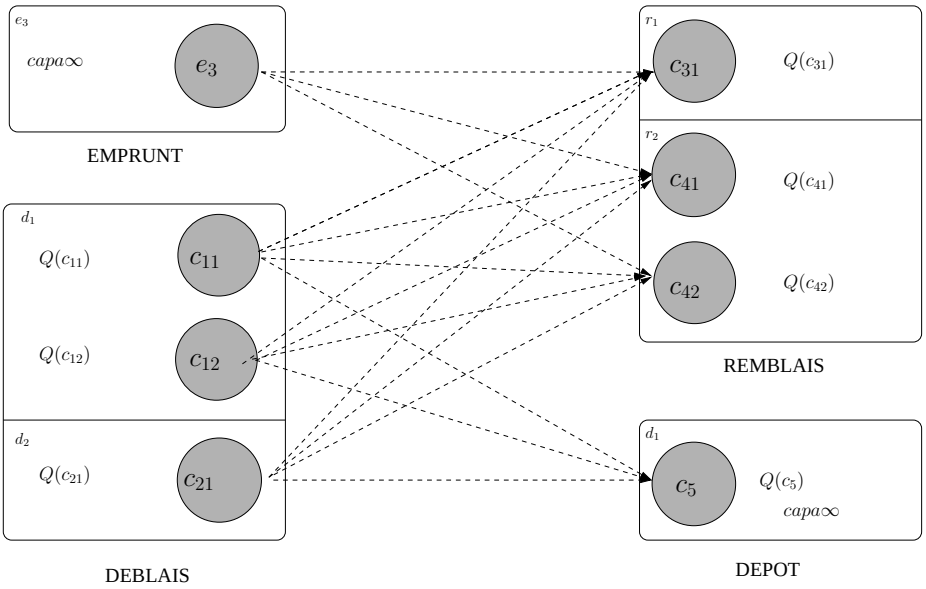


Un mouvement de terre d'une couche d'un déblai vers une couche d'un remblai.

FIGURE 3.3 – Mouvement de Terre.

avec les déblais en place. De même, pour garantir la faisabilité du problème, des dépôts sont ouverts, de capacité infinie, qui peuvent s'avérer nécessaires afin de stocker des surplus de matériaux. Ces dépôts $b \in B$ sont définis eux aussi comme des ouvrages avec une abscisse $A(b)$ et une liste de couches $C(b)$ ayant chacune une capacité d'accueil $Q(C(b))$. On a donc $O = D \bigcup R \bigcup E \bigcup B$. Les dépôts ne jouent le rôle que de remblais. Tous les dépôts ne sont pas forcément vides à l'issue de la réalisation d'un chantier. Le Schéma 3.4 présente le problème de flot de coût minimum.

Le Modèle 3.1 présente une modélisation mathématique du problème.



Chaque type de bloc est représenté : les déblais, les remblais, le dépôt et l'emprunt.

FIGURE 3.4 – Flot de coût minimal.

$$
\begin{cases}
min \displaystyle\sum_{\substack{i \in C(u) \,/\, u \in D \cup E, \\ j \in C(v) \,/\, v \in R \cup B}} q(m(i,\, j)) \,.\, |\, A(u) \,-\, A(v) \,| & (E1) \\
s.t. & \\
\forall\, u \,\in\, D,\, \forall\, i \,\in\, C(u), \displaystyle\sum_{\substack{j \in C(v) \\ / \, v \in R \cup B}} q(m(i,\, j)) \,=\, Q(i) & (E2) \\
\forall\, v \,\in\, R,\, \forall\, j \,\in\, C(v), \displaystyle\sum_{\substack{i \in C(u) \\ / \, u \in E \cup D}} q(m(i,\, j)) \,=\, Q(j) & (E3) \\
\forall\, v \,\in\, B,\, \forall\, j \,\in\, C(v), \displaystyle\sum_{\substack{i \in C(u) \\ / \, u \in D}} q(m(i,\, j)) \,\leq\, Q(j) & (E4)
\end{cases}
\tag{3.1}
$$

L'objectif (*E1*) est donc de minimiser le moment total de transport. (*E2*) indique que la somme des quantités partant d'un déblai vers les remblais ou les dépôts doit être égale à sa capacité. (E3) indique que la somme des quantités arrivant à un remblai depuis un déblai ou un emprunt doit être égale à sa capacité. (E4) indique que la somme des quantités arrivant à un dépôt depuis un déblai ou un emprunt doit être inférieure à sa capacité. Par définition,

les mouvements de terre ne peuvent pas quitter un remblai ou un dépôt, ni arriver à un déblai ou un emprunt. La résolution de ce premier problème d'optimisation permet d'établir de manière optimale la liste des mouvements de terre à effectuer sur le chantier avec un volume associé. Si $q(m(u,v)) = 0$, aucun mouvement entre u et v n'est créé.

Ce problème est résolu de manière efficace et optimale en utilisant la programmation linéaire. Chaque couche est définie par l'ordre d'exécution par rapport aux autres couches du même bloc. Pour des raisons physiques logiques, les couches supérieures d'un déblai seront faites avant toutes les autres des couches inférieures, et inversement pour les remblais. Ces contraintes physiques génèrent des précédences comme nous le verrons en Section 3.3 qui peuvent entraîner des cycles de précédences. Ces éventuels problèmes de cycles sont réglés manuellement en utilisant des critères métiers, par exemple en activant ou désactivant des dépôts ou des emprunts supplémentaires. Dans la section suivante, nous décrirons le problème d'affectation des mouvements de terre générés aux ressources, avant de s'intéresser dans un second temps à la stratégie de résolution de ce problème d'optimisation multi-ressources.

3.3 Planification des mouvements de terre

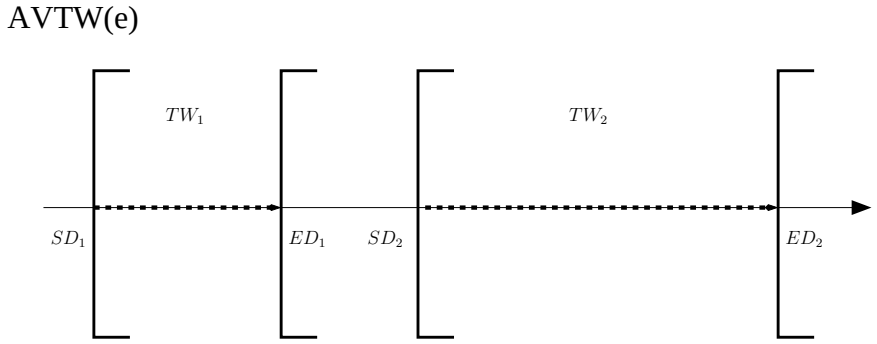
3.3.1 Modélisation

Les mouvements de terre générés par le problème de transport optimal de masse servent de données d'entrées au problème de planification. L'ensemble M représente l'ensemble des mouvements de terre m à planifier. La réalisation totale d'un mouvement de terre peut varier de quelques jours à plusieurs mois, en fonction du volume de matériaux à déplacer. Une des difficultés majeures du problème réside dans le fait que les matériaux constituant les ouvrages en déblais et en remblais doivent être déplacés et mis en œuvre dans un ordre qui ne correspond pas toujours à l'ordre d'extraction (la couche inférieure rocheuse d'un déblai, c'est-à-dire la dernière extraite, peut être utilisée en fond de remblai, soit la première mise en œuvre).

Planifier les mouvements de terre d'un chantier consiste à générer des tâches notées $t \in T$. On note $m(t) \in M$ le mouvement de terre associé à la tâche t qui consiste à déplacer le matériau d'une couche d'un ouvrage vers une autre en utilisant la capacité de travail d'un des échelons $e(t) \in \mathit{echelons}(m(t)) \subseteq E$ autorisés pour ce mouvement. Un échelon est constitué d'un engin de production et de plusieurs moyens de transport (par exemple, une pelle reliée à 5 camions). L'association d'un échelon $e(t)$ à un mouvement de terre $m(t)$ définit la cadence d'une tâche t, notée $C(t)$, qui s'exprime en volume déplacé par heure. Chaque mouvement de terre m peut être réalisé par plusieurs tâches qui doivent toutes être définies dans les intervalles de réalisation $TW(m)$. Ces fenêtres de temps sont plutôt larges (plusieurs semaines) et permettent d'exprimer des contraintes de disponibilités suivant les zones géographiques du chantier. En effet, la construction de certains ouvrages (voie d'accès,

ponts, ...) peut bloquer certains mouvements de terre pendant une période plus ou moins grande.

Pour chaque mouvement de terre $m \in M$, on connaît la liste des échelons $echelons(m)$ en mesure de le réaliser. Par exemple, des échelons sont plus adaptés à travailler la roche, d'autres sont uniquement utilisés pour déplacer de la terre meuble. Chacun des échelons e est disponible suivant un calendrier définissant des heures de travail journalier, qu'on matérialise par des fenêtres de temps $AVTW(e)$. Cette liste d'intervalles temporels est définie par une date de début (inclue) et une date de fin (exclue), comme indiqué sur la Figure 3.5. En fonction des conditions météorologiques ou des contraintes de disponibilités des ressources, certaines plages horaires sont indisponibles. Des contraintes physiques additionnelles, comme par exemple la finalisation d'un ouvrage d'art, génèrent des délais partiels et donc des contraintes additionnelles de disponibilités. Chaque tâche t est définie par une date de début $debut(t)$, une date de fin $fin(t)$, avec $debut(t) < fin(t)$. On note $A_s(t)$ (resp. $A_d(t)$), l'abscisse de la source (resp. de la destination) du mouvement associé.



Les dates de début de chaque intervalle sont incluses et les dates de fin sont exclues.

FIGURE 3.5 – Les fenêtres de disponibilité.

Enfin, $debut(m)$ et $fin(m)$ définissent respectivement le début et la fin d'un mouvement de terre m, c'est-à-dire le minimum des $debut(t)$ et le maximum des $fin(t)$ de toutes les tâches $t \in T(m)$ satisfaisant le mouvement de terre m. On note une précédence $p(m_1; m_2) \in P$ lorsqu'un mouvement de terre m_1 doit avoir lieu avant un mouvement de terre m_2. Si la réalisation d'un mouvement de terre entre deux points facilite un autre mouvement de terre plus grand passant par cette zone géographique, il sera planifié avant. Sur le schéma 3.6, le mouvement de terre m_2 contribuera à la réalisation du mouvement de terre m_1 et une précédence stricte est donc posée : $p(m_2,\ m_1)$. Cependant, si les mouvements de terre sont croisés, on ne peut pas imposer de conditions car on ne peut pas déduire quel mouvement sera plus intéressant.

Le Modèle 3.2 propose une formalisation de ce problème de planification. L'équation (E6) indique que pour chaque mouvement de terre, la quantité transportée par ses tâches doit être égale à la quantité attendue. Comme précisé en (E7), chaque tâche ne peut être réalisée

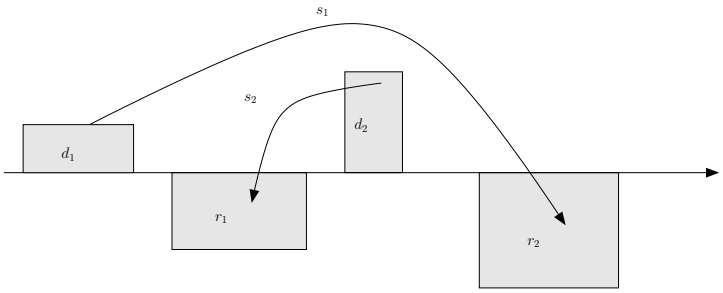


FIGURE 3.6 – Mouvement de terre inclus dans un autre.

que pendant les fenêtres de temps autorisées pour ce mouvement de terre, tandis que la contrainte (E8) garantit que les fenêtres de disponibilités des ressources sont respectées. Les équations (E9) et (E10) définissent les variables $debut(m)$ et $fin(m)$ pour chaque mouvement de terre m. La contrainte (E11) permet d'assurer que les tâches ne se chevauchent pas. Les contraintes sur les précédences sont spécifiées en (E12) : tous les mouvements de terre des couches du même ouvrage ayant un ordre plus grand que la couche en cours doivent être faits après ceux de cette couche. Le graphe de précédences est enrichi avec des contraintes de précédences supplémentaires visant à modéliser les contraintes d'aplanissement de la route ainsi que des contraintes additionnelles fixées par les opérationnels.

$$
\begin{cases}
\sum_{t \in T} \; (\; fin(t) \; - \; debut(t) \;) \; . \; C(t) \; = \; \sum_{m \in M} Q(\; m(t) \;) & (E6) \\
\forall \; t \; \in \; T, \; t \; \subseteq \; TW(m(t)) & (E7) \\
\forall \; t \; \in \; T, \; t \; \subseteq \; AVTW(e(t)) & (E8) \\
\forall \; m \; \in \; M, \; debut(s) \; = \; \min_{t \in T(s)} \; debut(t) & (E9) \\
\forall \; m \; \in \; M, \; fin(s) \; = \; \max_{t \in T(s)} \; fin(t) & (E10) \\
\forall \; e \; \in \; E, \; \forall \; t_1, \; t_2 \; \in \; T(e), \\
(\; fin(t_1) \; < \; debut(t_2) \;) \; \bigvee \; (\; debut(t_1) \; > \; fin(t_2) \;) & (E11) \\
\forall \; p(m(t_1), \; m(t_2)) \; \in \; P, \; fin(t_1) \; \leq \; debut(t_2) & (E12)
\end{cases}
\tag{3.2}
$$

Les contraintes entre couches génèrent un graphe de précédences entre les mouvements de terre. Le graphique 3.7 illustre un ensemble de précédences entre mouvements de terre sous la forme d'un treillis, que nous obtenons par analyse des précédences entre couches. Chaque nœud du graphe représente un des mouvements de terre potentiellement réalisable par la ressource et chaque arc orienté, une relation de précédence entre les deux mouvements

de terre.

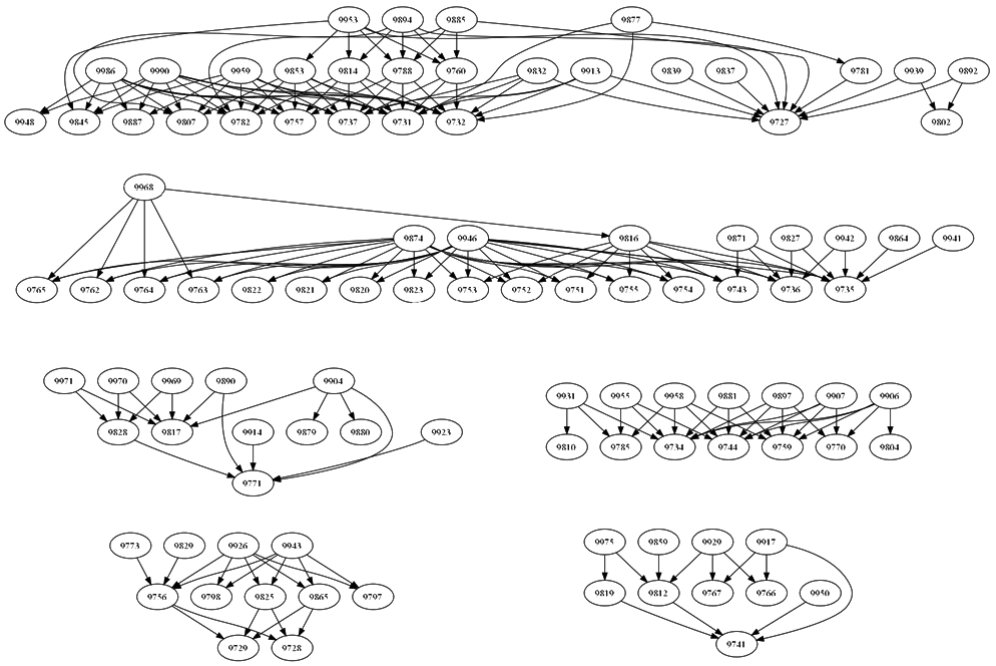


FIGURE 3.7 – Graphes de précédences d'une ressource

Nous cherchons ici à optimiser l'utilisation des ressources. Pour cela, on tente de minimiser le nombre total de ressources ouvertes, notée *nbResources* c'est-à-dire le nombre total de ressource réalisant au moins une activité sur le chantier. On définit aussi la distance cumulée *totalDistance* parcourue par la ressource entre les sources de chacun de ses mouvements.

$$totalDistance = \sum_{e \in E} \sum_{\substack{t \in T(e) \ / \\ t(e,m) \neq t_0(e,\ m)}} | A_s(t(e,\ m)) - A_s(precT(t(e,\ m))) | + A_s(t_0(e,\ m)) \tag{3.3}$$

La notation $precT(t(e,\ m))$ représente la tâche précédant $t(e,\ m)$ dans l'ensemble $T(e)$ et $t_0(e,\ m)$ est la tâche avec $debut(t(e,\ m))$ la plus petite de $T(m)$. Enfin, on note *coutTemporel* le coût temporel, c'est-à-dire la somme pour toutes les ressources du nombre d'heures passées sur le chantier entre la date d'arrivée et la date de départ multipliée par le coût horaire $c(e)$ de l'échelon e.

$$coutTemporel = \sum_{e \in E} c(e) \times (fin(Derniere(T(e))) - debut(Premiere(T(e)))) \tag{3.4}$$

Premiere() et *Derniere*() représentent respectivement les premières et dernières tâches de l'ensemble de tâches $T(e)$ de l'échelon e. Ces trois composantes du coût total des ressources sont combinées en utilisant des coefficients linéaires. L'objectif global s'écrit donc :

$$min\ (\alpha\ .\ nbResources\ +\ \beta\ .\ totalDistance\ +\ \gamma\ .\ totalDuration\ ) \tag{3.5}$$

L'optimisation de plannings de mouvements de terre est donc un problème en variables mixtes. Les aspects combinatoires concernent ici l'affectation des mouvements aux machines de terrassements (en respectant leur planning de disponibilités) et les aspects continus sont relatifs aux quantités de terre déplacées et au placement de l'activité dans le planning de la ressource. Ce modèle étant décrit en détails, nous présentons dans la section suivante la complexité du problème et l'état de l'art sur ce sujet.

3.3.2 État de l'art et complexité

Le problème de la littérature se rapprochant le plus du problème de planification des mouvements de terre est le Problème de planification $P|prec; pmtn; r_i|C_{max}$ qui est une généralisation du Problème $P|prec; pmtn; p_i = 1|C_{max}$ prouvé comme NP-difficile par Ullman [107]. Si on considère que nous avons un seul matériau, que le graphe de précédences est acyclique, que les durées des activités p_i sont toutes identiques, alors si $\alpha = 0$ et $\beta = 0$, le problème est NP-Difficile. Dans le cas β non nul, $\alpha = 0$ et $\gamma = 0$, alors le problème est aussi NP-Difficile. Dans ce cas, on a une seule ressource qui fait une unique tournée, en visitant notamment un point p_0 devant être vu avant tous les autres points et placé à l'origine de l'axe et un point p_n devant être visité après tous les autres et placé à l'extrémité de l'axe. À noter que si le graphe de précédences prend la forme d'un "out-tree", le problème est dans P.

Parmi les travaux de la littérature traitant de la planification des mouvements de terre, les travaux publiés par Mayer *et al.* [88] proposent une approche par programmation linéaire. Leur objectif est de générer les mouvements de terre tout en minimisant le nombre de déplacement de matériel entre les sections non-adjacentes. Dans notre algorithme, nous retrouvons cet objectif qui minimise le nombre de kilomètres parcourus par les ressources entre deux activités.

Dans les travaux présentés par l'équipe de Marzouk *et al.* [86, 87], Moselhi *et al.* [96] et ensuite dans la thèse de Alshibani [5], les auteurs présentent une approche à base de métaheuristiques s'appuyant sur des algorithmes génétiques et la programmation linéaire en nombres entiers afin d'optimiser les plannings des mouvements de terre. Ce sont les travaux de la littérature qui se rapprochent le plus de l'étude présentée ici. Les premiers travaux de recherche de Marzouk *et al.* [86] utilisent un algorithme génétique. Les travaux qui ont suivi [5, 87, 96] présentent un système d'information géographique couplé à un moteur d'optimisation, afin d'aider les planificateurs dans la recherche d'une solution fonctionnelle. Dans

cette modélisation, les plannings des ressources et des sous-traitants, les contraintes de budget et les caractéristiques des équipements sont pris en compte. Deux exemples numériques sont présentés pour des instances comptant environ 3 millions de mètres cube de terre. Les durées de simulation sont de l'ordre d'une heure. Leur outil permet de tester de nouvelles hypothèses en termes de ressources, de coûts et de lancer une ré-optimisation à la suite de mesures sur le terrain, en utilisant aussi les relevés GPS de positionnement des engins. L'ensemble du système d'information comporte un module de simulation, une base de données d'équipements avec leur coûts et un module de génération de rapports.

Dans la même lignée des outils d'aide à la décision, Askew *et al.* [7] décrivent un outil pour les planificateurs. À partir des données d'entrées saisies et d'une base de connaissances, l'ensemble des mouvements de terre est simulé en génèrant les activités à réaliser. Même si le modèle semble prendre en compte un certain nombre de contraintes, peu d'informations sont fournies concernant les techniques de résolution utilisées. Un système expert est utilisé avec des règles de type : les mouvements de petite taille sont planifiés d'abord, les points de départ doivent être situés près de points d'accès adaptés, les mouvements de chargements sont évités en amont, etc. Un graphique présente des instances de 2,6 millions m^3 sur un chantier de 25 km mais ne détaille par les résultats numériques. Henderson *et al.* [61] présentent un modèle utilisant une heuristique de recuit simulé qui optimise la façon dont les équipements sont utilisés en réduisant la distance totale couverte par les équipements, permettant ainsi de réduire la consommation de carburant et les durées d'interventions.

Enfin, plusieurs travaux de la littérature ont cherché à inclure la phase d'établissement de la ligne rouge aux phases de génération et de planification des mouvements de terre (Easa *et al.* [41], Moreb [94], Kataria *et al.* [75]) en couplant optimisation et simulation. Avant de présenter l'algorithme de recherche locale développé pour résoudre ce problème de manière efficace et robuste, nous nous intéressons au problème avec une unique ressource.

3.4 Étude du cas à une ressource

3.4.1 Introduction

Nous nous intéressons dans cette section au problème de planification des mouvements de terre, avec une unique ressource. Les contraintes temporelles imposées par les fenêtres de temps sont ici ignorées. Ce problème de décision est un problème de voyageur de commerce linéaire. Après l'énoncé de quelques définitions et propriétés relatives à ce problème, nous nous intéresserons à l'étude de la NP-difficulté de ce problème, ce qui permet de démontrer que le problème d'optimisation des mouvements de terre reste NP-difficile dans le cas à une ressource (comme nous l'avons déjà vu dans la section précédente). Une recherche arborescente permet de déterminer une solution pour les petites instances. Nous montrerons qu'une approche par programmation dynamique permet de résoudre des instances de plus grande taille. Cette étude se focalise aussi sur des bornes inférieures dont nous expliquerons le mode de calcul.

Le problème de décision du voyageur de commerce linéaire consiste à visiter un ensemble de points (ou nœuds) positionnés sur un axe, en réalisant un parcours de longueur inférieure à une longueur K. Trivial sous cette forme, le problème devient plus complexe si l'on y ajoute des précédences entre les points (TSP-UP). Étant donné un ensemble P de n points ayant chacun une abscisse X_i ($i \in [1,\ n]$) et un ensemble C de m contraintes de précédences $(i,\ j) \in P^2$, le problème consiste à trouver une permutation V des points S telle que :

$$\sum_{i=1}^{n} | X_{V(i)} - X_{V(i-1)} | \leq K, \text{ avec } X_{V(0)} = 0$$
$$\forall (i,\ j) \in C,\ V^{-1}(i) < V^{-1}(j), \text{ avec } V^{-1}(i) \text{ la position de l'objet } i$$

On notera dans la suite de cette section $N_2 \succ N_1$, lorsque le point N_1 doit être visité avant le point N_2. Le Schéma 3.8 présente un exemple de TSP-UP avec quatre points. Le jeu de précédences entre ces points est défini par le graphe situé à gauche. Le graphique avec les pointillés matérialise un parcours de visites des points à chaque abscisse. À notre connaissance, ce problème n'a pas encore été étudié dans la littérature. Après quelques définitions et propriétés introduites en Section 3.4.2, nous nous intéresserons à la complexité de ce problème dans la section 3.4.3, puis la Section 3.4.4 permet de montrer qu'en considérant le nombre minimal de traversées de chaque tronçon entre deux abscisses consécutives, on peut calculer une borne du problème. Nous présentons ensuite en Section 3.4.5 une recherche arborescente, exploration que nous rendons plus efficace en définissant un algorithme de programmation dynamique.

3.4.2 Définitions et propriétés

Lemme. Le problème de planification des mouvements de terre sur un chantier linéaire avec une ressource et sans fenêtre de temps est un TSP-UP.

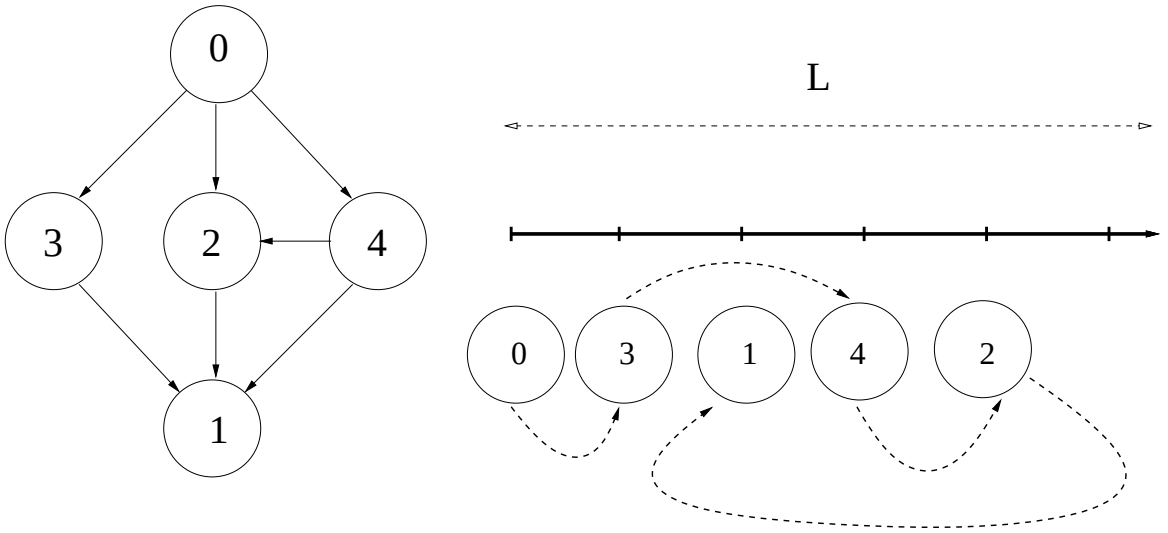


FIGURE 3.8 – Un problème de voyageur de commerce linéaire avec précédences.

Preuve. S'il n'y a qu'une ressource, la minimisation du nombre de ressources n'est plus un objectif.

En l'absence de fenêtre de temps, tout planning peut être modifié en supprimant les éventuels temps d'attente, c'est-à-dire en plaçant le début de chaque tâche à la fin de la tâche précédente. Cette transformation n'a pas d'impact sur le nombre de kilomètres parcourus (l'ordre étant préservé) et ne peut que diminuer la largeur de la fenêtre d'utilisation de la ressource : celle-ci atteint alors son minimum, à savoir la somme des durées nécessaires pour chaque mouvement de terre. Les termes *nbResources* et *totalDuration* de la fonction objectif sont donc des constantes et l'objectif se réduit à la minimisation de la distance totale. Notons enfin que si un mouvement de terre est réalisé par plusieurs tâches, la solution peut encore être transformée en repoussant toutes ces tâches juste avant la dernière de cet ensemble. Cette seconde transformation préserve le respect des précédences (puisqu'un mouvement de terre n'est terminé qu'à la fin de sa dernière tâche) et ne peut que diminuer les kilomètres parcourus. Ces deux relations de dominance prouvent que chercher une solution optimale au problème de planification des mouvements de terre est équivalent à chercher une permutation des tâches respectant les précédences et minimisant le nombre de kilomètres parcourus, ce qui est la définition du TSP-UP. ■

Lemme. Le problème de TSP-UP appartient à NP.

Preuve. Si on donne une permutation, le calcul de sa longueur est linéaire. Vérifier que toutes les précédences sont respectées se fait en temps linéaire en le nombre d'arêtes du graphe de précédences (soit en $O(|C|)$). Le problème est donc dans NP. Nous démontrerons dans la Section 3.4.3 que le TSP-UP est NP-complet. ■

Définition. Une solution est dite *non-procrastinante* si à chaque passage à une abscisse,

tous les points qui sont à cette abscisse, et pour lesquels on a visité tous les prédécesseurs, sont visités.

Lemme. L'ensemble des solutions *non-procrastinante* est dominant pour le problème.

Preuve. Soit une permutation V contenant au moins une procastination, c'est-à-dire un point d qui soit traversé sans être visité alors que tous ses prédécesseurs ont déjà été visités. Nous allons montrer que la solutions est strictement dominée. Appelons a et b ces deux points consécutifs dans la permutation tels que :

$$X_d \in [X_a, X_b[\quad ou \quad X_d \in]X_b, X_a] \tag{3.6}$$

et que tous les tous les prédécesseurs de d (dont éventuellement a) sont supposés être avant a dans la permutation V (voir Figure 3.9).

Par hypothèses, le point d est visité ultérieurement, entre 2 points c et e. Notons que c peut éventuellement être égal à b et que e peut ne pas exister si d est le dernier point visité, sans que cela n'affecte la validité du raisonnement qui va suivre.

La longueur associée à la permutation V' est inférieure ou égale à la longueur de V car :

- le trajet $a \rightarrow d \rightarrow b$ est égal au trajet $a \rightarrow b$ (d étant entre a et b).
- le trajet $c \rightarrow e$ est inférieur ou égal au trajet $c \rightarrow d \rightarrow e$ puisque les distances sur un axe respectent l'inégalité triangulaire (et en l'absence de e le trajet $c \rightarrow d$ est simplement supprimé.

En répétant cette transformation, toutes les procastinations peuvent être éliminées, tout en conservant ou diminuant la longueur de la permutation V. On peut donc construire à partir de V une permutation V' non procastinante de longueur inférieure ou égale à celle de V.

Ainsi, il existe pour tout problème une permutation *dominante* optimum. ■

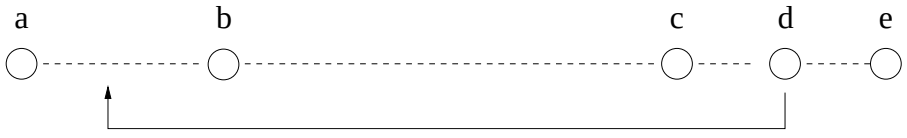


FIGURE 3.9 – Règle de dominance.

Corollaire. Une solution du TSP-UP avec n points peut aussi s'écrire sous la forme d'une liste de t abscisses avec $t \leq n$ (on parle ici de chemin).

La séquence des abscisses induit une séquence de visite des noeuds (unique à l'ordre près des noeuds d'une même abscisse). Le TSP-UP peut ainsi se re-formuler en un problème de décision : existe-t-il une séquence d'abscisses (ou chemin) qui ait une longueur plus petite que K et qui visite tous les points ?

3.4.3 Complexité

Pour établir la NP-complétude du problème de voyageur de commerce linéaire avec précédences, nous démontrons qu'il existe une réduction polynomiale du problème de la plus courte super-séquence commune en un problème de TSP-UP.

Plus courte super-séquence commune. X est une super-séquence de Y si et seulement si toutes les lettres de Y apparaissent dans X dans le même ordre mais pas forcément de manière consécutive, en d'autres termes si on supprime un certain nombre de caractères de X, on retrouve la séquence Y. Le problème de la plus courte super-séquence commune (SCS, de l'anglais *shortest common supersequence*) est défini par n séquences $X_i = < x_1, ..., x_m >$. Une séquence $U = < u_1, ..., u_k >$ est une super-séquence commune aux X_i si U est individuellement une super-séquence de chacun des séquences X_i. L'objectif du problème est de savoir s'il existe une super séquence commune de k lettres au plus.

Maier [84] ont prouvé la NP-complétude du problème pour un alphabet de taille supérieur à 5. Räihä et Ukkonen [99] ont prouvé la NP-complétude du problème pour un alphabet de taille supérieure ou égale à 2.

Réduction du SCS en TSP-UP

Soit un problème de plus courte super-séquence commune avec n séquences et un alphabet binaire. Cet alphabet de 2 lettres est décrit ainsi : D pour "Droite" et G pour "Gauche".

Propriété. À partir d'un problème de SCS à alphabet binaire avec n séquences, un problème de TSP-UP peut être construit linéairement.

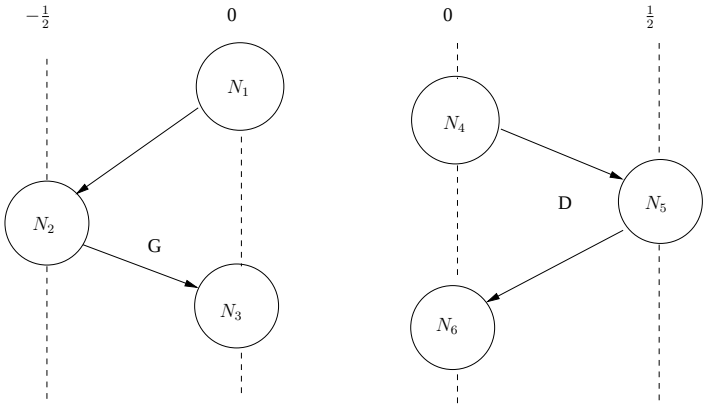
Preuve. On se donne trois abscisses sur un axe : $(-\frac{1}{2};\ 0;\ \frac{1}{2})$. On créé un point de départ sur le point d'abscisse 0. Pour chaque séquence i de longueur k_i, on construit une chaîne de $(2k_i)$ points (appartenant chacun à une de ces 3 abscisses) et $(2k_i)$ arêtes. Chaque lettre est codée sur le modèle suivant, comme illustré sur la Figure 3.10 :

- pour G, N_1 est le dernier point de la chaîne (nécessairement à l'abscisse 0), on créé le point N_2 à l'abscisse $-\frac{1}{2}$ et le point N_3 à l'abscisse 0, on définit deux précédences entre ces 3 points : $N_1 \succ N_2$ et $N_2 \succ N_3$.
- pour D, N_4 est le dernier point de la chaîne (nécessairement à l'abscisse 0), on créé le point N_5 à l'abscisse $\frac{1}{2}$ et le point N_6 à l'abscisse 0, on définit deux précédences entre ces 3 points : $N_4 \succ N_5$ et $N_5 \succ N_6$.

Les n séquences i de longueur k_i définissent $\sum_{i=1}^{n} 2k_i + 1$ points et $\sum_{i=1}^{n} 2k_i$ arêtes. Le parcours est donc linéaire en le nombre de caractères dans la chaîne, donc linéaire en la taille du nombre de séquences. ■

Lemme. Si le problème SCS a une solution de longueur L alors le TSP-UP a une solution de longueur L.

Preuve : La solution de SCS de longueur L est une séquence de L lettres qui peuvent chacune être remplacée par une paire d'abscisses avec la règle suivante : chaque lettre G est



Exemple de séquence “Gauche” (G) et séquence “Droite” (D)

FIGURE 3.10 – Séquence de précédences à 1 caractère.

remplacée par la séquence d'abscisses $(-\frac{1}{2};\ 0)$ et chaque lettre D par $(\frac{1}{2};\ 0)$. On débute la séquence par l'abscisse 0 et on définit ainsi une suite de visites d'abscisse de l'axe. On a donc une séquence de $2L+1$ abscisses dont chacune est à une distance $1/2$ de son prédécesseur (soit une longueur totale égale à L). les séquences d'abscisses de chacune des chaînes construites dans l'instance de TSP-UP construite à partir de l'instance SCS apparaissent donc dans la séquence d'abscisses construite ci-dessus, donc cette séquence permet de visiter tous les points. Elle est donc une solution (de longueur L) du TSP-UP. ■

Propriété. Si le problème TSP-UP a une solution de longueur L, alors l'instance originale SCS a aussi une solution de longueur L.

Preuve. Une solution du TSP-UP est une suite de n points de longueur L exprimée comme à une liste d'abscisses appartenant à l'ensemble $A = (-\frac{1}{2};\ 0;\ \frac{1}{2})$, sans perte d'informations : toute sous-séquence $(-\frac{1}{2};\ \frac{1}{2})$ peut être exprimée comme $(-\frac{1}{2};\ 0;\ \frac{1}{2})$ et inversement $(\frac{1}{2};\ -\frac{1}{2})$ peut être remplacée par $(\frac{1}{2};\ 0;\ -\frac{1}{2})$. Pour aller d'un point externa à l'autre, le point central 0 doit être visité sans que cela ne rallonge la longueur du trajet.

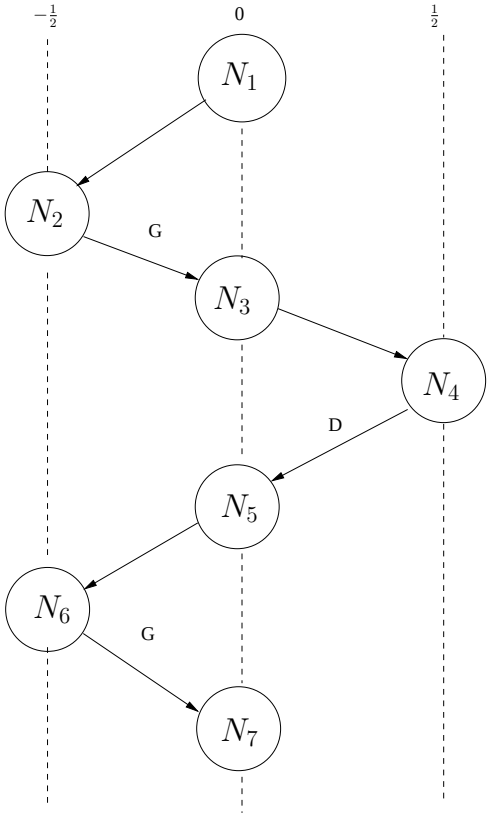
On obtient alors une liste d'abscisses dont un élément sur deux est 0 (dont le premier), donc chaque abscisse est à une distance de $\frac{1}{2}$ de son prédécesseur. La longueur de ce chemin étant L il y a donc $2L+1$ abscisses dans cette séquence. A partir de la deuxième abscisse, on remplace chaque séquence $(-\frac{1}{2};\ 0)$ par la lettre G et chaque séquence $(+\frac{1}{2};\ 0)$ par la lettre D. On obtient ainsi un mot de longueur L sur l'alphabet G, D.

Puisque la séquence d'abscisses est une solution du TSP-UP, elle visite tous les points et contient donc la liste des abscisses de chaque chaîne. Pour chaque mot X du problème SCS la séquence contient donc la séquence d'abscisses correspondante, donc le mot de longueur L construit ci-dessus est une superséquence du mot X. Ce mot de longueur L est donc une superséquence commune à tous les mots de l'instance SCS considérée. ■

Conclusion. Il existe une réduction polynomiale du problème de la plus courte super-

séquence commune ("Shortest Common Supersequence") en un problème de TSP-UP. Ce problème étant NP-complet ([84, 99]), le TSP-UP est donc lui aussi un problème NP-complet.

Exemple à trois caractères. Soit une séquence GDG. Chaque lettre d'une séquence définit une décision prise à l'abscisse 0, D si on repart vers la droite (i.e. vers l'abscisse $\frac{1}{2}$) ou G vers la gauche (i.e. vers l'abscisse $-\frac{1}{2}$). On débute à un nœud de départ N_1 et on a donc une première décision qui nous conduit en N_2 $(-\frac{1}{2})$, puis on revient au centre en N_3 car on n'a pas d'autre choix, ensuite on a une seconde décision qui nous conduit en $N_4(\frac{1}{2})$, puis on revient au centre en N_5. La dernière décision amène à visiter le point $N_6(-\frac{1}{2})$ et à revenir au centre en N_7. La Figure 3.11 résume cet exemple simple.



Exemple de séquence "Gauche-Droite-Gauche"

FIGURE 3.11 – Exemple de séquences à 3 caractères.

3.4.4 Bornes inférieures

On cherche une borne inférieure pour le problème TSP-UP. On note G le graphe acyclique des précédences où chaque nœud correspond à un point i ayant pour abscisse X_i sur l'axe

du chantier.

Borne triviale B_0. Une première borne inférieure du problème est la longueur L, correspondant à la distance entre les abscisses minimale et maximale.

Définition. Un *tronçon* $[X_g;\ X_d]$ est une partie de l'axe linéaire entre deux abscisses voisines. L'axe est donc découpé en tronçons disjoints : partant de l'origine, un nouveau tronçon est créé dès qu'il existe au moins un nœud à cette abscisse (Voir le graphe de gauche sur la Figure 3.14). La longueur de tronçon t est :

$$L(t) \ = \ | \ X_d \ - \ X_g \ | \tag{3.7}$$

Nous proposons ici de raffiner la borne B_0 en définissant une borne B_1 calculée par tronçon.

Cas particulier à 2 abscisses. Dans le cas où le graphe ne compte que deux abscisses (voir Figure 3.12), le calcul est trivial. Il n'y a qu'une seule solution possible : une visite du point de gauche ne peut conduire qu'à un déplacement à droite et inversement une visite du point de droite ne peut conduire qu'à un déplacement à gauche. Comme démontré précédemment, il est toujours optimal de traiter tous les nœuds disponibles à cette abscisse lors de la visite d'une abscisse.

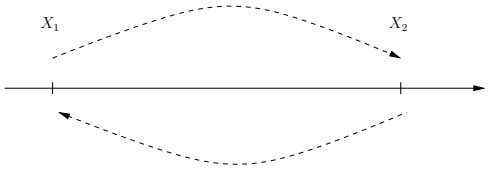


FIGURE 3.12 – Cas particulier à 2 abscisses.

Cas à n abscisses ($n \ > \ 2$) : dans le cas à n abscisses, on calcule une borne inférieure à partir de la résolution de $|T|$ problèmes à 2 abscisses (avec $|T| \ \leq \ n-1$). Pour chaque tronçon $t \in T$, le problème s'écrit sous la forme d'un problème à 2 abscisses, en transformant le graphe de précédences avec la règle suivante. Tous les nœuds à gauche de l'abscisse gauche du tronçon t sont considérés comme étant des nœuds à l'abscisse 0 et les nœuds à droite de l'abscisse droite comme étant des nœuds à l'abscisse $L(t)$. L'algorithme présenté dans la section suivante permet de déduire le nombre minimum de fois qu'un tronçon sera traversé en partant du nœud initial puis en alternant les déplacements de droite à gauche, tout en visitant à chaque abscisse tous les nœuds disponibles.

Algorithme. L'Algorithme 2 présente le calcul de cette borne. Le parcours du graphe pour modifier les abscisses du problème à n abscisses afin de traiter $|T|(\leq n-1)$ problèmes à 2 abscisses est réalisé en $O(|P|)$. On définit $opt_g(t)$ (resp. $opt_d(t)$), le nombre minimum de parcours du tronçon t (voir Figure 3.13) en considérant que le nœud terminal (autrement dit, le dernier point de la permutation) se trouve à une abscisse située à gauche (resp. à droite) du tronçon.

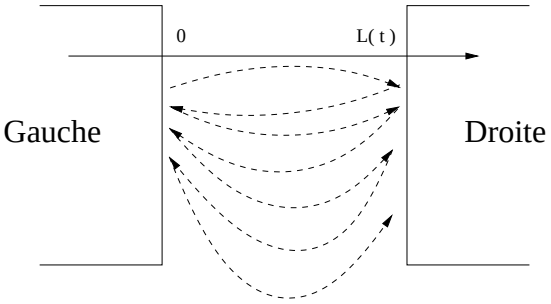


FIGURE 3.13 – Parcours d'un tronçon.

La borne $B(k)$ correspond à la borne dans le cas où la dernière abscisse est l'abscisse k. Elle est définie par la formule de récursion suivante :

$$B(0) = \sum_{t \in T} opt_g(t)$$
$$\forall\, k \in [1,\ |T|],\ B(k) = B(k-1) - opt_g(k-1) + opt_d(k-1)$$

Soit X_f l'ensemble des abscisses qui ont au moins un point sans successeurs. La borne du problème est définie ainsi :

$$Borne = \min_{k \in X_f} B(k) \tag{3.8}$$

Pour chaque tronçon $t \in T$, le calcul de $opt_g(t)$ est réalisé en $O(\ n + |C|\ )$ avec n le nombre de points et $|C|$ le nombre de précédences entre ces points. Le calcul de B_0 se fait en $O(\ |T|\ )$ avec $|T| \leq n$ et les calculs des B_t se font en $O(1)$. Le calcul de cette borne est en $O(\ |T|\ (\ n + |C|\ )\ )$. Comme $|T|$ est inférieur à n et $|C|$ est inférieur à n^2, la calcul se fait donc en n^3 dans le pire des cas.

Exemple. Soit un TSP linéaire avec précédences, composé de 6 nœuds régis par le graphe de précédences de la Figure 3.14. Trois nœuds terminaux sont ici candidats : N_1, N_4 et N_5. L'Algorithme 2 considère les cinq tronçons un par un. Sur la gauche de la Figure 3.14, le tronçon t_2 est isolé. Le graphe de précédences est ensuite considéré pour obtenir un problème à deux abscisses sur le tronçon t_2 entre les abscisses 2 et 5. Il en résulte le nouveau graphe de précédences en bas à droite de la Figure 3.14 défini entre 0 et $L(t_3) = 3$.

Algorithme 2: BORNE-PAR-TRONÇON

input : L'ensemble des abscisses, des points et le graphe de précédences
output : Une borne inférieure de la distance parcourue

begin
  for *Chaque tronçon t du graphe de précédences* **do**
    Créer le problème à deux abscisses associé avec le nœud de départ à l'abscisse 0
    Distance = 0
    AbscisseCourante = 0
    Visiter tous les nœuds disponibles à AbscisseCourante
    while *Tous les nœuds n'ont pas été visités* **do**
      Changer AbscisseCourante pour l'autre abscisse
      $Distance \mathrel{+}= L(t)$
      Visiter tous les nœuds disponibles à AbscisseCourante
    if $AbscisseCourante = X_d$ **then**
      $opt_g(t) = Distance + L(t)$ $opt_d(t) = Distance$
    else
      $opt_g(t) = Distance$ $opt_d(t) = Distance + L(t)$
  $B_0 = \sum_{t \in T} opt_g(t)$
  for $k \in [1, |T|]$ **do**
    $B_k = B_{k-1} - opt_g(t_{k-1}) + opt_d(t_{k-1})$
  Retourner $\min_{k \in X_F} B_k$

Le calcul de la sous-borne sur le tronçon t_2 débute par la visite du nœud de départ N_0 à l'abscisse de gauche 2. N_1 ne peut pas être visité puisque ce nœud est encore lié par deux contraintes de précédences à N_2 et N_3. Une première traversée du tronçon s'impose donc. Une fois à l'abscisse de droite 5, N_2 et N_3 peuvent être visités car la contrainte de précédence n'est plus active suite à la visite du nœud N_0. Ces deux visites permettent de visiter ensuite N_4 et N_5. Il n'y a plus de nœud candidat à cette abscisse, mais il reste encore un nœud à visiter à l'abscisse 2. Le tronçon est donc traversé une deuxième fois et N_1 est visité, ce qui achève le parcours du graphe. Donc $B_2 = 2 \times 3 = 6$.

Pour chaque tronçon t, $opt_g(t)$ et $opt_d(t)$ valent donc :

$$\begin{array}{llll}
opt_g(t_1) = 2 \times 2 = 4 & et & opt_d(t_1) = 1 \times 2 = 2 \\
opt_g(t_2) = 2 \times 3 = 6 & et & opt_d(t_2) = 3 \times 3 = 9 \\
opt_g(t_3) = 2 \times 2 = 4 & et & opt_d(t_3) = 3 \times 2 = 6 \\
opt_g(t_4) = 2 \times 1 = 2 & et & opt_d(t_4) = 1 \times 1 = 1 \\
opt_g(t_5) = 2 \times 2 = 4 & et & opt_d(t_5) = 1 \times 2 = 2
\end{array}$$

et on en déduit les B_k suivants :

$$\begin{array}{l}
B_0 = \sum_{t \in T} opt_g(t) = 4 + 6 + 4 + 2 + 4 = 20 \\
B_1 = B_0 + opt_g(t_0) + opt_g(t_0) = 20 - 4 + 2 = 18 \\
B_2 = B_1 + opt_g(t_1) + opt_g(t_1) = 18 - 6 + 9 = 21 \\
B_3 = B_2 + opt_g(t_2) + opt_g(t_2) = 21 - 4 + 6 = 23 \\
B_4 = B_3 + opt_g(t_3) + opt_g(t_3) = 23 - 2 + 1 = 22 \\
B_5 = B_4 + opt_g(t_4) + opt_g(t_4) = 22 - 4 + 2 = 20
\end{array}$$

Les nœuds terminaux candidats sont N_1, N_4 et N_5, donc la valeur finale de la borne vaut donc $min(B_1,\ B_4,\ B_5) = 18$.

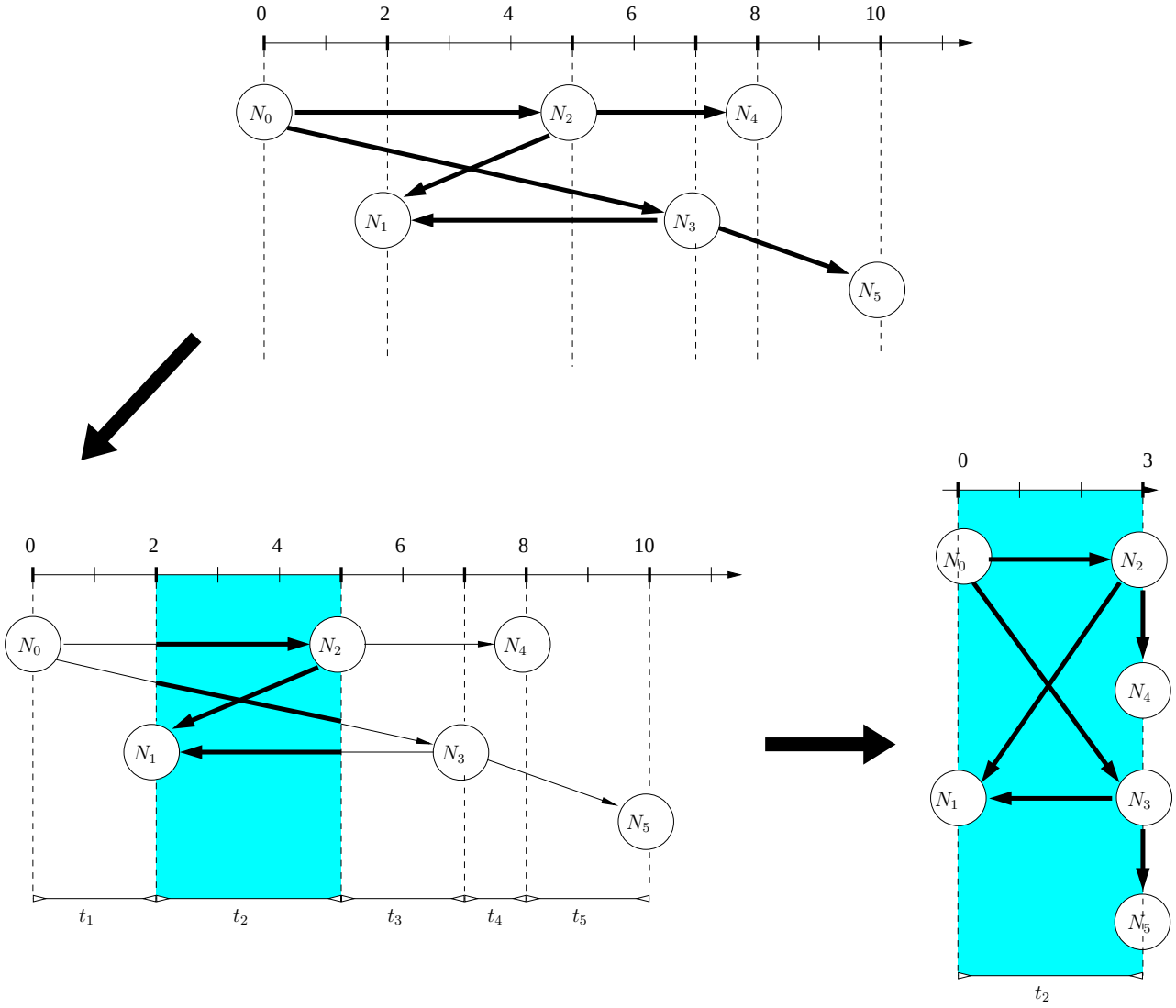


FIGURE 3.14 – Graphe de précédences, isolement pour calcul du tronçon t_2.

Contre-exemple. Le cas suivant est un exemple de non-optimalité de cette borne. Soient deux chemins croisés, entre des nœuds appartenant à trois abscisses différentes comme présenté sur la Figure 3.15. Il n'y a ici qu'un seul nœud terminal possible (N_7). Le calcul sur le tronçon t_2 donne une borne de 6 et le calcul sur t_3 donne 4, soit une borne totale de 10. Or, le chemin optimal respectant les précédences est de 20 : $N_0 \ -2 \rightarrow \ N_2 \ -2 \rightarrow \ N_3 \ -3 \rightarrow \ N_1 \ -3 \rightarrow \ N_4 \ -2 \rightarrow \ N_6 \ -2 \rightarrow \ N_7 \ -3 \rightarrow \ N_5 \ -3 \rightarrow \ N_7$:

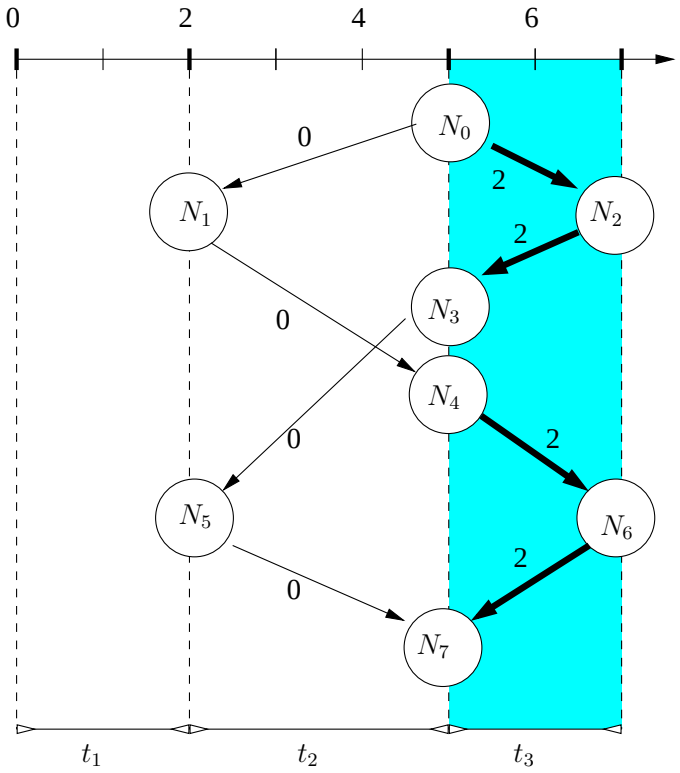


FIGURE 3.15 – Calcul de bornes sous-optimal.

Cette borne, que l'on notera B_0 dans la suite, permet d'obtenir de très bons résultats pour des temps de calcul très courts. Nous présenterons la comparaison des résultats de cette borne comparés à ceux de la recherche locale dans la Section 3.4.6.

3.4.5 Algorithme exact

On définit dans cette section une recherche arborescente dans laquelle on utilise la borne B_1 introduite à la section précédente. Une solution au problème de TSP-UP est une suite de décisions droite / gauche prises à chaque abscisse visitée. En partant de l'origine, l'arbre de décision contient donc 2^n points et non $n!$. Afin de diminuer encore le nombre de nœuds visités par rapport à une énumération complète, on fait appel à une programmation dynamique de type Held et Karp [59] stockant les états déjà rencontrés. On fait appel à la valeur de l'état enregistrée pour éviter de visiter à nouveau le sous-arbre associé. De plus, on garde en mémoire la meilleure solution courante et on évite de visiter un sous-arbre si la valeur courante est déjà supérieure à cette meilleure solution (voir M.G. De La Banda et al. [37]). Le pire cas de l'algorithme de type Held et Karp est $n\ 2^n$, la programmation dynamique n'améliore pas le pire cas mais nous montrerons en pratique son efficacité.

On définit un état $s_u(t)$ comme une paire $(X,\ P)$ où X est un identifiant de l'abscisse courant et P est l'ensemble des points à visiter. Comme illustré dans la Figure 3.16, la valeur de Bellman enregistrée à chaque état correspond à la plus courte distance nécessaire pour satisfaire entièrement le problème de TSP-UP à partir de ce point, c'est-à-dire la distance minimale nécessaire pour visiter les k points restants. Un état correspond donc à une abscisse et la liste des nœuds restants à visiter. À chaque état, on stocke la meilleure longueur courante restant à parcourir. Si on retrouve l'état courant, on ne parcourt pas à nouveau la branche mais on utilise cette valeur. On évite ainsi un nouveau parcours du sous-arbre associé, pour lequel la valeur optimale a déjà été calculée.

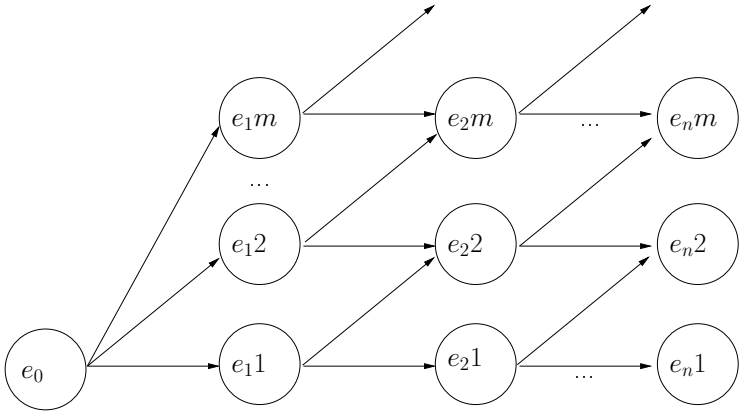


FIGURE 3.16 – Programmation dynamique.

L'Algorithme 3 résume le fonctionnement de ce parcours arborescent. Une coupe utilisant la borne $borneInf$ permet d'accélérer la recherche. On appelle cette fonction récursive avec les paramètres suivants : on débute à l'abscisse $X = 0$, avec l'ensemble des points à visiter P contenant initialement tous les points et une borne supérieure $U = +\infty$. On note $plusPresDroit(X,\ P)$ (resp. $plusPresGauche(X,\ P)$) la première abscisse à droite (resp. à

gauche) de l'abscisse courante X où il y a des points à visiter (c'est-à-dire une abscisse avec des points sans prédécesseur ou avec des prédécesseurs qui ont déjà été visités), $dispo(X)$ la liste des points disponibles à l'abscisse X (ces points peuvent avoir des précédences entre eux mais on sait qu'on pourra les traiter tous en venant en X). Les lignes qui ont le label $PgDyn$ sont celles qui sont spécifiques à la programmation dynamique. Elles font appel à des structures de données permettant de stocker des états : pour une abscisse X et une liste de points restants à visiter P, $optimaStockes[X,\ P]$ enregistre la meilleure solution déjà rencontrée au cours de l'exploration, tandis que $borneInfStockee[X,\ P]$ stocke la borne inférieure. La ligne labelisée BI permet d'élaguer les branches de l'arbre qui ne permettront pas d'obtenir une solution strictement meilleure que U. L'Algorithme 4 détaille le calcul de cette borne. On utilise pour cela la fonction $plusAGauche()$ (resp. $plusADroite()$) qui renvoie l'abscisse la plus à gauche (resp. à droite) de l'axe avec des points candidats. La valeur de ces deux fonctions est maintenue dynamiquement, mise à jour quand on termine les points le plus à gauche et à droite. La borne présentée en Section 3.4.4 pourrait être utilisée ici à la place de cette borne qui est moins précise, mais sa complexité est quadratique en temps.

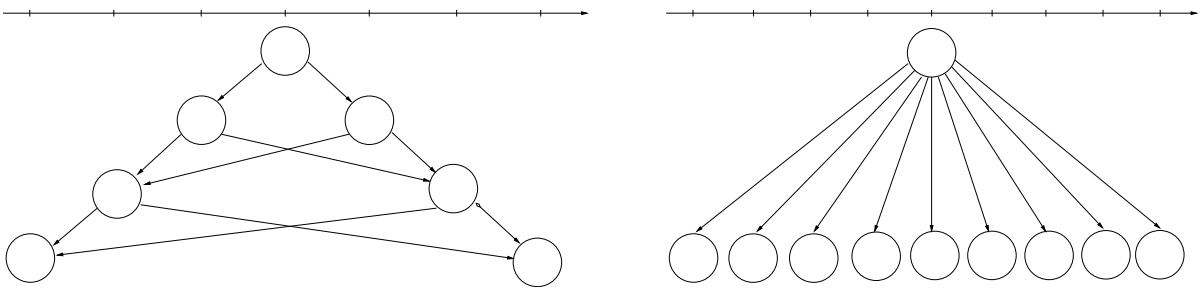


FIGURE 3.17 – Cas particuliers de graphes de précédences.

La Figure 3.17 présentent deux cas particuliers de graphes de précédences. Dans le cas de précédences fortes (à gauche), l'énumération complète se fait en visitant 2^n nœuds (un choix binaire par étage), tandis que la programmation dynamique visite 4 n nœuds (4 états par étage). Dans le cas d'un arbre distribué autour d'un point central placé au milieu de l'axe (à droite), l'énumération complète visite 2^n nœuds et la programmation dynamique n^2 nœuds. Dans le cas trivial où il n'y a aucune précédence et où toutes les abscisses sont positives, l'énumération et la programmation dynamique visitent n nœuds.

3.4.6 Résultats numériques

Le Tableau 3.1 décrit les neuf instances A utilisées pour l'évaluation de ces bornes. Les instances réelles ont été modifiées afin de les transformer en instances mono-ressource. Une des ressources est sélectionnée, une cadence de travail importante lui est affectée et ses fenêtres de temps sont fusionnées pour en créer une unique, débutant à la date d'ouverture du chantier et terminant à sa date finale. Toutes les fenêtres de temps contraignant la réalisation

Algorithme 3: LONGUEURMIN(X,P,U)

input : L'abscisse courante X, les points restants à visiter P, une borne sup. U
output : La longueur de la meilleure solution trouvée si $< U$, sinon U

begin

if $P = \varnothing$ **then Return** 0

PgDyn **if** *optimaStockes*$[X, P]$ **then Return** optimaStockes[X, P]

PgDyn **if** *not(borneInfStockee*$[X, P]$*)* **then** borneInfStockee[X, P] = borneInf(X, P)

PgDyn **if** *borneInfStockee*$[X, P] \geq U$ **then Return** U

BI **if** *borneInf*$(X, P) \geq U$ **then Return** U

$X_R = plusPresDroit(X, P)$

$X_L = plusPresGauche(X, P)$

$Meilleure = U$

if $X_R \neq +\infty$ **then**

$distanceDroite = X_R - X$

$Meilleure = distanceDroite +$
$longueurMin(X_R, P \setminus dispo(X_R), Meilleure - distanceDroite)$

if $X_L \neq -\infty$ **then**

$distanceGauche = X - X_L$

$Meilleure = distanceGauche +$
$longueurMin(X_L, P \setminus dispo(X_L), Meilleure - distanceGauche)$

PgDyn **if** $Meilleure < U$ **then** *optimaStockes*$[X, P] = Meilleure$

PgDyn **else** *borneInfStockee*$[X, P] = U$

Return min $Meilleure, U$

des mouvements de terre ont subi le même traitement afin de s'assurer que la résolution du problème s'abstrait de toutes contraintes temporelles. La première colonne (Inst) indique le nom de l'instance, puis nous indiquons des données décrivant le chantier : le nombre de blocs (nbBlocs), le nombre de couches total (nbCou), la longueur du chantier (long), le nombre de mouvements de terre à planifier à l'aide de cette ressource (nbMvtr) c'est-à-dire le nombre de points (ou nœuds), le nombre de précédences (m) et le nombre d'abscisses (x).

Le Tableau 3.2 présente la borne simple de longueur du chantier (l) en mètres, la borne calculée par l'algorithme des tronçons (Tr) en mètres et le temps nécessaire à son calcul (Tr tps) en secondes, l'optimum obtenu par programmation dynamique (DP), le nombre de de points (DP pts) et de nœuds (DP nds) visités, et le temps nécessaire pour calculer cet optimum (DP tps). Enfin, on retrouve la valeur obtenue par recherche arborescente (RA), le nombre de points (RA pts) et de visites d'abscisses (RA vst) et le temps nécessaire pour calculer cet optimum (RA tps).

Algorithme 4: BORNEINF(X,P)

input : L'abscisse courante X, les points restant à visiter P
output : La longueur de la meilleure solution trouvée si inferieure a U, sinon U

begin
  if $P = \varnothing$ **then** **Return** 0
  $L = plusAGauche(P, X)$
  $R = plusADroite(P, X)$
  Return $R - L + min(X - L, R - X)$

Inst	nbBlocs	nbCou	nbMvtr	m	x	long
A01	234	234	270	853	91	33,6
A02	61	89	99	763	22	8
A03	15	21	8	35	8	5,5
A04	57	63	55	205	15	13,215
A05	30	30	27	55	12	2,705
A06	46	85	83	185	20	5,2
A07	37	37	32	65	13	3,5
A08	200	219	239	493	75	52,1
A09	99	104	130	317	37	48,8

TABLE 3.1 – Benchmarks : Description des instances mono-ressource.

Le deuxième Tableau 3.3 présente une comparaison entre la solution obtenue sur ces instances mono-ressources. Dans la première colonne, on présente le nom de l'instance, puis la solution obtenue par l'algorithme glouton (G), par recherche locale (LS), la longueur du chantier (L), la borne par tronçon (TR), l'optimum par programmation dynamique (DP). L'instance *A01* ne termine pas avec la recherche arborescente sans programmation dynamique.

Inst	L(m)	Tr	Tr tps (s)	DP	DP pts	DP nds	DP tps (s)	RA	RA pts	RA vst	RA tps (s)
A01	33 290	50 138	4 670,253	66 580	153 850	307 974	130,721	79 654	Stop	Stop	4500
A02	8 000	17 152,5	580,206	18 165	1 448	3 081	0,533	18 165	809 327	2 278 037	1,778
A03	5 500	5 500	0,99	5 500	8	19	0,053	5 500	8	19	0,018
A04	13 215	13 410	3,5	19 140	399	901	0,118	19 140	9 541	21 715	0,076
A05	2 705	2 705	0,272	2 705	12	28	0,0603	2 705	12	28	0,0603
A06	5 200	12 770	1,964	12 770	141	418	0,176	12 770	1 634	4 284	0,116
A07	3 510	3 790	0,482	4 070	15	37	0,076	4 070	41	93	0,018
A08	52 145	52 145	1 027,407	52 145	75	240	1,664	52 145	75	240	0,119
A09	48 800	68 975,5	182,259	84 700	117	326	0,175	84 700	1 439	3 189	0,012

TABLE 3.2 – Résultats du calcul des bornes (1).

Inst	L	TR	DP	G	LS
A01	33 290	50 138	66 580	2 887 803	112 556
A02	8 000	17 152,5	18 165	99 185	18 165
A03	5 500	5 500	5 500	5 500	5 500
A04	13 215	13 410	19 140	65 310	37 740
A05	2 705	2 705	2 705	27 830	2 705
A06	5 200	12 770	12 770	103 700	12 700
A07	3 510	3 790	4 070	35 210	4 070
A08	52 145	52 145	52 145	1 833 701,5	56 000
A09	48 800	68 975,5	84 700	1 461 197	84 700

TABLE 3.3 – Résultats du calcul des bornes (2).

3.5 Approche par recherche locale

Après cette section consacrée uniquement au problème mono-ressource sans contrainte temporelle, nous nous intéressons de nouveau au cas général. L'objectif de cette section est de présenter une heuristique à base de recherche locale pour le résoudre de manière efficace et robuste. Ce problème d'optimisation en variables mixtes se prête parfaitement à l'élaboration de la stratégie décrite dans le chapitre 1. Dans cette section, nous décrirons tout d'abord la stratégie de recherche, puis les mouvements adaptés à la structure du problème. Enfin, nous préciserons l'algorithmique d'évaluation des mouvements en détaillant notamment quelques structures de données utiles pour rendre cette recherche locale plus efficace.

3.5.1 Stratégie et heuristique

La stratégie de recherche est ici de type *first-improvement descent* avec choix stochastique des mouvements. Aucune métaheuristique avec des cœfficients-paramètres n'est utilisée ici. L'algorithme glouton d'initialisation ne peut pas garantir l'obtention d'une solution où tous les mouvements de terre sont affectés mais un premier objectif est de trouver une première solution avec tous les mouvements de terre affectés aux ressources. Ensuite, on s'intéresse à l'optimisation des coûts associés aux ressources. Chaque phase dispose de son propre ensemble de transformations dédié à l'optimisation de son objectif. La liste des mouvements utilisés dans chaque phase sera précisée ultérieurement ainsi que l'algorithme d'initialisation. L'Algorithme 5 résume le fonctionnement de cette heuristique de descente.

Algorithme 5: STOCHASTIC-DESCENT

input : Liste des mouvements de terre, ressources, blocs et couches
output : Liste des tâches planifiées

begin
  Calcul d'une solution initiale S avec l'algorithme glouton
  Optimisation des quantités non affectées
  while *quantiteNonAffectee* $>$ 0 *et LimiteTemps n'est pas atteinte* **do**
    Choisir aléatoirement une transformation T dans l'ensemble $\mathcal{T}_{MISS}$
    if *le gain n'est pas négatif* **then** Valider T
    else Annuler T
  Optimisation des ressources
  while *coutsRessources* $>$ 0 *et LimiteTemps n'est pas atteinte* **do**
    Choisir aléatoirement une transformation T dans l'ensemble $\mathcal{T}_{RES}$
    if *le gain n'est pas négatif* **then** Valider T
    else Annuler T
  Retourner la liste des tâches créées

3.5.2 Solution initiale

L'algorithme d'initialisation est un algorithme glouton qui ordonne les mouvements de terre (voir $ClasserMouvements()$ dans 6) selon les valeurs décroissantes des critères suivants :

- le rang de la couche source des mouvements et celui de la couche destination en cas d'égalité ;
- la distance entre les centres de gravité du bloc source et du bloc destination ;
- la date de début de la fenêtre de temps cible ;
- la quantité de matériau à déplacer.

Une fois ordonnés, l'algorithme essaye d'affecter ces mouvements de terre aux ressources autorisées, elles-même ordonnées suivant les critères suivants (cf $ClasserRessources(e)$ dans l'Algorithme 6) : on privilégie tout d'abord les ressources déjà "ouvertes", c'est-à-dire celles à qui on a déjà affecté du travail. Le critère de tri consiste à prendre d'abord les ressources ayant la première date de disponibilité la plus petite, puis celles qui ont déjà le plus grand nombre de tâches. Pour chaque ressource disponible pouvant réaliser le mouvement de terre e ($RessourcesDispos(e)$), on cherche la première date de disponibilité afin d'affecter la plus grande quantité de matériau déplacée à cette ressource. Pour chaque fenêtre de temps continue, une tâche est créée. Quand il n'y a plus de disponibilité de cette ressource dans l'intervalle de temps autorisé par ce mouvement de terre, on passe à la ressource suivante. Les fonctions permettant d'accéder à la prochaine date de disponibilité et d'occupation d'une ressource r sont notées respectivement $ProchaineDateDispo(r)$ et $ProchaineDateNonDispo(r)$.

Cet algorithme glouton calcule une première affectation en moins d'une seconde, même pour les instances de grande taille. Mais il n'y a aucune garantie d'obtenir une solution avec toutes les quantités affectées. En pratique, une première solution sans quantité non affectée peut être rapidement trouvée pour la plupart des instances. Les résultats présentés en Section 3.6 montre que l'utilisation de cet algorithme glouton en tant que phase d'initialisation permet d'améliorer les résultats de la recherche locale. L'algorithme d'initialisation permet de débuter la recherche locale sur cette solution où les activités ne sont pas éparpillées géographiquement dès le début.

3.5.3 Mouvements

Les transformations consistent à créer, déplacer, supprimer une tâche entière, une partie ou même un groupe de tâches. Les déplacements de quantité de travail peuvent être effectués entre plusieurs ressources ou au sein même d'une ressource (Voir la Figure 3.18). Le premier mouvement est l'*insertion* qui consiste à ajouter une tâche à une ressource. On sélectionne un mouvement de terre puis une ressource autorisée pour ce mouvement. L'objectif est d'affecter à cette ressource la quantité maximale de travail à la première date autorisée par

Algorithme 6: GLOUTON-INITIALISATION

input : Les mouvements de terre, ressources, blocs et couches
output : Liste des tâches à effectuer

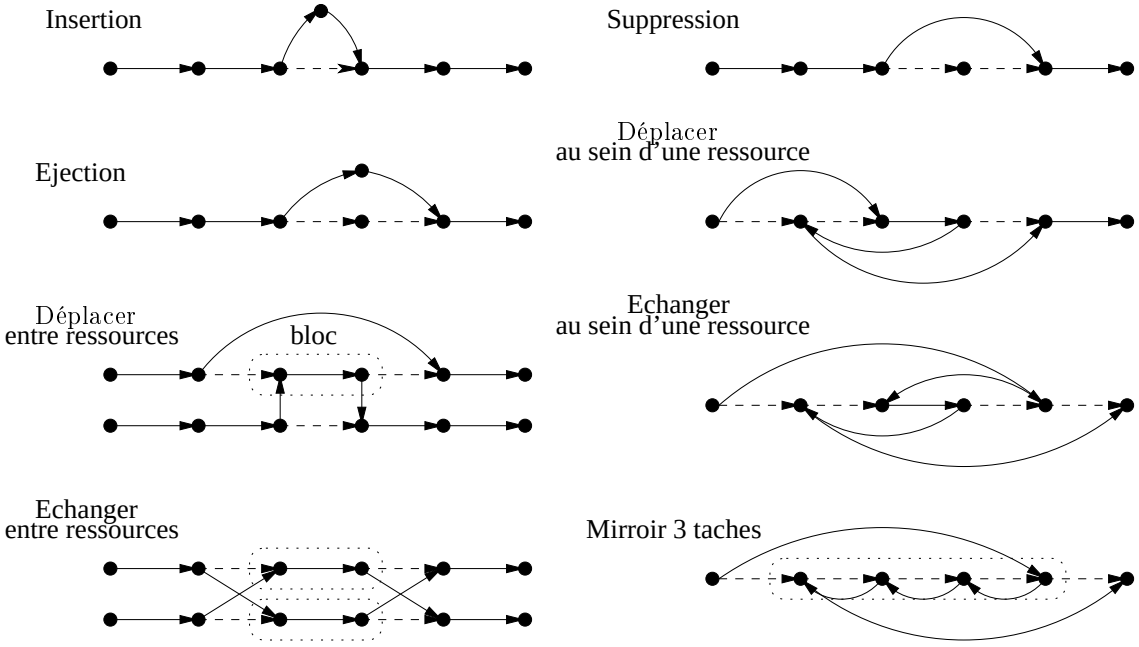
```
begin
    ClasserMouvements()
    for e ∈ E do
        while QuantiteesManq(e) non nulle And RessourcesDispos(e) non nulle do
            ClasserRessources(e)
            for r ∈ RessourcesDispo(e) do
                debut = ProchaineDateDispo(r)
                while QuantiteesManq(e) non nulle And debut non nul do
                    fin = ProchaineDateNonDispo(r, QuantiteesManq(e))
                    CreationTache(r, debut, fin)
                    debut = ProchaineDateDispo(r)
    Retourner la liste des tâches créées
```

la ressource (ou à une date tirée aléatoirement entre cette première date et la dernière date de disponibilité), en respectant l'intervalle de réalisation du mouvement.

Ensuite, plusieurs mouvements concernent le déplacement de tout ou partie d'une tâche au sein du planning d'une ressource ou entre ressources. Le premier mouvement est le *déplacement partiel mono-ressource* : la transformation choisit une ressource, une tâche de cette ressource et on tente de déplacer dans le temps une partie de cette tâche, en la ré-affectant à cette même ressource avant ou après sa date actuelle, dans les intervalles de temps autorisé. Ensuite, le *déplacement complet mono-ressource* sélectionne une ressource, une tâche de cette ressource et on tente de la déplacer entièrement dans le temps, en la ré-affectant à cette même ressource avant ou après sa date courante. Ces deux transformations sont déclinées pour le déplacement d'activité entre ressources. Le *déplacement partiel multi-ressources* choisit une ressource, une tâche de cette ressource et tente de déplacer une partie de cette tâche vers une ou plusieurs autres ressources, afin que la totalité de la quantité déplacée soit ré-affectée. Et le *déplacement complet multi-ressources* sélectionne une ressource, une tâche de cette ressource et tente de déplacer la totalité de cette tâche vers une ou plusieurs autres ressources, afin que tout le travail soit ré-affecté. La *purge d'une ressource* consiste à sélectionner une ressource et à tenter de déplacer toutes ses tâches en les ré-affectant à une ou plusieurs ressources.

Les échanges entre ressources se focalisent sur des déplacements de volume de travail entre échelons. Premièrement, l'*échange de tâches mono-ressource* sélectionne une ressource, deux tâches de cette ressource et tente d'inverser ces deux tâches dans le temps. Deuxièmement, l'*échange de tâches entre-ressources* choisit deux ressources, une tâche pour chacune de ces

deux ressources et tente d'inverser les tâches en faisant réaliser la tâche de la première par la deuxième ressource et inversement. Une autre famille de transformations est le *miroir à K tâches*. Dans ces transformations, on sélectionne une ressource et une chaîne de K tâches de cette ressource, c'est-à-dire K tâches consécutives et on tente de les inverser par un effet miroir : la première échange sa place avec la dernière, la deuxième prend l'avant dernière place, *etc.* Ce mouvement correspond à l'amélioration 2-opt bien connue pour le problème de voyageur de commerce. Nous disposons aussi d'une transformation consacrée à *la fusion de tâches*. On choisit une ressource et on regroupe deux tâches si elles sont juxtaposées et qu'elles concernent le même mouvement de terre, ce qui permet d'en faire une unique tâche.



Les suites de tâches originales sont données par des arcs pleins, les arcs supprimés sont représentés par des traits pointillés, les arcs courbés ou verticaux sont ajoutés par la transformation.

FIGURE 3.18 – Les transformations sur les tâches des ressources.

Les deux derniers mouvements *déplacement mono-ressource par décalage* et *déplacement multi-ressource par décalage* sont les plus complexes. Le premier consiste à sélectionner une ressource puis une tâche au hasard, puis à déplacer cette tâche au sein même de la ressource avant sa date de réalisation, en décalant les autres tâches. Le second réalise le même processus mais en déplaçant la tâche dans le planning d'une autre ressource. Une option supplémentaire complexifiant encore le mécanisme consiste à propager les décalages à toutes les tâches ayant un lien de précédences avec les tâches déplacées dans la ressource modifiée. La Figure 3.19 résume le mode de fonctionnement de ce mouvement qui décale les dates de réalisation des tâches concernées au cours du déplacement de la tâche.

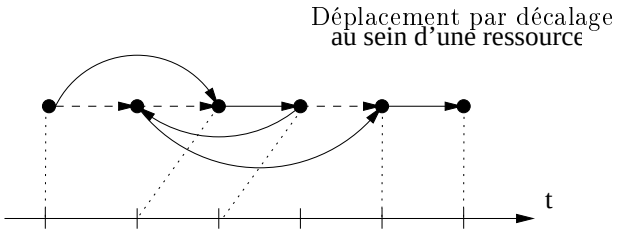


FIGURE 3.19 – Le déplacement mono-ressource par décalage.

Quelques unes de ces transformations ont été spécialisées, notamment en opérant une sélection en priorité des ressources ayant peu de tâches. Une autre spécialisation consiste à sélectionner les premières ou les dernières tâches des ressources pour y appliquer des transformations, afin entre autre de réduire la largeur totale de la fenêtre d'activité d'une ressource. Cette spécialisation permet d'améliorer le pourcentage d'acceptation de ces transformations par rapport à celui de leur version générique. En effet, à chaque application d'une de ces transformations, il s'agit d'évaluer si on a été en mesure de réaliser la transformation tout en continuant de respecter le jeu de contraintes (fenêtres de temps, précédences, *etc.*).

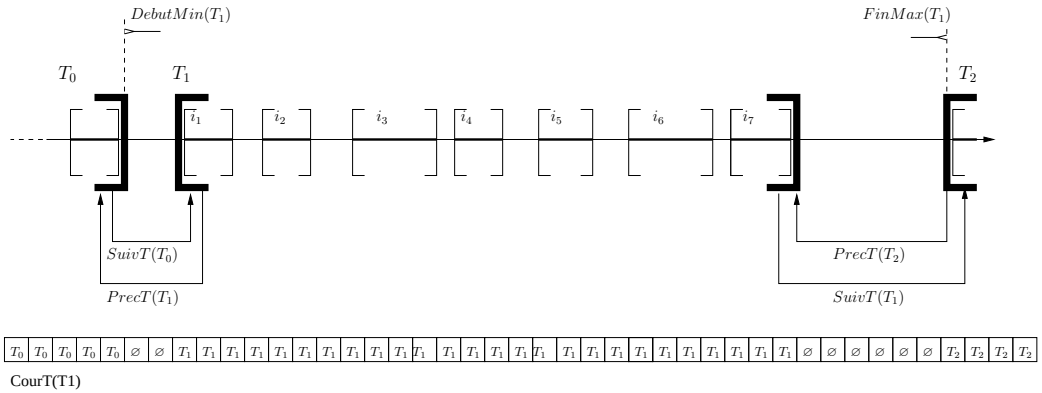
Chacune des deux phases d'optimisation dispose de son propre ensemble de transformations plus ou moins spécialisées en fonction des besoins de la phase. Ces spécialisations ont été validées à l'aide de tests sur différentes instances, afin de montrer que leur ajout permettait d'accélérer la convergence globale de la recherche locale. Elles permettent d'augmenter le nombre de transformations évaluées ainsi que le pourcentage de transformations acceptées. Certaines transformations sont même désactivées pendant la phase de minimisation des quantités de matériaux non affectés, c'est le cas de l'*échange de tâches entre-ressources*, car même si ce mouvement a beaucoup de chance d'être accepté et d'améliorer rapidement l'objectif, il a un effet négatif : les tâches sont morcelées en micro-activités et la recherche locale a des difficultés à les regrouper par la suite. Cette transformation a tendance à améliorer la densité du planning mais cela dégrade l'objectif de minimisation de la distance parcourue. On choisit de ne pas autoriser cette transformation pendant la phase d'optimisation des quantités non affectées. Elle est autorisée uniquement lorsqu'on optimise les coûts associés aux ressources, en la couplant avec le mouvement de fusion des tâches. Aucun voisinage large n'est employé dans cette recherche locale. Les voisinages explorés ici ont une taille d'environ $O(n^2)$ avec n le nombre de tâches et de ressources dans la solution courante.

3.5.4 Machinerie d'évaluation

Cette section a pour objectif de fournir des détails concernant l'implémentation de la recherche locale. La recherche locale s'appuie sur trois fonctions cruciales qui sont l'*évaluation* (du gain ou de la perte d'une transformation appliquée sur la solution courante), la *validation* (afin de mettre à jour la solution courante par modification des données impactées) et enfin le

retour arrière (qui nettoie les champs qui ont été modifiés au cours de l'évaluation et revient à l'état précédent). L'évaluation est rendue ici complexe à cause de la présence des précédences entre tâches et de la multitude des fenêtres de temps sur les ressources et les mouvements de terre. Toute modification d'une tâche a un impact sur le planning de la ressource, du mouvement de terre et de la couche associée. Sans maintien des invariants appropriés, toute modification locale nécessite des calculs coûteux pour évaluer le mouvement. On maintient donc des indicateurs et des valeurs mis à jour uniquement s'ils sont perturbés par la recherche locale. Leur mise à jour est ainsi minimisée.

Tout d'abord, nous pré-calculons un certain nombre de structures de données. Par exemple, pour chaque tâche affectée à une ressource, nous stockons la tâche précédente et la suivante. Une structure dynamique est aussi maintenue en déterminant la tâche courante de la ressource pour des pas de temps qui découpent le temps continu (voir le tableau $courT$ sur la Figure 3.20, on note $\varnothing$ si le pas de temps ne compte pas de tâche courante). Ainsi, lorsqu'on souhaite connaître pour n'importe quelle date du planning la tâche courante, le pas de temps associé est calculé et la tâche courante est récupérée en $O(1)$. L'évaluation d'une transformation travaillant sur les tâches d'une ressource ne modifie pas cette structure de données qui est mise à jour uniquement lors de la validation de la transformation si elle est acceptée. De même, chaque ressource dispose d'un tableau discrétisant le temps en pas de temps pour lesquels on stocke chacun la fenêtre courante d'autorisation de travail.



Une tâche est composée d'une série d'intervalles de fenêtres de temps.

FIGURE 3.20 – Une tâche.

Si une tâche est déplacée ou modifiée par une transformation, la fenêtre de temps courante où elle peut être déplacée doit être strictement respectée en fonction de ses tâches précédentes et suivantes. Pour cela, la fenêtre de temps dans laquelle peut se déplacer la tâche est maintenue à tout instant, en utilisant deux données :

– *la plus petite date de début :* définie comme le maximum entre la date de fin de la tâche précédente de cette ressource et la plus grande date de fin de toutes les tâches des mouvements de terre précédant le mouvement de terre courant.

– *la plus grande date de fin :* définie comme le minimum entre la date de début de la tâche suivante de cette ressource et la plus petite date de début de toutes les tâches des mouvements de terre suivant le mouvement de terre courant.

Si une transformation vient ajouter ou modifier une tâche, alors automatiquement, ces deux bornes s'en retrouvent modifiées, pour une ou plusieurs tâches. Mais seules les structures temporaires sont mises à jour au cours de l'évaluation, afin de fournir le résultat de l'évaluation sans réaliser des calculs non nécessaires. Pour chaque ressource, on fait très fréquemment appel à la fonction permettant, depuis une date, de trouver la prochaine date de disponibilité comme présenté dans l'Algorithme 7. Cela correspond à la prochaine date où du travail peut être affecté à cette ressource.

Algorithme 7: CALCUL-DATE-DISPO

input : Une ressource, une date courante
output : Une date de disponibilité

begin
  On calcule pour la date courante la première date incluse dans une fenêtre de temps
  Calculer l'intervalle de disponibilité I_f de la ressource, contenant f
  while *La date courante ne vaut pas la dernière date du planning* **do**
    Récupérer la tâche T en cours pour cette ressource à cette (en O(1))
    if *T non nulle* **then**
      Calcul-Date-Dispo(ressource(T), dateFin(T)) **else**
        Vérifier si une nouvelle tâche T' n'existe pas à cette date
        if *T' non nulle* **then** Calcul-Date-Dispo(ressource(T'), dateFin(T'))
        else Retourner la date courante

De même, pour chaque ressource, la date de fin d'une tâche nouvellement ajoutée au sein de son emploi du temps est calculée à l'aide d'une dichotomie en $O(n \log n)$ en temps, avec n son nombre de fenêtres de temps. L'algorithme sous-jacent est détaillé dans l'Algorithme 8.

Le calcul du temps de travail disponible entre deux dates s'appuie aussi sur des structures de données incrémentales mises à jour uniquement lors d'une validation de transformation. L'Algorithme 9 détaille son mode de fonctionnement. À noter que $Cumul(I)$ est une donnée maintenue dynamiquement au cours de la recherche locale.

À l'aide de ces routines optimisées et de ces structures de données incrémentales, environ 120 000 transformations sont évaluées en moyenne chaque seconde dans l'espace des solutions, avec un taux d'acceptation de l'ordre de 8 % et un taux d'amélioration de 0.016 %. Ces performances permettent d'assurer une très bonne diversification de la recherche locale et la section suivante a pour objectif de fournir des résultats sur les jeux de données testés.

Algorithme 8: CALCUL-DATE-DE-FIN

input : Une ressource et ses tâches, un mouvement de terre, une nouvelle tâche, une date de début d, une date de fin maximale f

output : Une ressource avec un nouveau planning de tâches, la quantité de travail affectée

begin

Calculer l'intervalle de disponibilité I_d de la ressource, contenant d
Calculer l'intervalle de disponibilité I_f de la ressource, contenant f
$MinDate = Max(\ Max(\ debut(\ I_d\ ),\ d\ )\ ;\ debut(\ I_f\ )\ )$
$MaxDate = Min(\ fin(\ I_d\ )\ ;\ Min(\ fin(\ I_f\ ),\ f\ )\ )$
Calcul-Temps-Travail-Dispo($MinDate$, $MaxDate$)
if *Cette quantité est suffisante* **then**
    Déduire la date de fin effective de la tâche entre $MinDate$ et $MaxDate$ par affectation de la quantité de travail à la ressource
    else Retourner MaxDate

Algorithme 9: CALCUL-TEMPS-TRAVAIL-DISPO

input : Une ressource et ses tâches, deux dates d_1 et d_2

output : Le temps de travail disponible entre les deux dates

begin

Calculer l'intervalle de disponibilité I_{d_1} de la ressource, contenant d_1
Calculer l'intervalle de disponibilité I_{d_2} de la ressource, contenant d_2
if $date1\ >\ debut(\ I_{d_1}\ )$ **then** $Cumul\ -=\ d_1\ -\ debut(\ I_{d_1}\ )$
if $date2 > debut(I_{d_2})$ **then** $Cumul\ +=\ d_2\ -\ debut(\ I_{d_2}\ )$
Retourner Cumul

3.6 Résultats numériques

Cette section a pour objectif de présenter les résultats obtenus sur onze instances B issues de données réelles. L'algorithme complet de recherche locale a été implémenté en C# 2.0 (pour être exécuté dans l'environnement Microsoft .NET 2.0). Le programme compte 9 000 lignes de code, dont 1 700 dédiées à la vérification de la validité des structures de données (en mode débugage). Toutes les statistiques et les résultats présentés ici ont été obtenus (sans parallélisation) sur un ordinateur équipé du système d'exploitation Windows Vista et d'un processeur Intel Xeon X5365 64 bits (CPU 3 GHz, L1 cache 64 Ko, L2 cache 4 Mo, RAM 8 Go).

Le Tableau 3.4 présente dans sa première colonne (Inst) le nom de l'instance, puis les données décrivant le chantier sont indiquées : le nombre de blocs (nbBlocs), le nombre de

couches total (NbCouches), la longueur du chantier (long), le nombre de jours max du planning (nbJours), la quantité totale à déplacer (quant). Ensuite, nous précisons les informations relatives aux ressources : le nombre de familles de ressources (nbFamRes), le nombre de ressources disponibles (nbResDisp), le nombre de couples ressources / mouvements de terre (nbResMvtr) et le temps total disponible pour toutes les ressources en minutes (tps). La colonne (nbMvtr) indique le nombre de mouvements de terre à planifier à l'aide de ces ressources.

Inst	nbBlocs	NbCouches	nbFamRes	nbResDisp	nbMvtr	nbResMvtr	long	nbJours	quant	tps
B01	234	234	4	19	270	2 390	33,6	788	15 286 188	273 334 770
B02	61	89	3	15	99	495	8	364	1 364 276	40 365 981
B03	15	21	2	2	18	18	5,5	182	840 000	4 743 342
B04	57	63	3	4	55	79	13,2	152	652 386	12 117 545
B05	73	106	3	60	110	2 200	9,8	879	1 581 314	139 391 890
B06	30	30	3	15	27	135	2,7	102	255 930	4 004 613
B07	46	85	3	15	83	415	5,2	364	479 155	43 624 717
B08	37	37	3	6	32	81	3,5	637	639 506	9 406 048
B09	200	219	4	6	239	375	52,1	758	3 975 009	261 217 201
B10	99	104	1	10	130	1 300	50	1 825	7 413 905	323 073 950
B11	174	251	6	11	246	551	17,35	486	2 105 442	172 514 634

TABLE 3.4 – Benchmarks : Description des instances.

Dans le Tableau 3.5, les résultats détaillés obtenus par l'algorithme glouton d'initialisation sont présentés, avec tout d'abord la quantité à déplacer (q), la quantité déplacée (mq), le pourcentage déplacé (mq/q), le temps total de travail (min), la distance totale parcourue (d), le nombre de ressources utilisées (nR), le nombre de tâches créées (nT) et enfin le coût global des ressources (C). Ces résultats sont obtenus en moins d'une seconde pour toutes les instances.

Inst	total m^3	m^3	% m^3	min	d	nR	nT	C
B01	15 286 188	13 628 221,66	89,15%	18 576 498	2 929 938	19	292	190 594 918
B02	1 364 276	1 364 276	100,00%	1 440 486	144 903,5	14	105	15 949 763,5
B03	840 000	840 000	100,00%	303 973	32 700	2	18	3 272 430
B04	652 386	626 721	96,07%	404 518	75 945	4	54	4 521 125
B05	1 581 314	1 408 115	89,05%	942 648	266 310	24	106	12 092 790
B06	255 930	255 930	100,00%	175 248	20 965	15	27	3 273 445
B07	479 154,23	479 154,23	100,00%	700 182	108 590	3	90	7 410 410
B08	639 506	634 949,33	99,29%	918 521	24 310	6	36	9 809 520
B09	3 975 008.92	3 975 008,92	100,00%	3 767 118	2 027 339,5	6	239	40 298 519,5
B10	7 413 904,62	6 198 226,67	83,60%	6 440 878	1 589 338	8	126	66 798 118
B11	2 105 442	1 528 993,73	72,60%	3 369 379	1 356 105	11	245	36 149 895

TABLE 3.5 – Résultats de l'algorithme d'initialisation.

Ce troisième tableau de résultats détaille le nombre de transformations évaluées au total ($nbEval$), par seconde ($nbEvalSec$), diversifiantes ($nbDivers$), le pourcentage par rapport au nombre d'évaluations totales ($\%divers$), le nombre de transformations améliorantes ($nbAmelio$), le pourcentage par rapport au nombre d'évaluations totales ($\%amelio$).

Inst	nbEval	nbEvalSec	nbDivers	%divers	nbAmelio	%amelio
B01	1 132 000	45 280	74 309	6,56%	898	0,08%
B02	1 144 000	45 760	54 205	4,74%	99	0,01%
B03	4 284 000	171 360	144 883	3,38%	4	9,33 E-5%
B04	2 547 000	101 880	144 018	5,65%	104	4,1 E-03%
B05	2 315 000	92 600	234 893	10,15%	111	4,8 E-03%
B06	6 479 000	259 160	53 608	0,83%	32	4,9 E-04%
B07	1 209 000	48 360	157 753	13,05%	55	4,54 E-03%
B08	2 938 000	117 520	182 773	6,22%	16	5,45 E-04%
B09	1 536 000	61 440	100 430	6,54%	270	0,02%
B10	981 000	39 240	86 283	8,80%	490	0,05%
B11	1 541 000	61 640	77 385	5,02%	602	0,04%

TABLE 3.6 – Nombre d'évaluations, diversifications et améliorations.

Dans le Tableau 3.7, les solutions obtenues par recherche locale sont présentées, avec le coût total des ressources RC, en détaillant les sous-coûts associés : ($nbRes$) le nombre de ressources utilisées, (Km) le nombre de kilomètres parcourus et (W) la somme cumulée des temps de travail des ressources. Pour toutes les solutions présentées ici, MQ est supposé nulle, ce qui signifie que toutes les quantités de terre ont été affectées à l'issue de la recherche locale. La colonne ($\%greedyMq$) présente le pourcentage de quantité de terre non affectée et ($\%gRc$) l'écart entre le coût total des ressources à la fin de l'algorithme glouton d'initialisation et la valeur à la fin de la recherche locale. Les écarts sont donnés pour le nombre de ressources utilisées ($\%ggNbRes$) le nombre total de kilomètres parcourus, ($\%ggKm$) et la somme cumulée des temps de travail des ressources ($\%ggW$), entre les solutions de cet algorithme d'initialisation et les solutions à la fin de la recherche locale.

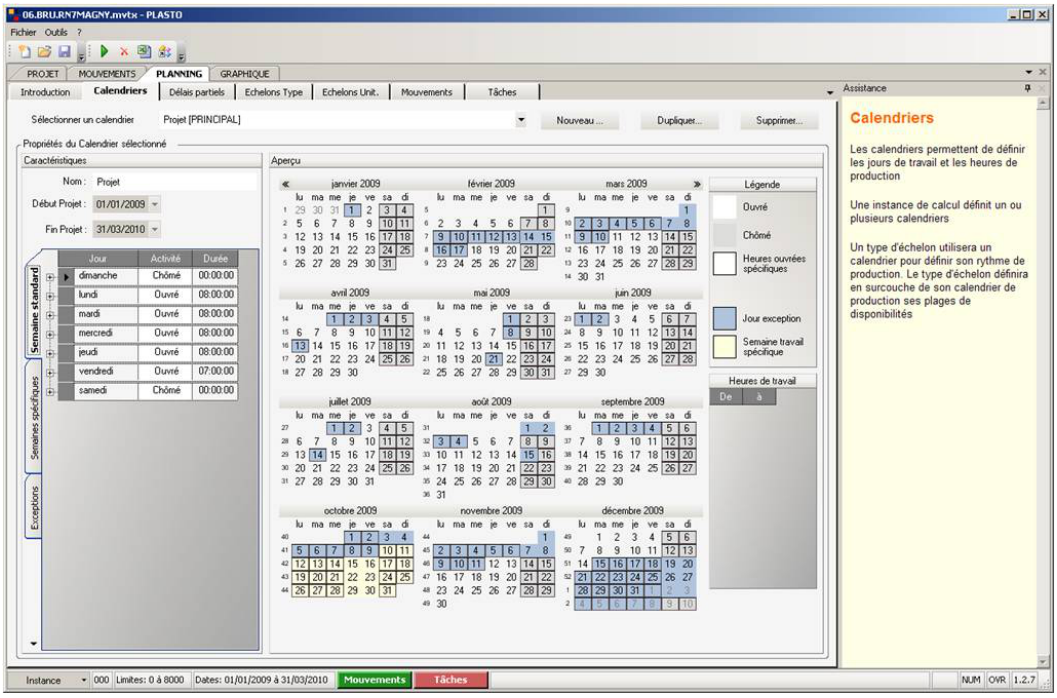
Inst	RC	nbRes	Km	W	%gMq	gRc	gNbRes	gKm	gW
B01	45 868 315,64	19	1 358 145,00	18 680 858,00	10,80%	415,53%	100%	216%	99%
B02	2 560 950,23	8	7 623,50	966 160,00	0,00%	623,98%	175%	187%	149%
B03	868 662,27	2	28 100,00	303 477,00	0,00%	376,72%	100%	116%	100%
B04	1 182 720,13	4	70 980,00	400 130,00	3,90%	382,26%	100%	107%	101%
B05	1 581 313,00	3	121 200,00	950 745,00	4,30%	756,54%	100%	207%	120%
B06	402 879,38	9	13 205,00	187 349,00	0,00%	812,51%	167%	159%	94%
B07	1 486 219,30	3	47 710,00	478 406,00	0,00%	498,61%	100%	228%	146%
B08	2 438 411,06	6	20 020,00	763 461,00	0,70%	402,29%	100%	121%	120%
B09	8 668 828,45	6	634 705,00	3 722 328,00	0,00%	464,87%	100%	319%	101%
B10	17 520 277,84	8	750 178,50	8 039 261,00	16,40%	381,26%	100%	212%	80%
B11	7 519 636,23	11	443 145,00	3 049 219,00	27,40%	480,74%	100%	306%	110%

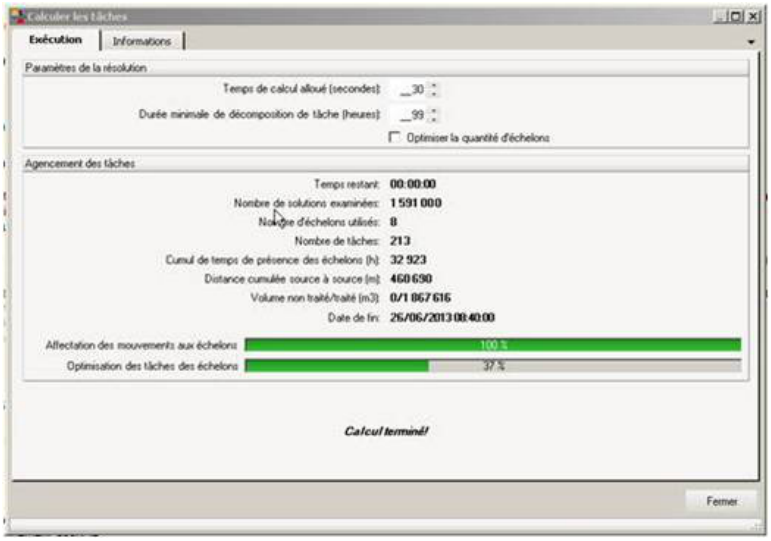
TABLE 3.7 – Valeurs de la fonction objectif de la recherche locale et de l'algorithme glouton d'initialisation.

3.7 Synthèse

Dans ce chapitre était présentée une modélisation du problème de planification des mouvements de terre sur chantier linéaire. Ce problème prend comme donnée d'entrée les mouvements de terre obtenus par la résolution du problème de génération des mouvements de terre sur chantier linéaire. Ce dernier est résolu optimalement par une programme linéaire consistant à minimiser le moment total de transport, c'est-à-dire le produit de la somme des quantités de terre transportée par les distances parcourues. Pour résoudre le problème de planification, nous utilisons une heuristique directe de recherche locale. Notre algorithme fournit en des temps très court (moins de 1 minute) des solutions de qualité qui se rapprochent de près de solutions d'experts obtenues après une analyse de plusieurs heures. Afin de valider notre approche, nous avons isolé le cas à une ressource, que nous avons analysé afin de trouver des solutions exactes et des bornes inférieures, permettant de prouver que notre recherche locale, dans ce cas à une ressource, donnait des résultats satisfaisant.

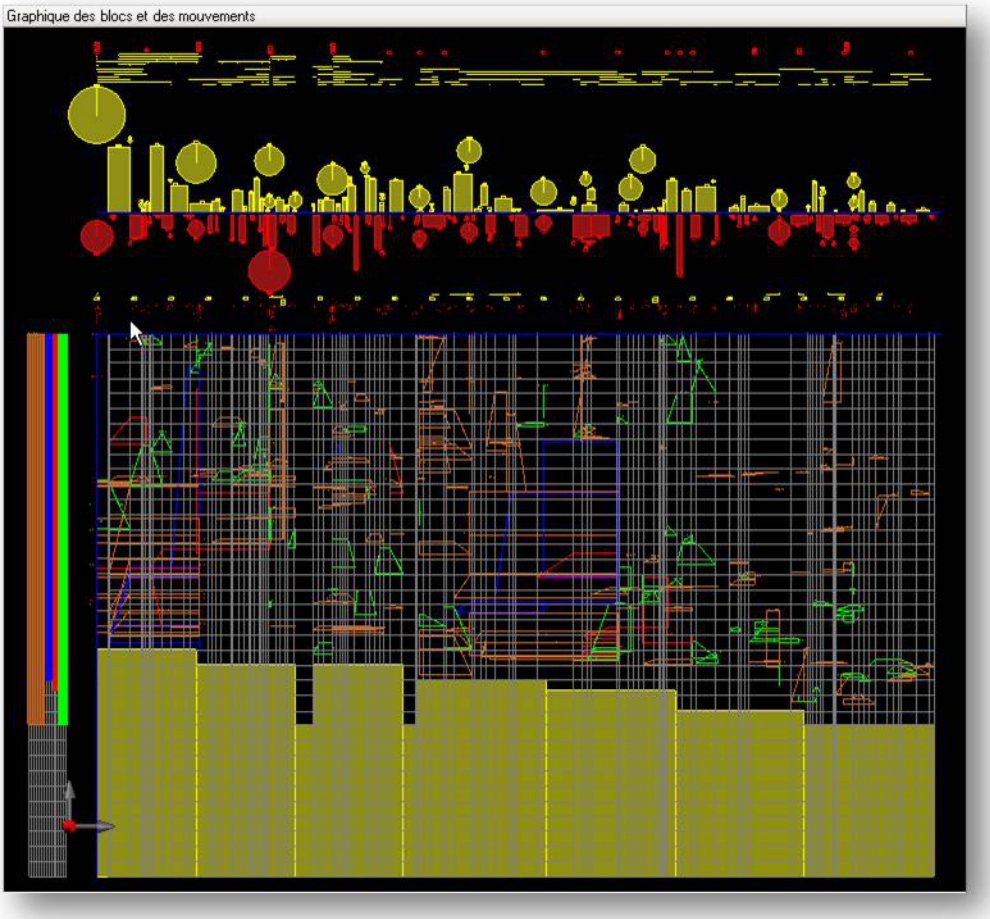
Un logiciel exploitant cet algorithme est aujourd'hui en exploitation et permet de planifier de grands chantiers de terrassement en quelques minutes. Une interface de saisie permet de faire des ajustements d'emplois du temps des ressources, de modifier la compositions des équipements, de placer des délais partiels, de saisir aussi des contraintes de précédences complémentaires (voir la Figure 3.21 et la Figure 3.22). Cet outil d'aide à la décision permet à l'utilisateur final de tester de multiples scénarios avant de rendre son planning chemin de fer final décrivant l'ensemble des activités pratiquées tout au long de la vie du chantier. Les diagrammes chemin de fer sont construits ainsi : en abscisse, on retrouve l'axe du chantier en kilomètres et en ordonnée le temps en jours. En quelques minutes, l'utilisateur peut tester plusieurs scénarios et décider d'ajouter ou supprimer des contraintes temporelles ou spatiales. L'efficacité de cette recherche locale réside notamment dans la mécanique des transformations qui permettent aux quantités de terre déplacées de passer d'une tâche à l'autre, alors que les dates d'affectation de ces tâches sont aussi modifiées en même temps.

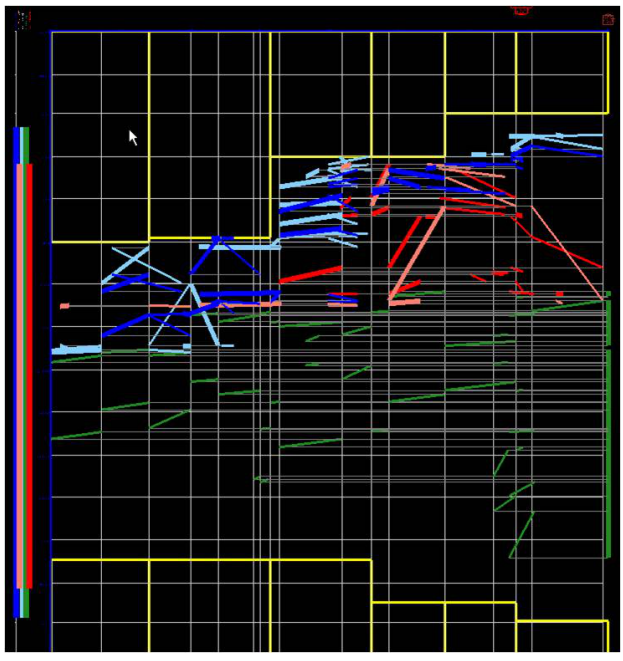




Une interface utilisateur permet de saisir les données (en haut, les notions de calendrier de ressources), de les modifier, de lancer le calcul (en bas) et d'afficher ensuite les diagrammes chemin de fer (2 derniers graphiques).

FIGURE 3.21 – Interface utilisateur (1)





L'interface permet ensuite d'afficher les diagrammes chemin de fer, qui ont pour abscisse l'axe du chantier en kilomètres et pour ordonnée le temps en jours.

FIGURE 3.22 – Interface utilisateur (2)

Chapitre 4

Optimisation de tournées de véhicules avec gestion des stocks

On s'intéresse ici à l'optimisation de la logistique de distribution de produits industriels par camion [1]. Ce problème est une généralisation d'un problème connu en anglais sous le nom d'*Inventory Routing Problem* (IRP) [27, 28], traduit en français par problème d'optimisation de tournées de véhicules avec gestion des stocks. La principale différence avec le *Vehicle Routing Problem* (VRP) est qu'ici les stocks des clients sont gérés par le fournisseur. Il s'agit de planifier, sur un horizon fini et à moindre coût, les tournées de réapprovisionnement des clients. Une solution au problème est un ensemble de tournées visitant des clients et livrant une certaine quantité d'un produit à chacun de ces arrêts. Le problème est rendu déterministe par la connaissance des prévisions de consommation des clients. En plus des contraintes de routage, les assèchements doivent être évités, ce qui signifie que la quantité de produit en stock chez chaque client doit rester au dessus d'un certain seuil de sécurité. L'objectif économique du problème est de minimiser les coûts de ces tournées.

Après une présentation des principales caractéristiques du modèle du problème d'optimisation de tournées de véhicules avec gestion des stocks (Section 4.1), nous soulignerons les principales contributions de ces travaux par rapport aux études précédentes (Section 4.2). Basée sur des bornes inférieures des coûts de livraison, une nouvelle fonction objectif est proposée pour modéliser le problème à court terme, améliorant significativement l'optimisation à long terme (Section 4.3). Nous appliquons ensuite la méthodologie introduite au début de cette thèse pour résoudre ce problème d'optimisation mixte. Nous proposons une approche à base de recherche locale pure et directe pour aborder ce problème à court terme (Section 4.4). Nous utilisons ici une large variété de mouvements travaillant aussi bien sur les

1. Les travaux de recherche présentés dans ce chapitre ont été réalisés dans le cadre d'un projet de recherche et développement mené pour un client du Groupe Bouygues de 2008 à 2009. Ces travaux ont été menés en collaboration avec Thierry Benoist (Bouygues e-lab), Frédéric Gardi (Bouygues e-lab) et Bertrand Estellon (Laboratoire d'Informatique Fondamentale de Marseille). Ils ont fait l'objet de plusieurs publications [12, 19, 20, 21, 22, 23]

aspects combinatoires que continus du problème et nous faisons appel à une algorithmique incrémentale pour évaluer les mouvements. Nous détaillerons en particulier les algorithmes d'ordonnancement des tournées et d'affectation des volumes. Une étude détaillée sur un large jeu de données (Section 4.5) démontre que nos solutions fournissent sur le long terme des réductions de coûts excédant 20 % en moyenne par rapport à des solutions construites par des planificateurs experts ou même par rapport à un algorithme glouton classique.

4.1 Présentation du problème

Par souci de concision, le problème n'est pas décrit complètement et formellement ici mais les principales caractéristiques du modèle sont présentées. Pour plus de détails, nous invitons le lecteur à se référer à la description détaillée du modèle en Annexe 6.2 qui présente la liste des contraintes portant sur le routage, les ressources, la gestion des stocks, les tournées et les voyages.

4.1.1 Données

Dans une zone géographique, des clients consomment un produit provenant d'usines qui le produisent. Chaque site (client ou usine) est ouvert durant des fenêtres de temps. Chaque client dispose d'un stockage avec une certaine capacité ; de même, chaque usine possède un stockage duquel peut être prélevé le produit. Les prévisions de production des usines sont connues sur un certain horizon. Côté client, deux types d'approvisionnement sont pris en charge. Le premier type de contrat, dit "à la prévision", correspond aux clients de l'ensemble pour lesquels nous possédons les prévisions de consommation sur un certain horizon. Les stocks de ces clients doivent être approvisionnés par camion de façon à ne jamais descendre en dessous d'un seuil de sécurité. La Figure 4.1 illustre une livraison qui est effectuée trop tard provoquant un assèchement chez le client, dont le niveau d'inventaire passe sous le seuil critique. Le deuxième type de contrat, dit "à la commande", correspond aux clients de l'ensemble passant des commandes définies chacune par une quantité à livrer et une fenêtre de temps durant laquelle la livraison doit être effectuée. Certains clients peuvent appartenir aux deux types à la fois : leur stock est géré de façon prévisionnelle mais ils peuvent passer commande lorsqu'ils le souhaitent (pour faire face à une augmentation imprévue de leur consommation, par exemple). Une livraison chez un client (resp. un chargement en usine) est une opération définie par une date d'arrivée sur le site, une date de départ, ainsi que la quantité livrée (resp. chargée).

Le transport est effectué à l'aide de trois types de ressources : les chauffeurs, les tracteurs et les remorques. Chaque ressource est attachée à une base. Un véhicule correspond à l'association d'un chauffeur, d'un tracteur et d'une remorque. Tous les triplets de ressources ne sont pas admissibles. Chaque ressource possède des fenêtres de disponibilité pendant

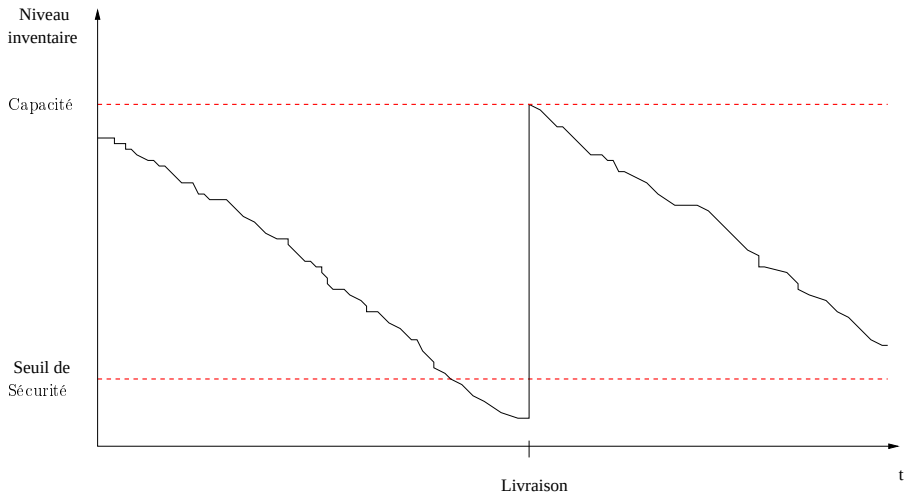


FIGURE 4.1 – Une livraison après assèchement.

lesquelles celle-ci peut être utilisée. Chaque site n'est accessible qu'à un sous-ensemble des ressources. Ainsi, planifier la tournée d'un véhicule consiste à définir : une base, un triplet de ressources (chauffeur, tracteur, remorque) et un ensemble d'opérations définies chacune par un triplet (site, date, quantité) correspondant aux livraisons/chargements effectués sur la tournée. Une tournée doit démarrer depuis une base où toutes les ressources sont présentes et se clore par un retour à cette même base. Les temps de travail et de conduite des chauffeurs sont limités ; dès qu'un maximum est atteint, le chauffeur doit prendre une pause d'une durée minimale. Les sites visités durant la tournée doivent être accessibles aux ressources composant le véhicule. L'intervalle de temps pendant lequel est utilisée chaque ressource doit être contenu dans une de ces fenêtres de disponibilité. Enfin, chaque opération doit être réalisée dans une des fenêtres d'ouverture du site. Notons ici que la durée d'une opération ne dépend pas de la quantité livrée ou chargée ; en effet, il a été convenu que cette durée serait fixée en fonction du site sur lequel l'opération a lieu, l'approximation faite étant couverte par les incertitudes pesant sur les temps de parcours.

Le problème que nous traitons est déterministe. Les consommations des clients et les productions des usines (prévisionnelles) sont données à court terme de façon discrète (l'horizon est découpé en H pas de temps régulier). Les prévisions de consommation et de production sont données (et considérées comme fiables) pour un horizon de 15 jours. Ainsi, les tournées d'approvisionnement sont planifiées jour après jour sur un horizon glissant de 15 jours. Chaque jour, un plan de distribution est construit pour les 15 prochains jours, mais seules les tournées démarrant le premier jour de l'horizon (c'est-à-dire le lendemain) sont fixées. Le jour suivant, l'horizon est décalé d'un jour et un nouveau planning est construit pour la quinzaine à venir, tenant compte du fait que les ressources mobilisées par les tournées fixées sont indisponibles et que les niveaux des stocks des sites visités par ces tournées sont déjà décidés.

Deux matrices sont fournies : la matrice des distances entre les sites et la matrice temporelle correspondant aux temps de transport entre deux sites. Les 2 matrices ne sont pas symétriques, mais nous supposons qu'elles satisfont l'inégalité triangulaire. Les différentes unités de mesure utilisées pour les quantités, durées et distances sont indépendantes de la modélisation du problème mais elles doivent être cohérentes. Pour mesurer les quantités, nous préférons souvent utiliser des poids plutôt que des volumes pour gérer le vrac en logistique. Le temps est représenté comme une ligne continue d'horizon fini T. En d'autres termes, chaque instant est donné par un point dans l'intervalle $[0, T]$. Ainsi, toutes les dates définies dans le modèle peuvent être exprimées avec ce degré de précision. Ici, nous travaillerons avec T fixé à 15 jours et une unité temporelle égale à 1 minute. Des restrictions physiques ne permettent pas de connaître les prévisions de manière continue. Nous choisissons donc un pas de temps de taille U, avec $U \times H = T$ où H est le nombre de pas de temps jusqu'à l'horizon. Dans notre cas, la granularité adoptée pour U vaut 1 heure. Sauf mention particulière, chaque intervalle de temps (notamment les fenêtres de temps de l'instance) est défini avec la date de début inclue et la date de fin exclue. Une instance est définie par le nombre de clients, le nombre d'usines de production, le nombre de dépôts, le nombre de conducteurs, le nombre de tracteurs, le nombre de remorques, le nombre de commandes et le nombre de pas de temps pour lesquels nous connaissons les consommations / productions jusqu'à l'horizon. La taille des problèmes que nous avons à traiter ici est considérable. En effet, une zone géographique peut contenir jusqu'à 1500 clients, 50 usines, 50 bases, 100 chauffeurs, 100 tracteurs, 100 remorques. Toutes les dates et les durées sont exprimées en minutes (l'horizon de planification de 15 jours comporte donc 21600 minutes) ; comme détaillé ci-dessous, la dynamique des stocks utilise un pas de temps d'une heure. Le temps imparti pour calculer un planning de distribution à 15 jours est de 5 minutes sur des ordinateurs standards.

4.1.2 Contraintes de routage

Trouver une solution à ce problème consiste à construire un ensemble de tournées valides, c'est-à-dire respectant à la fois les contraintes de routage, ainsi que les commandes et les niveaux de sécurité demandés.

En termes de routage, toutes les tournées doivent commencer à un dépôt et se terminer à ce même dépôt. L'intervalle induit par toute tournée doit être contenu dans un intervalle de disponibilité de chaque ressource affectée à la tournée et la durée de la tournée doit être plus petite ou égal à l'amplitude maximale du conducteur de la tournée. Les tournées réalisées par une ressource ne peuvent pas se chevaucher dans le temps (c'est-à-dire, les intervalles de temps induits par ces tournées sont deux à deux disjoints). Les trois ressources (conducteur, tracteur, remorque) affectées à la tournée doivent avoir comme dépôt celui de la tournée. Le temps écoulé entre la fin d'une opération et le début de la prochaine opération doit être plus grand ou égale au temps de trajet entre ces deux points (donné par la matrice de temps). À noter que cette inégalité autorise du temps d'attente au cours d'une tournée, par exemple entre la fin d'un trajet et l'ouverture d'un site. Toute opération doit avoir lieu pendant les

heures d'ouverture du site visité.

En pratique, un conducteur est seulement autorisé à utiliser une liste restreinte de tracteurs et de remorques, en fonction de son permis de conduire notamment. De même, la possibilité d'attacher une remorque à un tracteur est régie par des contraintes techniques. De plus, la configuration du site peut le rendre inaccessible à certains types de tracteurs ou de remorques (par exemple, en cas de portes étroites ou d'équipements non adaptés) et des accréditations spéciales peuvent être requises pour accéder à certains sites. Ces relations binaires (ressource/ressource et ressource/site) sont représentées par des matrices booléennes.

Comme mentionné dans Archetti *et al.* [6], les dispositions légales ne peuvent être ignorées dans un modèle d'IRP opérationnel. En effet, pour des raisons de sécurité, les conducteurs ne peuvent pas travailler plus d'une certaine durée sans faire une pause obligatoire. Plus précisément, une durée maximale de conduite et de travail sont définies pour chaque conducteur(dépendant du pays, ces durées sont généralement comprises entre 10 et 15 heures). Pour chaque conducteur, le temps de conduite cumulé depuis la fin de la dernière pause ou depuis le début de la tournée ne peut pas dépassé cette durée maximale. De même, pour le temps de travail cumulé. En d'autres termes, une pause doit être définie dès qu'un des deux maxima est atteint. Une opération ne peut pas être stoppée pour une pause (les opérations ne sont pas préemptives). De plus, deux tournées consécutives affectées à un même conducteur doivent être séparées par une durée de pause minimale. Les temps de pause ne sont pas comptés comme des temps de travail, c'est-à-dire que la durée de temps de travail pour une tournée est sa largeur moins la durée des pause pendant la tournée. Cela signifie que des temps d'attente sont potentiellement comptabilisés dans ce temps de travail.

Au lieu d'utiliser une matrice de temps unique, nous définissons une matrice pour chaque type de tracteur (10 vitesses de tracteurs différentes sont prises en compte). Des opérations de vérifications doivent être effectuées au début et à la fin de chaque tournée, tout comme avant et après chaque pause ; ces durées fixes sont notées respectivement *tpsAvant* et *tpsApres*. Tout comme les opérations, ces tâches ne peuvent pas être interrompues pour une pause. Le schéma 4.2 donne deux représentations graphiques d'une tournée avec ces durées. $durees(a, b)$ correspond au temps nécessaire pour se déplacer du point a au point b, *tpsAvant* et *tpsApres* sont les temps de traitement avant et après chaque pause, $dureeOperation(s)$ est le temps de réalisation d'une activité de chargement ou de déchargement au site s, $debut(s)$ et $fin(s)$ sont respectivement les dates de début et de fin de la tournée s, $arrivee(c)$ et $depart(c)$ correspondent aux dates d'arrivée et de départ d'un site c pour une livraison ou un chargement.

Le coût des tournées n'est pas proportionnel aux durées de travail et aux distances parcourues. Divers coûts fixes doivent être comptabilisés. En particulier, $coutChargement(d)$ et $coutLivraison(d)$ sont définis pour chaque conducteur d, représentant respectivement les coûts fixes de chargement et de livraison. Un coût fixe est aussi ajouté à chaque fois qu'une pause est prise.

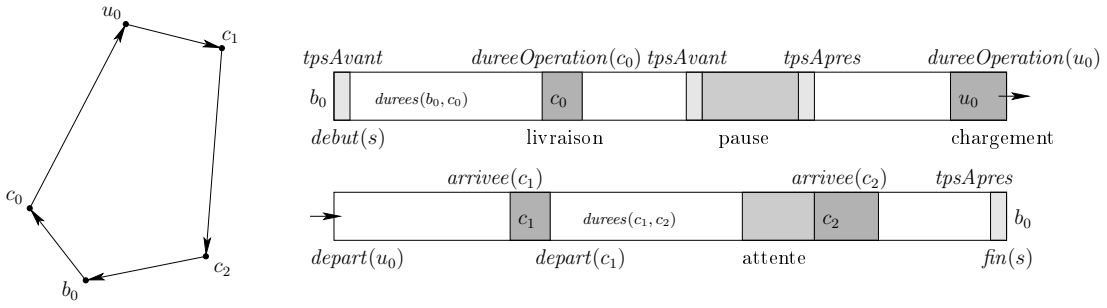


FIGURE 4.2 – Exemple d'une TOURNÉE

Enfin, le dernier jeu de contraintes concerne les commandes. Une commande r d'une quantité $quantite(r)$ est considérée comme satisfaite si une opération o est planifiée en satisfaisant $quantite(o) \geq quantite(r)$ et $arrivee(o) \in [dateMin(r), dateMax(r)[$, avec $arrivee(o)$ la date de début de l'opération o, $dateMin(r)$ (resp. $dateMax(r)$) la date minimale (resp. maximale) à laquelle la commande doit être satisfaite.

4.1.3 Contraintes d'inventaire

Nous l'avons vu précédemment, deux types d'inventaires sont à gérer : les réservoirs sur les sites (les clients et les usines) et les remorques. Dans tous les cas, la quantité du stockage doit rester entre zéro et sa capacité. Pour les clients, la quantité à chaque pas de temps h est égale à la quantité contenue dans le réservoir au pas de temps précédent $h - 1$, moins la consommation au cours du pas de temps h, plus toutes les livraisons réalisées pendant h. À noter que les quantités livrées aux clients doivent être positives (les chargements sont interdits chez le client). Plus formellement, les inventaires dynamiques pour un client c peuvent être exprimés sous la forme suivante : $quantiteReservoir(c, -1) = quantiteInitiale(c)$ et pour chaque pas de temps $h \in \{0, \ldots, H-1\}$,

$$\begin{cases} quantiteReservoir(c,\ h) = quantiteReservoir(c,\ h-1) - previsions(c,\ h) \\ + \sum_{o \in OPERATIONS(c,\ h)} quantite(o) \\ \\ si\ quantiteReservoir(c,\ h) < 0,\ alors\ quantiteReservoir(c,\ h) = 0 \end{cases}$$

avec $quantiteReservoir(c,\ h)$, la quantité contenue dans le réservoir d'un client c au pas de temps h, $OPERATIONS(c,\ h)$ l'ensemble des opérations réalisées chez un client c dont la date de début appartient au pas de temps h et $previsions(c,\ h)$ la prévision de consommation d'un client c pendant le pas de temps h. La même formule peut s'appliquer pour les usines, puisque les prévisions de production et les chargements de quantités ont des

valeurs négatives (les livraisons sont interdites aux usines). Cependant, pour une usine u, quand la formule ci-dessus fournit une quantité plus grande que la capacité $capacite(u)$, le résultat est minoré par $capacite(u)$. Ces dépassements de capacité ne sont pas pénalisés, car les aspects "production" ne sont pas pris en charge dans ce modèle.

Pour les remorques, les inventaires dynamiques sont plus simples puisque les opérations réalisées par les remorques ne peuvent se chevaucher. Comme la quantité d'une remorque w n'est pas définie pour chaque pas de temps mais après chacune de ses opérations (et fait référence à $quantiteRemorque(w, o)$). Initialement égal à $quantiteInitiale(w)$, ce niveau est incrémenté par les chargements et décrémenté par les livraisons. Plus formellement, nous pouvons écrire :

$$quantiteRemorque(w,\ o)\ =\ quantiteInitiale(w)\ -\ quantite(o) \tag{4.1}$$

pour o la première opération de la remorque w, et :

$$quantiteRemorque(w,\ o)\ =\ quantite(prec(o))\ -\ quantite(o) \tag{4.2}$$

autrement dit, avec $prec(o)$ l'opération précédente réalisée par w et $quantite(o)$ la quantité de produit traité par l'opération o.

Enfin, l'interdiction des assèchements à chaque client c est obtenu en définissant pour chaque pas de temps h la contrainte :

$$\forall\ h,\ quantiteReservoir(c,\ h)\ \geq\ seuilSecurite(c) \tag{4.3}$$

4.1.4 Objectifs

Deux jeux de contraintes sont définis comme souples : les contraintes imposant que la totalité des commandes passées par les clients soient satisfaites, ainsi que les contraintes de maintien du niveau des stocks au-dessus des niveaux de sécurité. En effet, l'existence d'une solution admissible n'est pas garantie en conditions opérationnelles, mais les situations dans lesquelles il est difficile de satisfaire celles-ci sont très rares, car ne pas honorer une commande ou bien assécher un client est totalement inacceptable.

L'objectif de ce problème, défini *sur le long terme* (plus de 90 jours), est donc scindé en trois objectifs. Tout d'abord, nous cherchons à éviter les commandes non satisfaites. Ensuite, le deuxième objectif est de minimiser les assèchements. Enfin, le dernier objectif consiste à minimiser le ratio logistique. Les coûts de production ne sont pas pris en compte ici, mais uniquement les coûts de distribution.

Le premier terme de la fonction objectif s'intéresse aux commandes non satisfaites. Une commande est satisfaite si une opération est affectée avec une quantité livrée supérieure ou

égale à la quantité attendue et une date d'arrivée inclue dans la fenêtre de temps escomptée. Pour chaque client c, on note $nbCommandesManquees(c)$ le nombre de commandes non-satisfaites et $C_{cns}(p)$ le coût associé à chacune des ces commandes non livrées. Nous en déduisons que le coût total des commandes non-satisfaites MO est donné par :

$$MO = \sum_{c \in CLIENTS} C_{cns}(c) \times n_{cns}(c)$$

Le second coût optimisé ici concerne les assèchements. Un assèchement apparaît quand le niveau d'inventaire d'un client c (ne fonctionnant pas "à la commande") se retrouve sous le seuil de sécurité pour un pas de temps h, soit $quantiteReservoir(c, h) < seuilSecurite(c)$. Pour chaque client c, on note $n_{ass}(c)$ le nombre de pas de temps passé en assèchement et $C_{ass}(c)$ le coût associé aux assèchements. Le coût total des assèchements SO s'écrit donc :

$$SO = \sum_{c \in CLIENTS} C_{ass}(c) \times n_{ass}(c)$$

Pour éviter les effets de bords de fin de planning, les commandes non-satisfaites et les assèchements sont comptabilisés sur l'horizon de temps raccourci T', ce qui correspond à $T - \max_{d \in CONDUCTEURS} amplitudeMax(d)$ (afin d'être sûr que les exigences qui se posent à la fin de l'horizon puissent toujours être satisfaites). $amplitudeMax(d)$ représente l'amplitude maximale d'une tournée d'un conducteur d.

Le troisième et dernier terme est le ratio logistique $RL = SC/DQ$, avec SC le coût total des tournées et DQ la quantité totale livrée sur l'horizon considéré (sauf si $DQ = 0$, alors $RL = 0$). Ainsi, DQ vaut la somme des quantités livrées pour toutes les tournées. La distance d'une tournée s, notée $distanceTournee(s)$, correspond à la somme des longueurs des arcs induites par la tournée ; la durée de la tournée s, notée $distanceTournee(s)$, correspond au temps passé par le conducteur à travailler (autrement dit $fin(s) - debut(s)$ moins la somme de la durée des pauses). Le nombre total de livraisons (resp. chargements, pauses) pendant une tournée s est noté $nbLibraisons(s)$ (resp. $nbChargements(s)$, $nbPauses(s)$). Ainsi, le coût $SC(s)$ d'une tournée s est composé de cinq termes :

$$\begin{aligned} SC(s) &= coutDistance(tracteur(s)) \times distanceTournee(s) \\ &+ coutTemporel(conducteur(s)) \times distanceTournee(s) \\ &+ coutLivraison(conducteur(s)) \times nbLibraisons(s) \\ &+ coutChargement(conducteur(s)) \times nbChargements(s) \\ &+ coutPause(conducteur(s)) \times nbPauses(s) \end{aligned}$$

Ainsi, le coût total SC est donné par $SC = \sum_{s \in TOURNEES} SC(s)$.

Il est important de voir que les trois termes MO, SO et RL de la fonction objectif sont optimisés dans un ordre lexicographique : $MO \succ SO \succ RL$ mais les solutions avec $MO = 0$ et $SO = 0$ peuvent être trouvées facilement en pratique.

4.2 État de l'art et contributions

Bell *et al.* [10] introduisent le problème d'optimisation des tournées avec gestion des stocks en décrivant le problème rencontré chez le producteur de gaz industriels Air Products. Ces premières publications ont donné naissance à de nombreux travaux dans ce domaine et en particulier une longue série de papiers publiés par Campbell *et al.* [27, 28], Campbell et Savelsbergh [29, 30, 31], Savelsbergh et Song [102, 103] concernant les problématiques rencontrées chez le producteur de gaz industriels Praxair. Cependant, dans de nombreuses entreprises, les problèmes d'optimisation de tournées de véhicules avec gestion des stocks sont toujours résolus à la main ou par des logiciels régis par des règles basiques comme servir les clients proche de l'assèchement tout en maximisant le nombre de tournées avec chargement complet. Nous invitons le lecteur à consulter les papiers récents de Savelsbergh et Song [102, 103] qui offrent un résumé détaillé des travaux réalisés dans le domaine de l'optimisation des tournées avec gestion des stocks ces 25 dernières années. Ci-dessous sont présentées les trois contributions de notre approche :

- Les contraintes, les fonctions de coûts et la taille du problème proviennent d'un modèle réel ;
- Une nouvelle fonction objectif est proposée pour prendre en compte la gestion de l'horizon de temps glissant ;
- Aucune décomposition n'est utilisée ici dans l'approche par recherche locale, grâce à l'utilisation d'un algorithme innovant d'affectation des volumes.

Modélisation d'un problème réel. Le problème abordé ici est proche de la problématique réelle à laquelle les planificateurs doivent faire face au quotidien. À notre connaissance, peu de problèmes d'optimisation de tournées de véhicules avec gestion des stocks de cette échelle ont été abordés dans la littérature. De plus, de nombreuses contraintes incluses dans notre modélisation ne sont pas présentes dans les problèmes décrits dans la littérature. Tout d'abord, notre modèle prend en compte aussi bien les clients à la prévision que ceux à la commande, ce qui complique considérablement plusieurs sous-problèmes liés à la planification des tournées et à l'allocation des ressources sur les tournées. Un autre aspect intéressant est la capacité à aller plus loin dans l'optimisation de la logistique en assouplissant les contraintes de routage, ce que Savelsbergh et Song [102, 103] appellent les "mouvements continus", encore appelés "installations satellitaires" dans Bard *et al.* [8] : le véhicule peut arbitrairement charger ou décharger du produit au cours de ses tournées et les chargements peuvent être faits de manière arbitraire dans différentes usines. De plus, quand un conducteur atteint une limite de temps de travail ou de conduite maximale, il peut poursuivre cette même tournée après sa pause. Cela permet de créer des tournées s'étalant sur plusieurs jours et couvrant des zones géographiques très étendues. Enfin, nous prenons en compte des consommations et productions quotidiennes non-linéaires, grâce à la gestion des inventaires à l'heure. Les prévisions sont supposées connues à chaque pas de temps ce qui rend le problème déterministe sur tout l'horizon. Chaque client dispose d'un profil de consommation qui lui est

propre auquel viennent s'ajouter des commandes de produit : un client peut exiger d'être livré d'une certaine quantité à une date donnée.

À notre connaissance, les seuls travaux publiés sur ce problème utilisant des données réelles sont ceux de Campbell *et al.* [28], Campbell et Savelsbergh [29], Savelsbergh et Song [102, 103]. L'unique particularité non prise en compte dans notre modèle et inclue dans les modélisations proposées dans la littérature concerne la variabilité des temps de chargements / livraisons en fonction de la quantité de produit.

Fonction objectif modifiée. Alors que l'objectif de l'IRP est d'optimiser la distribution sur le long terme, les prévisions de consommation sont quant à elles connues généralement sur le court terme et révisées en continu. C'est pourquoi des horizons glissant sont généralement utilisés. Un planning à court terme sur 15 jours est construit par exemple et seules les tournées du premier jour sont fixées. Puis le jour suivant, un nouveau planning est construit en prenant en compte les tournées du jour d'avant et en ajoutant une journée à l'horizon temporel.

Comme mentionné par Campbell *et al.* [27, 28] ainsi que par Bell *et al.* [10], la première difficulté qui survient quand nous modélisons le problème d'optimisation de tournées de véhicules avec gestion des stocks est la définition appropriée des objectifs à court terme afin d'obtenir de bons résultats à long terme. Un objectif utilisé couramment dans la littérature est de maximiser le volume par kilomètre sur le long terme [27, 28, 102, 103], obtenu en divisant la quantité totale livrée à tous les clients par la distance totale parcourue. Au lieu d'utiliser uniquement la distance parcourue, nous prenons en compte dans notre approche le coût réel des tournées, en utilisant une modélisation précise du coût de chaque tournée en fonction de la distance parcourue, du temps de trajet, du nombre de chargements, du nombre de livraisons et du nombre de pauses. Pour cela, nous utilisons une fonction objectif de substitution permettant de minimiser le coût par unité de produit livré, appelé *Ratio Logistique* dans ce papier. Une de nos contributions est ici de résoudre le problème de planification à court terme avec une fonction objectif de substitution permettant une optimisation à long terme. La planification à court terme est construite pour 15 jours dans les détails mais nous conservons uniquement les tournées débutant le premier jour avant de faire glisser l'horizon planning. La fonction objectif de substitution fait appel à des bornes inférieures obtenues au préalable pour chaque client. Nous détaillerons par la suite comment ces bornes sont obtenues. Des expérimentations sur les jeux de données réels montrent que cette fonction objectif modifiée apporte des gains significatifs sur le long terme en comparaison à une stratégie d'optimisation à court terme.

Approche directe. Avant de présenter notre approche de résolution, nous détaillons les résultats obtenus par Campbell *et al.* [28], Campbell et Savelsbergh [29] pour résoudre le problème d'optimisation de tournées de véhicules avec gestion des stocks avec une seule usine, et les résultats de Savelsbergh *et al.* [102, 103] pour la résolution avec plusieurs usines.

Les méthodes de résolution des deux premiers papiers sont assez similaires, même si le papier de 2004 propose un modèle plus complexe car il prend en compte des contraintes réelles additionnelles. Les méthodes sont déterministes et s'appuient sur une décomposition en deux phases. Dans une première phase utilisant des techniques de programmation en nombres entiers, il est décidé quels clients sont livrés et quel est leur premier volume cible de livraison. Puis une deuxième étape résolue à l'aide d'heuristiques d'insertion permet de prendre en compte les contraintes de capacités des véhicules, les fenêtres d'ouverture, les restrictions de conduite, *etc.* La première phase est résolue à l'aide d'une heuristique faisant appel à des techniques de programmation linéaire en nombres entiers, tandis que la seconde phase est résolue avec des heuristiques spécifiques d'insertion [31], comme dans les papiers décrivant des problèmes de tournées de véhicules avec des fenêtres de temps (Solomon, [105]). Le premier problème (Campbell *et al.* [28]) concerne des plannings sur un horizon glissant de 5 jours et une vision agrégée sur 4 semaines. Les 2 instances comportent 50, puis 87 clients avec 4 véhicules. La solution est comparée à un algorithme glouton (similaire à celui présenté dans [28]) : elle représente un gain de 8.11 % de volume par mile et une meilleure utilisation des ressources. Dans le deuxième papier (Campbell et Savelsbergh, [29]) travaillent sur une période de 10 jours (3 jours détaillés et 7 jours agrégés) glissant sur 1 mois. À chaque itération, la première phase est résolue par programmation entière sur 3 jours avec le niveau de détails le plus fin plus une semaine sous la forme agrégée puis la deuxième phase est résolue à l'aide de l'heuristique d'insertion en utilisant l'information du programme linéaire pour les deux premiers jours. Puis les routes résultantes sont fixées et nous décalons l'horloge de deux jours à chaque fois. Le temps de calcul pour effectuer une itération est limité à 10 minutes (avec un processeur de 366 MHz). L'auteur compare son approche à un algorithme glouton similaire à celui décrit dans Campbell *et al.* [28]. Les jeux de données sont composés de deux instances comportant au moins 100 puis 50 clients respectivement (les ressources disponibles ne sont pas détaillées). Le gain moyen sur un mois est de 2,7 % pour le volume par mile avec une meilleure utilisation des volumes des camions et des tournées plus courtes (meilleure optimisation du remplissage des camions, longueur moyenne des tournées plus petite).

Dans Savelsbergh *et al.* [102, 103], les auteurs développent deux approches pour résoudre le problème d'optimisation de tournées de véhicules avec gestion des stocks avec plusieurs usines. Plusieurs caractéristiques réelles prises en compte dans Campbell et Savelsbergh [29] sont relâchées dans ce dernier modèle. En particulier, les ressources sont modélisées par une remorque se déplaçant entre des réservoirs, permettant de faire apparaître un flot à variables entières. La première approche ([102]) est fondée sur une heuristique d'insertion qui livre les clients en les ordonnant selon leur urgence tout en minimisant les assèchements et les coûts de transport. Cette approche est déclinée en trois algorithmes gloutons : un algorithme basique (appelé BGH) où les insertions sont seulement réalisées à la fin des tournées, un algorithme amélioré (EGH) où les insertions peuvent être réalisées n'importe où dans la tournée après le dernier chargement et enfin l'algorithme amélioré aléatoire (RGH) où l'approche EGH est améliorée à l'aide d'une procédure adaptative de recherche aléatoire [47]. Puis un post-

traitement est appliqué en utilisant la programmation linéaire pour maximiser les quantités livrées sur les tournées résultantes (afin de maximiser le volume par mile). Les auteurs présentent des résultats sur 20 instances dérivées à partir d'une instance de 200 clients, 7 usines, 7 véhicules. Sur un horizon de 10 jours, l'amélioration moyenne pour les coûts d'assèchement et de transport entre l'approche BGH et EGH (resp. entre EGH et RGH) est de 15,2 % (resp. 6,8 %). Les temps de calcul pour les deux premiers algorithmes sont de quelques secondes alors qu'il faut environ 12 minutes pour le RGH. La post-optimisation permet d'augmenter la quantité livrée de 2,8 % en moyenne sur les mêmes jeux de données (avec un temps de calcul de moins d'une seconde). D'autres expérimentations faites sur un horizon glissant de 5 mois (avec 10 jours planifiés, 5 jours fixés) montre que le volume livré pendant la post-optimisation permet de réduire les coûts d'environ 3 % (en utilisant RGH comme l'algorithme de référence). La seconde approche de Savelsbergh *et al.* [103] consiste à résoudre de manière heuristique le programme de flot en nombres entiers entre plusieurs sites (en utilisant des techniques de programmation entière). Les auteurs présentent les résultats obtenus sur 25 jeux de données dérivés d'instances avec 200 clients utilisés comme données de base dans Savelsbergh *et al.* [102]. L'amélioration moyenne par rapport à RGH pour les coûts d'assèchement et de transport est de 4.1 %, tandis que le temps de calcul moyen est de plus de 31 heures (avec un processeur de 900 MHz). Puisque les besoins en temps de calcul sont trop grands en pratique, les auteurs présentent un programme linéaire en nombres entiers pour visiter des voisinages larges dans une recherche locale (voir [43, 44] pour une application de cette technique au problème d'ordonnancement de véhicules). Cette technique consiste à ré-optimiser les emplois du temps de deux véhicules dans le planning en résolvant un programme en nombres entiers avec les autres emplois du temps figés. Ainsi, toutes les paires de véhicules sont ré-optimisées itérativement. Les auteurs rapportent une amélioration moyenne par rapport au RGH de 3.1 %, avec un temps de calcul moyen inférieur à 3 minutes et un nombre moyen d'itérations améliorantes de 3. Malheureusement, il n'est pas mentionné dans Savelsbergh *et al.* [102, 103] de statistiques détaillées à propos du volume par mile résultant sur le long terme.

L'heuristique originale de recherche locale décrite pour résoudre le problème de planification à court terme suit la méthodologie d'ingénierie algorithmique introduite par Estellon *et al.* [45]. Cette méthode a été utilisée avec succès pour résoudre d'autres problèmes industriels tels que l'ordonnancement de chaînes de montage de véhicules pour Renault (Estellon *et al.* [43, 44]) ou encore la planification d'activités pour France Telecom (Estellon *et al.* [45]). On retrouve dans la littérature une approche par recherche locale pour résoudre un problème d'IRP avec fenêtre de temps. Cependant, la modélisation utilisée décompose le problème : la gestion des stocks est optimisée dans une première phase puis les tournées de livraisons sont planifiées. Comme nous l'avons déjà dit en introduction de la thèse, il n'y a aucune décomposition dans l'approche présentée ici : les 15 jours de planning sont traités directement et optimisés dans leur globalité par la recherche locale. Des tests sur un nombre important d'instances ont prouvé que cette approche était à la fois efficace et robuste et qu'elle permettait de fournir des gains à long terme dépassant les 20 % en moyenne, en com-

paraison avec des solutions calculées par des planificateurs experts ou même par rapport à un algorithme glouton fondé sur l'urgence. Nous appliquons ici la méthodologie en 3 couches présentée dans l'introduction de cette thèse. La première couche correspond à la stratégie de recherche. Nous utilisons ici une heuristique de descente à la première amélioration avec une sélection aléatoire des mouvements (une solution initiale est calculée en utilisant une heuristique gloutonne d'insertion fondée sur l'urgence des demandes). La deuxième couche correspond à la définition de l'ensemble des mouvements qui vont définir le voisinage. On retrouve ici plus d'une centaine de transformations différentes en tout, qui peuvent être regroupées en une douzaine de familles (pour les opérations : l'insertion, la suppression, l'éjection, le déplacement, l'inversion ; pour les opérations : l'insertion, la suppression, le déplacement, l'inversion, la fusion et la séparation). Nous détaillerons ces mouvements dans la Section 4.4.3. Enfin, la troisième couche concerne le moteur de la recherche locale, qui s'appuie sur les 3 procédures d'évaluation, de validation et de retour en arrière. Puisque la durée d'une opération ne dépend pas de la quantité chargée ou livrée, la procédure d'évaluation peut être séparée en 2 sous-fonctions : la première calcule le planning des tournées et l'autre les affectations de volumes. Ces fonctions, dont le temps de calcul est critique pour les performances, s'appuient sur des structures de données spécialement conçues pour exploiter les invariants des transformations. En moyenne, notre algorithme visite plus de 10 millions de solutions dans l'espace de recherche pendant les 5 minutes de temps de calcul, avec un taux de diversification de presque 5 % (c'est-à-dire le nombre de transformations validées sur le nombre de transformations évaluées), qui permet d'atteindre rapidement des optima locaux de grande qualité. Nous verrons en Section 4.4.4 quelles structures et quels algorithmes ont été utilisés pour rendre efficace l'évaluation, la validation ou l'annulation de ces transformations.

4.3 Objectif de substitution à court terme

Minimiser le ratio logistique à court terme ne conduit pas nécessairement à de bonnes solutions sur le long terme. En effet, si nous considérons un client que l'on peut livrer à moindre coût pendant notre période de 15 jours mais qui n'en a pas la nécessité puisqu'aucun assèchement n'apparaît pendant cette période, la fonction objectif à court terme ne verra pas d'intérêt à réaliser une livraison. Ce manque d'anticipation lors de la minimisation du ratio logistique sur un horizon court terme nous a fait introduire une fonction objectif modifiée. Son objectif peut se résumer ainsi : "ne jamais remettre à demain ce qu'on peut faire de façon optimale aujourd'hui". L'objectif à court terme devient alors de minimiser le coût global additionnel par unité de produit livré, comparé au ratio logistique optimal RL^*. Si nous notons $RL^*(c)$ le ratio optimal du client c et

$$\mathrm{CT}^*(s) = \sum_{clients\ c\ livres\ pendant\ t} RL^*(c) \times quantite(c)$$

le coût optimal d'une tournée s dépendant de la quantité livrée à chaque client au court de la tournée. Alors le ratio logistique modifié RL^* est défini comme :

$$RL^* = \frac{\sum_s (CT(s) \; - \; CT^*(s))}{QT}$$

Les coûts par unité de produit livré $RL^*(p)$ sont calculés en phase préliminaire en sélectionnant pour chaque client la partie de tournée livrant ce client, avec un coût logistique le plus petit. Nous divisons ce coût par la capacité maximale des capacités de tous les camions accédant à ce client et nous obtenons une borne inférieure sur le coût par unité livrée à ce client.

Nous cherchons à calculer une borne inférieure sur le ratio logistique, notée RL_{min}, en faisant l'hypothèse qu'il n'y a ni commandes insatisfaites, ni assèchements dans la solution. La fonction objectif ayant un ordre lexicographique, le triplet (0, 0, RL_{min}) est aussi une borne inférieure du problème global. Nous considérons donc pour la suite que le nombre de commandes non satisfaites et d'assèchements est nul et le but est de calculer RL_{min}.

Tout d'abord, une borne inférieure pour $RL^*(c)$ est donnée pour chaque client c. Un voyage est défini comme une sous-partie d'une tournée (voir la Figure 4.3) : c'est une séquence de visites démarrant à une usine (ou un dépôt), livrant un ou plusieurs clients, et finissant à une usine (ou à un dépôt). En d'autres termes, un voyage $v \in VOYAGES(s)$ dans une tournée s correspond à un intervalle $[debut(v), fin(v)[$ avec $debut(v)$ (resp. $fin(v)$) la date de début depuis l'usine ou le dépôt (resp. la date de départ depuis l'usine ou le dépôt visité dans le voyage suivant). Ainsi, les coûts d'une tournée s peuvent être décomposés suivant ceux de ses voyages, de telle manière que le coût d'un voyage corresponde aux coûts (distance, temps, livraisons, chargements, pauses) accumulés pendant $[debut(s), fin(s)[$. Par ailleurs, le coût de chaque voyage peut être distribué aux clients visités proportionnellement aux quantités livrées. Pour chaque client p, une borne inférieure $RL_{\min}(p)$ peut être ainsi calculée en divisant le coût du plus petit voyage visitant c par la capacité maximale des camions capables de réaliser ce voyage. Puisque la matrice de distance respecte l'inégalité triangulaire, le moins coûteux des voyages consiste donc à visiter uniquement le client c. Par conséquence, $RL_{\min}(c)$ est calculé en $O((D+U)^2)$ en temps pour chaque client c, avec D le nombre de dépôts et U le nombre d'usines.

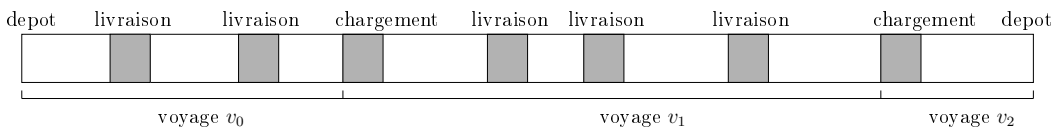


FIGURE 4.3 – Les voyages d'une tournée.

Maintenant, nous sommes en mesure d'utiliser cette borne inférieure locale $RL_{\min}(p)$ dans le calcul d'une borne inférieure globale $RL_{\min}$. Pour chaque client c, nous notons $Q_{\min}(c)$ la

quantité minimale à livrer à c afin d'éviter qu'il tombe sous son seuil de sécurité avant la fin du planning. D'un autre côté, nous notons $Q_{\max}(c)$ la quantité maximale de produit livrable d'ici la fin du planning sans provoquer de dépassement de la capacité du réservoir du client c. Enfin, nous notons $q(c)$ la quantité qui peut être livrée à chaque client c afin d'éviter un assèchement avec :

$$Q_{\min}(c) \leq q(c) \leq Q_{\max}(c)$$

De plus, on a :

$$\begin{aligned} Q_{\min}(c,\ h) &= \mathit{SeuilSecurite}(c) - (\mathit{quantiteInitiale}(c) - \textstyle\sum_h \mathit{previsions}(c,\ h)) \\ Q_{\max}(c,\ h) &= \mathit{Capacite}(c) - (\mathit{quantiteInitiale}(c) - \textstyle\sum_h \mathit{previsions}(c,\ h)) \end{aligned}$$

Si $Q_{\min}(c) = 0$ pour tous les clients c, alors la solution vide (pas de tournée) est optimale. Maintenant, faisons l'hypothèse qu'au moins un client c existe tel que $Q_{\min}(c) > 0$. Une première borne inférieure $RL_{\min}$ est donnée par :

$$RL_{\min} = \frac{\sum_c (RL_{\min}(c) \times Q_{\min}(c))}{\sum_c Q_{\max}(c)}$$

Chaque terme du numérateur correspond à un coût minimal des tournées nécessaires pour livrer la quantité $Q_{\min}(c)$ au client c au cours du planning. Mais une meilleure borne peut être obtenue en résolvant le programme mathématique suivant :

$$\begin{aligned} &min\ \frac{\sum_c (RL_{\min}(c) \times q(c))}{\sum_c q(c)} \\ &q(c) \in [Q_{\min}(c), Q_{\max}(c)] \quad \forall c \end{aligned}$$

Nous notons un vecteur solution du programme $Q = (q(1), \ldots, q(n))$ et son coût associé par $f(Q)$. Tout d'abord, une solution optimale Q^* de ce programme est prouvée comme extrême : le vecteur Q est tel que $q(c) = Q_{\min}(c)$ ou $q(c) = Q_{\max}(c)$ pour chaque composante c. De plus, une solution Q^* est optimale si et seulement si aucune solution Q existe telle que $g(Q) = \sum_c ((RL_{\min}(c) - f(Q^*)) \times q(c)) < 0$. On suppose qu'un index c existe de telle façon que $q(c)$ n'est pas un extremum de $[Q_{\min}(c), Q_{\max}(c)]$. Si $RL_{\min}(c) - f(Q^*) \leq 0$ (resp. $RL_{\min}(c) - f(Q^*) \geq 0$), alors fixer $q(c) = Q_{\max}(c)$ (resp. $q(c) = Q_{\min}(c)$) permet d'atteindre une solution ayant un coût plus petit ou égal à $f(Q^*)$. Comme cette opération peut être réalisée de manière indépendante pour chaque index c (parce que $g(Q)$ est séparable par addition), le lemme est prouvé.

Maintenant, tout vecteur extremum optimal peut être normalisé en ordonnant ses composantes afin que les constantes correspondantes $RL_{\min}(c)$ soient non décroissantes. Suivant la même logique que précédemment, un extremum optimal Q^* a une forme normée

$(Q_{\max}(1), \ldots, Q_{\max}(c^*), Q_{\min}(c^*+1), \ldots, Q_{\min}(n))$, avec $RL_{\min}(1) \leq \cdots \leq RL_{\min}(n)$. Par conséquence, le calcul d'un extremum optimal est réduit au calcul d'un index $c^* \in \{1, \ldots, n\}$ pour lequel la forme normalisée a un coût minimum (opération réalisée en temps linéaire). En conclusion, le calcul d'un vecteur optimum Q^* dans notre algorithme est fait en $O(n \log n)$ en temps et en espace linéaire, avec n le nombre de composantes du vecteur.

Pour résumer, les bornes inférieures locales $RL_{\min}(c)$, définies pour chaque client c, sont calculées en $O(C(D+U)^2)$ en temps et $O(C)$ en espace, avec C (resp. U, D) le nombre de clients (resp. usines, dépôts). Alors, la borne inférieure globale $RL_{\min}$ est calculée en $O(C \log C)$ en temps et $O(C)$ en espace.

4.4 Approche par recherche locale

De par sa complexité et sa taille, ce problème est typique des grands problèmes industriels d'optimisation en variables mixtes. Dans cette section, nous décrivons comment nous utilisons les principes mentionnés en introduction de cette thèse pour résoudre ce problème de manière efficace et robuste. Nous utilisons une décomposition de la conception : une stratégie de recherche simple, des mouvements adaptés à la structure du problème et une algorithmique poussée d'évaluation des mouvements.

4.4.1 Stratégie et heuristique

La stratégie de recherche est ici de type descente standard avec choix stochastiques des mouvements. Notons que nous n'utilisons pas ici de métaheuristique. Le mécanisme global de l'heuristique est résumé dans l'Algorithme 10.

L'heuristique est ici divisée en trois phases d'optimisation : la première (MO) consiste à minimiser le coût relatif aux commandes, la seconde (SO) consiste à minimiser le coût relatif aux assèchements et enfin la troisième (RL) consiste à optimiser l'objectif relatif au ratio logistique. En pratique, le temps total d'exécution est divisé ainsi : 10 % pour l'optimisation de MO, 40 % pour l'optimisation de SO, 50 % pour l'optimisation de RL. De même, la procédure qui évalue le gain d'une transformation est séparée en trois niveaux (voir Figure 4.4). Il faut noter que l'acceptation de solution à coût égal est cruciale pour assurer une bonne diversification de la recherche et ainsi converger vers des solutions de grande qualité.

Le gain résultant de l'application d'une transformation T est la différence entre la valeur du coût avant l'application de T sur la solution courante S (l'ancienne) et sa valeur après son application (nouvelle). Expliquons maintenant comment sont calculés les gains à chaque étape du schéma d'évaluation.

Un coût de substitution sur les commandes non traitées MO' est défini pour lisser l'objectif réel MO afin de faciliter la convergence de la recherche locale. Pour cela, nous introduisons

Algorithme 10: DESCENTE-STOCHASTIQUE

input : Une instance de l'IRP
output : Une solution de l'IRP

begin
- Initialiser S avec Algorithme-glouton;
- **Optimisation des commandes (MO)**;
- **while** *coutCommandesManquees > 0 et* $TempsMax_C$ *non dépassé* **do**
  - Choisir aléatoirement une transformation t dans l'ensemble T_C;
  - Évaluer le gain de l'application de t à S;
  - **if** *le gain n'est pas négatif* **then** Appliquer t à S;
  - **else** Revenir en arrière à l'état courant;
- **Optimisation des assèchements (SO)**;
- **while** *coutAssechement >0 and* $TempsMax_A$ *non dépassé* **do**
  - Choisir aléatoirement une transformation t dans l'ensemble T_A;
  - Évaluer le gain de l'application de t à S;
  - **if** *le gain n'est pas négatif* **then** Appliquer t à S;
  - **else** Revenir en arrière à l'état courant;
- **Optimisation des coûts logistiques (RL)**;
- **while** $TempsMax_{CL}$ *non dépassé* **do**
  - Choisir aléatoirement une transformation t dans l'ensemble T_{CL};
  - Évaluer le gain de l'application de t à S;
  - **if** *le gain n'est pas négatif* **then** Appliquer t à S;
  - **else** Revenir en arrière à l'état courant;

un état intermédiaire appelé "non-satisfaite" entre l'état "commande manquée" et "commande satisfaite". Une commande est "non-satisfaite" même si une opération existe et satisfait la fenêtre de temps de la commande, mais pas sa quantité. Ainsi, une commande est "satisfaite" (resp. "manquée") quand aussi bien les dates et les quantités sont respectées par au moins une opération (resp. par aucune opération). Si on note le nombre de commandes non-satisfaites par UO, la valeur du $gain_{MO'}$ est calculée ainsi :

$$\begin{aligned} si\ MO_{\text{old}} \neq MO_{\text{new}}\ alors\ gain_{MO'} &= MO_{\text{old}} - MO_{\text{new}} \\ sinon\ gain_{MO'} &= UO_{\text{old}} - UO_{\text{new}} \end{aligned}$$

Par conséquent, une transformation ne peut pas être acceptée si le nombre de commandes manquées ou si le nombre de commandes non satisfaites est détérioré. Le gain relatif aux assèchements est calculé ainsi : $gain_{SO} = SO_{\text{old}} - SO_{\text{new}}$. Au final, le signe de $gain_{RL'}$ est obtenu en évaluant l'expression suivante :

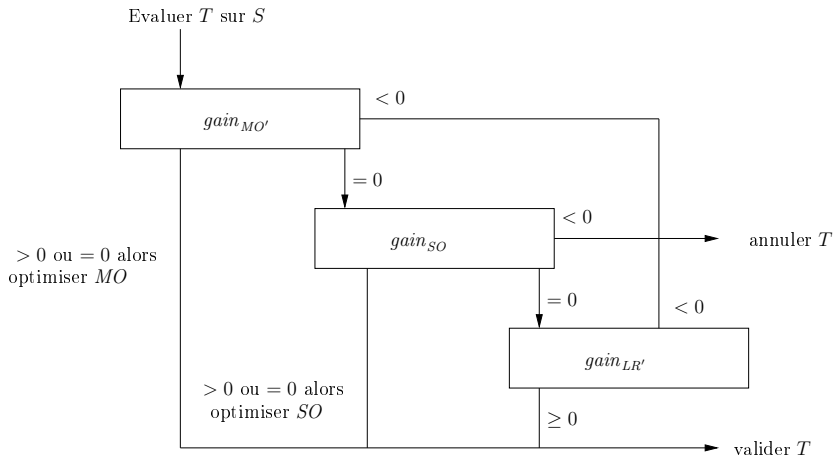


FIGURE 4.4 – Schéma d'évaluation d'une transformation.

$$\frac{SC_{\text{old}} - SC^*_{\text{old}}}{DQ_{\text{old}}} - \frac{SC_{\text{new}} - SC^*_{\text{new}}}{DQ_{\text{new}}}$$

ou son équivalent : $DQ_{\text{new}}(SC_{\text{old}} - SC^*_{\text{old}}) - DQ_{\text{old}}(SC_{\text{new}} - SC^*_{\text{new}})$ ce qui réduit les imprécisions dues aux arrondis quand l'expression tend vers zéro. L'efficacité pratique de l'heuristique présentée ici s'appuie sur 2 points cruciaux : les mouvements variés et les algorithmes efficaces d'évaluation. Avant de fournir des détails concernant les mouvements, nous introduisons l'algorithme permettant d'obtenir une solution initiale.

4.4.2 Solution initiale

Nous initialisons la recherche locale à l'aide d'un algorithme glouton formalisé dans l'Algorithme 11. Pour chaque client, on calcule la première date d'assèchement à partir de son niveau initial et de ses prévisions de consommation. De plus, on connaît pour chacun la liste des commandes avec leur date associée. L'algorithme considère un à un ces deux types de demandes ordonnées suivant leur date d'occurrence. Pour chacune, l'insertion la moins coûteuse à la fin d'une tournée existante est évaluée puis comparée au coût de la création d'une tournée dédiée pour servir ce client avant cette date.

Dans la première hypothèse d'insertion du client à une tournée existante, il s'agit de vérifier que l'amplitude temporelle maximum de la tournée ne sera pas dépassée à cause de cette insertion. De plus, le triplet de ressources utilisées dans cette tournée doit être disponible pour cet ajout et autorisé à visiter ce client. Si le niveau du camion en fin de tournée ne permet pas de satisfaire le besoin du client, on ajoute un passage à une usine.

Dans le cas d'une création de tournée dédiée, on construit une tournée dédiée à l'aide d'une heuristique simplifiée : l'objectif est de trouver une base et une usine ouverte suffisam-

ment proches pour former une première tournée élémentaire consistant à quitter cette base, à visiter l'usine et le client puis à revenir à la base. L'heuristique cherche aussi un triplet de ressources disponibles en ces points pour satisfaire les contraintes spatio-temporelles. Des coupes précises permettent de trouver rapidement un résultat parmi les ressources disponibles et les sites accessibles. Dans le cas des assèchements, on cherche à livrer juste avant que le niveau ne tombe sous le seuil de sécurité, afin de livrer en une fois la plus grande quantité possible. En quelques secondes, une première solution respectant le jeu de contraintes décrit en Section 4.1 est trouvée et peut être comparée au meilleur coût de l'insertion dans une tournée existante.

Algorithme 11: ALGORITHME-GLOUTON

input : Une instance de l'IRP
output : Une solution de l'IRP (un ensemble de tournées)

begin
  $S \leftarrow \varnothing$;
  Initialiser l'ensemble des demandes D avec les assèchements de tous les clients;
  while D *n'est pas vide* **do**
    Sélectionner la demande $d \in D$ avec la plus petite date d'occurence;
    Créer une livraison L la moins coûteuse possible pour satisfaire d;
    if L *existe* **then**
      Insérer o dans une tournée S (créer si besoin une nouvelle tournée);
      Calculer le prochain assèchement après la livraison L;
      Mettre à jour la deadline de d en conséquence;
    else Retirer d de l'ensemble D;

Cet algorithme ne revient jamais ni sur les décisions prises sur les dates et ni sur les volumes. Une fois cette première solution obtenue, on est en mesure de débuter la descente par recherche locale, en utilisant une série de mouvements que nous détaillerons par la suite.

4.4.3 Mouvements

Les mouvements (ou transformations) sont classés en deux catégories distinctes : les premières travaillant sur les opérations, les autres sur les tournées. Ayant introduit les différentes transformations, nous nous focaliserons ensuite sur la description de leurs déclinaisons (c'est-à-dire la manière dont les objets modifiés sont sélectionnés). Alors que la création de transformations "orthogonales" (c'est-à-dire des transformations produisant des voisinages disjoints) permet de diversifier la recherche et ainsi d'atteindre des solutions de meilleures qualités, les transformations spécialisées selon des spécificités du problème permettent quant à elles d'intensifier la recherche et d'accélérer la convergence de l'heuristique.

Les mouvements sur les *opérations* sont regroupés dans les types suivants : insertion, suppression, éjection, déplacement et échange. La Figure 4.5 présente le mode de fonctionnement de ces transformations. Les tournées originales sont données par des arcs pleins, les arcs supprimés sont représentés par des traits pointillés, les arcs courbés ou verticaux sont ajoutés par la transformation. Deux types d'*insertion* sont définis : le premier type consiste à insérer une opération à l'intérieur d'une tournée existante ; le deuxième consiste à insérer un chargement suivi d'une livraison à l'intérieur d'une tournée (on choisit l'usine à insérer la plus proche du client inséré). La *suppression* consiste à supprimer un bloc d'opérations (c'est-à-dire un ensemble d'opérations consécutives) dans une tournée. Une *éjection* a pour but de remplacer une opération existante par une nouvelle d'un site différent. Le *déplacement* extrait un bloc d'opérations d'une tournée existante et le réinsère à une autre position. Deux types de déplacements peuvent avoir lieu : celui déplaçant les opérations d'une tournée dans une autre, ou ceux déplaçant les opérations au sein de la même tournée. Un *échange* consiste à inverser deux blocs d'opérations. Comme pour le *déplacement*, il y a plusieurs types d'échanges possibles : l'échange de blocs d'une même tournée, entre deux tournées ou encore le *miroir* consistant à faire une inversion chronologique d'un bloc d'opérations dans une tournée. La transformation miroir correspond à la transformation connue sous le nom de "2-opt" notamment utilisée dans les problèmes de voyageur de commerce (Voir le livre de Aarts *et al.* [1]).

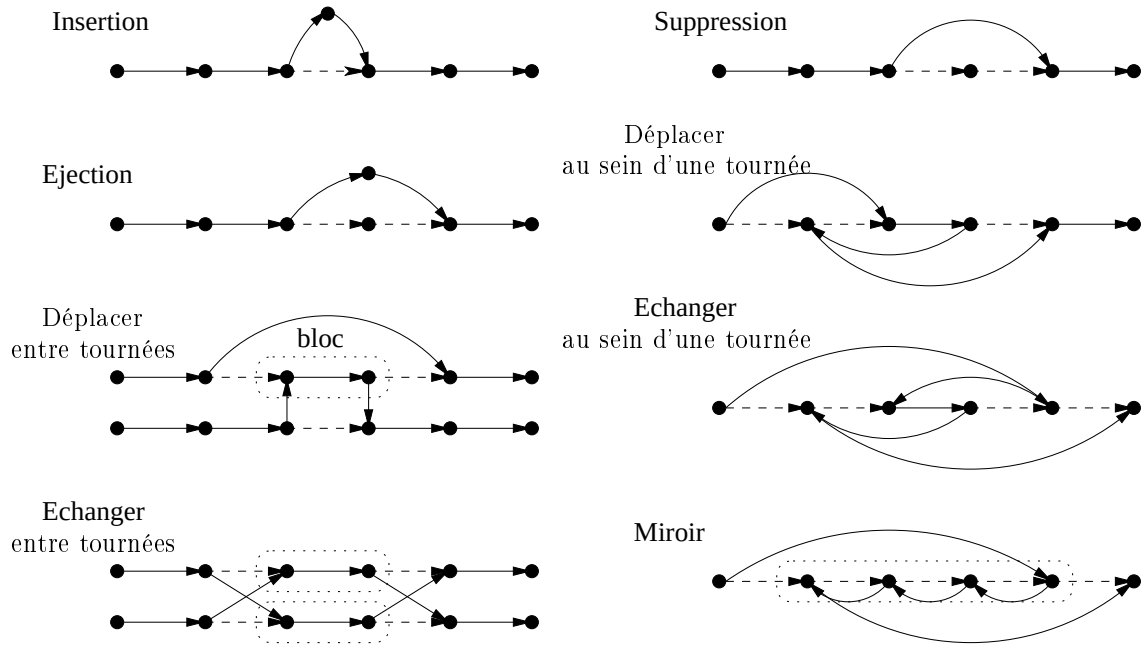


FIGURE 4.5 – Les transformations sur les opérations.

Les mouvements sur les *tournées* appartiennent quant à eux aux types suivants : insertion, suppression, glissement, déplacement, échange, la fusion et la séparation. Comme

pour les opérations, deux types d'*insertion* sont définis : l'insertion d'une tournée contenant une opération (chargement ou livraison) ou l'insertion d'une tournée avec une opération de chargement suivie d'une livraison. La *suppression* consiste à retirer une tournée existante. Le *glissement* permet la translation d'une tournée dans le temps. Le *déplacement* permet d'extraire une tournée du planning de certaines de ses ressources pour la ré-insérer dans le planning d'autres ressources (une telle transformation autorise le changement de certaines ressources et la date de départ). L'*échange* est défini de manière similaire : les ressources des tournées sont échangées et leur date de début peut subir une translation temporelle. La *fusion* de deux tournées en une seule tournée ainsi que la *séparation* d'une tournée en deux autres est aussi disponible ici.

Toutes ces transformations sont déclinées suivant différentes stratégies. La première option concerne la taille maximale des blocs pour les transformations où les blocs d'opérations sont utilisés. Ainsi, des transformations plus génériques sont définies afin d'augmenter la diversification si nécessaire : la *(k,l)-éjection* remplace k opérations existantes par l nouvelles sur des sites différents, le *k-déplacement* déplace un bloc de k opérations, le *(k,l)-échange* réalise un échange un bloc de k opérations par un bloc de l autres et enfin le *k-miroir* inverse un bloc de k opérations.

Ensuite, la deuxième option permet de spécialiser les transformations en fonction de la phase d'optimisation. Par exemple, l'insertion d'une livraison à un client qui n'a pas de commande non satisfaite n'est pas intéressante quand on est en train de minimiser les coûts associés. C'est aussi le cas pour les assèchements. Dans la même idée, échanger deux opérations entre deux sites très éloignés a peu de chance de se solder par un succès quand on est en phase de minimisation des coûts logistiques. Plusieurs déclinaisons ont été mises en place qui diffèrent légèrement d'une transformation à l'autre : celles permettant un choix de position favorisant les livraisons satisfaisant une commande, celles favorisant la résolution des assèchements et enfin celles permettant de choisir des clients proches à insérer ou à échanger afin de baisser les coûts logistiques.

Puis, la troisième option concerne la direction du calcul des dates pour les tournées modifiées : les dates peuvent être calculées vers l'avant en considérant que la date de début est fixée et que les dates des opérations suivantes se déduisent de celle-ci, soit vers l'arrière en fixant la date de fin et en remontant dans le temps. Cette option est valable pour toutes les transformations, sauf pour la suppression des tournées. Pour les transformations modifiant deux tournées en même temps (par exemple le déplacement d'une opération entre deux tournées), il en résulte quatre instanciations possibles : "en arrière/en arrière", "en arrière/en avant", "en avant/en arrière", "en avant/en avant".

Enfin, la quatrième et dernière option permet d'augmenter le nombre d'opérations dont la quantité sera modifiée par la routine d'affectation du volume. L'opération de recalcul des volumes après une transformation peut permettre de résoudre des assèchements mais augmente considérablement le temps nécessaire à l'évaluation de la transformation par rapport à une simple application de la transformation (ce point sera détaillé dans la section suivante).

Il est important de noter ici qu'aucun voisinage large n'est employé. Les voisinages explorés ici ont une taille d'environ $O(n^2)$ avec n le nombre d'opérations et de tournées dans la solution courante, mais la constante cachée dans la notation O est grande. Le nombre de transformations dans $\mathcal{T}_{MO}$, $\mathcal{T}_{SO}$, $\mathcal{T}_{RL}$ pour les trois phases MO, SO puis RL sont respectivement de 47, 49 et 71. Le Tableau 6.21 et le Tableau 6.22 de l'Annexe 6.3 listent les transformations appliquées à chaque phase. Pour chaque phase, la transformation à appliquer est choisie aléatoirement dans chaque ensemble de transformations. Les distributions non-uniformes n'apportent pas d'amélioration significative et ne sont donc pas utilisées ici ce qui facilite aussi la maintenance, les évolutions et la robustesse.

4.4.4 Machinerie d'évaluation

Cette section a pour but de présenter des détails concernant deux routines du moteur de calcul. Elles sont appelées à chaque évaluation d'un mouvement puis lors de sa validation, sur la tournée modifiée ainsi que sur toutes celles transformées par le mouvement. Tout d'abord, on doit calculer les dates d'une tournée depuis une opération, jusqu'au début ou à la fin de l'opération (nous parlerons alors de "calcul en avant" ou "calcul en arrière"). Ensuite, il s'agit d'affecter tous les volumes livrés (resp. chargés) à chaque client (resp. usine) d'une tournée étant donné le niveau initial du camion.

Ordonnancement des tournées. Les mouvements modifient les tournées de la solution courante (au plus deux). Quand une tournée est modifiée par une transformation (par exemple, une opération est insérée dans la tournée), les dates de début et de fin de ses opérations doivent être calculées à nouveau. Soit une tournée $\tilde{s} = (o_1; \ldots; o_n)$ et faisons l'hypothèse qu'une opération $\tilde{o}$ est insérée dans la tournée s entre les opérations o_i et o_j. La tournée résultante $\tilde{s}$ est maintenant composée des opérations $(o_1; \ldots; o_i; \tilde{o}; o_j; \ldots; o_n)$. Ensuite, deux possibilités se présentent : planifier à nouveau toutes les dates "en avant" ou "en arrière". La stratégie "en avant" (resp. "en arrière") consiste à fixer la date de fin de o_i (resp. la date de début de o_j) afin de calculer à nouveau les dates de début (resp. de fin) des opérations $(\tilde{o}\ ;\ \ldots\ ;\ o_n)$ (resp. $(o_1\ ;\ \ldots\ ;\ \tilde{o}\ )$). Le calcul des dates ici peut être fait sans affectation des volumes, car la durée des opérations ne dépend pas des quantités livrées / chargées. Comme le calcul des dates "en avant" et "en arrière" est fait de manière complètement symétrique en représentant les tournées comme des listes doublement chaînées, la présentation peut être réduite au cas "en avant".

Plus formellement, on doit résoudre le problème de décision suivant, appelé ordonnancement de tournées avec pause : soit une date de début pour la tournée ordonnée $s = (o_1; \ldots; o_n)$, l'objectif est de déterminer s'il existe des dates pour chaque opération telles que la tournée soit admissible. Les deux problèmes d'optimisation équivalents sont : ayant fixé sa date de début, construire une tournée avec une date de fin au plus tôt ou avec un coût minimum. Un problème similaire, appelé *Truckload-Trip-Scheduling*, a été

étudié récemment par Archetti et Savelsbergh [6]. Ce dernier problème est plus restrictif au sens où seule une fenêtre de temps est considérée pour chaque site à visiter et le temps de pause doit être égal à la durée légale. Archetti et Savelsbergh [6] présentent un algorithme en $O(n^2)$ en temps pour résoudre le problème *Truckload-Trip-Scheduling*, avec n le nombre de sites à visiter. Pour des raisons d'efficacité, un algorithme linéaire en temps et en espace a été mis en place pour résoudre heuristiquement le problème d'ordonnancement des tournées. L'Algorithme 12 résume le fonctionnement de cette fonction d'affectation des dates.

Algorithme 12: CALCUL-DATES

input : Une liste d'opérations sur des sites dans l'ordre de visite
output : Une tournée dont les dates sont affectées en respectant les contraintes.

```
begin
    Soit une tournée vide
    forall the Sites à visiter (dans l'ordre des indices) do
        Conduire jusqu'au prochain site (en posant les pauses au plus tard si
        nécessaire)
        if Un temps d'attente est nécessaire (à cause de la fenêtre d'ouverture du site)
        then
            if une pause a été posée au cours du dernier arc then
                Agrandir une des pauses pour absorber le temps d'attente
            else if Une pause est nécessaire (à cause du temps d'attente) ou que le
            temps d'attente est plus grand que le temps de pause then
                Prendre une pause jusqu'à l'ouverture du site (pour absorber le temps
                d'attente s'il y en a)
            else
                Attendre l'ouverture du prochain site
            Réaliser l'opération au prochain site et l'ajouter à la tournée
        if l'amplitude maximale de la tournée est dépassée then Retourner VIDE
        (Infaisabilité)
        Retourner la tournée (Faisabilité)
```

Cette heuristique est "gloutonne" au sens où les opérations sont affectées chronologiquement sans revenir sur les décisions prises. Chaque boucle est réalisée en temps et espace constant (si les pauses ne sont pas stockées explicitement) et l'algorithme complet tourne en $O(n)$. L'algorithme CALCUL-DATES est valide par construction. La clé du problème d'ordonnancement de tournées est de minimiser le temps d'improductivité au cours de la tournée. C'est pourquoi, l'idée sous-jacente de l'algorithme est de prendre les pauses aussi tard que possible au cours de la tournée et d'éviter autant que possible les temps d'attente dus aux fenêtres de temps des heures d'ouverture des lieux. Ici, nous essayons de supprimer les temps d'attente en les convertissant en temps de pause (Voir la Figure 4.6), mais seulement sur l'arc

courant, ce qui est sous-optimal. En effet, notre algorithme pourrait être renforcé en essayant de convertir le temps d'attente en temps de pause sur les arcs précédents (comme introduit par Archetti et Savelsbergh [6]). Mais une telle modification nous conduirait à un algorithme quadratique en temps, ce qui n'est pas souhaitable ici, d'autant plus que l'optimalité ne serait pas garantie à cause des fenêtres de temps multiples. De plus, nous avons observé que le temps d'attente est rarement généré en pratique puisque de nombreuses tournées sont réalisées en une journée et même en une demi-journée, ce qui permet de garantir l'optimalité de l'Algorithme CALCUL-DATES dans pratiquement tous les cas. À notre connaissance, la complexité du problème d'ordonnancement de tournées reste inconnue.

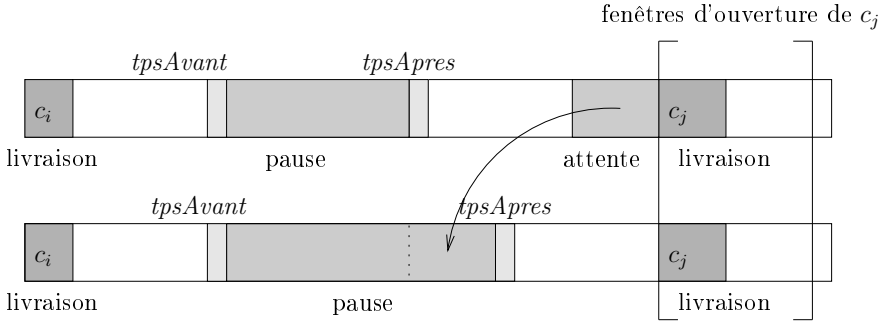


FIGURE 4.6 – Un exemple de conversion de temps d'attente en temps de pause.

Affectation des volumes. Si l'affectation des dates se solde par un succès, l'évaluation est poursuivie en réalisant une affectation des quantités. Cette sous-routine est appelée moins souvent que le calcul des dates précédemment décrit car toute transformation est abandonnée si l'affectation des dates est un échec. Par contre, s'il est validé, il s'agit de trouver la quantité livrée/prélevée pour chaque site, la date de visite de chaque site étant fixée.

Comme dans le sous-problème précédent, on considère ici une unique tournée définie par une base, un ensemble ordonné d'opérations effectuées chacune à une date fixée dans des usines et chez des clients. Cette tournée est réalisée par un triplet de ressources. Le volume initial du camion est supposé connu. En termes de théorie des graphes, ce problème peut être ramené à un problème de flot maximum dans un graphe orienté acyclique. Le problème du calcul des volumes peut être résolu $O(n^3)$ en utilisant un algorithme classique de flot maximum. Cependant, une telle complexité n'est pas acceptable ici, même si cet algorithme permet d'obtenir la solution optimale. Ainsi, une heuristique gloutonne faisant appel à des variables incrémentales maintenues tout au long de la résolution a été mise en place. Cet algorithme, de complexité $O(n \log n)$ en temps, tend à affecter une quantité maximale de produit livré/prélevé à chaque site. Il est donc important que les sites soient ordonnés par date afin de garantir la conservation des flux et le respect des capacités.

Comme le nombre d'opérations peut devenir particulièrement grand (dans le pire des cas, le nombre de livraisons peut atteindre les 2 fois par jour pour les 1500 clients, soit 45000

opérations), un juste milieu doit être trouvé entre une bonne complexité temporelle et une bonne affectation des volumes. Cela revient à trouver un compromis entre une affectation minimale et complète. L'affectation minimale consiste à ne mettre à jour que les volumes des opérations impactées (c'est-à-dire les opérations dont la date de début a été modifiée par le mouvement). Cela peut être effectué en repérant ces opérations par un marqueur lors de l'application de chaque mouvement. Cependant, modifier la quantité prélevée / livrée doit se faire avec la plus grande précaution sous peine de provoquer des assèchements ou des dépassements de capacités sur les opérations futures. On présente ici comment calculer le volume maximal à livrer aux clients. Le volume maximal pouvant être chargé en usine peut être calculé de manière symétrique.

Soient $q_{max}(c, o)$ la quantité maximale livrable à un client c lors de l'opération o afin d'éviter les dépassements de capacité entre la fin de l'opération et l'horizon du planning, $q_{min}(c, o)$ la quantité minimale afin d'éviter un assèchement et $niveauRemorque(r, o)$ le niveau de la remorque r juste avant le début de l'opération o. La quantité livrable au client c lors de l'opération o, notée $q(o)$, est donc bornée par $\min(niveauRemorque(r, o), q_{max}(c, o))$. Cependant, pour s'assurer qu'aucun assèchement n'aura lieu entre cette livraison et le prochain rechargement du camion à une usine, on introduit une nouvelle variable, notée $besoinsFuturs(r, o)$ qui correspond aux besoins futurs de la remorque r après l'opération o. Cela correspond au volume minimal nécessaire pour satisfaire tous les $q_{min}(c', o)$ des clients suivants. La borne peut être alors affinée ainsi :

$$\mathrm{q}(o) \leq \min(niveauRemorque(r, o) - besoinsFuturs(r, o), q_{max}(c, o))$$

L'algorithme glouton met à jour toutes ces données en $O(n)$ en temps et en espace. Ensuite, à chaque transformation, la liste des opérations impactées est ordonnée par ordre chronologique et chacune se voit affecter la quantité maximale à livrer/prélever.

Une fois les dates mises à jour sur les tournées modifiées, il faut mettre à jour les volumes des opérations associées. Les dates étant désormais fixées, le problème consiste uniquement à décider de l'affectation des volumes de manière à respecter les contraintes de gestion des stocks tout en maximisant la quantité totale livrée à l'échelle de toutes les tournées. Un problème similaire, appelé Delivery-Volume-Optimization a été présenté par Campbell et Savelsbergh [30]. Dans ce problème, les auteurs considèrent seulement des livraisons sans chargement et des durées d'opérations dépendantes des quantités livrées.

Ce problème polynomial est un problème de flot maximum dans un réseau acyclique orienté. Le problème du calcul des volumes peut être résolu en $O(n^3)$ en temps en utilisant un algorithme classique de flot maximum Cormen *et al.* [33], avec n le nombre d'opérations. Cependant, une telle complexité n'est pas acceptable ici, même si ce modèle garantit l'optimalité. En pratique, les implémentations naïves avec une complexité dépendant du nombre H de pas de temps (360 en pratique) sont aussi à éviter ici. En effet, quand la granularité devient plus petite qu'une journée, le nombre de pas de temps excède largement le nombre d'opérations pour un site (deux par jour dans le pire des cas).

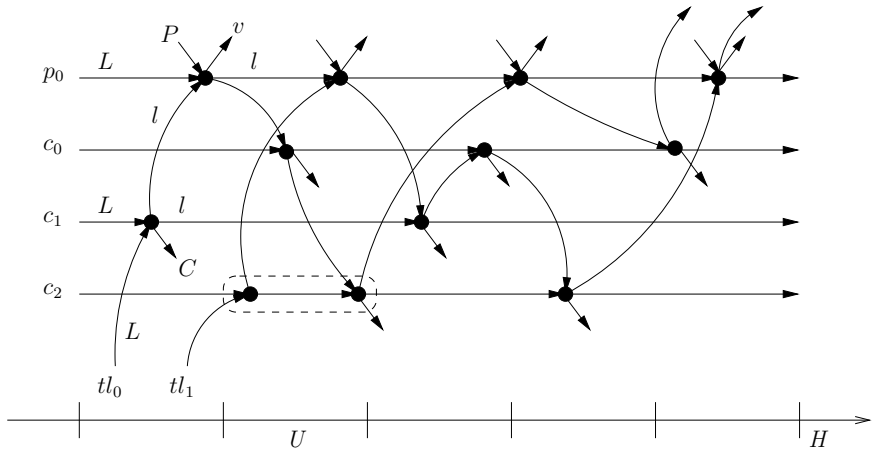


FIGURE 4.7 – Un exemple de réseau de flot pour l'affectation du volume

La Figure 4.7 représente un exemple de réseau de flot pour l'affectation du volume. Les opérations sont représentées par des nœuds, le flot entrant L correspond au niveau initial pour chaque inventaire (remorque, client, usine), le flot entrant C (resp. P) correspond à la consommation du client c_1 (resp. la production de l'usine p_0) pour chaque pas de temps entre l'opération courante et la précédente, les flots L correspondent aux niveaux d'inventaire (remorque, client, usine) entre deux opérations, les flots v autorisent un dépassement aux usines. Les flots sur les arcs représentant les niveaux d'inventaires sont des bornes supérieures sur la capacité des inventaires ; pour les clients, les flots sont aussi bornés inférieurement par les niveaux de sécurité. À noter que si des opérations consécutives apparaissent sur le même pas de temps, les flots entrants correspondant à la consommation ou à la production sont cumulés à la dernière opération de ce pas de temps.

Ainsi, ce besoin d'efficacité algorithmique a motivé le développement d'une heuristique en $O(n \log n)$ en temps pour résoudre efficacement ce problème d'affectation des volumes. L'idée sous-jacente est simple : une fois les opérations triées par ordre chronologique (c'est-à-dire suivant l'augmentation des dates de début), on affecte la quantité maximale dans cet ordre en suivant une règle gloutonne. À chaque opération, on maximise ainsi la quantité livrée/chargée, ce qui est une politique en phase avec le ratio logistique modifié (ceci rejoint certaines idées développées par Campbell et Savelsbergh [30]). Cet ordre est crucial pour garantir le respect des contraintes relatives aux stocks (conservation des flots et contraintes de capacité). En théorie des graphes, cet algorithme revient à pousser un maximum de flot dans un réseau acyclique orienté en respectant l'ordre topologique des nœuds (assurant qu'aucun nœud n'est visité deux fois).

Cependant, n peut devenir tellement grand qu'il est nécessaire de trouver un compromis entre la complexité en temps (même linéaire) et la qualité de l'affectation des volumes. Afin d'introduire un peu de flexibilité ici, l'algorithme a été conçu de manière à faire des

affectations partielles, entre une affectation *minimale* et *complète*. Une affectation minimale consisterait à changer uniquement les volumes des opérations modifiées (celles dont la date de début a été modifiée par la transformation courante) alors que l'affectation complète réaffecte les volumes de toutes les opérations. Le but est donc de marquer comme modifiées plus d'opérations que dans l'affectation minimale, afin d'étendre la réaffectation des volumes, sans pour autant faire une réaffectation complète. En effet, changer la quantité livrée d'une opération est un processus délicat : augmenter (resp. diminuer) la quantité livrée (resp. chargée) peut entraîner un dépassement de capacité (resp. une pénurie) d'une ou plusieurs opération(s) future(s). La détermination du volume maximal à livrer (ou à charger) n'est donc pas un calcul trivial.

Pour chaque site p, on note $\bar{n}_p$ le nombre d'opérations entre la première modifiée et la dernière avant la fin du planning (les opérations sont dans l'ordre chronologique). Si aucune opération n'est modifiée sur le site p, on a $\bar{n}_p = 0$. On définit le nombre total d'opérations impactées $\bar{n} = \sum_p \bar{n}_p$. Quand l'ensemble des opérations modifiées ne contient que des opérations dont la date a été modifiée par la transformation, on observe en pratique que $\bar{n} \ll n$, car chaque transformation modifie au plus deux tournées (le nombre de sites visités par une tournée est généralement petit). Par conséquent, il est plus intéressant de fournir un algorithme qui tourne en un temps linéaire en $O(\bar{n})$, et pas seulement en $O(n)$. Avant de présenter l'algorithme en lui-même, nous précisons comment sont calculées les quantités maximales livrables (les quantités chargeables se déduisant de manière symétrique).

On note $\mathit{niveauClient}(c, o)$ (resp. $\mathit{niveauRemorque}(r, o)$) le niveau du client c (resp. de la remorque r) avant le début de l'opération o et par $\mathit{eviterDebord}(c, o)$ la quantité maximale qui peut être livrée à un client c pendant une opération o sans provoquer de débordement jusqu'à la fin du planning. Ainsi, la quantité livrable à une opération o, notée $\mathit{livrable}(o)$, est bornée supérieurement par $\min\{\mathit{niveauRemorque}(r, o), \mathit{eviterDebord}(c, o)\}$. Cette borne est renforcée car la quantité restante dans la remorque après une livraison doit être suffisante pour éviter les assèchements de clients visités jusqu'au prochain chargement. On note $\mathit{eviterAssech}(c, o)$, la quantité minimale à livrer à une opération o pour éviter un assèchement avant la fin du planning. La quantité minimale $\mathit{necessaire}(r, o)$ qui doit rester dans la remorque r après une opération o pour éviter un assèchement ensuite est calculée en sommant $\mathit{eviterAssech}(c, o)$ pour toutes les opérations entre la courante et le prochain chargement. On a donc :

$$\mathit{livrable}(o) \leq \min\{\mathit{niveauRemorque}(r, o) - \mathit{necessaire}(r, o), \mathit{eviterDebord}(c, o)\}$$

Connaissant la liste d'opérations modifiées ordonnées chronologiquement pour chaque remorque, client et usine, toutes les structures de données mentionnées ci-dessus sont calculables en $O(\bar{n})$ en temps. La mise à jour de $\mathit{niveauClient}(c, o)$ (resp. $\mathit{niveauRemorque}(r, o)$) pour chaque opération o est réalisée en balayant en avant les opérations livrant le client c (resp. réalisées par la remorque r). Celle de $\mathit{eviterDebord}(c, o)$ et de $\mathit{eviterAssech}(c, o)$ pour chaque opération o est fait en balayant en arrière les opérations du client c (on stocke pour

cela les consommations cumulées jusqu'à la fin du planning). Enfin, calculer $necessaire(r, o)$ pour chaque opération o se fait en balayant en arrière les opérations des tournées réalisées par la remorque r. L'Algorithme 13 résume le mode de fonctionnement de cette affectation gloutonne des volumes.

Algorithme 13: AFFECTATION-GLOUTONNE-VOLUME

input : L'ensemble $\mathcal{E}$ de $\bar{n}$ opérations impactées

begin
  Ordonner l'ensemble $\mathcal{E}$ chronologiquement
  Mettre à jour *niveauClient*, *niveauRemorque*, *eviterDebord*, *eviterAssech*, *necessaire*
  for *Chaque opération dans* $\mathcal{E}$ **do**
    Affecter la quantité maximale livrable / chargeable pour cette opération

Un exemple simple est donné pour illustrer la mise à jour des quantités livrées et la mise à jour des structures de données. Un client a 4 opérations planifiées $A = (4, 2)$, $B = (10, 5)$, $C = (14, 8)$, $D = (18, 12)$ avec le premier nombre de la paire représentant le pas de temps auquel l'opération a lieu et le second la quantité livrée. Nous avons ici un planning comptant 24 pas de temps. La capacité du réservoir est de 14 et le niveau de sécurité est de 2. Le Tableau 4.1 donne pour chaque pas de temps h la consommation prévisionnelle (2e ligne) et le niveau du réservoir résultant (4e ligne). Alors, les valeurs des structures de données *eviterAssech* et *eviterDebord* sont détaillées. Ces valeurs sont calculées en arrière par rapport à la fin du planning en appliquant la récurrence suivante :

$$eviterAssech(h) = \max\{eviterAssech(h+1), niveauSecurite - niveauClient(h) + consommation(h)\}$$

$$eviterDebord(h) = \min\{eviterDebord(h+1), capacite - niveauClient(h) + consommation(h)\}$$

Maintenant, considérons qu'une opération B est impactée par une transformation : seule la quantité de l'opération B peut être modifiée. La quantité livrée à l'opération B est fixée à 0. Alors, pour affecter une nouvelle quantité à l'opération B, seules les valeurs de la colonne 10 sont à mettre à jour : $niveauClient(10)$ est diminué de 5, tandis que $eviterAssech(10)$ et $eviterDebord(10)$ sont augmentés de 5. Ainsi, les nouvelles valeurs (temporaires) sont : $niveauClient(10) = 2$, $eviterAssech(10) = 4$ et $eviterDebord(10) = 6$. Cela signifie qu'une

quantité minimale de 4 doit être livrée pour éviter les assèchements pendant toute la durée du planning (ici le pas de temps 17 est critique) et une quantité maximale de 6 peut être livrée sans induire un dépassement (ici le pas de temps 18 est critique). Nous insistons sur le fait que les autres colonnes ne sont mises à jour uniquement si une transformation est acceptée et validée.

h	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
consommation	0	1	0	1	1	1	2	3	2	2	0	1	1	1	2	1	3	3	2	2	1	2	3	0
livraison	-	-	-	-	2	-	-	-	-	-	5	-	-	-	8	-	-	-	12	-	-	-	-	-
niveauClient	13	12	12	11	12	11	9	6	4	2	7	6	5	4	10	9	6	3	13	11	10	8	5	5
eviterAssech	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-3	-3	-3	-3	-3	-3
eviterDebord	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	4	6	9	9

TABLE 4.1 – Les structures de données d'affectation des volumes.

Les cinq structures de données servant au calcul de la quantité maximale livrable sont mises à jour en $O(\bar{n})$ en temps. L'ordonnancement de l'ensemble des opérations impactées est en $O(\bar{n} \log \bar{n})$ en temps dans le pire des cas (en utilisant le "heapsort algorithm" présenté dans [33]). Toute cette affectation de volume est donc faite en temps linéaire, puisque le calcul des quantités maximales livrables / chargeables nécessite un temps constant grâce à l'utilisation de structures de données adéquates. L'algorithme global a donc une complexité de $O(\bar{n} \log \bar{n})$ en temps.

En théorie, l'algorithme glouton est loin d'être optimal. La Figure 4.8 donne les plus petites configurations pour laquelle l'algorithme glouton n'arrive pas à trouver l'affectation optimale. Deux conditions sont suffisantes pour rendre l'algorithme glouton optimal : la première concerne le cas où chaque client est servi au plus une fois pendant la durée du planning. Cette condition est intéressante parce qu'elle est souvent rencontrée en pratique. La seconde condition correspond au cas où chaque tournée visite seulement un client. Par exemple, cette condition est satisfaite dans le cas où les clients ont des capacités de stockage infinies.

Des expérimentations ont été menées pour évaluer la performance de cette fonction critique. En pratique, son temps d'exécution est constant en fonction du nombre d'opérations : il est 100 fois plus rapide que l'application totale de l'algorithme glouton (c'est-à-dire, considérant que toutes les opérations sont impactées, impliquant ainsi que $\bar{n} = n$) et 2000 fois plus rapide que l'algorithme exact. Les tests ont été ici réalisés avec l'algorithme simplexe de la librairie de programmation linéaire GLPK 4.24. Il est important de voir que le volume total livré par la fonction est très proche de l'affectation optimale, en particulier lorsqu'aucun assèchement n'apparaît (la différence moyenne entre cet algorithme glouton et l'algorithme optimale est inférieure à 2 %).

Finalement, une fois ces volumes affectés, le calcul du gain d'une transformation peut se faire efficacement : l'évaluation de la variation du coût des tournées est calculée pendant la phase d'ordonnancement et celle de la variation de volume livré est obtenue pendant la

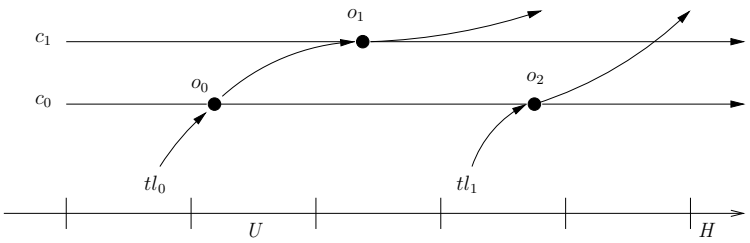


FIGURE 4.8 – Mauvaise configuration pour l'affectation gloutonne des volumes.

phase d'affectation des volumes, sans ajout de complexité. On déduit aussi les variations du nombre d'assèchements ou de commandes non satisfaites pendant l'affectation. Ce dernier calcul nécessite par contre une boucle en $O(\log H)$ en temps, avec H le nombre de pas de temps jusqu'à l'horizon final (on recherche le premier pas de temps sous le seuil de sécurité).

4.4.5 Détails d'implémentation

Tous les ensembles (ordonnés ou non, fixes ou dynamiques) utilisés au cours de cette recherche locale sont implémentés sous la forme de tableaux, afin d'améliorer la gestion de la mémoire cache. Les allocations de mémoires sont évitées autant que possible : toutes les structures de données sont allouées au démarrage de la recherche locale. Un tableau de capacité n représentant une liste dynamique d'objets est étendu si nécessaire par réallocation d'un bloc mémoire plus grand de taille $n + k$ (avec $k \approx 10$).

Comme le taux de réussite des transformations est faible en moyenne (quelques pourcents), la fonction "de retour arrière" doit être très efficace. Ainsi, les variables de décision du problème (par exemple, les dates de début et de fin d'une opération) sont dupliquées. Seules les données temporaires sont modifiées par la transformation. La fonction "de retour arrière" consiste uniquement à réécrire les données temporaires dans les données courantes (ce qui correspond à un retour à la solution courante). Mais cela est compliqué par le fait que durant la transformation, plusieurs objets (en particulier les opérations et les tournées) sont susceptibles d'être déplacés des tableaux dans lesquels ils étaient stockés.

Nous utilisons des piles d'objets afin d'éviter la création de nouveaux objets "tournée" ou "opération" au cours de la recherche locale. Pour optimiser les suppressions/insertions d'objets dans les listes, on utilise ici des listes doublement chaînées mises à jour elles-aussi en temps $O(1)$. Enfin, pour garantir l'efficacité des algorithmes, nous faisons appel à des structures de données stockant pour chaque opération, son successeur/prédécesseur dans la tournée et pour cette ressource (ces listes sont mises à jour uniquement lors d'une acceptation). La Figure 4.9 illustre les échanges d'opérations entre tournées. Les opérations $o_{i,3}$, $o_{i,4}$ de la tournée s_i sont échangées avec les opérations $o_{j,3}$, $o_{j,4}$ de la tournée s_j : les liens courants (avant transformation) sont pleins, les liens temporaires (après transformation) sont

en pointillés.).

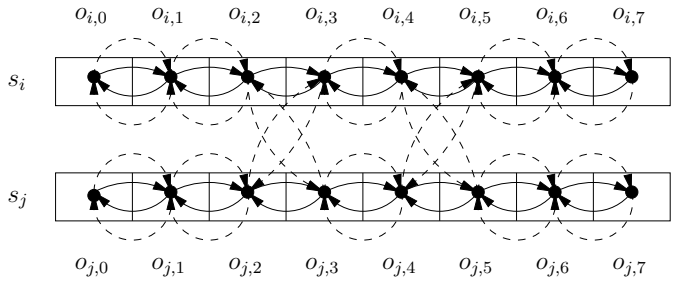


FIGURE 4.9 – Représentation des tournées et des opérations.

Les principales structures de données sont bâties de telle manière qu'elles supportent les routines classiques (chercher, insérer, supprimer, effacer) en $O(1)$ en temps, même si elles sont implémentées comme des tableaux. Par exemple, la structure de données classique utilisée pour implémenter une liste non-ordonnée d'objets, comme la liste des clients qui subissent un assèchement dans la solution courante ou encore la liste des clients ayant au moins une commande non satisfaite, *etc.* Ces listes sont implémentées comme des tableaux de longueur L, avec $L.size$ le nombre courant d'éléments dans L et $L.capacite$ le nombre maximum d'éléments pouvant être stockés dans la liste L sans excéder la mémoire allouée. Tout élément e stocké dans L a un pointeur $e.indexL$ sur sa position dans le tableau L. Si le nombre maximal d'éléments stockés dans L est connu en avance et pas trop grand (plusieurs milliers), alors L peut être allouée avec une capacité égale à ce nombre (pour éviter les allocations fréquentes de mémoire), sinon l'extension de L est faite en augmentant sa capacité de $L \times K$ éléments quand cela est nécessaire. Les fonctions TROUVER, INSÉRER, SUPPRIMER, et NETTOYER sont implémentées ainsi et tournent donc en $O(1)$ en temps.

Algorithme 14: TROUVER

input : Le tableau L, l'élément e à trouver

begin
  $i = e.indexL$
  if $i \geq 0$ *and* $i < L.size$ *and* $L[i] = e$ **then**
    if *une pause a été posée au cours du dernier arc* **then** retourne VRAI
    else retourne FAUX

Pour clore cette section, nous nous focalisons sur une structure de données qui est particulièrement critique pour l'efficacité. Les opérations ou les tournées correspondent à des intervalles de temps, pour lesquels nous avons les besoins suivants : soit une date sur l'horizon, trouver l'opération précédente, courante ou suivante effectuée sur un site si il y en a une. Le même problème se pose pour trouver les tournées réalisées par une ressource. Dans ce cas,

Algorithme 15: INSÉRER

input : Le tableau L, l'élément e à insérer

```
begin
    i = e.indexL
    if TROUVER(L,e) then Stop
    if L.size = L.capacite then
        L.capacite = L.capacite + L.K
        Allouer à L sa nouvelle capacité L.capacite
    L[L.size] = e, e.indexL = L.size, L.size = L.size + 1
```

Algorithme 16: SUPPRIMER

input : le tableau L, l'élément e à supprimer

```
begin
    if not FIND(L,e) then Stop
    i = e.indexL, e' = L[L.size − 1]
    L[i] = e', e'.indexL = i, L.size = L.size − 1
```

la structure de données employée est la suivante. Soient n opérations stockées dans une liste ordonnée L. L'horizon est divisé en m intervalles $U_0, \ldots, U_{m-1}$ de longueur u donnée (avec T divisible par u). Soit un tableau I, défini tel que I_i fait référence à la première opération dont la date de début est plus grande que le point le plus à gauche de U_i. L'opération suivant la date d est trouvée en cherchant les opérations entre celle pointée par I_i avec $i = \lfloor d/u \rfloor$ et celle pointée par I_{i+1}. En opérant ainsi, la recherche est réalisée en $O(k)$ en temps dans le pire cas, avec k le nombre d'opérations contenues dans l'intervalle U_i. Si u correspond à l'horizon entier ($m = 1$), alors $k = n$; Dans le cas contraire, si u correspond à la plus petite granularité pour exprimer le temps (ici en minutes, ce qui donne $m = 21600$), alors $k = 1$. Si on fait l'hypothèse que les dates de début des opérations effectuées sur un site sont uniformément distribuées sur l'horizon, le nombre k vaut n/m. Dans ce cas, la recherche se fait en $O(n/m)$ en temps (à l'évaluation de la transformation), mais le tableau I nécessite $O(m)$ en espace pour être stocké et $O(m)$ en temps pour être mis à jour (lors de la validation d'une transformation). Cela implique deux compromis : temps pour l'évaluation contre temps pour la validation et temps pour l'évaluation contre espace.

D'un point de vue théorique, la meilleure valeur m^* pour résoudre ce compromis sur le temps d'exécution correspond à trouver le minimum de la fonction $T(m) = \frac{EN}{m} + Cm$, avec N le nombre moyen d'opérations par client et E (resp. C) un coefficient relatif à la proportion des appels de la fonction d'évaluation (resp. de la fonction de validation) par client. Un simple calcul par différenciation donne $m^* = \sqrt{\frac{EN}{C}}$. Le tableau montre les valeurs de m^* pour différentes configurations réalistes des paramètres N, E, C. En pratique, on

Algorithme 17: NETTOYER

begin
  Entrées : le tableau L
  $L.size = 0$

choisit $m^* = 15$, ce qui correspond à un intervalle U_i d'une journée : une telle valeur de m confère un bon compromis entre temps de calcul (même dans les pires cas) et génère une petite empreinte mémoire.

N	2	2	2	10	10	10	30	30	30
E	90	95	99	90	95	99	90	95	99
C	10	5	1	10	5	1	10	5	1
m^*	4,2	6,2	14,1	9,5	13,8	31,5	16,4	23,9	54,5

TABLE 4.2 – Valeurs théoriques de m^*.

4.5 Résultats numériques

L'algorithme complet de recherche locale a été implémenté en C# 2.0 (pour être exécuté dans l'environnement Microsoft .NET 2.0). Le programme compte 30000 lignes de code, dont 6000 dédiées à la vérification de la validité des structures de données incrémentales (20 %), à chaque itération de l'algorithme (en mode débugage). Toutes les statistiques et les résultats présentés ici ont été obtenus (sans parallélisation) sur un ordinateur équipé du système d'exploitation Windows Vista et d'un processeur Intel Xeon X5365 64 bits (CPU 3 GHz, L1 cache 64 Ko, L2 cache 4 Mo, RAM 8 Go).

Comme la recherche locale est stochastique, cinq exécutions ont été réalisées avec une graine différente pour chaque lancement. Toutes les statistiques présentées sont donc la moyenne de ces 5 exécutions. Les résultats nécessitant une explication particulière sont marqués avec une astérisque (*) dans les tableaux de données numériques, et les explications sont notées en dessous.

L'analyse des solutions du problème a été facilitée par la création d'une interface de visualisation, développée spécialement en Winform pour le projet. Cet outil permet de visualiser les tournées depuis plusieurs points de vue, tout comme les inventaires. Les figures 6.1, 6.2, 6.3, 6.4, 6.5 et 6.6 donnent quelques aperçus de cet outil de visualisation.

4.5.1 Benchmarks sur un horizon court terme

L'algorithme de recherche locale a été testé sur des benchmarks à court terme (15 jours) avec différentes caractéristiques : réalistes (c'est-à-dire respectant les conditions opérationnelles), pathologiques (par exemple, usines qui tombent en panne sans produire pendant plusieurs jours), de grande envergure (par exemple, 1500 clients et 100 usines). Tous ces tests nous ont permis de vérifier et d'améliorer la performance et la robustesse de cet algorithme de recherche locale. Une partie des résultats est présentée pour 61 benchmarks à court terme décomposés en 3 types A, B, C. Le Tableaux 6.1, 6.2 et 6.3 de l'Annexe 6.3 donne les caractéristiques de chaque instance : le nombre de clients, le nombre d'usines, le nombre de bases, le nombre de conducteurs, le nombre de tracteurs, le nombre de remorques, le nombre de clients qui fonctionnent à la commande, le nombre de commandes sur la période. Quand le nombre de clients à la commande est nul, cela signifie que toutes les commandes sont passées par des clients qui ont les deux modes de réapprovisionnement. Les benchmarks A et B ne gèrent pas les commandes (les clients sont en réapprovisionnement à la prévision), tandis que le benchmark C comporte de tels clients. Le Tableau 4.3 donne les résultats pour les 17 instances de type A.

	clients	usines	dépôts	conducteurs	tracteurs	remorques
minimum	46	1	1	10	5	10
maximum	500	5	2	50	50	50
moyenne	138	1,9	1,2	28	19	25

TABLE 4.3 – Benchmark à court terme (type A) : caractéristiques moyennes des 17 instances.

Les résultats obtenus par l'algorithme glouton sur les benchmarks A, B et C sont présentés dans les Tableaux 6.4, 6.5, et 6.6 et les Tableaux 6.7, 6.8 et 6.9 de l'Annexe 6.3. Deux types de résultats sont présentés pour l'heuristique par recherche locale : ceux obtenus par l'optimisation directe du ratio logistique RL (notés LS-RL et présentés dans les Tableaux 6.10, 6.11 et 6.12 et Tableau 6.13 en Annexe 6.3) et ceux obtenus par l'optimisation du ratio logistique modifié RL' (notés LS-RL' et présentés dans les Tableaux 6.14 et 6.15) en Annexe 6.3. Afin de comparer les solutions avec des commandes non satisfaites ou des assèchements, on introduit le coût global $GC = MO + SO + RL$.

Certaines instances sont particulières : A11, B01, B28, B29, B30, et plus particulièrement B31 sont classifiées comme des instances de grande taille (plus de 500 clients) ; B28 et B29 sont des instances où la production a été stoppée à l'usine (aucun produit n'est chargeable) ; B30 contient des clients en assèchement dès le début de l'horizon ; les clients de l'instance C03 et C04 ont des horaires d'ouverture très serrés (par exemple, un client peut très bien être accessible uniquement 6 heures au cours des 15 jours). Presque tous les clients de l'instance C12 sont des clients à la commande (ce qui implique plus de 600 commandes à satisfaire). Le temps d'exécution de l'algorithme glouton est de l'ordre de quelques secondes pour des

instances très grandes (12 secondes pour l'instance B31). Les statistiques de performance de cette recherche locale sont données dans le tableau 6.16 en Annexe 6.3. La colonne "tentat." correspond au nombre de transformations évaluées au cours de l'exécution. Les colonnes "accept." (resp. "amélio.") correspondent au nombre de transformations acceptées (resp. strictement améliorante) ; de plus, le ratio correspondant pour 100 (resp. pour 10000) transformations évaluées est spécifié. À noter que les valeurs moyennes données en bas de tableau sont calculées en omettant les résultats exceptionnels obtenus sur l'instance C04 (des horaires d'ouverture très serrés entraînent des rejets précoces et donc plus de tentatives). L'algorithme de recherche locale évalue plus de 10000 transformations à la seconde, même sur des instances très grandes. En moyenne, notre algorithme visite plus de 10 millions de solutions au cours de 5 minutes de temps de calcul autorisé (temps maximal désiré dans les conditions opérationnelles). Quand on planifie sur un horizon de 15 jours, la mémoire allouée par le programme n'excède pas les 30 Mo pour des instances de taille moyenne (plusieurs centaines de sites, une dizaine de ressources), et 300 Mo sur des instances de taille plus importante (plusieurs centaines de sites, plusieurs centaines de ressources). Le taux d'acceptation moyen, qui correspond au nombre de transformations acceptées (c'est-à-dire le nombre de transformations qui n'améliorent pas strictement la solution courante) sur le nombre de transformations évaluées, varie entre 1 et 10 %, avec une valeur moyenne de 5 % sur toutes les instances de A, B et C. Ce taux est quasi-constant au cours de la recherche (soit les 5 minutes de temps d'exécution), autorisant ainsi une large diversification de la recherche (ceci sans métaheuristique). De plus, le nombre de transformations strictement améliorantes est de plusieurs centaines, ce qui correspond à un taux d'amélioration de presque 2 sur 10000 tentatives. Le choix de l'objectif (LS-RL ou LS-RL') n'affecte en aucun cas les performances de l'heuristique de recherche locale. Le Tableau 4.4 fournit ces résultats minima, maxima et moyens pour les 17 instances de type A.

	tentat.	accept.		amélio.	
minimum	3,706 M	117333	2,4 %	389	0,5 c%
maximum	15,184 M	536327	6,8 %	2218	5,2 c%
moyenne	7,830 M	303620	4,1 %	946	1,5 c%

TABLE 4.4 – Benchmark à court terme (type A) : statistiques pour LS-LR (gauche) et LS-LR' (droite). M = million, c% = un centième de pourcent.

La colonne "gain GC" (resp. "gain RL", "gain RL'") des Tableaux 6.10, 6.11, 6.12 et 6.14 en Annexe 6.3 présente le gain pour GC (resp. RL, RL') en comparaison à la solution trouvée par l'algorithme glouton. Le gain sur GC autorise à évaluer d'une manière plus globale le gain sur MO et SO. À noter que le gain sur RL peut être négatif quand la minimisation de MO et SO impose de détériorer sévèrement la qualité de la solution trouvée par l'algorithme glouton en terme de ratio logistique. Dans les 2 cas (LS-RL ou LS-RL'), l'heuristique de recherche locale améliore considérablement la qualité des solutions fournies par l'algorithme glouton. Sur les instances de la base 1 (pour laquelle une comparaison entre l'algorithme

glouton et la recherche locale est juste en terme de ratio logistique), la moyenne du gain obtenue par LS-RL (resp. LS-RL') est de 29.2 % (resp. 22.6 %), et de 24.1 % (resp. 20.0 %) en considérant le ratio logistique global (c'est-à-dire, la somme des coûts des tournées pour toutes les instances divisée par la somme des quantités livrées pour toutes les instances). Cette dernière mesure est intéressante mais pas complètement juste, puisque tous les coûts ne sont pas toujours exprimés dans la même unité. Les quantités sont quant à elles toujours exprimées en kilogramme ici. Dans le tableau 6.14 en Annexe 6.3, on note que les gains sur RL' sont corrélés à ceux sur RL. Le Tableau 4.5 fournit ces résultats minima, maxima et moyens pour les 17 instances de type A.

	LS-LR	LS-LR'
gain min sur le glouton	20,8 %	28,5 %
gain max sur le glouton	39,1 %	65,6 %
gain moyen sur le glouton	29,2 %	45,4 %

TABLE 4.5 – Benchmark à court terme (type A) : résultats aggrégés de la recherche locale.

Les tableaux 6.13 et 6.15 en Annexe 6.3 donnent plus de statistiques sur les solutions trouvées par recherche locale. La colonne "nb tour." (resp. "nb oper.") des tableaux présentent le nombre de tournées (resp. d'opérations) dans la solution. La colonne "avg oper." (resp. "avg livr.", "avg charg.", "avg pause") présente le nombre moyen d'opérations (resp. de livraisons, de chargements, de pauses) par tournée. Au final, la colonne "avg dist." (resp. "avg dur.") présente la distance moyenne parcourue (resp. la durée) par tournée. Beaucoup de tournées, et encore plus d'opérations, sont inclues aussi bien dans les solutions de LS-RL que dans celle de LS-RL'. En moyenne, le nombre de tournées (resp. d'opérations) augmente de presque 25 % (resp. 50 %). Ainsi, le nombre moyen d'opérations par tournée augmente de presque 4 à plus de 6. La distance moyenne et la durée moyenne des tournées diminuent légèrement dans le cas LS-RL, alors que celles-ci augmentent un peu dans le cas LS-RL'.

4.5.2 Benchmarks sur un horizon long terme

La recherche locale a été aussi testée sur des benchmarks à long terme, en particulier pour vérifier que l'optimisation en utilisant l'objectif de substitution permettait d'atteindre des solutions de meilleure qualité sur le long terme. Des résultats sont présentés pour 5 benchmarks associés à des cas réels, chacun d'eux ayant 105 jours. Le processus opérationnel de planification est simulé ainsi : la simulation débute au jour 0 en calculant un planning pour les 15 prochains jours, avec 5 minutes de temps de calcul. Puis, on conserve uniquement les tournées débutant le premier jour de ce planning à court terme (les niveaux des usines et des clients visités par ces tournées sont mis à jour, les ressources travaillant sur ces tournées deviennent non disponibles). Le processus est ensuite itéré les jours suivants.

Les caractéristiques des 5 benchmarks sont présentées dans le Tableau 4.6. Les statistiques complètes sont données pour chaque heuristique dans le Tableau 6.18, le Tableau 6.19 et le Tableau 6.20 en Annexe 6.3. La principale remarque est que le nombre moyen d'opérations par tournée augmente au cours de la recherche locale. En effet, le nombre de tournées dans les solutions trouvées par recherche locale est légèrement plus petit que dans l'algorithme glouton, tandis que le nombre d'opérations augmente. Cependant, les solutions obtenues par recherche locale sont caractérisées par une durée moyenne de voyage par tournée plus grande, grâce à l'utilisation plus fréquente des pauses. Les ratios logistiques moyens (marqués d'une astérisque dans le tableau) sont calculés comme la somme des coûts des tournées pour les 5 benchmarks divisés par la somme de toutes les quantités livrées.

Les gains obtenus par LS-RL' sont reportés dans la partie droite du Tableau 4.6. La colonne "pire 1 mn" présente le pire gain RL en %, obtenu sur 5 lancements de 1 minutes pour chaque itération de planning. La colonne "moy 1 mn" (resp. "moy 5 mn", "moy 1 h") présente la moyenne du gain en % pour RL obtenu par recherche locale en limitant à 1 minute (resp. 5 minutes, 1 heure) de calcul pour chaque itération de planning. À noter que les solutions trouvées par l'algorithme de recherche locale comprennent des assèchements. Le gain RL obtenu par LS-RL' avec 1 minute de temps de calcul par itération de planning est d'environ 20 % en moyenne. Non seulement ces statistiques montrent que l'heuristique fournit des solutions de bonne qualité, mais elles prouvent qu'elle est robuste et rapide (avec une convergence inversement exponentielle).

données	clients	plants	bases	conducteurs	tracteurs	remorques	commandes	pire 1 mn	moy 1 mn	moy 5 mn	moy 1 h
L1	75	6	1	35	21	5	56	23,8 %	24,6 %	26,3 %	26,5 %
L2	75	6	1	35	21	5	55	22,3 %	23,5 %	24,9 %	25,2 %
L3	175	8	1	35	21	12	189	5,2 %	5,8 %	8,3 %	11,2 %
L4	165	4	1	24	11	7	167	9,9 %	11,2 %	14,0 %	18,9 %
L5	198	8	7	12	12	12	40	32,5 %	34,2 %	35,7 %	35,9 %
moyenne	138	6	2	28	17	8	101	18,7 %	19,9 %	21,8 %	23,5 %

TABLE 4.6 – Benchmarks sur le long terme : caractéristiques et gains en terme de ratio logistique RL, avec différentes limites de temps.

Les solutions fournies par les experts en planification sont présentées dans le tableau 4.8 (en Annexe 6.3). La comparaison entre les résultats des experts et les trois heuristiques n'est pas totalement équitable : en effet, trouver une solution sans assèchement (en fonction des niveaux courants de sécurité) est difficile et fastidieux, et les experts ont été ici autorisés à modifier les benchmarks initiaux à long terme afin de fournir une solution sans commande non satisfaite ni assèchement. C'est pourquoi on note un résultat négatif de l'algorithme glouton sur les instances L1, L2, L4.

4.5.3 Influence de l'objectif de substitution

Les caractéristiques clés pour comparer LS-RL et LS-RL' sont données dans le tableau 6.17 (en Annexe 6.3, pour les benchmarks à court terme) et sur la partie droite du tableau 4.7 (pour les benchmarks à long terme). "moy DQ" correspond à la moyenne de la quantité totale livrée et "QLiv. moy" à la quantité livrée moyenne (par opération). Pour les benchmarks à court terme, la quantité totale livrée aussi bien dans les solutions de LS-RL que dans celles de LS-RL' augmente de plus de 50 % en moyenne. Mais il faut noter aussi que la quantité moyenne par livraison augmente de presque 5 % dans les solutions LS-RL' comparées à celles LS-RL.

Pour les benchmarks à long terme, on observe que LS-RL' permet d'augmenter la quantité moyenne par livraison, ce qui a pour conséquence de générer des solutions meilleures sur le long terme. Sur l'instance L1 (resp. L2), l'augmentation est de 21 % (resp. 13 %), permettant ainsi de gagner +8 % (resp. +5 %) par rapport à RL. De plus, LS-RL' est capable de produire des solutions sans assèchement sur l'instance L4, contrairement à LS-RL. Les valeurs marquées d'une astérisque dans les tableaux 4.7 et 4.8 sont calculées globalement, pour les 5 benchmarks (comme pour les ratios logistiques des tableaux 6.19 et 6.20 présentés en Annexe 6.3).

données	$RL_{\min}$
L1	0,061025
L2	0,059855
L3	0,019176
L4	0,012256
L5	0,005576

données	gain LS-RL	gain LS-RL'	QLiv. moy glouton	QLiv. moy LS-RL	QLiv. moy LS-RL'
L1	17,7 %	26,3 %	12460	9344	11391
L2	19,1 %	24,9 %	12205	9759	11035
L3	7,0 %	8,3 %	14997	14706	14833
L4	15,0 %	14,0 %	13018	11885	11876
L5	31,8 %	35,7 %	15755	11044	11078
moyenne	*14,1 %	*16,3 %	*14146	*12537	*12864

TABLE 4.7 – Benchmarks sur le long terme : bornes inférieures (gauche) et gains obtenus par recherche locale comparés au glouton (droite).

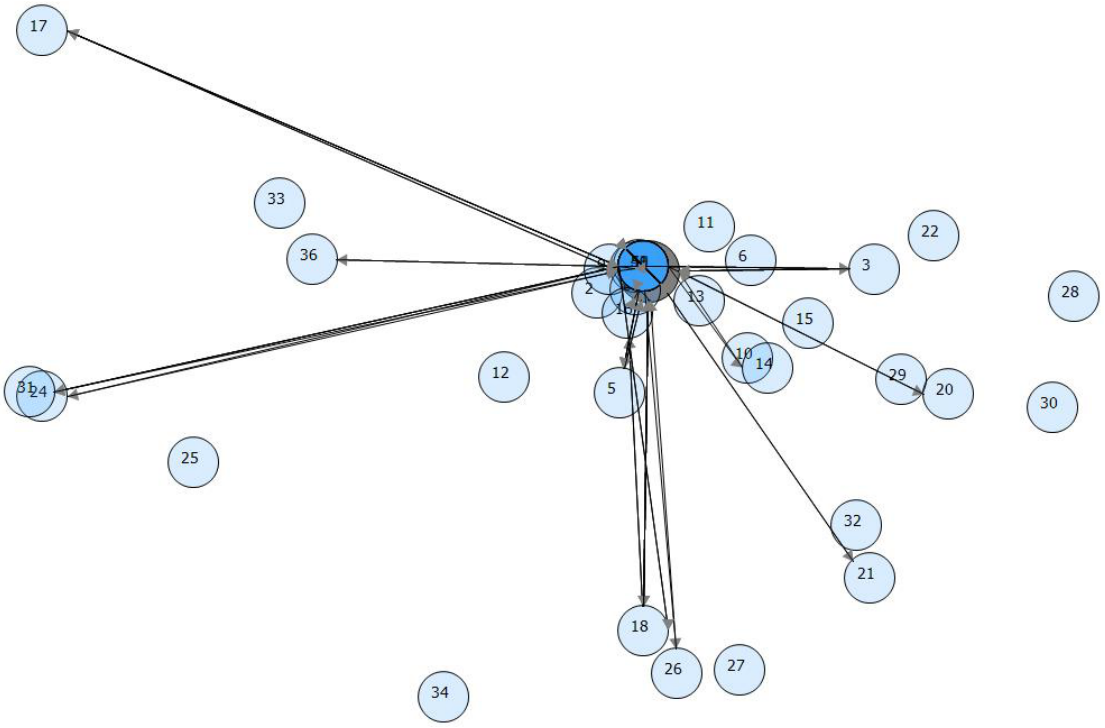
données	SO	SC	DQ	RL	gain glouton	gain LS-RL	gain LS-RL'
L1	0	378778	3725847	0,101662	-6,1 %	12,7 %	21,9 %
L2	0	328364	3567370	0,092047	-15,6 %	6,5 %	13,1 %
L3	0	1257354	32667576	0,038489	11,2 %	17,4 %	18,6 %
L4	0	788893	18683473	0,042224	-3,9 %	11,7 %	10,7 %
L5	0	290921	10398050	0,027978	41,2 %	59,9 %	62,2 %
moyenne	0	608862	13808463	*0,044093	*2,9 %	*16,6 %	*18,7 %

TABLE 4.8 – Benchmarks sur le long terme : gains obtenus par recherche locale comparés à ceux des experts.

Les colonnes de gauche des Tableaux 6.4, 6.5 et 6.6 et Tableau 4.7 présentent les bornes inférieures pour SO et RL. Le calcul de la borne inférieure $SO_{\min}$ sur le nombre d'assèchements est fondé sur deux observations basiques. Des assèchements apparaissent sur un client si : la plus petite date d'arrivée chez un client pendant ses horaires d'ouverture est

plus grande que la date du premier assèchement ; la somme des consommations du client au cours des horaires de fermetures dépasse la capacité de sa plus grande remorque. Le calcul de $RL_{\min}$ est fait comme dans la description de la Section 4.3. Dans les Tableaux 6.4, 6.5, 6.6, 6.10, 6.11, 6.12 et 6.14, une astérisque est apposée sur la colonne SO si la valeur est égale à la borne inférieure $SO_{\min}$.

Le ratio logistique obtenu après optimisation par recherche locale reste loin de la borne inférieure calculée $RL_{\min}$ (la différence est de plus de 10 % pour les instances B06, B20, B25). Le schéma 4.10 montre qu'une solution quasi-optimale est obtenue sur l'instance B06, avec principalement des tournées "aller-retour" avec livraisons complètes. Cependant, le lecteur notera que cette borne a été obtenue en relaxant fortement les contraintes du problème et qu'il conviendrait de continuer les recherches sur cette thématique.



Une majorité de tournées "aller-retour" avec livraison complète.

FIGURE 4.10 – Solution quasi-optimale de l'IRP

4.6 Synthèse

Dans ce chapitre étaient présentées une modélisation du problème d'optimisation de tournées de véhicules avec gestion des stocks et une méthode de résolution efficace pour le résoudre. Deux contributions ont été présentées : tout d'abord, nous avons introduit une

fonction objectif de substitution fondée sur des bornes inférieures permettant d'assurer une optimisation sur le long terme tout en construisant des sous-plannings sur le court terme. Puis, nous avons détaillé une heuristique de recherche locale afin de résoudre de manière efficace et robuste le problème sur le court terme (15 jours en détails complets), même sur des instances de très grande taille pouvant atteindre plusieurs milliers de sites et une centaine de ressources. Les résultats numériques obtenus sur différentes instances démontrent la performance et la robustesse de cet algorithme de recherche locale. L'étude démontre que l'algorithme permettait de réaliser des économies dépassant les 20 % en moyenne, comparées aux solutions construites par des planificateurs experts ainsi que par rapport aux solutions fournies par un algorithme glouton fondé sur l'urgence.

L'algorithme de recherche locale réalise presque 50 000 mouvements par seconde, même pour des instances plus de grande taille. L'algorithme visite plus de 10 millions de solutions pendant la période de résolution de 5 minutes. Le taux d'acceptation des mouvements de la recherche locale varie entre 1 % et 10 % suivant les instances et les phases d'optimisation. Ce taux est quasiment constant tout au long de la recherche et permet d'assurer une riche diversification de la solution, sans usage de métaheuristique.

Une solution d'aide à la décision intégrant cette heuristique de recherche locale est désormais exploitée par le partenaire industriel de ce projet. Des économies conséquentes sur le long terme ont été confirmées en opérationnel.

D'autres travaux de recherche sont toujours en cours sur cette thématique dans l'équipe du e-lab :

- Enrichir le modèle et le nombre de contraintes traitées dans ce problème d'optimisation de tournées de véhicules avec gestion des stocks (notamment, ajouter des coûts supplémentaires par type de produit et respecter les habitudes existantes des conducteurs) ;
- Améliorer l'algorithme de recherche locale en ajoutant de nouvelles transformations (notamment, avec des voisinages larges pour améliorer la rapidité de convergence) ;
- Améliorer les bornes inférieures, en intégrant les tournées visitant plusieurs sites et les contraintes de ressources.

Une dernière perspective, plus ambitieuse et tout aussi prometteuse, est de procéder par étape afin d'intégrer petit à petit l'intégralité de la chaîne logistique, en abordant aussi bien au sein du même modèle les problématiques de production que celles de distribution, voire les problématiques d'approvisionnement. En effet, notre approche par recherche locale est parfaitement adaptée pour aborder directement de tels problèmes mixtes.

Chapitre 5

Conclusion

5.1 Bilan

Les problèmes d'optimisation en variables mixtes sont souvent résolus par décomposition en sous-problèmes quand ils sont de grande taille. Qu'elles reposent sur les spécificités intrinsèques au problème ou qu'elles fassent appel à la programmation mathématique, les approches par décomposition ont cependant des inconvénients en pratique : difficulté de garantir la qualité voire l'admissibilité des solutions, complexité technique de mise en œuvre, complexité logicielle en cas d'utilisation de solveurs de programmation mathématique. Dans cette thèse, nous avons proposé une *approche directe*, par recherche locale, pour l'optimisation en variables mixtes. Notre méthodologie se focalise sur deux points. Tout d'abord, nous définissons une *large variété de mouvements randomisés* travaillant simultanément sur les dimensions combinatoire et continue du problème, afin d'assurer une exploration diversifiée de l'espace de recherche. Ensuite, nous accélérons l'évaluation de ces mouvements en nous appuyant sur une *algorithmique incrémentale et approchée*.

Nous avons traité en premier lieu un problème purement combinatoire concernant l'optimisation des stocks de banches sur des chantiers de construction résidentielle. Ce problème est résolu en utilisant trois méthodes que nous comparons sur un jeu de trente-six instances. L'approche par recherche locale permet de conserver des temps d'exécution très courts tout en produisant des solutions à 5 % de l'optimum en moyenne. Nous avons ensuite appliqué notre méthodologie pour résoudre un problème d'ordonnancement de mouvements de terre sur des chantiers linéaires pour DTP Terrassement. L'objectif de ce problème est de planifier sur plusieurs années des centaines de mouvements de terre à l'aide d'une cinquantaine de ressources avec fenêtres de temps journalières. Nous avons utilisé pour cela une heuristique de recherche locale dont les mouvements pouvaient modifier les quantités de terre déplacées par chaque tâche tout en changeant l'ordonnancement de ces tâches et leur affectation aux ressources. Le logiciel en exploitation a permis la planification de grands chantiers en quelques minutes. Nous nous sommes aussi focalisés, dans ce chapitre consacré aux mouve-

ments de terre, à l'étude du cas à une ressource, correspondant à un problème de voyageur de commerce linéaire avec précédences, prouvé ici comme étant NP-difficile. Nous avons obtenu des bornes inférieures et proposé un algorithme par programmation dynamique.

Enfin, le dernier chapitre de cette thèse concernait un problème de tournées de véhicules avec gestion de stock pour un client industriel du Groupe Bouygues. Ce problème consiste en l'optimisation des coûts de distribution d'un produit par camion, sur des zones géographiques comptant plusieurs centaines de clients, dont la gestion des stocks est confiée au fournisseur. Une heuristique de recherche locale est proposée pour résoudre le problème à court terme (15 jours) en quelques minutes. Dans cette approche, les quantités livrées au cours des tournées peuvent être modifiées par les mouvements, alors même que les tournées sont modifiées géographiquement ou temporellement. Nous nous appuyons également sur un algorithme approché de réaffectation des volumes, un millier de fois plus rapide en pratique qu'un algorithme exact de flot maximum. L'exploitation de ce logiciel, quotidienne et à l'échelle mondiale, a permis une réduction des coûts logistiques de l'ordre de 15 %.

Nous avons démontré dans ces deux exemples l'efficacité et la robustesse de notre approche. Pour chacun des problèmes, nous avons défini une famille de mouvements adaptés aux spécificités du problème. Nous nous sommes attachés à mettre en place une évaluation incrémentale de ces transformations à l'aide de fonctions travaillant simultanément sur les deux dimensions (combinatoire et continue) des problèmes. Ceci nous a permis d'évaluer plusieurs dizaines de milliers de transformations à la seconde dans les deux recherches locales, assurant ainsi une très large diversification et une descente rapide vers des solutions de qualité.

5.2 Perspectives

Nous avons identifié trois autres problèmes ou familles de problèmes d'optimisation en variables mixtes. Ils se révèlent être des problèmes mixtes potentiellement intéressants pour appliquer la méthodologie de résolution par recherche locale décrite dans cette thèse.

Planification des corps d'état secondaire. Le premier problème mixte identifié est un problème rencontré chez Bouygues Habitat Résidentiel, une filiale de Bouygues Construction spécialisée dans la construction de bâtiments résidentiels. Il s'agit d'optimiser le planning des activités de construction après la phase du gros œuvre, c'est-à-dire les travaux de peinture, électricité, plomberie, chapes, circuits d'eau, *etc*, que l'on appelle les corps d'état secondaire. Ce problème a déjà été traité en 2006 en utilisant des heuristiques fondées sur un environnement de programmation par contraintes [14]. Il consiste à déterminer les jours de début et de fin de chacune des tâches en minimisant (entre autres) les dépassements de capacité de chaque ressource. Les contraintes combinatoires de cette planification sont les précédences entre les tâches et les contraintes d'exclusion mutuelles entre certaines tâches au sein d'un

même appartement. Les variables continues décrivent pour chaque jour la répartition de la capacité de travail de chaque ressource (exprimée par exemple m^2 de carrelage par jour) entre les différentes tâches en cours. Un exemple de mouvement que nous pourrions implémenter dans une recherche locale consisterait à évaluer le déplacement d'une partie du travail effectué par une ressource un jour donné, en affectant ce volume de travail un autre jour sur une autre ressource. Plusieurs milliers de tâches élémentaires sont effectuées par une cinquantaine de ressources sur un planning d'environ 6 mois, ce qui justifie l'usage de structures incrémentales dans l'algorithmie sous-jacente, afin d'évaluer un nombre important de mouvements au cours de la résolution.

Optimisation de la gestion de l'éclairage public. La planification prévisionnelle dans le cadre d'un appel d'offre fait également intervenir des variables mixtes. Elle consiste à décider et planifier les opérations de renouvellement et de maintenance des équipements d'éclairage d'une ville sur 20 ans, de façon à minimiser le coût global pour la collectivité (travaux, facture énergétique et maintenance). Afin d'améliorer la rapidité et la qualité de la réponse à l'appel d'offre, un outil opérationnel de simulation a été développé en 2007, pour ETDE, filiale de Bouygues Construction, avec une modélisation purement combinatoire du problème [85]. Des variables de décision continue apparaissent dans ce problème dès que l'on considère les montants éventuellement empruntés par le concessionnaire chaque année. Dès lors, une approche par recherche locale pourrait modifier simultanément la planification et les budgets annuels, c'est-à-dire les variables discrètes et les variables continues.

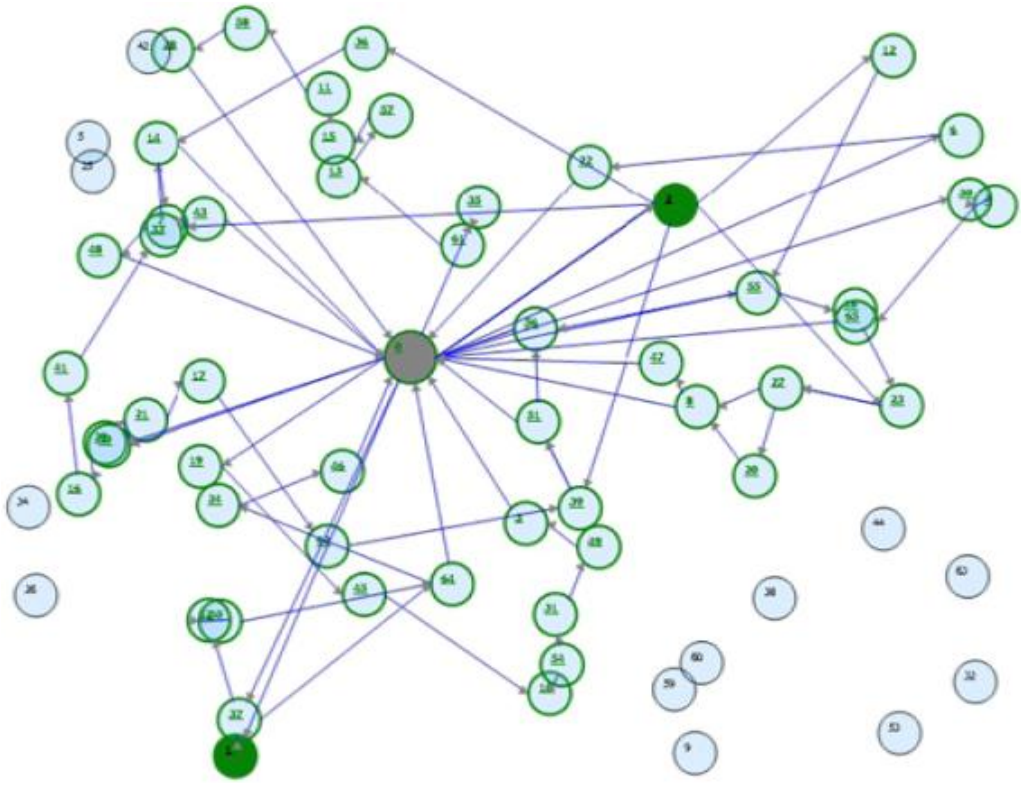
Energie. Le problème de planification des arrêts de centrales nucléaires, évoqué en introduction, est un autre problème mixte sur lequel l'approche présentée dans cette thèse a montré son efficacité [49]. Plus généralement les problèmes liés à la gestion de l'énergie sont très souvent des problèmes mixtes puisqu'ils combinent des décisions discrètes (comme l'arrêt ou le démarrage de la climatisation d'un bâtiment) et des décisions continues correspondant généralement à des puissances consommées ou produites.

Nous pensons que notre approche par recherche locale directe et pure, à base de mouvements travaillant simultanément sur les variables discrètes et continues du problème, peut se révéler être une technique efficace afin d'obtenir pour ces problèmes de grande taille des résultats de qualité, tout en gardant des temps de calcul très courts.

Chapitre 6

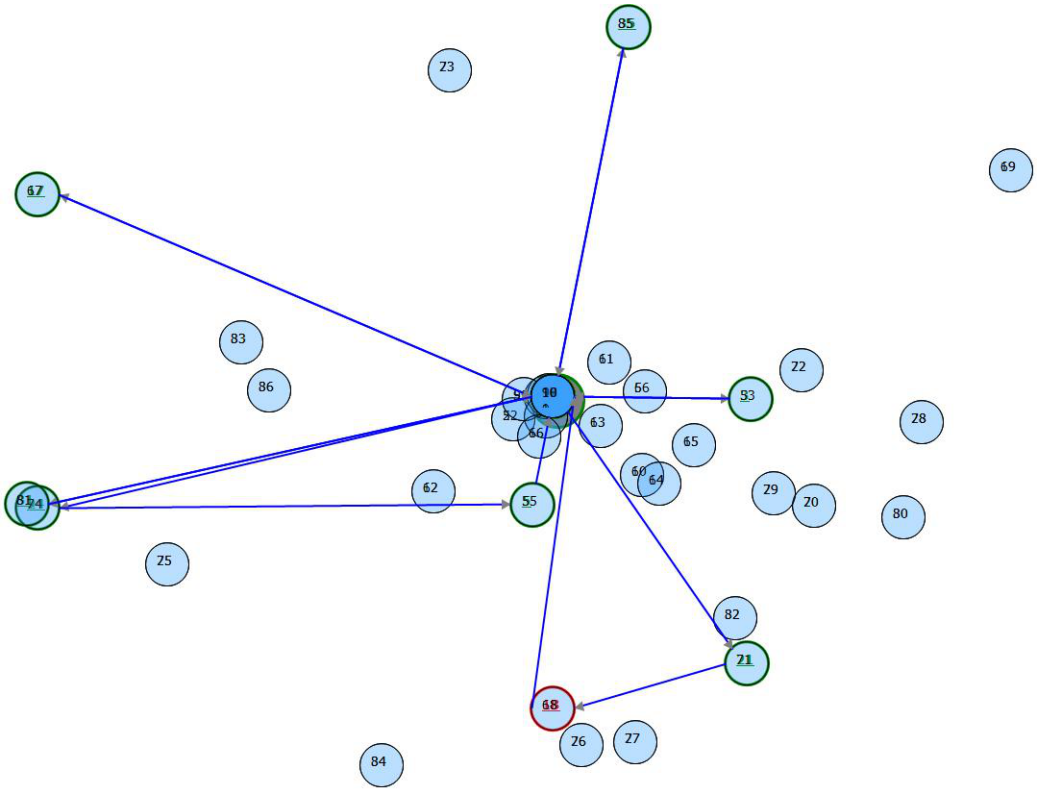
Annexes

6.1 Annexe 1 : Interface du logiciel d'IRP



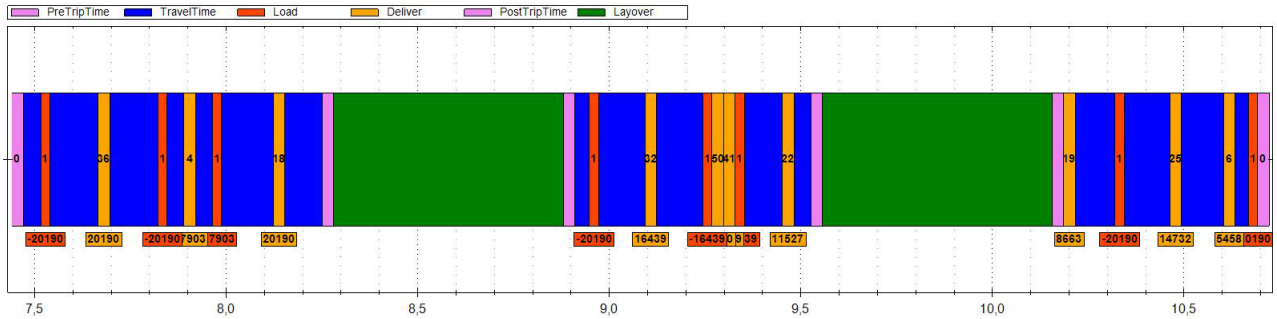
Une vue géographique d'une solution de l'IRP.

FIGURE 6.1 – Une vue géographique d'une solution de l'IRP.



Une vue géographique d'une "longue" tournée (18 opérations : 10 livraisons, 8 chargements, 2 pauses) construit par recherche locale.

FIGURE 6.2 – Une vue géographique d'une solution de l'IRP et d'une longue tournée.



À noter que la deuxième pause de la tournée était anticipée pour attendre les heures d'ouverture du client.

FIGURE 6.3 – La vue chronologique de la "longue" tournée.

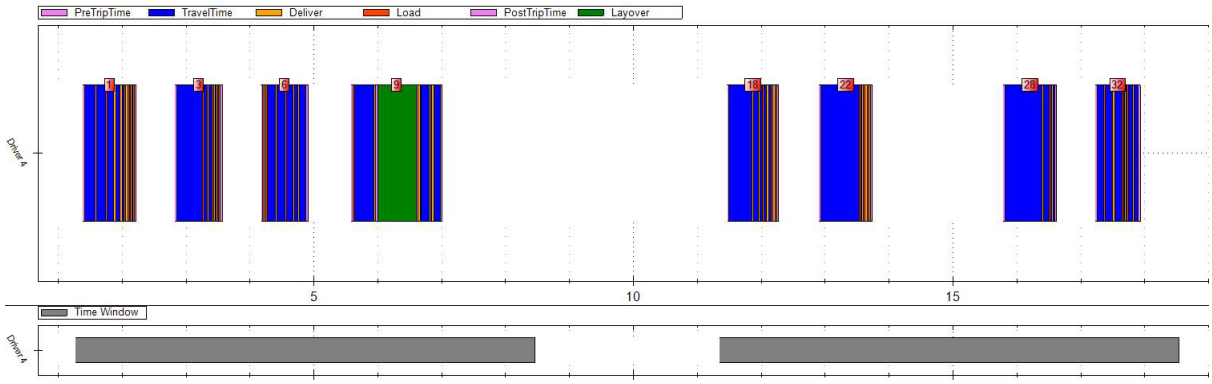
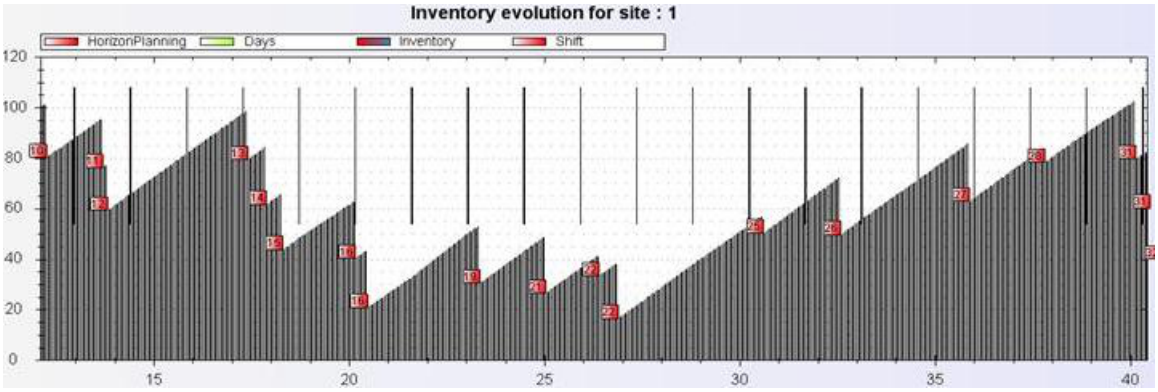
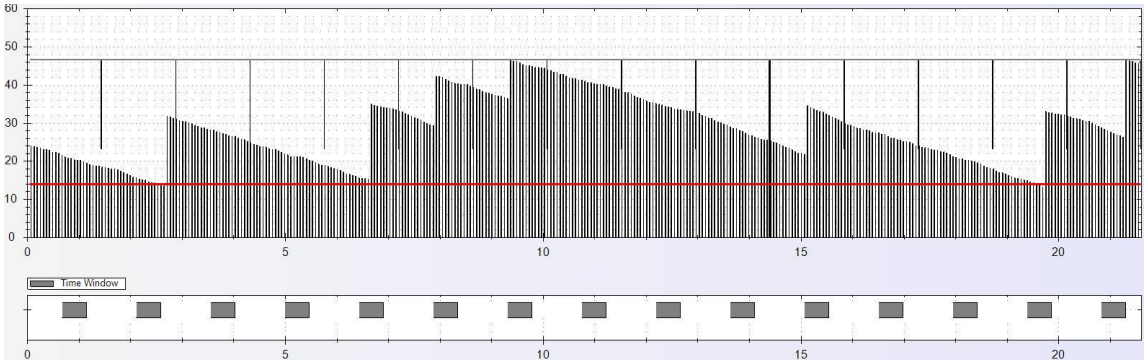


FIGURE 6.4 – La liste des tournées générées par un conducteur.



Les chargements à l'usine 1.

FIGURE 6.5 – Les chargements à l'usine 1.



Le niveau de sécurité est marqué horizontalement par une ligne et les jours du planning par une ligne verticale. Toutes les livraisons, qui sont caractérisées par une augmentation du niveau d'inventaire, ont bien lieu pendant les heures d'ouverture.

FIGURE 6.6 – Le niveau d'inventaire pour un client sur 15 jours.

6.2 Annexe 2 : Modèle détaillé de l'IRP

Cette section a pour objectif de proposer une modélisation détaillée du modèle du problème d'optimisation de tournées de véhicules avec gestion des stocks (Voir chapitre 4 de cette thèse). Nous présentons l'ensemble du modèle avec les contraintes portant sur le routage, les ressources, la gestion des stocks, les tournées et les voyages (ensemble de tournées). Nous détaillons ici tout d'abord les notations utilisées dans la modélisation de ce problème avant de préciser ensuite les différentes contraintes.

6.2.1 Notations

Les ressources. Les ressources sont de trois types : les conducteurs, les tracteurs et les remorques. Une ressource *conducteur* représente soit un conducteur, soit une paire de conducteurs (c'est le cas dans les zones géographiques étendues, comme au Canada par exemple, pour faire des tournées très longues). Pour chaque conducteur $d \in$ CONDUCTEURS, nous connaissons son dépôt noté $base(d)$ auquel il est rattaché. Il quitte ce dépôt en début de tournée pour la terminer toujours à ce même dépôt. L'ensemble $disponibilites(d)$ représente les fenêtres de disponibilité du conducteur jusqu'à l'horizon. L'ensemble $tracteur(d)$ stocke tous les tracteurs que le conducteur d est en mesure de conduire. Afin de calculer les coûts d'utilisation de cette ressource, nous utilisons différents facteurs : tout d'abord, le coût par unité de temps de travail noté $coutTemporel(d)$, ensuite le coût d'une opération d'un chargement à une usine de production noté $coutChargement(d)$, puis symétriquement le coût d'une opération de livraison noté $coutLivraison(d)$ et enfin le coût fixe d'une pause prise par le conducteur $coutPause(d)$. Nous adaptons ces coûts et ces contraintes, si d est une paire de conducteurs. Ensuite, les ressources peuvent être aussi du type *tracteurs*. Pour un tracteur noté $tr \in$ TRACTEURS, nous connaissons son dépôt de rattachement noté $base(tr)$ et son ensemble de fenêtres de temps de disponibilité $disponibilites(tr)$ (certains tracteurs peuvent être parfois en panne ou en maintenance). $remorques(tr)$ représente la liste des remorques compatibles avec ce tracteur. Nous connaissons aussi pour chaque tracteur tr leur vitesse $tS(tr)$ et le coût par unité de distance parcourue $coutDistance(tr)$. Enfin, les *remorques* sont les dernières ressources du triplet. Une remorque $w \in$ REMORQUES est définie par son dépôt de rattachement $base(w)$, son ensemble de fenêtres de temps de disponibilité $disponibilites(w)$, sa capacité $capacite(w)$ (c'est-à-dire la quantité maximale chargeable dans le camion et donc livrable au client) et sa quantité initialement présente dans la remorque au début de la période, notée $quantiteInitiale(w)$.

Les sites. Les sites sont de trois types : *base*, *client* et *usine*. Une *base* $b \in$ BASES est un site affecté à une ressource, utilisé comme point de départ et d'arrivée de toute tournée. Un *client* $c \in$ CLIENTS est représenté par sa capacité de stockage $capacite(c)$, c'est-à-dire la quantité maximale livrable, son seuil de sécurité $seuilSecurite(c)$ qui correspond à la

quantité minimale de produit qui doit être maintenue dans le stock pour éviter des coûts d'assèchement, sa quantité initiale de produit dans le réservoir au début de la période notée *quantiteInitiale*(c), ses prévisions de consommation *previsions*(c, h) pour chaque pas de temps h, son ensemble de fenêtres de temps d'ouverture *disponibilites*(c), ses ensembles de conducteurs *conducteursAutorises*(c) autorisés à entrer sur le site (de même l'ensemble de tracteurs *tracteursAutorises*(c) et de remorques *remorquesAutorises*(c)), son temps fixe d'opération *dureeOperation*(c) pour réaliser une livraison et enfin sa liste de commandes *commandes*(c) passées par le client. Pour chaque client c, nous connaissons aussi les coûts associés à chaque commande non traitée *coutCommandesManquees*(c) et ceux associés à chaque pas de temps passé sous le seuil de sécurité *coutAssechement*(c). Un client peut avoir une propriété supplémentaire : l'attribut *premierApresSource*(c) indique s'il doit ou non être livré juste après une opération de chargement dans la tournée (utile pour garantir la pureté d'un produit avant sa livraison). Une *commande* m est caractérisée par sa quantité *quantite*(m) demandée par le client, sa plus petite *dateMin*(m) et plus grande date de livraison *dateMax*(m) définissant la fenêtre de temps pendant laquelle la livraison doit avoir lieu. Une usine $u \in$ USINES est définie de manière similaire à un client, sans les attributs relatifs aux commandes et au niveau de sécurité : *coutAssechement*(u), *commandes*(u), *premierApresSource*(u). Les consommations d'un client (resp. d'une usine de production) sont représentées par des valeurs positives (resp. négatives).

Les matrices. Deux matrices sont fournies : *distances*(c_1, c_2) donne la distance entre deux sites c_1 et c_2. *durees*(c_1, c_2, k) correspond aux temps de transport entre deux sites c_1 et c_2 utilisant un tracteur i avec l'indice $tS(i)$ qui vaut l'index k. Les 2 matrices ne sont pas symétriques, mais nous assumons qu'elles satisfont l'inégalité triangulaire. Des opérations de vérification doivent être effectuées au début et à la fin de chaque tournée, tout comme avant et après chaque pause ; ces durées fixes sont respectivement notées *tpsAvant* et *tpsApres*. *dureeOperation*(c) est le temps de réalisation d'une activité de chargement ou de déchargement au site c. Les dates de début et de fin de la tournée s sont respectivement *debut*(s) et *fin*(s) alors que *arrivee*(c) et *depart*(c) correspondent aux dates d'arrivée et de départ d'un site c pour une livraison ou un chargement.

Les données de sortie. Une solution à ce problème est un ensemble de tournées valides. Une *tourne* $s \in$ TOURNEES est définie par son triplet de ressources (son conducteur *conducteur*(s), son tracteur *tracteur*(s) et sa remorque *remorque*(s)), la date de début *debut*(s) où elle quitte la base *base*(s), sa quantité initiale de produit présent dans le camion *quantiteRemDebut*(s) au début de la tournée et *quantiteRemFin*(s) celle à la fin de la tournée, sa date de fin *fin*(s) où elle y revient et par son ensemble d'opérations OPERATIONS(s) ordonnées chronologiquement, correspondant aux livraisons / chargements effectués sur la tournée. Une *opération* o est définie par sa tournée d'appartenance *tournee*(o), son site où elle a lieu *point*(o), la commande m satisfaite s'il y en a une, son type (chargement/livraison), la quantité livrée ou chargée *quantite*(o) (positive pour les livraison,

négative pour les chargements), la commande satisfaite par cette opération (si il y en a une), ses dates de début $arrivee(o)$ et de fin $depart(o)$ (fenêtre de temps qui doit être inclue dans une période d'ouverture du site) et sa liste de pauses $pausesAvant(o)$ depuis l'opération précédente. Chaque tournée contient une opération finale fictive permettant d'enregistrer la liste des pauses entre le dernier site et la base. Une *pause* p est un intervalle de temps de non-activité entre deux sites, définie par sa date de début $debut(p)$, sa date de fin $fin(p)$, son temps de conduite $tpsConduiteAvant(p)$ depuis la dernière opération ou la dernière pause (au cas où il y a plusieurs pauses entre deux sites). Si $tpsConduiteAvant(p) = 0$, cela signifie que la pause est prise au site précédent. Les niveaux d'inventaire (pour les clients, les usines et les remorques) peuvent être calculés à l'aide des quantités livrées ou chargées dans les tournées. On note $quantiteReservoir(i, h)$ la quantité de produit du réservoir d'un site i à un instant h et par $quantiteRem(r, o)$ la quantité de produit dans la remorque r à la fin de l'opération o. Si une opération o est la dernière d'une tournée s, on doit avoir $quantiteRemFin(s) = quantiteRem(r, o)$. Toutes les données d'entrée / sortie relatives au volume sont entières.

Temps de travail et de conduite. Pour des raisons de sécurité et pour respecter la loi, les conducteurs ne peuvent pas travailler plus d'une certaine durée sans faire une pause obligatoire. Plus précisément, une durée maximale de conduite $dureeMaxConduite(d)$ et une durée maximale de conduite $dureeMaxConduite(d)$ sont définies pour chaque conducteur (dépendant du pays, ces durées sont généralement comprises entre 10 et 15 heures). Pour chaque conducteur d, $tpsConduiteCumule(d, t)$ au temps t correspond au temps de conduite cumulé depuis la fin de la dernière pause ou depuis le début de la tournée. En même temps, $tpsTravailCumule(d, t)$ correspond au temps écoulé depuis la fin de la dernière pause ou le début de la tournée. À tout instant t de la tournée, $tpsConduiteCumule(d, t)$ (resp. $tpsTravailCumule(d)$) ne peut pas dépasser $dureeMaxConduite(d)$ (resp. $dureeMaxConduite(d)$).

Des opérations de vérification doivent être effectuées au début et à la fin de chaque tournée, tout comme avant et après chaque pause ; ces durées fixes sont notées respectivement $tpsAvant$ et $tpsApres$. Tout comme les opérations, ces tâches ne peuvent pas être interrompues pour une pause. Le schéma 4.2 donne deux représentations graphiques d'une tournée avec ces durées. $durees(a, b)$ correspond au temps nécessaire pour se déplacer du point a au point b, $tpsAvant$ et $tpsApres$ sont les temps de traitement avant et après chaque pause,

Le coût des tournées n'est pas proportionnel aux durées de travail et aux distances parcourues. Divers coûts fixes doivent être comptabilisés. En particulier, $coutChargement(d)$ and $coutLivraison(d)$ sont définis pour chaque conducteur d, représentant respectivement les coûts fixes de chargement et de livraison. Un coût fixe est aussi ajouté à chaque fois qu'une pause est prise.

Enfin, le dernier jeu de contraintes concerne les commandes. Une commande est considérée comme satisfaite si une opération o est planifiée en satisfaisant $quantite(o) \geq quantite(r)$ et $arrivee(o) \in [dateMin(r),\ dateMax(r)[$, avec $arrivee(o)$ la date de début de l'opération

o, $dateMin(r)$ (resp. $dateMax(r)$) la date minimale (resp. maximale) à laquelle la commande doit être satisfaite.

6.2.2 Contraintes

Les contraintes de routage. Sont listées ici les contraintes concernant les ressources de la tournée, appelées "contraintes de routage". Les trois ressources (conducteur, tracteur et remorque) affectées à la tournée doivent être localisées à la base d'une tournée. La remorque w de la tournée doit appartenir à l'ensemble des remorques autorisées $remorques(tr)$ par le tracteur tr qui lui même doit faire partie de l'ensemble des tracteurs autorisés $tracteurs(d)$ par le conducteurs d. L'intervalle $[debut(s),\ fin(s)[$ induit par toute tournée s doit être inclus dans un intervalle de disponibilité pour chaque ressource affectée à s. Enfin, les tournées réalisées par une ressource ne peuvent pas se chevaucher dans le temps (c'est-à-dire, les intervalles de temps induits par les tournées sont disjoints deux à deux). On retrouve des contraintes spécifiques aux conducteurs. Pour tout conducteur d, deux tournées s_1 et s_2 successives affectées à d doivent être séparées d'une durée d'au moins $dureePauseMin(d)$. On a donc :

$$\forall\, d \in \text{CONDUCTEURS},\ \forall\, s_1,\ s_2 \in \text{TOURNEES}(d),$$
$$debut(s2) \geq fin(s1) + dureePauseMin(d) \vee debut(s1) \geq fin(s2) + dureePauseMin(d)$$

De plus, deux tournées s_1 et s_2 successives affectées à d ne peuvent pas se chevaucher :

$$\forall\, d \in \text{CONDUCTEURS},\ \forall\, s1,\ s2 \in \text{TOURNEES}(d),$$
$$debut(s2) \geq fin(s1) + dureePauseMin(d) \vee$$
$$debut(s1) \geq fin(s2) + dureePauseMin(d)$$

La durée de la tournée d'un conducteur d ne peut pas dépasser l'amplitude maximale $amplitudeMax(d)$:

$$\forall\, s \in \text{TOURNEES},\ fin(s) - debut(s) \leq amplitudeMax(d)$$

Pour chaque tournée s et chaque opération o (y compris les opérations initiales et finales de chaque tournée), la durée totale de conduite cumulée est soit celle depuis le site précédent plus le temps de trajet depuis ce point, soit le temps de trajet depuis la dernière pause si cette pause a été prise depuis la visite du dernier site. ($precDansTour(o)$ représente l'opération précédent l'opération o dans la tournée).

---

begin

$\forall\, s \in$ TOURNEES, $\forall\, o \in$ OPERATIONS(s), ;

if $PAUSESAVANT(o) = 0$ **then**

$$tpsConduiteCumule(o) = tpsConduiteCumule(precDansTour(o)) + tpsTrajet(o,s)$$

else

$$tpsConduiteCumule(o) = tpsTrajet(o,s) - \sum_{\substack{l \in \\ \text{PAUSESAVANT}(o)}} tpsConduiteAvant(l)$$

Comme $tpsConduiteCumule(o)$ doit être non-négatif, la contrainte ci-dessus fixe une borne supérieure sur la somme des $tpsConduiteAvant(l)$. De plus, pour chaque opération, le temps cumulé de conduite avant une opération ne doit pas excéder le temps de conduite maximum autorisé :

$$\forall\, s \in \text{TOURNEES}, \forall\, o \in \text{OPERATIONS}(s),$$
$$0 \leq tpsConduiteCumule(o) \leq dureeMaxConduite(\text{CONDUCTEURS}(s))$$

Pour la première pause avant une opération o, le temps de conduite depuis le dernier point additionné au temps de conduite avant cette pause ne peut pas excéder le temps maximal de conduite autorisé. Pour les autres pauses avant o, le temps de conduite ne peut pas excéder la durée maximale de conduite autorisée :

---

begin
  $\forall\, s \in$ TOURNEES, $\forall\, o \in$ OPERATIONS(s), $\forall\, l \in LayoversBefore(o)$, ;
  if $l = premier(pausesAvant(o))$ **then**

$$tpsConduiteCumule(precDansTour(o)) + tpsConduiteAvant(l) \leq dureeMaxConduite(\text{CONDUCTEURS}(s));$$

  else
    $tpsConduiteAvant(l) \leq dureeMaxConduite(\text{CONDUCTEURS}(s))$;

Pour chaque opération o, on définit la variable $finDernierePause(o)$ comme étant la fin soit de la dernière pause avant cette opération, soit de celle du dernier point :

---

begin
  $\forall\, s \in$ TOURNEES, $\forall\, o \in$ OPERATIONS$(s) \bigcup final(s)$,
  if $pausesAvant(o) = 0$ **then**
    $finDernierePause(o) = finDernierePause(precDansTour(o))$;
    else $finDernierePause(o) = fin(derniere(pausesAvant(o)))$;

Ainsi, pour chaque opération o, le temps écoulé depuis la fin de la dernière pause jusqu'au début de l'opération o ne doit pas dépasser le temps de travail maximal autorisé :

$$\forall\, s \in \text{TOURNEES}, \forall\, o \in \text{OPERATIONS}(s),$$
$$depart(o) - finDernierePause(o) \leq dureeMaxConduite(\text{CONDUCTEURS}(s))$$

De même, pour chaque pause avant une opération o, le temps écoulé depuis la fin de la dernière pause jusqu'au début de cette pause ne peut pas dépasser ni le temps de travail maximal autorisé et ni la durée minimale de travail avant de prendre une pause :

---

begin
  $\forall\, s \in$ TOURNEES, $\forall\, o \in$ OPERATIONS(s), $\forall\, l \in pausesAvant(o)$,
  if $l = premier(pausesAvant(o))$ **then**

$$dureeMinTravailAvantPause \leq debut(l) - finDernierePause(precDansTour(o)) \leq dureeMaxConduite(\text{CONDUCTEURS}(s));$$

  else

$$dureeTravailMinAvantPause \leq debut(l) - fin(precedent(l, PAUSESAVANT(o)) \leq dureeMaxConduite(\text{CONDUCTEURS}(s)));$$

La durée minimale d'une pause est $dureePauseMin$.

$$\forall\, s \in \text{TOURNEES}, \forall\, o \in \text{OPERATIONS}(s), \forall\, l \in pausesAvant(o), \quad fin(l) - debut(l) \leq dureePauseMin(conducteur(s))$$

Pour chaque pause avant une opération o, le temps écoulé depuis la fin de la dernière pause (ou depuis le début du dernier point) jusqu'au début de la pause ne peut pas être plus petit que le temps de conduite requis pour ce trajet (plus un temps de traitement avant et après, quand on n'est pas à la base) :

A noter que si un conducteur représente en réalité une paire de conducteurs, alors les règles de conduite / travail ne s'appliquent pas identiquement. Par exemple, pour une paire de conducteurs qui seraient chacun sujet aux règles 11/14/10 $heures$, on aura $dureeMaxConduite(d) = dureeMaxConduite(d) = amplitudeMax(d)$ car nous considérons que les deux conducteurs alternent leur période de conduite / travail (le second conducteur travaille quand le premier conducteur s'arrête, et vice versa). Par contre, la durée des tournées doit rester contrainte par le paramètre $amplitudeMax(d)$.

Les contraintes de remorques. Une première contrainte sur les remorques est d'éviter les chevauchements de tournées. Ainsi, pour chaque remorque, si on considère deux tournées

```
begin
    ∀ s ∈ TOURNEES, ∀ o ∈ OPERATIONS(s), ∀ l ∈ pausesAvant(o),
    if l = premier(pausesAvant(o)) then
        debut(l) − depart(point(precDansTour(o))) ≤ tpsConduiteAvant(l) + tpsApres;
        else
            debut(l) − fin(precedent(l, pausesAvant(o))) ≤ tpsConduiteAvant(l)
                                                          + tpsApres + tpsAvant;
    ∀ s ∈ TOURNEES, ∀ l ∈ pausesAvant(initial(s)),
    if l = premier(pausesAvant(o)) then
            debut(l) − depart(point(precDansTour(o))) ≤ tpsConduiteAvant(l);
        else
            debut(l) − fin(precedent(l, pausesAvant(o))) ≤ tpsConduiteAvant(l);
```

r_1 et r_2 réalisées par cette remorque, soit r_1 finit avant le début de r_2 ou r_2 finit avant le début de r_1 :

$$\forall\, r \in \text{REMORQUES}, \forall\, s_1,\ s_2 \in \text{TOURNEES}(r), \quad debut(s_2) \geq fin(s_1) \vee debut(s_1) \geq fin(s_2)$$

Pour chaque tournée r, l'intervalle complet $[debut(r),\ fin(r)]$ doit correspondre à une des fenêtres de temps de la remorque :

$$\forall\, s \in \text{TOURNEES}, \exists\, tw \in disponibilites(remorque(s)), \quad debut(s) \geq debut(tw) \vee fin(tw) \geq fin(s)$$

Pour chaque tournée r, la remorque affectée doit être compatible et appartenir à la liste des remorques autorisées à être tractées par le tracteur :

$$\forall\, s \in \text{TOURNEES},\ remorque(s) \in remorques(tracteur(s))$$

Les contraintes de tracteurs. De même que pour les remorques, une première contrainte sur les tracteurs est d'éviter les chevauchements de tournées. Ainsi, pour chaque tracteur, si on considère deux tournées r_1 et r_2 réalisées par ce tracteur, soit r_1 finit avant le début de r_2 ou r_2 finit avant le début de r_1 :

$$\forall\, tr \in \text{TRACTEURS},\ \forall\, s_1,\ s_2 \in tours(tr),\\ debut(s_2) \geq fin(s_1) \vee debut(s_1) \geq fin(s_2)$$

Pour chaque tournée r, l'intervalle complet $[debut(r),\ fin(r)]$ doit correspondre à une des fenêtres de temps du tracteur.

$$\forall\, s \in \text{TOURNEES},\ \exists\, tw \in disponibilites(tracteur(s)),\\ debut(s) \geq debut(tw) \vee fin(tw) \geq fin(s)$$

Pour chaque tournée r, le tracteur affecté doit être compatible et appartenir à la liste des tracteurs autorisés à être conduits par le conducteur $conducteur(r)$:

$$\forall\, s \in \text{TOURNEES},\ tracteur(s) \in TRACTEURS(conducteur(s))$$

Les contraintes de gestion des stocks. En plus de ces contraintes de ressources, le problème nécessite de satisfaire les contraintes d'inventaires. Les niveaux d'inventaires pour chaque client, usine et remorque sont calculables à partir des quantités livrées ou chargées. Bien évidement, ces inventaires doivent rester à tout pas de temps positifs et inférieurs à la capacité du stockage :

$$\forall\, p \in \text{SOURCES},\ \forall\, h \in [0,\ H-1],\ 0 \leq quantiteReservoir(p,h) \leq capacite(p)\\ \forall\, c \in \text{CLIENTS},\ \forall\, h \in [0,\ H-1],\ 0 \leq quantiteReservoir(c,h) \leq capacite(c)$$

Pour chaque source p, la quantité présente $quantiteReservoir(p,h)$ à chaque pas de temps h dans le réservoir est égale à la quantité dans le réservoir au pas de temps $h-1$, plus la production prévue pour le pas de temps h, moins l'ensemble des chargements qui ont eu lieu pendant le pas de temps h (avec $previsions(p,h)$ et $quantite(o)$ qui sont négatives pour les sources). Les équations basiques d'inventaire dynamique pour les sources (avec $quantiteReservoir(p,h) \geq 0$) s'écrivent donc :

$$\forall\, p \in \text{SOURCES}$$
$$quantiteReservoir(p,\ -1) \ =\ quantiteInitiale(p)$$
$$\forall\, h \in [0,\ H-1],\ quantiteReservoir(p,\ h) \ =\ min(dyn,\ capacite(p))$$
$$avec :\ dyn \ =\ quantiteReservoir(p,\ h{-}1) - previsions(p,\ h) + \sum_{o \text{ OPERATIONS}(p,\ h)} quantite(o)$$

Si *dyn* excède la capacité du réservoir, cela signifie qu'il y a dépassement de la source. Cela n'est pas pénalisé dans la fonction objectif, mais ce dépassement pourrait y être intégré.

Symétriquement pour chaque client c, on a donc (avec $quantiteReservoir(c, h) \ \leq\ capacite(c)$) :

$$\forall\, c \in \text{CLIENTS}$$
$$quantiteReservoir(c,\ -1) \ =\ quantiteInitiale(c)$$
$$\forall\, h \in [0,\ H-1], quantiteReservoir(c,\ h) \ =\ max(dyn,\ 0)$$
$$avec :\ dyn \ =\ quantiteReservoir(c,\ h{-}1) - previsions(c,\ h) + \sum_{o \text{ OPERATIONS}(c,\ h)} quantite(o)$$

Si *dyn* est sous 0, cela signifie que le client est en assèchement. On pénalise cela dans la fonction objectif à l'aide d'un coût d'assèchement payé lorsque :

$$quantiteReservoir(c,\ h) \ <\ niveauSecurite(c)$$

Les contraintes des tournées. En plus de ces contraintes portant sur les ressources et les volumes, on retrouve ici des contraintes associées à la construction même des tournées. Tout d'abord, on retrouve des contraintes sur les dates. On distingue la date de départ de la date de début de tournée, le départ devant avoir lieu après un temps de préparation :

$$\forall\, s \in TOURNEES,\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v),\ depart(s) \ \geq\ debut(s) + tpsAva$$

De même, un temps de traitement doit être prévu en fin de tournée afin d'assurer que le conducteur dispose d'un temps suffisant pour clôturer sa journée :

$$\forall\, s \in TOURNEES,\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v),\ fin(s) \ \geq\ arrivee(s) + tpsApre$$

Ensuite, à chaque point visité, il faut s'assurer qu'on a bien respecté le temps de conduite depuis le dernier point. On utilise ici des inégalités afin d'autoriser du temps d'attente entre la fin d'un voyage et la date d'arrivée à un point, ce dernier matérialisant la date d'entrée sur le site (qui peut être mise en attente à cause des horaires d'ouvertures) :

$$\begin{aligned}\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v)\\ arrivee(o) \geq depart(tourneePrec(o)) + tpsTrajet(o, v)\\ arrivee(v) \geq depart(derniere(\text{OPERATIONS}(v))\\ + durees[point(derniere(\text{OPERATIONS}(v))),\ base(s)]\end{aligned}$$

Tout chargement ou déchargement à un même site prend un temps constant, quelque soit le voyage v :

$$\begin{aligned}\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v),\\ depart(o) = arrivee(o) + dureeOperation(point(o))\end{aligned}$$

Les opérations de chargement et de déchargement doivent être effectuées pendant les horaires d'ouverture des sources et des clients, respectivement. Ainsi, pour toutes les opérations, l'intervalle $[arrivee(o),\ depart(o)]$ doit être totalement inclus dans une des fenêtres de temps d'ouverture du site :

$$\begin{aligned}\forall\, o \in \text{OPERATIONS}(s),\ tw \in disponibilites(point(o)),\\ arrivee(o) \leq debut(tw) \leq fin(tw) \leq depart(o)\end{aligned}$$

Les opérations de chargement et de déchargement requièrent que le site soit accessible à ce conducteur, à ce tracteur et à cette remorque :

$$\begin{aligned}\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v)\\ conducteur(s) \in conducteursAutorises(point(o))\\ tracteur(s) \in tracteursAutorises(point(o))\\ remorque(s) \in remorquesAutorisees(point(o))\end{aligned}$$

La quantité contenue après chaque opération dans la remorque ne peut pas être négative ou excéder la capacité de la remorque :

$$\begin{aligned}\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v) \setminus final(s),\\ quantiteRem(o) = quantiteRem(remorque(s),\ s, tourneePrec(o))\\ - quantite(o)\\ quantiteRem(o) \geq 0\\ quantiteRem(o) < capacite(remorque(s))\end{aligned}$$

La quantité initiale contenue dans la remorque pour une tournée est la quantité finale de la tournée précédente de cette remorque :

---

begin
  $\forall\, s \in$ TOURNEES,
  if $r = premier(\text{TOURNEES}(remorque(s)))$ **then**
    $quantiteRemDebut(s) = quantiteInitiale(remorque(s));$
  else
    $quantiteRemDebut(s) = quantiteRemFin(precedent(s,\ tours(remorque(s))));$

On définit donc $quantiteRemFin(s)$ comme la quantité finale restant dans la remorque de la tournée s. Certains clients nécessitent de se faire livrer juste après le chargement à une source. Pour toutes ces livraisons, on sait que l'opération précédente (si elle existe) doit être un chargement. Autrement si la livraison est la première opération de la tournée, la dernière opération de la tournée précédente à cette tournée doit être un chargement.

$$\begin{aligned}\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v)\\ (premierApresSource(\ point(o)\ ) = 0\ )\\ \vee\ (\ point\ (tourneePrec(o)\ ) \in \text{SOURCES}\ )\\ \vee\ (o = premier(\text{OPERATIONS}(s))\\ \vee\ (premier(\text{TOURNEES}(remorque(r)) \neq r \wedge\\ derniere(\text{OPERATIONS}(precedent(r,\ tours(remorque(r))))) \in \text{SOURCES})\\ \vee\ (premier(\text{TOURNEES}(remorque(r)) = r \wedge JusteCharge(remorque(r)) = true))\end{aligned}$$

Pour chaque source, les quantités chargées doivent être positives et pour chaque client, les quantités livrées doivent être aussi positives :

$$\forall\, p \in \text{CLIENTS},\ \forall\, h \in [0,\ H-1],\ \ \forall\, o \in \text{OPERATIONS}(p,h)\ \mathit{quantite}(o) \geq 0$$
$$\forall\, p \in \text{SOURCES},\ \forall\, h \in [0,\ H-1],\ \ \forall\, o \in \text{OPERATIONS}(p,h)\ \mathit{quantite}(o) \geq 0$$

Pour chaque opération de livraison o d'une tournée s, la quantité chargée / livrée doit être plus grande qu'une quantité minimale connue :

$$\forall\, s \in \text{TOURNEES},\ \forall\, o \in \text{OPERATIONS}(s),\ |\mathit{quantite}(o)| \leq \mathit{quantiteMinOperation}(\mathit{point}(o))$$

La quantité dans la remorque lorsqu'on arrive à un client doit excéder une quantité donnée afin d'assurer une distribution optimale. Nous rappelons que la quantité $\mathit{quantiteRem}(r,\ o)$ fait référence à la quantité de produit présente dans la remorque r après la livraison o :

$$\forall\, s \in \text{TOURNEES},\ \forall\, o \in op \in \text{OPERATIONS}(s)\ ,\ \mathit{point}(op) \in \text{CLIENTS}$$
$$\mathit{if}\ (o \neq \mathit{premier}(\text{OPERATIONS}(s))\ \mathit{then}$$
$$\mathit{quantiteRem}(\mathit{tourneePrec}(o)) \geq \mathit{quantiteMinRemorque}(\mathit{remorque}(s))$$
$$\mathit{else}\ \mathit{quantiteRemDebut}(s) \geq \mathit{quantiteMinRemorque}(\mathit{remorque}(s))$$

En plus de ces contraintes volumétriques entre tournées, on retrouve des contraintes relatives à l'accrochage et au décrochage de la remorque au tracteur. Pour chaque tournée r, si le précédent tracteur de la remorque diffère du tracteur de leur tournée ou si la remorque précédente du tracteur diffère de la remorque de la tournée, alors on active le booléen $\mathit{accroche}(r)$ que l'on fixe à *vrai*.

$$\forall\, s \in \text{TOURNEES},\ \mathit{accroche}(s) = (\mathit{precedentTracteur}(\mathit{remorque}(s)) \neq \mathit{tracteur}(s)$$
$$\vee\, \mathit{precedenteRemorque}(\mathit{tracteur}(s)) \neq \mathit{remorque}(s))$$

Pour chaque livraison, la dernière source visitée par une remorque doit être approuvée par le client livré, quelque soit la tournée s :

$$\forall\, s \in \text{TOURNEES},\ \forall\, o \in\ op \in \text{OPERATIONS}(s)\ ,\ \mathit{point}(op) \in \text{CLIENTS},$$
$$((\mathit{sourcesApprouvees}(\mathit{point}(o)) = \emptyset) \vee (\mathit{point}(\mathit{precedentChargement}(o))$$
$$\in \mathit{sourcesApprouvees}(\mathit{point}(o)))$$

Enfin, la base de chaque tournée s correspond à la base du conducteur $conducteur(s)$, à la base du tracteur $tracteur(s)$ et à la base de la remorque $remorque(s)$.

$$\forall\, s \in \text{TOURNEES},\quad base(s) = base(conducteur(s)) = base(remorque(s)) = base(tracteur(s))$$

Les contraintes des voyages (ensembles de tournées). Une tournée s pourra être constituée de plusieurs voyages $v \in VOYAGES(s)$, correspondant à un trajet base $\rightarrow$ clients / sources $\rightarrow$ base. Pour chaque voyage v et opération o (incluant les opérations initiale et finale de chaque tournée), l'arrivée à un point d'opération ne peut pas avoir lieu plus tôt qu'après avoir réalisé le temps de conduite depuis la dernière pause (plus le temps de traitement après quand on n'est pas à la base) :

$$\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\ \forall\, o \in \text{OPERATIONS}(v),\quad Arrival(o) \geq finDernierePause(o) + tpsAvant + tpsConduiteCumule(o)$$
$$et$$
$$Arrival(initial(v))\ ) \geq finDernierePause(initial(v))$$

Enfin, pour chaque voyage v, l'intervalle complet $[debut(v),\ fin(v)]$ doit correspondre à une des fenêtres de temps des conducteurs :

$$\begin{aligned}\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\quad & \exists\, tw \in disponibilites(\text{CONDUCTEURS}(s)),\\ & debut(v) \geq debut(tw) \wedge fin(tw) \geq fin(v)\\ & \wedge\ (tw \cap HeureDebut(conducteur(s)) = 0\\ & \vee\ \exists\, st \in (tw \cap HeureDebut(conducteur(s)),\\ & \mid debut(v) - st \mid \leq tpsAvantDemarrage(base(conducteur(s))))\end{aligned}$$

Pour chaque voyage v d'une tournée s, la base et le conducteur doivent être les mêmes :

$$\forall\, s \in \text{TOURNEES},\ \forall\, v \in \text{VOYAGES}(s),\\ (\mathit{base}(s) = \mathit{base}(v)) \wedge (\mathit{conducteur}(s) = \mathit{conducteur}(v))$$

La fenêtre de temps d'un voyage est l'enveloppe des fenêtres de temps de ses tournées. Tous les voyages v d'une tournée s doivent être inclus dans les fenêtres de temps $[\mathit{debut}(s),\ \mathit{fin}(s)]$:

$$\forall\ s \in \text{TOURNEES},\ \mathit{debut}(s) = \mathit{min}\ \ \mathit{debut}(v),\ v\ \mathit{in}\ \text{VOYAGES}(s)$$

$$\forall\ s \in \text{TOURNEES},\ \mathit{fin}(s) = \mathit{max}\ \ \mathit{fin}(v),\ v\ \mathit{in}\ \text{VOYAGES}(s)$$

6.3 Annexe 3 : Résultats numériques de l'IRP

données	clients	usines	dépots	cond.	tract.	rem.	commandes
A01	80	2	2	20	10	20	0
A02	108	1	1	35	18	35	0
A03	132	1	1	20	17	15	0
A04	130	2	1	17	10	20	0
A05	125	2	1	20	18	20	0
A06	46	1	2	50	50	50	0
A07	80	2	2	20	10	20	0
A08	75	1	1	10	10	10	0
A09	150	2	1	20	20	20	0
A10	250	5	1	30	30	30	0
A11	500	4	2	50	20	50	0
A12	108	1	1	35	18	32	0
A13	100	1	1	35	35	35	0
A14	70	1	1	50	5	10	0
A15	132	1	1	20	17	15	0
A16	130	2	1	17	10	20	0
A17	135	3	1	20	18	20	0

TABLE 6.1 – Benchmarks A sur le court terme : caractéristiques et résultats de l'algorithme glouton.

données	clients	usines	dépots	cond.	tract.	rem.	commandes
B01	500	4	2	50	40	50	0
B02	200	1	1	13	9	12	0
B03	100	1	1	35	35	35	0
B04	70	1	1	10	5	10	0
B05	135	3	1	20	18	20	0
B06	50	1	1	5	5	5	0
B07	200	1	1	13	9	12	0
B08	100	2	1	20	15	15	0
B09	124	2	1	20	18	20	0
B10	99	3	2	20	14	30	0
B11	75	1	1	10	10	10	0
B12	75	1	1	10	10	10	0
B13	75	1	1	10	10	10	0
B14	75	1	1	10	10	10	0
B15	198	3	1	10	10	8	0
B16	198	3	1	10	10	8	0
B17	198	3	1	10	10	8	0
B18	198	3	1	10	10	8	0
B19	50	1	1	5	5	5	0
B20	50	1	1	10	10	10	0
B21	50	1	1	5	5	5	0
B22	50	1	1	5	5	5	0
B23	50	1	1	5	5	5	0
B24	50	1	1	5	5	5	0
B25	50	1	1	5	5	5	0
B26	20	1	1	1	1	1	0
B27	99	3	1	20	20	20	0
B28	500	1	3	60	60	60	0
B29	500	1	3	60	60	60	0
B30	783	16	4	78	49	42	0
B31	1500	50	50	100	100	100	0

TABLE 6.2 – Benchmarks sur le court terme : caractéristiques et résultats de l'algorithme glouton.

données	clients	usines	dépots	cond.	tract.	rem.	commandes
C01	75	6	1	35	21	3	9
C02	75	6	1	34	21	3	4
C03	75	6	1	35	21	5	9
C04	75	6	1	35	21	5	2
C05	75	6	1	35	21	5	6
C06	75	6	1	35	21	5	6
C07	122	1	1	6	6	6	10
C08	122	1	1	6	6	6	15
C09	175	8	1	35	21	12	20
C10	175	8	1	31	21	12	28
C11	215	1	5	18	15	34	25
C12	272	17	9	57	27	27	616
C13	100	1	1	50	50	50	41

TABLE 6.3 – Benchmarks sur le court terme : caractéristiques et résultats de l'algorithme glouton.

données	$SO_{\min}$	$\mathrm{RL}_{\min}$
A01	0	0,014630
A02	0	0,009551
A03	0	0,013192
A04	0	0,039239
A05	0	0,043731
A06	0	0,031722
A07	0	0,015470
A08	0	0,194255
A09	0	0,194276
A10	0	0,147122
A11	0	0,032681
A12	0	0,010978
A13	0	0,013444
A14	0	0,016950
A15	0	0,009364
A16	0	0,048491
A17	0	0,032739

données	MO	SO	SC	DQ	RL
A01	-	0	57178	2317462	0,024673
A02	-	0	64472	3534883	0,018239
A03	-	0	77651	3531172	0,021990
A04	-	0	202829	2951746	0,068715
A05	-	0	199190	2173870	0,091629
A06	-	0	172410	1095217	0,157421
A07	-	0	45092	1602654	0,028136
A08	-	0	307783	875106	0,351710
A09	-	0	788566	2100705	0,375382
A10	-	0	1310183	3758893	0,348556
A11	-	0	595733	9410181	0,063307
A12	-	0	89696	5020209	0,017867
A13	-	0	64197	2429315	0,026426
A14	-	0	41885	1468461	0,028523
A15	-	0	54049	2945628	0,018349
A16	-	0	376060	3070358	0,122481
A17	-	0	334667	3247942	0,103040

TABLE 6.4 – Benchmarks sur le court terme A : bornes inférieures (gauche) et résultats de l'algorithme glouton (droite).

données	$SO_{\min}$	$\mathrm{RL}_{\min}$
B01	45	0,006406
B02	0	0,012301
B03	16	0,009767
B04	10	0,006043
B05	21	0,040597
B06	7	0,014001
B07	0	0,009816
B08	297	0,050168
B09	7	0,047395
B10	3	0,028094
B11	0	0,157607
B12	0	0,159432
B13	12	0,159432
B14	26	0,159362
B15	0	0,060412
B16	0	0,061183
B17	0	0,060412
B18	0	0,061692
B19	278	0,006026
B20	23	0,196649
B21	0	0,036080
B22	0	4,302140
B23	0	0,036080
B24	0	0,036080
B25	0	2,895112
B26	4	0,022161
B27	0	0,078626
B28	37	0,013677
B29	37	0,013677
B30	968	0,004329
B31	47	0,057477

données	*MO*	*SO*	*SC*	*DQ*	RL
B01	-	51	196307	13909999	0,014113
B02	-	196	131242	3905673	0,033603
B03	-	*16	160038	13809196	0,011589
B04	-	*10	23082	1379000	0,016738
B05	-	*25	291179	3583893	0,081247
B06	-	10	16638	509922	0,032629
B07	-	58	84933	3565791	0,023819
B08	-	8312	475665	5595653	0,085006
B09	-	55	362308	3227554	0,112255
B10	-	* 3	117994	1361526	0,086663
B11	-	3	293817	742473	0,395728
B12	-	7	237479	588415	0,403590
B13	-	*12	277024	586285	0,472508
B14	-	69	315868	769653	0,410404
B15	-	7600	471325	1619909	0,290958
B16	-	4356	505750	2322558	0,217756
B17	-	8802	405570	1213195	0,334299
B18	-	9458	418735	1315944	0,318201
B19	-	6971	10822	46100	0,234751
B20	-	1244	292095	685681	0,425993
B21	-	262	55080	631287	0,087250
B22	-	262	6557325	609803	10,753186
B23	-	357	60865	612436	0,099382
B24	-	258	98020	605441	0,161899
B25	-	265	3723050	572758	6,500215
B26	-	1088	23470	328763	0,071389
B27	-	927	529644	1038549	0,509985
B28	-	46322	155168	1986943	0,078094
B29	-	46298	149518	1986944	0,075250
B30	-	74094	124298	9880399	0,012580
B31	-	*47	4206864	33035582	0,127343

TABLE 6.5 – Benchmarks sur le court terme B : bornes inférieures (gauche) et résultats de l'algorithme glouton (droite).

données	$SO_{\min}$	$\mathrm{RL}_{\min}$
C01	57	0,445947
C02	57	0,411399
C03	0	0,038796
C04	37	0,027532
C05	0	0,041293
C06	0	0,042624
C07	0	0,024255
C08	140	0,023984
C09	13	0,217204
C10	13	0,223018
C11	0	0,005883
C12	0	0,900584
C13	0	0,035972

données	*MO*	*SO*	*SC*	*DQ*	RL
C01	6	2698	414736	379165	1,093814
C02	3	819	258656	251252	1,029469
C03	0	309	25570	240408	0,106362
C04	0	*37	1464	17827	0,082148
C05	0	31	30147	244708	0,123194
C06	0	41	22566	225504	0,100069
C07	0	1789	366799	214009	1,713942
C08	0	5271	281550	205571	1,369601
C09	0	1290	1043204	1972743	0,528809
C10	3	4085	1496640	2488223	0,601490
C11	1	10449	24271	1317292	0,018425
C12	95	14	3226530	2036466	1,584377
C13	4	5318	221965	3610200	0,061483

TABLE 6.6 – Benchmarks sur le court terme C : bornes inférieures (gauche) et résultats de l'algorithme glouton (droite).

données	nb tournee	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
A01	98	433	2,4	1,4	1,1	0,2	179	316
A02	76	481	4,3	2,4	1,9	0,6	254	540
A03	86	573	4,7	2,5	2,1	0,0	273	415
A04	46	370	6,0	3,3	2,8	0,1	501	598
A05	51	335	4,6	2,6	2,0	0,1	417	549
A06	87	413	2,7	2,2	0,6	0,2	929	857
A07	61	316	3,2	1,8	1,4	0,0	227	381
A08	25	159	4,4	2,2	2,1	0,2	493	651
A09	47	341	5,3	2,8	2,4	0,3	699	920
A10	70	626	6,9	3,8	3,2	0,5	933	1332
A11	116	1183	8,2	4,5	3,7	0,2	545	776
A12	69	632	7,2	3,8	3,4	0,0	391	581
A13	66	451	4,8	3,0	1,8	0,0	430	542
A14	53	273	3,2	1,7	1,5	0,0	244	362
A15	109	570	3,2	1,8	1,5	0,0	146	272
A16	54	433	6,0	3,4	2,6	0,6	810	1300
A17	52	463	6,9	3,8	3,1	0,5	720	1113

TABLE 6.7 – Benchmarks sur le court terme A : résultats de l'algorithme glouton (statistiques sur les tournées).

données	nb tournee	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
B01	335	2084	4,2	2,3	1,9	0,7	167	644
B02	125	780	4,2	2,4	1,8	0,6	321	594
B03	112	1839	14,4	7,5	6,9	0,0	550	864
B04	48	246	3,1	1,7	1,4	0,0	141	259
B05	65	500	5,7	3,2	2,5	0,1	482	699
B06	23	105	2,6	1,4	1,1	0,0	225	312
B07	49	588	10,0	5,7	4,3	0,2	517	834
B08	151	958	4,3	2,3	2,0	0,0	245	380
B09	52	446	6,6	3,7	2,8	0,6	788	1312
B10	46	226	2,9	2,0	0,9	0,1	400	475
B11	24	149	4,2	2,2	2,0	0,2	453	704
B12	22	127	3,8	2,0	1,8	0,1	425	574
B13	24	133	3,5	2,0	1,6	0,2	407	695
B14	29	175	4,0	2,3	1,7	0,3	398	770
B15	76	357	2,7	1,5	1,2	0,3	587	810
B16	79	446	3,6	1,9	1,7	0,3	602	817
B17	63	284	2,5	1,4	1,1	0,3	617	824
B18	78	313	2,0	1,1	1,0	0,2	498	729
B19	16	64	2,0	1,0	1,0	0,2	182	586
B20	65	691	8,6	4,3	4,3	0,4	239	925
B21	15	109	5,3	2,9	2,3	0,3	259	713
B22	12	97	6,1	3,4	2,7	0,4	319	887
B23	16	114	5,1	2,8	2,3	0,3	254	639
B24	16	108	4,8	2,6	2,2	0,4	238	802
B25	15	101	4,7	2,7	2,1	0,4	245	884
B26	18	71	1,9	1,0	0,9	0,1	36	254
B27	75	351	2,7	1,9	0,8	0,8	400	1849
B28	63	207	1,3	1,3	0,0	0,0	398	458
B29	62	206	1,3	1,3	0,0	0,0	389	408
B30	307	1184	1,9	1,0	0,8	0,0	197	417
B31	360	3975	9,0	5,8	3,3	0,1	474	566

TABLE 6.8 – Benchmarks sur le court terme B : résultats de l'algorithme glouton (statistiques sur les tournées).

données	nb tournee	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
C01	28	112	2,0	1,1	0,9	0,3	452	833
C02	20	77	1,9	1,1	0,8	0,2	387	696
C03	20	78	1,9	1,2	0,8	0,2	379	705
C04	2	8	1,5	1,5	0,0	0,5	196	1847
C05	14	75	2,7	1,7	1,0	0,7	638	1356
C06	13	55	2,2	1,4	0,8	0,5	517	1078
C07	63	252	2,0	1,0	1,0	0,2	260	577
C08	51	204	2,0	1,0	1,0	0,3	254	639
C09	84	406	2,8	1,6	1,2	0,2	298	683
C10	106	524	2,9	1,7	1,2	0,2	364	786
C11	58	279	2,8	1,8	1,0	0,0	348	568
C12	252	1355	3,4	2,1	1,3	0,0	150	645
C13	70	450	4,4	2,6	1,8	0,0	321	433

TABLE 6.9 – Benchmarks sur le court terme C : résultats de l'algorithme glouton (statistiques sur les tournées).

données	*MO*	*SO*	*SC*	*DQ*	RL	gain *GC*	gain RL
A01	-	0	61347	3140722	0,019533	20,8 %	20,8 %
A02	-	0	67627	5389262	0,012548	31,2 %	31,2 %
A03	-	0	75443	4617278	0,016339	25,7 %	25,7 %
A04	-	0	214050	4255173	0,050303	26,8 %	26,8 %
A05	-	0	187720	3130162	0,059971	34,5 %	34,5 %
A06	-	0	184128	1634342	0,112662	28,4 %	28,4 %
A07	-	0	49354	2357186	0,020938	25,6 %	25,6 %
A08	-	0	338629	1582160	0,214029	39,1 %	39,1 %
A09	-	0	981784	3666094	0,267801	28,7 %	28,7 %
A10	-	0	1600926	6656780	0,240496	31,0 %	31,0 %
A11	-	0	729420	15765833	0,046266	26,9 %	26,9 %
A12	-	0	93166	7084289	0,013151	26,4 %	26,4 %
A13	-	0	61572	3012796	0,020437	22,7 %	22,7 %
A14	-	0	37755	1963049	0,019233	32,6 %	32,6 %
A15	-	0	47438	3824474	0,012404	32,4 %	32,4 %
A16	-	0	376239	4398543	0,085537	30,2 %	30,2 %
A17	-	0	356575	5135668	0,069431	32,6 %	32,6 %

TABLE 6.10 – Benchmarks sur le court terme A : résultats LS-RL (statistiques sur les coûts).

données	*MO*	*SO*	*SC*	*DQ*	RL	gain *GC*	gain RL
B01	-	48	208995	17147972	0,012188	5,9 %	13,6 %
B02	-	69	132251	4654644	0,028413	64,7 %	15,4 %
B03	-	*16	169664	15643641	0,010846	0,0 %	6,4 %
B04	-	*10	17700	1853051	0,009552	0,7 %	42,9 %
B05	-	*21	353487	5243479	0,067415	16,0 %	17,0 %
B06	-	10	14348	958772	0,014965	13,3 %	54,1 %
B07	-	0	93094	5720379	0,016274	97,3 %	31,7 %
B08	-	1152	690985	9136192	0,075632	86,1 %	11,0 %
B09	-	* 7	390235	4661929	0,083707	76,8 %	25,4 %
B10	-	* 3	120401	2367933	0,050846	30,7 %	41,3 %
B11	-	3	293645	1392672	0,210850	43,4 %	46,7 %
B12	-	7	245273	1217519	0,201453	42,7 %	50,1 %
B13	-	*12	264905	1300288	0,203728	45,4 %	56,9 %
B14	-	*26	346686	1592901	0,217645	56,6 %	47,0 %
B15	-	1378	557875	4586864	0,121624	81,8 %	58,2 %
B16	-	834	535275	4374700	0,122357	80,7 %	43,8 %
B17	-	1537	580345	4607614	0,125953	82,5 %	62,3 %
B18	-	2580	522365	3664172	0,142560	72,7 %	55,2 %
B19	-	6472	6773	46400	0,145970	7,2 %	37,8 %
B20	-	374	392495	1872138	0,209651	69,9 %	50,8 %
B21	-	0	47685	1131970	0,042126	100,0 %	51,7 %
B22	-	0	6438345	1157627	5,561675	84,9 %	48,3 %
B23	-	0	52320	1139022	0,045934	99,9 %	53,8 %
B24	-	0	126960	1204818	0,105377	99,6 %	34,9 %
B25	-	0	3197695	1078086	2,966085	91,0 %	54,4 %
B26	-	137	24405	816473	0,029891	87,2 %	58,1 %
B27	-	0	545077	1506760	0,361754	100,0 %	29,1 %
B28	-	28834	336092	1986943	0,169150	37,8 %	-116,6 %
B29	-	28892	353603	1986944	0,177963	37,6 %	-136,5 %
B30	-	15659	141026	15978096	0,008826	78,9 %	29,8 %
B31	-	*47	5016050	46282014	0,108380	0,4 %	14,9 %

TABLE 6.11 – Benchmarks sur le court terme B : résultats LS-RL (statistiques sur les coûts).

données	*MO*	*SO*	*SC*	*DQ*	RL	gain *GC*	gain RL
C01	3	530	639297	563692	1,134125	72,3 %	-3,7 %
C02	3	74	233825	322556	0,724912	63,5 %	29,6 %
C03	0	0	33215	533379	0,062273	99,8 %	41,5 %
C04	0	*37	2901	87268	0,033239	1,3 %	59,5 %
C05	0	0	29851	505713	0,059028	98,2 %	52,1 %
C06	0	0	24280	466894	0,052003	98,8 %	48,0 %
C07	0	0	319365	341328	0,935655	95,2 %	45,4 %
C08	0	2407	266655	350180	0,761480	54,1 %	44,4 %
C09	0	33	1285520	3469495	0,370521	94,8 %	29,9 %
C10	0	149	2063977	5304062	0,389131	95,8 %	35,3 %
C11	1	235	53706	3827663	0,014031	97,7 %	23,8 %
C12	89	0	3020317	2114443	1,428422	6,5 %	9,8 %
C13	1	7	372072	6513105	0,057127	98,0 %	7,1 %

TABLE 6.12 – Benchmarks sur le court terme C : résultats LS-RL (statistiques sur les coûts).

données	nb tournee	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
A01	115	565	2,9	1,7	1,2	0,2	159	298
A02	135	790	3,9	2,1	1,7	0,2	143	342
A03	133	830	4,2	2,4	1,8	0,0	163	341
A04	54	503	7,3	4,1	3,3	0,0	416	528
A05	53	404	5,6	3,3	2,4	0,1	348	513
A06	102	523	3,1	2,4	0,7	0,1	795	751
A07	78	449	3,8	2,3	1,5	0,0	188	394
A08	25	254	8,2	4,4	3,8	0,1	490	677
A09	55	546	7,9	4,3	3,6	0,3	712	975
A10	83	994	10,0	5,3	4,6	0,5	895	1360
A11	153	1865	10,2	5,4	4,8	0,2	456	744
A12	138	988	5,2	2,7	2,4	0,0	194	380
A13	77	548	5,1	3,2	1,9	0,0	327	466
A14	81	398	2,9	1,6	1,3	0,0	138	302
A15	157	821	3,2	1,8	1,4	0,0	81	246
A16	61	586	7,6	4,4	3,2	0,5	673	1172
A17	60	659	9,0	5,0	4,0	0,4	617	1008
B01	386	2506	4,5	2,4	2,1	0,7	150	638
B02	164	986	4,0	2,3	1,7	0,4	242	488
B03	154	2140	11,9	6,2	5,7	0,0	403	685
B04	69	357	3,2	1,8	1,4	0,0	66	228
B05	89	736	6,3	3,6	2,6	0,3	388	739
B06	35	188	3,4	2,0	1,4	0,0	116	295
B07	82	993	10,1	5,8	4,3	0,2	317	700
B08	229	1534	4,7	2,5	2,2	0,0	234	364
B09	58	607	8,5	4,8	3,7	0,5	736	1193
B10	53	353	4,7	2,9	1,8	0,1	315	462
B11	25	237	7,5	4,1	3,4	0,1	398	596
B12	24	209	6,7	3,7	3,0	0,0	335	487
B13	24	224	7,3	4,0	3,3	0,0	365	525
B14	32	271	6,5	3,7	2,8	0,2	354	664
B15	84	773	7,2	4,1	3,1	0,2	600	843
B16	91	760	6,4	3,6	2,8	0,2	525	739
B17	86	790	7,2	4,1	3,1	0,2	612	847
B18	72	647	7,0	4,1	2,9	0,3	669	908
B19	13	61	2,7	1,5	1,2	0,2	199	420
B20	71	1580	20,3	10,1	10,1	0,3	274	985
B21	11	161	12,6	7,0	5,6	0,2	283	712
B22	9	156	15,3	8,3	7,0	0,3	378	916
B23	11	164	12,9	7,0	5,9	0,3	294	790
B24	12	165	11,8	6,3	5,5	0,2	297	679
B25	16	189	9,8	6,2	3,6	0,6	197	1086
B26	7	111	13,9	7,4	6,4	0,0	162	528
B27	78	439	3,6	2,7	0,9	0,7	346	1937
B28	102	570	3,6	3,6	0,0	0,2	492	666
B29	101	591	3,9	3,7	0,1	0,3	511	795
B30	300	1529	3,1	2,0	1,1	0,0	206	500
B31	531	5402	8,2	5,0	3,2	0,1	358	461
C01	29	173	4,0	2,7	1,3	0,7	649	1451
C02	14	81	3,8	2,6	1,1	0,4	464	981
C03	24	115	2,8	1,8	1,0	0,3	397	935
C04	6	23	1,7	1,3	0,3	0,0	129	268
C05	21	120	3,3	2,2	1,0	0,4	401	934
C06	19	100	3,2	2,2	1,0	0,3	357	862
C07	65	354	3,4	1,7	1,7	0,1	360	585
C08	43	265	4,2	2,1	2,0	0,2	398	771
C09	99	588	3,9	2,4	1,6	0,2	254	737
C10	156	926	3,9	2,4	1,5	0,1	273	690
C11	114	704	4,2	2,7	1,5	0,1	383	738
C12	256	1389	3,4	2,2	1,2	0,0	138	706
C13	215	1051	2,9	1,7	1,2	0,0	144	282
average	93	706	6,3	3,6	2,7	0,2	360	694

TABLE 6.13 – Benchmarks sur le court terme : résultats LS-RL (statistiques sur les coûts).

données	*MO*	*SO*	*SC*	*DQ*	RL	gain *GC*	gain RL′	gain RL
A01	-	0	72227	3392822	0,021288	13,7 %	37,3 %	13,7 %
A02	-	0	72213	5383578	0,013414	26,5 %	50,1 %	26,5 %
A03	-	0	96672	5112877	0,018908	14,0 %	28,5 %	14,0 %
A04	-	0	231448	4299676	0,053829	21,7 %	51,1 %	21,7 %
A05	-	0	205122	3218972	0,063723	30,5 %	65,6 %	30,5 %
A06	-	0	191645	1647962	0,116292	26,1 %	33,5 %	26,1 %
A07	-	0	55130	2465328	0,022362	20,5 %	47,7 %	20,5 %
A08	-	0	346527	1397463	0,247969	29,5 %	51,9 %	29,5 %
A09	-	0	1055789	3906338	0,270276	28,0 %	54,5 %	28,0 %
A10	-	0	1881203	7387093	0,254661	26,9 %	49,8 %	26,9 %
A11	-	0	817071	16925051	0,048276	23,7 %	51,2 %	23,7 %
A12	-	0	116102	7513201	0,015453	13,5 %	33,4 %	13,5 %
A13	-	0	67251	3174802	0,021183	19,8 %	41,1 %	19,8 %
A14	-	0	56302	2317519	0,024294	14,8 %	40,7 %	14,8 %
A15	-	0	68054	4395307	0,015483	15,6 %	36,8 %	15,6 %
A16	-	0	423711	4807925	0,088128	28,0 %	49,7 %	28,0 %
A17	-	0	377020	5298991	0,071149	30,9 %	48,7 %	30,9 %
B01	-	49	204693	15950475	0,012833	3,9 %	16,3 %	9,1 %
B02	-	70	132788	4660904	0,028490	64,3 %	21,3 %	15,2 %
B03	-	*16	169280	15211761	0,011128	0,0 %	15,9 %	4,0 %
B04	-	*10	32526	2084416	0,015604	0,2 %	43,9 %	6,8 %
B05	-	*21	398876	5606842	0,071141	16,1 %	24,7 %	12,4 %
B06	-	10	27538	1301185	0,021164	5,2 %	66,3 %	35,1 %
B07	-	0	103538	5930657	0,017458	99,1 %	41,7 %	26,7 %
B08	-	1172	687755	9030922	0,076156	85,9 %	22,7 %	10,4 %
B09	-	* 7	431889	4937110	0,087478	82,9 %	41,3 %	22,1 %
B10	-	* 3	143157	2613767	0,054770	37,5 %	59,1 %	36,8 %
B11	-	3	365400	1550112	0,235725	52,7 %	62,6 %	40,4 %
B12	-	7	279055	1217253	0,229250	43,4 %	62,5 %	43,2 %
B13	-	*12	347743	1499034	0,231978	50,8 %	76,8 %	50,9 %
B14	-	*26	458784	1906122	0,240690	63,6 %	68,1 %	41,4 %
B15	-	1549	600530	4900575	0,122543	79,6 %	71,8 %	57,9 %
B16	-	841	555560	4406363	0,126081	80,6 %	60,8 %	42,1 %
B17	-	1508	572385	4583682	0,124875	82,8 %	75,6 %	62,6 %
B18	-	2648	511925	3639688	0,140651	72,0 %	68,4 %	55,8 %
B19	-	6472	7028	46400	0,151466	7,2 %	35,3 %	35,5 %
B20	-	374	356665	1595901	0,223488	69,9 %	66,7 %	47,5 %
B21	-	0	64315	1253926	0,051291	100,0 %	72,2 %	41,2 %
B22	-	0	8364065	1342321	6,231047	95,3 %	70,0 %	42,1 %
B23	-	0	62915	1175160	0,053537	100,0 %	75,4 %	46,1 %
B24	-	0	157870	1380915	0,114323	99,7 %	39,0 %	29,4 %
B25	-	0	6190560	1491728	4,149925	99,3 %	83,0 %	36,2 %
B26	-	137	31785	847726	0,037494	87,4 %	83,1 %	47,5 %
B27	-	0	555017	1530477	0,362643	100,0 %	34,2 %	28,9 %
B28	-	28846	344463	1986943	0,173363	37,7 %	-147,5 %	-122,0 %
B29	-	28831	338685	1986944	0,170455	37,7 %	-156,5 %	-126,5 %
B30	-	15354	154313	16925694	0,009117	79,3 %	37,5 %	27,5 %
B31	-	*47	5036543	45573555	0,110515	0,3 %	20,0 %	13,2 %
C01	3	611	650584	562753	1,156073	70,8 %	-22,8 %	-5,7 %
C02	3	82	264592	303151	0,872806	63,7 %	14,7 %	15,2 %
C03	0	0	37177	511392	0,072698	99,9 %	61,7 %	31,7 %
C04	0	*37	13535	199939	0,067696	1,1 %	83,8 %	17,6 %
C05	0	0	44035	594459	0,074075	99,7 %	80,3 %	39,9 %
C06	0	0	42185	556139	0,075854	99,8 %	78,3 %	24,2 %
C07	0	0	350576	348089	1,007144	95,0 %	41,9 %	41,2 %
C08	0	2383	267431	350192	0,763669	54,6 %	45,2 %	44,2 %
C09	0	33	1275703	3418723	0,373152	96,4 %	49,3 %	29,4 %
C10	0	147	1989237	5155110	0,385877	96,3 %	54,1 %	35,8 %
C11	1	340	52607	3513443	0,014973	96,7 %	41,5 %	18,7 %
C12	89	0	3009688	2104364	1,430213	6,5 %	10,5 %	9,7 %
C13	1	9	420256	7116669	0,059052	98,1 %	-8,3 %	4,0 %

TABLE 6.14 – Benchmarks sur le court terme : résultats LS-RL′ (statistiques sur les coûts).

données	nb tournee	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
A01	128	599	2,7	1,5	1,2	0,1	171	286
A02	103	709	4,9	2,6	2,3	0,4	203	457
A03	117	799	4,8	2,6	2,2	0,0	246	413
A04	46	474	8,3	4,4	3,9	0,1	557	694
A05	46	391	6,5	3,8	2,7	0,1	460	670
A06	101	521	3,2	2,4	0,8	0,1	847	797
A07	88	465	3,3	1,9	1,4	0,0	189	363
A08	19	212	9,2	4,6	4,5	0,1	745	909
A09	52	549	8,6	4,6	4,0	0,3	835	1116
A10	91	1083	9,9	5,2	4,7	0,5	991	1457
A11	160	1966	10,3	5,4	4,9	0,1	506	766
A12	101	921	7,1	3,7	3,5	0,0	341	559
A13	70	552	5,9	3,7	2,2	0,0	405	556
A14	82	422	3,1	1,7	1,5	0,0	210	356
A15	151	816	3,4	1,8	1,6	0,0	130	281
A16	61	618	8,1	4,6	3,5	0,5	775	1228
A17	58	671	9,6	5,2	4,3	0,4	684	1134
B01	357	2310	4,5	2,4	2,1	0,7	161	651
B02	136	931	4,8	2,7	2,1	0,6	293	603
B03	110	1987	16,1	8,3	7,8	0,0	587	935
B04	65	351	3,4	1,8	1,6	0,0	146	278
B05	84	748	6,9	3,8	3,1	0,3	485	869
B06	40	223	3,6	2,0	1,6	0,0	208	374
B07	69	942	11,7	6,4	5,2	0,2	430	867
B08	225	1512	4,7	2,5	2,2	0,0	240	366
B09	65	651	8,0	4,5	3,5	0,6	732	1225
B10	68	406	4,0	2,4	1,5	0,1	295	416
B11	24	245	8,2	4,3	3,9	0,2	579	815
B12	21	199	7,5	3,9	3,6	0,0	496	659
B13	22	236	8,7	4,6	4,1	0,1	601	833
B14	28	289	8,3	4,4	3,9	0,4	617	1036
B15	82	808	7,9	4,4	3,4	0,3	662	961
B16	94	768	6,2	3,5	2,7	0,2	530	741
B17	84	763	7,1	4,0	3,1	0,2	623	834
B18	72	638	6,9	3,9	3,0	0,3	655	891
B19	16	67	2,2	1,3	0,9	0,1	165	356
B20	59	1312	20,2	10,1	10,1	0,3	335	1008
B21	14	178	10,7	5,8	4,9	0,1	340	671
B22	10	182	16,2	8,6	7,6	0,4	513	1073
B23	13	164	10,6	5,7	4,9	0,2	351	657
B24	11	177	14,1	7,4	6,7	0,5	560	1149
B25	19	230	10,1	5,7	4,4	0,5	322	1051
B26	7	112	14,0	7,6	6,4	0,0	280	621
B27	74	436	3,9	2,9	1,0	0,9	360	2026
B28	105	582	3,5	3,5	0,0	0,2	486	666
B29	99	576	3,8	3,7	0,1	0,2	494	795
B30	317	1630	3,1	2,0	1,1	0,0	215	516
B31	479	5110	8,7	5,3	3,4	0,1	414	507
C01	27	165	4,1	2,8	1,3	0,8	714	1613
C02	26	101	1,9	1,2	0,7	0,1	282	531
C03	28	118	2,2	1,4	0,9	0,3	388	877
C04	9	39	2,2	1,4	0,8	0,3	445	1012
C05	21	121	3,3	2,1	1,2	0,6	614	1413
C06	21	103	2,9	1,8	1,1	0,6	592	1246
C07	65	356	3,5	1,7	1,7	0,1	378	634
C08	42	257	4,1	2,1	2,0	0,3	422	826
C09	99	579	3,8	2,3	1,5	0,2	255	722
C10	149	889	4,0	2,4	1,6	0,1	273	697
C11	106	665	4,3	3,0	1,3	0,2	401	807
C12	255	1375	3,4	2,2	1,2	0,0	138	699
C13	225	1094	2,9	1,7	1,2	0,0	165	293
average	89	695	6,6	3,7	2,9	0,2	435	785

TABLE 6.15 – Benchmarks sur le court terme : résultats LS-RL′ (statistiques sur les tournées).

données	tentat.	acceptations		améliorations	
A01	11,175 M	483840	4,3 %	497	0,4 h%
A02	5,726 M	373977	6,5 %	1035	1,8 h%
A03	5,689 M	305816	5,4 %	923	1,6 h%
A04	6,786 M	247082	3,6 %	983	1,4 h%
A05	12,877 M	436890	3,4 %	1001	0,8 h%
A06	16,167 M	637190	3,9 %	787	0,5 h%
A07	8,552 M	540443	6,3 %	706	0,8 h%
A08	7,898 M	394354	5,0 %	593	0,8 h%
A09	8,912 M	301540	3,4 %	845	0,9 h%
A10	7,909 M	203519	2,6 %	1467	1,9 h%
A11	4,896 M	141962	2,9 %	2067	4,2 h%
A12	4,867 M	322731	6,6 %	1068	2,2 h%
A13	8,916 M	376197	4,2 %	821	0,9 h%
A14	10,698 M	705213	6,6 %	516	0,5 h%
A15	5,057 M	332914	6,6 %	703	1,4 h%
A16	10,776 M	347633	3,2 %	917	0,9 h%
A17	9,237 M	260568	2,8 %	1002	1,1 h%
B01	4,118 M	42669	1,0 %	1236	3,0 h%
B02	5,190 M	379446	7,3 %	1288	2,5 h%
B03	1,972 M	42669	2,2 %	570	2,9 h%
B04	12,925 M	204149	1,6 %	447	0,3 h%
B05	6,461 M	631576	9,8 %	2024	3,1 h%
B06	25,769 M	304123	1,2 %	325	0,1 h%
B07	3,992 M	175528	4,4 %	2547	6,4 h%
B08	4,822 M	294494	6,1 %	2044	4,2 h%
B09	7,446 M	715529	9,6 %	1649	2,2 h%
B10	46,834 M	384442	0,8 %	655	0,1 h%
B11	21,861 M	144446	0,7 %	461	0,2 h%
B12	21,920 M	196947	0,9 %	694	0,3 h%
B13	23,449 M	183145	0,8 %	516	0,2 h%
B14	5,976 M	959724	16,1 %	906	1,5 h%
B15	9,509 M	695514	7,3 %	2178	2,3 h%
B16	8,041 M	651256	8,1 %	1467	1,8 h%
B17	9,738 M	782531	8,0 %	2048	2,1 h%
B18	13,478 M	772540	5,7 %	1665	1,2 h%
B19	19,563 M	1859210	9,5 %	188	0,1 h%
B20	3,024 M	411119	13,6 %	1136	3,8 h%
B21	10,806 M	470509	4,4 %	645	0,6 h%
B22	10,262 M	414098	4,0 %	566	0,6 h%
B23	10,359 M	449538	4,3 %	718	0,7 h%
B24	10,229 M	481571	4,7 %	628	0,6 h%
B25	11,157 M	1349731	12,1 %	887	0,8 h%
B26	9,502 M	1235342	13,0 %	311	0,3 h%
B27	33,932 M	740801	2,2 %	839	0,2 h%
B28	14,570 M	924970	6,3 %	2449	1,7 h%
B29	11,249 M	677473	6,0 %	2940	2,6 h%
B30	12,115 M	607763	5,0 %	5337	4,4 h%
B31	1,952 M	22282	1,1 %	2594	13,3 h%
C01	21,673 M	780970	3,6 %	539	0,2 h%
C02	26,518 M	2010675	7,6 %	337	0,1 h%
C03	44,361 M	1405911	3,2 %	221	0,1 h%
C04	298,345 M	919179	0,3 %	49	0,0 h%
C05	29,659 M	1593056	5,4 %	268	0,1 h%
C06	27,087 M	1175693	4,3 %	237	0,1 h%
C07	15,622 M	955990	6,1 %	647	0,4 h%
C08	24,908 M	1966737	7,9 %	533	0,2 h%
C09	12,472 M	969246	7,8 %	1069	0,9 h%
C10	8,345 M	652189	7,8 %	2113	2,5 h%
C11	9,913 M	758558	7,7 %	1782	1,8 h%
C12	21,907 M	682664	3,1 %	607	0,3 h%
C13	5,876 M	582395	9,9 %	3855	6,6 h%
average*	13,112 M	619185	5,5 %	1168	1,7 h%

données	tentat.	accept.		amélio.	
A01	9,704 M	372849	3,8 %	506	0,5 h%
A02	5,288 M	323798	6,1 %	1292	2,4 h%
A03	4,564 M	228900	5,0 %	716	1,6 h%
A04	6,914 M	185590	2,7 %	710	1,0 h%
A05	11,190 M	313239	2,8 %	1315	1,2 h%
A06	15,184 M	488048	3,2 %	814	0,5 h%
A07	9,844 M	536327	5,4 %	690	0,7 h%
A08	8,437 M	439519	5,2 %	389	0,5 h%
A09	8,126 M	263955	3,2 %	773	1,0 h%
A10	7,001 M	167872	2,4 %	1438	2,1 h%
A11	4,287 M	117333	2,7 %	2218	5,2 h%
A12	3,706 M	251338	6,8 %	1238	3,3 h%
A13	8,096 M	271741	3,4 %	877	1,1 h%
A14	7,758 M	452062	5,8 %	434	0,6 h%
A15	4,324 M	278738	6,4 %	668	1,5 h%
A16	10,093 M	260081	2,6 %	1043	1,0 h%
A17	8,599 M	210142	2,4 %	968	1,1 h%
B01	4,316 M	43458	1,0 %	934	2,2 h%
B02	5,252 M	380335	7,2 %	2241	4,3 h%
B03	2,127 M	48612	2,3 %	332	1,6 h%
B04	13,186 M	213735	1,6 %	305	0,2 h%
B05	6,522 M	645239	9,9 %	2289	3,5 h%
B06	24,905 M	319034	1,3 %	371	0,1 h%
B07	3,833 M	153680	4,0 %	2905	7,6 h%
B08	4,357 M	272632	6,3 %	2033	4,7 h%
B09	7,179 M	687065	9,6 %	1866	2,6 h%
B10	46,760 M	323324	0,7 %	650	0,1 h%
B11	21,805 M	138727	0,6 %	307	0,1 h%
B12	22,693 M	192329	0,8 %	444	0,2 h%
B13	23,179 M	155999	0,7 %	447	0,2 h%
B14	6,190 M	967508	15,6 %	1044	1,7 h%
B15	10,292 M	712879	6,9 %	2039	2,0 h%
B16	8,429 M	678137	8,0 %	1484	1,8 h%
B17	9,777 M	754940	7,7 %	2377	2,4 h%
B18	13,651 M	744369	5,5 %	1828	1,3 h%
B19	19,429 M	1874974	9,7 %	217	0,1 h%
B20	3,189 M	429622	13,5 %	1003	3,1 h%
B21	12,828 M	486984	3,8 %	552	0,4 h%
B22	9,598 M	360746	3,8 %	557	0,6 h%
B23	12,057 M	518152	4,3 %	519	0,4 h%
B24	9,442 M	367871	3,9 %	725	0,8 h%
B25	9,447 M	1341553	14,2 %	1120	1,2 h%
B26	9,398 M	1247208	13,3 %	272	0,3 h%
B27	31,473 M	499668	1,6 %	983	0,3 h%
B28	15,499 M	917257	5,9 %	2477	1,6 h%
B29	12,007 M	657253	5,5 %	3187	2,7 h%
B30	11,793 M	552206	4,7 %	5751	4,9 h%
B31	1,879 M	17957	1,0 %	2297	12,2 h%
C01	22,068 M	785592	3,6 %	525	0,2 h%
C02	31,398 M	2027068	6,5 %	323	0,1 h%
C03	56,408 M	1390099	2,5 %	181	0,0 h%
C04	264,322 M	887057	0,3 %	43	0,0 h%
C05	28,102 M	1127520	4,0 %	291	0,1 h%
C06	27,010 M	1539526	5,7 %	186	0,1 h%
C07	14,234 M	861659	6,1 %	690	0,5 h%
C08	24,957 M	1965849	7,9 %	467	0,2 h%
C09	11,885 M	883356	7,4 %	1203	1,0 h%
C10	7,653 M	592909	7,7 %	2172	2,8 h%
C11	9,839 M	681057	6,9 %	1990	2,0 h%
C12	20,623 M	623018	3,0 %	644	0,3 h%
C13	5,699 M	564609	9,9 %	4320	7,6 h%
average*	13,091 M	581787	5,3 %	1211	1,8 h%

TABLE 6.16 – Benchmarks sur le court terme : statistiques sur les transformations pour LS-RL (gauche) et LS-RL′ (droite) M = million, h% = un centième pour cent.

données	moy *DQ* greedy	moy *DQ* LS-RL	moy *DQ* LS-RL′	QLiv. moy greedy	QLiv. moy LS-RL	QLiv. moy LS-RL′
A	3031400	4565518	4861465	15992	16066	17073
A+B+C	2897779	4398780	4517178	16770	13139	13718

TABLE 6.17 – Benchmarks sur le court terme : statistiques sur les volumes.

données	*SO*	*SC*	*DQ*	RL	nb tournees	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
L1	652	406443	3767868	0,107871	189	503	2,7	1,6	1,0	0,6	640	1320
L2	146	407379	3827560	0,106433	196	506	2,6	1,6	1,0	0,5	619	1235
L3	86	1092976	31989357	0,034167	790	3584	4,5	2,7	1,8	0,2	366	954
L4	257	808887	18433289	0,043882	590	2249	3,8	2,4	1,4	0,2	395	844
L5	85	145339	8830708	0,016458	295	1020	3,5	1,9	1,5	1,2	598	1760
average	245	572205	13369756	*0,042798	412	1572	3,4	2,0	1,3	0,5	524	1223

TABLE 6.18 – Benchmarks sur le long terme : résultats de l'algorithme glouton.

données	*SO*	*SC*	*DQ*	RL	nb tournees	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
L1	0	340767	3840502	0,088730	137	590	4,3	3,0	1,3	0,8	721	1618
L2	0	335661	3899780	0,086072	148	570	3,9	2,7	1,2	0,7	660	1445
L3	0	1019292	32079238	0,031774	839	3570	4,3	2,6	1,7	0,2	317	873
L4	17	697009	18694845	0,037283	605	2400	4,0	2,6	1,4	0,2	321	735
L5	0	106326	9475562	0,011221	110	1324	12,0	7,8	4,3	3,2	1256	4286
average	3	499811	13597985	*0,036756	368	1691	5,7	3,7	2,0	1,0	655	1792

TABLE 6.19 – Benchmarks sur le long terme : résultats LS-RL.

données	*SO*	*SC*	*DQ*	RL	nb tournees	nb oper	oper. moy	liv. moy	charg. moy	pauses moy	dist. moy	dur. moy
L1	0	321449	4045989	0,079449	148	542	3,7	2,4	1,3	0,7	632	1403
L2	0	321207	4016621	0,079969	140	541	3,9	2,6	1,3	0,7	669	1471
L3	0	1012191	32320180	0,031318	807	3583	4,4	2,7	1,8	0,2	327	890
L4	0	701139	18587949	0,037720	602	2396	4,0	2,6	1,4	0,2	325	744
L5	0	101913	9630979	0,010582	138	1352	9,8	6,3	3,5	2,2	945	3159
average	0	491580	13720344	*0,035829	367	1683	5,2	3,3	1,9	0,8	580	1533

TABLE 6.20 – Benchmarks sur le long terme : résultats LS-RL′.

6.4 Annexe 4 : Liste complète des mouvements de l'IRP

$\mathcal{T}_{MO}$	$\mathcal{T}_{SO}$
OperationDeletionBackwardBlockPropag	OperationDeletionBackwardBlockPropag
OperationDeletionForwardBlockPropag	OperationDeletionForwardBlockPropag
OperationInsertionOrderBackwardPropag	OperationInsertionCustomerRunoutBackwardBlockPropag
OperationInsertionOrderForwardPropag	OperationInsertionCustomerRunoutForwardBlockPropag
OperationInsertionSourceOrderBackwardPropag	OperationInsertionSourceCustomerRunoutBackwardBlockPropag
OperationInsertionSourceOrderForwardPropag	OperationInsertionSourceCustomerRunoutForwardBlockPropag
OperationEjectionCustomerNearBackwardBlockPropag	OperationEjectionCustomerNearBackwardBlockPropag
OperationEjectionCustomerNearForwardBlockPropag	OperationEjectionCustomerNearForwardBlockPropag
OperationEjectionSourceNearBackwardBlockPropag	OperationEjectionSourceNearBackwardBlockPropag
OperationEjectionSourceNearForwardBlockPropag	OperationEjectionSourceNearForwardBlockPropag
OperationEjectionOrderBackwardBlockPropag	OperationEjectionRunoutBackwardBlockPropag
OperationEjectionOrderForwardBlockPropag	OperationEjectionRunoutForwardBlockPropag
OperationMoveBetweenShiftsBackwardBackwardBlockPropag	OperationMoveBetweenShiftsBackwardBackwardBlockPropag
OperationMoveBetweenShiftsBackwardForwardBlockPropag	OperationMoveBetweenShiftsBackwardForwardBlockPropag
OperationMoveBetweenShiftsForwardBackwardBlockPropag	OperationMoveBetweenShiftsForwardBackwardBlockPropag
OperationMoveBetweenShiftsForwardForwardBlockPropag	OperationMoveBetweenShiftsForwardForwardBlockPropag
OperationMoveInsideShiftBeforeBackwardBlockPropag	OperationMoveInsideShiftBeforeBackwardBlockPropag
OperationMoveInsideShiftBeforeForwardBlockPropag	OperationMoveInsideShiftBeforeForwardBlockPropag
OperationMoveInsideShiftAfterBackwardBlockPropag	OperationMoveInsideShiftAfterBackwardBlockPropag
OperationMoveInsideShiftAfterForwardBlockPropag	OperationMoveInsideShiftAfterForwardBlockPropag
OperationSwapBetweenShiftsBackwardBackwardBlockPropag	OperationSwapBetweenShiftsBackwardBackwardBlockPropag
OperationSwapBetweenShiftsBackwardForwardBlockPropag	OperationSwapBetweenShiftsBackwardForwardBlockPropag
OperationSwapBetweenShiftsForwardBackwardBlockPropag	OperationSwapBetweenShiftsForwardBackwardBlockPropag
OperationSwapBetweenShiftsForwardForwardBlockPropag	OperationSwapBetweenShiftsForwardForwardBlockPropag
OperationSwapInsideShiftBackwardBlockPropag	OperationSwapInsideShiftBackwardBlockPropag
OperationSwapInsideShiftForwardBlockPropag	OperationSwapInsideShiftForwardBlockPropag
OperationMirrorInsideShiftBackwardBlockPropag	OperationMirrorInsideShiftBackwardBlockPropag
OperationMirrorInsideShiftForwardBlockPropag	OperationMirrorInsideShiftForwardBlockPropag
ShiftSlidingBackward	ShiftSlidingBackward
ShiftSlidingForward	ShiftSlidingForward
ShiftSlidingOrderBackward	ShiftSlidingRunoutBackward
ShiftSlidingOrderForward	ShiftSlidingRunoutForward
ShiftSlidingUnsatOrderBackward	ShiftSlidingFirstRunoutBackward
ShiftSlidingUnsatOrderForward	ShiftSlidingFirstRunoutForward
ShiftResourcesChangingBackward	ShiftResourcesChangingBackward
ShiftResourcesChangingForward	ShiftResourcesChangingForward
ShiftDeletion	ShiftDeletion
ShiftInsertionOrderBackwardPropag	ShiftInsertionCustomerFirstRunoutBackwardPropag
ShiftInsertionOrderForwardPropag	ShiftInsertionCustomerFirstRunoutForwardPropag
ShiftInsertionSourceOrderBackwardPropag	ShiftInsertionSourceCustomerRunoutBackwardPropag
ShiftInsertionSourceOrderForwardPropag	ShiftInsertionSourceCustomerRunoutForwardPropag
ShiftMoveBackward	ShiftInsertionSourceCustomerFirstRunoutBackwardPropag
ShiftMoveForward	ShiftInsertionSourceCustomerFirstRunoutForwardPropag
ShiftSwapBackwardBackward	ShiftMoveBackward
ShiftSwapBackwardForward	ShiftMoveForward
ShiftSwapForwardBackward	ShiftSwapBackwardBackward
ShiftSwapForwardForward	ShiftSwapBackwardForward
	ShiftSwapForwardBackward
	ShiftSwapForwardForward

TABLE 6.21 – L'ensemble des transformations pour les étapes $\mathcal{T}_{MO}$ et $\mathcal{T}_{SO}$.

La première option, utilisée pour élargir le voisinage induit par certaines transformations, est notée en utilisant le suffixe "Block". La deuxième option, utilisée pour spécialiser certaines transformations suivant un objectif, est notée avec les suffixes "Order" (spécialisées pour les commandes), "Runout" (spécialisées pour les assèchements), ou "Near" (à proximité). La troisième option, utilisée pour fixer la direction utilisée au cours de la replanification des tournées, est notée en utilisant les suffixes "Backward" (en arrière) ou "Forward" (en avant). Enfin, la quatrième option, utilisée pour faciliter la propagation du flot pendant la réaffectation des volume, est notée avec le suffixe "Propag".

$\mathcal{T}_{\mathrm{RL}}$	
OperationDeletionBackward	OperationMoveInsideShiftBeforeBackward
OperationDeletionForward	OperationMoveInsideShiftBeforeForward
OperationDeletionBackwardBlock	OperationMoveInsideShiftAfterBackward
OperationDeletionForwardBlock	OperationMoveInsideShiftAfterForward
OperationInsertionCustomerBackward	OperationMoveInsideShiftBeforeBackwardBlock
OperationInsertionCustomerForward	OperationMoveInsideShiftBeforeForwardBlock
OperationInsertionSourceBackward	OperationMoveInsideShiftAfterBackwardBlock
OperationInsertionSourceForward	OperationMoveInsideShiftAfterForwardBlock
OperationInsertionSourceCustomerBackward	OperationSwapBetweenShiftsBackwardBackward
OperationInsertionSourceCustomerForward	OperationSwapBetweenShiftsBackwardForward
OperationInsertionSourceCustomerNearBackward	OperationSwapBetweenShiftsForwardBackward
OperationInsertionSourceCustomerNearForward	OperationSwapBetweenShiftsForwardForward
OperationEjectionCustomerBackward	OperationSwapBetweenShiftsBackwardBackwardBlock
OperationEjectionCustomerForward	OperationSwapBetweenShiftsBackwardForwardBlock
OperationEjectionCustomerNearBackward	OperationSwapBetweenShiftsForwardBackwardBlock
OperationEjectionCustomerNearForward	OperationSwapBetweenShiftsForwardForwardBlock
OperationEjectionSourceBackward	OperationSwapInsideShiftBackward
OperationEjectionSourceForward	OperationSwapInsideShiftForward
OperationEjectionSourceNearBackward	OperationSwapInsideShiftBackwardBlock
OperationEjectionSourceNearForward	OperationSwapInsideShiftForwardBlock
OperationEjectionCustomerBackwardBlock	OperationMirrorInsideShiftBackwardBlock
OperationEjectionCustomerForwardBlock	OperationMirrorInsideShiftForwardBlock
OperationEjectionCustomerNearBackwardBlock	ShiftSlidingBackward
OperationEjectionCustomerNearForwardBlock	ShiftSlidingForward
OperationEjectionSourceBackwardBlock	ShiftResourcesChangingBackward
OperationEjectionSourceForwardBlock	ShiftResourcesChangingForward
OperationEjectionSourceNearBackwardBlock	ShiftDeletion
OperationEjectionSourceNearForwardBlock	ShiftInsertionSourceCustomerBackward
OperationMoveBetweenShiftsBackwardBackward	ShiftInsertionSourceCustomerForward
OperationMoveBetweenShiftsBackwardForward	ShiftMoveBackward
OperationMoveBetweenShiftsForwardBackward	ShiftMoveForward
OperationMoveBetweenShiftsForwardForward	ShiftSwapBackwardBackward
OperationMoveBetweenShiftsBackwardBackwardBlock	ShiftSwapBackwardForward
OperationMoveBetweenShiftsBackwardForwardBlock	ShiftSwapForwardBackward
OperationMoveBetweenShiftsForwardBackwardBlock	ShiftSwapForwardForward
OperationMoveBetweenShiftsForwardForwardBlock	

TABLE 6.22 – L'ensemble des transformations de la phase $\mathcal{T}_{\mathrm{RL}}$.

Bibliographie

[1] E. Aarts, J.K. Lenstra. 1997. Local Search in Combinatorial Optimization. *Wiley-Interscience Series in Discrete Mathematics and Optimization*, John Wiley and Sons, Chichester, England, UK.

[2] R. Acuna-Agost, M. Boudia, S. Gabteni, N. Jozefowiez, C. Mancel, F. Mora-Camino. 2010. Méthodes de recherche opérationnelle pour la gestion des perturbations dans le domaine aérien. *ROADEF 2010*. Toulouse.

[3] C. Aimé, B. Bequet. 2009. Méthodologie des Terassements 1, 2, 3. Cours de l'ESTP.

[4] A.E. Akay. 2006. Minimizing total costs of forest roads with computer-aided design model. *In Proceedings of the 1997 Winter Simulation Conference*, Sadhana Vol. 31, Part 5, October 2006, pp. 621-633.

[5] A. Alshibani. 2008. Optimizing and Controlling Earthmoving Operations Using Spatial Technologies. *Phd Thesis*, Concordia University (Canada), 259 pages, AAT NR45646.

[6] C. Archetti, M. Savelsbergh. 2007. The truckload trip scheduling problem. *TRISTAN VI, the 6th Triennial Symposium on Transportation Analysis*, Phuket Island, Thailand.

[7] W.H. Askew, S.H. Al-jibouri, M.J. Mawdesley, D.E. Patterson. 2002. Planning linear construction projects : automated method for the generation of earthwork activities *Automation in construction 11 (6)*, pp. 643-653.

[8] J.F. Bard, F. Huang, M. Dror, P. Jaillet. 1998. A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions 30(9)*, pp. 831-834.

[9] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M. Savelsbergh et P.H. Vance. 1998. Branch-and-Price : Column Generation for Solving Huge Integer Programs. *Operations Research 46 (3)*, pp. 316-329.

[10] W. Bell, L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, P. Prutzman. 1983. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces 13 (6)*, pp. 4-23.

[11] J.F. Benders. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math. 4 (3)*, pp. 238-252.

[12] T. Benoist, B. Estellon, F. Gardi, A. Jeanjean (2010). Recherche locale pour un problème d'optimisation de tournées de véhicules avec gestion des stocks. In *MOSIM 2010*, Hammamet, Tunisia, Mai 2010.

[13] T. Benoist, B. Estellon, F. Gardi, A. Jeanjean. 2009. High-Performance Local Search for Solving Real-Life Inventory Routing Problems. In *SLS 2009*, Brussels.

[14] T. Benoist, A. Jeanjean, G. Rochart, H. Cambazard, E. Grellier, N. Jussien. 2006. Subcontractors scheduling on residential buildings construction sites. In *ISS 2006*, the 3rd International Scheduling Symposium (sous la direction de T. Ibaraki), Technical Report JSME-06-203, pp. 32-37, Japan Society of Mechanical Engineers.

[15] T. Benoist, A. Jeanjean, P. Molin. 2007. Affectation du matériel de coffrage sur des chantiers de construction. In *Roadef'07*, Grenoble.

[16] T. Benoist. 2007. Towards optimal formwork pairing on construction sites. *In RAIRO Operations Research 41 (4)*, pp. 381-398.

[17] T. Benoist, A. Jeanjean. 2008. Etude du comportement d'annulation d'une contre-proposition d'un spot publicitaire. In Roadef'08, Clermont Ferrand.

[18] T. Benoist, A. Jeanjean, P. Molin. 2008. Minimum Formwork Stock Problem on residential buildings construction sites. *4OR* Volume 7, Number 3 / octobre 2009).

[19] T. Benoist, B. Estellon, F. Gardi, A. Jeanjean. 2009. High-performance local search for solving inventory routing problems. H. Hoos T. Stützle, M. Birattari, ed., *SLS 2009, the 2nd International Workshop on Engineering Stochastic Local Search Algorithms*, *Lecture Notes in Computer Science 5752*, pp. 105–109.

[20] T. Benoist, B. Estellon, F. Gardi, A. Jeanjean. 2009. Recherche locale haute performance pour l'optimisation des livraisons de gaz industriels par camion-citerne. *In Actes de ROADEF 2009, le 10ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, Paris, France.

[21] T. Benoist, B. Estellon, F. Gardi, A. Jeanjean. 2010. *Stochastic local search for real-life inventory routing*. To be published. *Transportation Sciences*

[22] T. Benoist, B. Estellon, F. Gardi, A. Jeanjean. 2010. Recherche locale pour un problème d'optimisation de tournées de véhicules avec gestion des stocks. *In MOSIM 2010*, Hammamet, Tunisia, Mai 2010.

[23] T. Benoist, B. Estellon, F. Gardi, S. Jain, A. Jeanjean,E. Patay. 2010. Inventory routing optimization for bulk gas transportation. *In In INFORMS 2009*, Annual Meeting. San Diego, US-CA.

[24] T. Benoist, F. Gardi, A. Jeanjean. 2011. Lessons learned from 15 years of operations research for French TV channel TF1. In *Submitted.*

[25] T. Benoist, A. Jeanjean, V. Jost. 2011. Le problème du voyageur de commerce unidimensionnel avec précédences. In *Roadef'2011*, Saint Etienne.

[26] A. Brunner. 2001. Zürich Airport Extension Project : Digital Support for Earthwork Construction *Springer-Verlag Berlin Heidelberg C.Y. Westort* (Ed.) : DEM 2001, LNCS 2181, pp. 1.

[27] A.Campbell, L. Clarke, A. Kleywegt, M. Savelsbergh. 1998. The inventory routing problem. *Fleet Management and Logistics*. T. Crainic, G. Laporte, eds., Kluwer Academic Publishers, Norwell, MA, pp. 95-113.

[28] A. Campbell, L. Clarke, M. Savelsbergh. 2002. Inventory routing in practice. P. Toth, D. Viego, eds., *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications 9, SIAM, Philadelphia, PA, 309–330.

[29] A. Campbell, M. Savelsbergh. 2004. A decomposition approach for the inventory-routing problem. *Transportation Sciences 38 (4)*, pp. 488-502.

[30] A. Campbell, M. Savelsbergh. 2004. Delivery volume optimization. *Transportation Sciences 38 (2)*, pp. 210-223.

[31] A. Campbell, M. Savelsbergh. 2004. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Sciences 38 (3)*, pp. 369-378.

[32] V. Chvátal. 1983. Linear Programming. In *W.H. Freeman & Company*, 478 pages.

[33] T. Cormen, C. Leiserson, R. Rivest, C. Stein. 2004. Introduction à l'Algorithmique. *Dunod*, Paris, France, French 2nd edition.

[34] F. Cornillier, F. F. Boctor, G. Laporte, J. Renaud. 2006. Heuristique pour le problème d'approvisionnement des stations d'essence sur plusieurs périodes. *Congrès conjoint SCRO / Journées de l'optimisation*. Montréal.

[35] G.B. Dantzig. 1949. Application of the simplex method to a transportation problem *In Activity Analysis of Production and Allocation*, Conference on Linear Programming, Chicago, Illinois, Tj.C. Koopmans.

[36] G.B. Dantzig, P. Wolfe. 1960. Decomposition Principle for Linear Programs. *Operations Research 8*, pp. 101-111.

[37] M.G. De La Banda, P.J. Stuckey. 2007. Dynamic Programming to Minimize the Maximum Number of Open Stacks. *INFORMS Journal on Computing 19(4)*, pp. 607-617.

[38] M. Desrochers, F.M. Solomon. 1989. A column generation approach to the urban transit crew scheduling problem. *Transportation Sciences 23*, pp. 1-13.

[39] M. Dorigo. 1992. Optimization, Learning and Natural Algorithms. *Phd Thesis*, Politecnico di Milano, Italie.

[40] M. Dror. 1994. Note on complexity of shortest path models for column generation in the vrptw. *Operations Research 42*, pp. 977-978.

[41] S.M. Easa. 1988. Earthwork allocation with linear unit costs. *Journal of Construction Engineering and Management 114 (2)* pp. 641-655.

[42] S.M. Ervin. 2001. Designed Landforms. *Digital Earth Moving Lecture Notes in Computer Science 2181 (2)*.

[43] B. Estellon, F. Gardi, K. Nouioua. 2006. Large neighborhood improvements for solving car sequencing problems. *RAIRO Operations Research 40 (4)*, pp. 355-379.

[44] B. Estellon, F. Gardi, K. Nouioua. 2006. Two local search approaches for solving real-life car sequencing problems. *European Journal of Operations Research 191 (3)*, pp. 928-944.

[45] B. Estellon, F. Gardi, K. Nouioua. 2009. High-performance local search for task scheduling with human resource allocation. H. Hoos T. Stützle, M. Birattari, ed., *SLS 2009, the 2nd International Workshop on Engineering Stochastic Local Search Algorithms*, *Lecture Notes in Computer Science 5752*, pp. 1-15. Springer, Berlin, Germany.

[46] D. Feillet. 2010. A tutorial on column generation and brand-and-price for vehicule routing problems. *4OR 8 (4)*.

[47] T. Feo, G. Resende. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization 6 (2)*, pp. 109-133.

[48] V. Gabrel, A. Knippel, M. Minoux. 1999. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters 25*, pp. 15-23.

[49] F. Gardi, K. Nouioua. 2011. Local search for mixed-integer nonlinear optimization : a methodology and an application. In *Proceedings of EvoCOP 2011, the 11th European Conference on Evolutionary Computation in Combinatorial Optimisation* (P. Merz, J.-K. Hao, eds.), *Lecture Notes in Computer Science 6622*, pp. 167-178. Springer, Berlin, Germany.

[50] M.R. Garey, D.S. Johnson, L. Stockmeyer. 1974. Some simplified NP-complete problems. *Proceedings of the sixth annual ACM symposium on Theory of computing*, pp. 47-63.

[51] M. Gendreau. 2003. An introduction to tabu search. *In Handbook of metaheuristics.* Glover F, Kochenberger GA, editors, pp. 37-54.

[52] A.M. Geoffrion. 1974. Lagrangean relaxation for integer programming. *Math. Programming Stud, 2*, pp. 82-114.

[53] F. Glover. 1990. Tabu Search : A Tutorial *Interfaces 20(4)*, pp. 74-94.

[54] S. Godskesen, T. Sejr, J. N. Kjeldsen, R. Larsen. 2010. Solving a real-life large-scale energy management problem. *Cornell University Librairy*, arxiv.org 2010.

[55] D.E. Goldberg. 1989. Genetic Algorithms in Search. *Optimization and Machine Learning.*Kluwer Academic Publishers, Boston, MA.

[56] T.F. Gonzalès. 2007. Handbook of Approximation Algorithms and Metaheuristics. *Chapman Hall/CRC.* 1432 pages.

[57] C. Guéret, C. Prins, M. Sevaux. 2000. Programmation linéaire - 65 problèmes d'optimisation modélisés et résolus avec Visual Xpress. Editions Eyrolles.

[58] P. Hansen, N. Mladenovic. 2006. First vs. best improvement : An empirical study *Journal Discrete Applied Mathematics 154 (5)*, Special issue : IV ALIO/EURO.

[59] M. Held, R.M. Karp. 1961. A dynamic programming approach to sequencing problems. In *Proceedings of the 16th ACM national meeting*, pp. 71.201-71.204.

[60] K. Helsgaun. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research 126 (1)*, pp. 106-130.

[61] D. Hendersona D. Vaughanb S. Jacobson R. Wakefieldd E. Sewelle. 2003. Solving the shortest route cut and fill problem using simulated annealing. *European Journal of Operational Research 145 (1)*, pp. 72-84.

[62] I.F.P.. 2007. L'IFP et le développement durable. *http ://www.ifpenergiesnouvelles.fr.*

[63] A. Jeanjean. 2004. Optimizing Transportation Decisions in a Manufacturing Supply Chain. *Master Thesis* - University of Oklahoma (USA).

[64] A. Jeanjean, B. Martin. 2009. Optimisation de placement de publicité internet par l'algorithme des bandits manchots. In *Roadef'09*, Nancy.

[65] A. Jeanjean. 2009. Optimisation de mouvements de terre sur des chantiers linéaires de terrassement. In *Digiteo Optimeo*, Supélec, Paris.

[66] A. Jeanjean, T. Hanin. 2010. Analyse et Classification d'évènements d'actualité pour l'amélioration de méthodes de prévision. In *V.S.S.T.'2010*, Colloque international de Veille Stratégique Scientifique et Technologique, Toulouse, Octobre 2010.

[67] A. Jeanjean. 2010a. Resource scheduling optimization in mass transportation problems. In *PMS 2010*, University of Tours, April 2010.

[68] A. Jeanjean. 2010b. Optimisation de mouvements de terre sur des chantiers linéaires de terrassement. In *Roadef'2010*, Toulouse.

[69] A. Jeanjean, B. Martin. 2010. Prévisions d'audiences et optimisation de plannings de publicités internet. In *Roadef'2010*, Toulouse.

[70] A. Jeanjean, J. Masson. 2011. Optimisation des prix et prévisions des fréquentations des hôtels du Groupe Louvre Hôtels. In *Roadef'2011*, Saint Etienne.

[71] E.L. Johnson. 1989. Modeling and strong linear programs for mixed integer programming. *Algorithms and Model Formulations in Mathematical Programming.* S.W.Wallace (ed) Springer-Verlag, pp. 1-43.

[72] L.V. Kantorovich. 1948. On a problem of Monge. *Uspekhi Mat. Nauk. 3*, pp. 225-226.

[73] G. Kannan, J.C. Martinez, M.C. Vorster. 1997. A framework for incorporating dynamic strategies in earth-moving simulations. In *Proceedings of the 1997 Winter Simulation Conference*. Ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson.

[74] R.M. Karp. 1972. Reducibility among combinatorial problems. *In Complexity of Computer Computations*, pp. 85-103.

[75] S. Kataria, S.A. Samdani, A.K. Singh. 2005. Ant Colony Optimization in Earthwork Allocation. *International Conference on Intelligent Systems*, Kuala Lumpur, Malaysia.

[76] I. Katriel, L. Michel, P. Van Hentenryck. 2005. Maintaining Longest Paths Incrementally In *Constraints 10 (2)*, pp. 159-183.

[77] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi. 1983. Optimization by Simulated Annealing. *Science 220 (4598)*, pp. 671-680.

[78] J.R. Koza. 1990. Non-Linear Genetic Algorithms for Solving Problems. *United States Patent 4935877.*

[79] F. Laburthe. 2000. Choco : implementing a cp kernel. *In Proceedings of TRICS*, a post-conference workshop of CP 2000, Singapore.

[80] V. Le Diagon. 2009. Optimisation des grands arrêts de raffinerie. *Rapport de recherche ASSETSMAN* - www.assetsman.com

[81] C. Le Pape, P. Baptiste. 1999. Optimisation du stock de banches sur des chantiers de construction. Unpublished work.

[82] S. Lin, B.W. Kernighan. 1973. An effective heuristic algorithm for the Traveling-Salesman Problem. *In Operations Research 21*, pp. 498-516.

[83] V. Lübbecke. 2010. Column Generation. In *Wiley Encyclopedia of Operations Research and Management Science (EORMS).*

[84] D. Maier. 1978. The complexity of some problems on subsequences and supersequences *J. ACM 25 (2)*, pp. 322-336.

[85] B. Martin, A. Jeanjean. 2007. Outils de simulation pour gestion de parc d'équipements. *In Roadef'07, Grenoble.*

[86] M. Marzouk, O. Moselhi. 2002. CEPM 1 : selecting earthmoving equipment fleets using genetic algorithms. *Proceedings of the 34th Winter Simulation Conference*, San Diego, California, pp. 1789-1796.

[87] M. Marzouk, O. Moselhi. 2004. Multiobjective Optimization of Earthmoving Operations. *Journal of Construction Engineering and Management 130 (1)* pp. 105-113.

[88] R. Mayer, R. Stark. 1981. Earthmoving logistics. *Journal of the Construction Division 107(2)*, pp. 297-312.

[89] M. Minoux. 1989. Network synthesis and optimum network design problems : models, solutions methods and applications. *Networks, 19.* pp. 313-360.

[90] M. Minoux. 2001. Discrete cost multicommodity network optimization problems and exact solution methods. *Annals of Operations Research, 106*, pp. 19-46.

[91] M. Minoux. 2003. Optimum network design models and algorithms in transportation and communication. *Research and Applications, 6*, pp. 5-15.

[92] M. Minoux. 2007. Programmation mathématique *Tec et Doc.* 710 pages.

[93] G. Monge. 1781. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Academie Royale des Sciences*, Paris.

[94] A.A. Moreb. 1996. Linear programming model for finding optimal roadway grades that minimize earthwork cost. *European Journal of Operation Research 93*, pp. 148-154.

[95] B.M.E. Moret. 2002. Towards a discipline of experimental algorithmics. In *Data Structures, Near Neighbor Searches, and Methodology : 5th and 6th DIMACS Implementation Challenges. DIMACS Monographs 59*, pp. 197-213. American Mathematical Society, Providence, RI.

[96] O. Moselhi, A. Alshibani. 2009. Optimization of Earthmoving Operations in Heavy Civil Engineering Projects. *J. Constr. Engrg. and Mgmt. 135 (10)*, pp. pp.948-954.

[97] F. Noonan, R. J. Giglio. 1977. Planning Electric Power Generation : A Nonlinear Mixed Integer Model Employing Benders Decomposition. *Management Sciences 23 (9)*, pp. 946-956.

[98] M. Porcheron, A. Gorge, O. Juan, T. Simovic, G. Dereu. 2010. ROADEF/EURO 2010 : a large-scale energy management problem with varied constraints EDF R&D, Clamart, France (February 2010), 27 pages.

[99] K.J. Räihä, E. Ukkonen. 1981. The shortest common supersequence problem over binary alphabet is NP-complete *Theoretical Computer Science 16 (2)*, pp. 187-198.

[100] K. Rana, R.G. Vickson. 1991. Routing Container Ships Using Lagrangean Relaxation and Decomposition. *Transportation Science 25 (3)*, pp. 201-214.

[101] M. Savelsbergh. 1997. A branch-and-price algorithm for the generalized assignment problem. *Operation Research 45*, pp. 831-841.

[102] M. Savelsbergh, J.-H. Song. 2007. Inventory routing with continuous moves. *Computers and Operations Research 34 (6)*, pp. 1744-1763.

[103] M. Savelsbergh, J.-H. Song. 2008. An optimization algorithm for the inventory routing with continuous moves. *Computers and Operations Research 35 (7)*, pp. 2266-2282.

[104] A. Schrijver. 2003. Combinatorial Optimization - Polyhedra and Efficiency. *Springer-Verlag*. Berlin.

[105] M.Solomon. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research 35 (2)*, pp. 254-265.

[106] E.G. Talbi. 2009. From Design to Implementation. *ed. Wiley*, 624 pages.

[107] J.D. Ullman. 1976. Complexity of sequencing problems. *Computer and Job / Shop Scheduling Theory*, John Wiley & Sons Inc., New York.

Laboratoire d'Informatique de l'X
(UMR CNRS 7161)
École Polytechnique
Modélisation et Optimisation des Systèmes (Sysmo)
91128 Palaiseau Cedex France

jeanjean@lix.polytechnique.fr

E-LAB – Bouygues SA
40 rue de Washington
75008 PARIS

antoinejeanjean@gmail.com

École Doctorale de l'X
(École Doctorale n° 447)
91128 Palaiseau Cedex, France

Résumé :

Les problèmes d'optimisation en variables mixtes sont souvent résolus par décomposition quand ils sont de grande taille, avec quelques inconvénients : difficultés de garantir la qualité voire l'admissibilité des solutions et complexité technique des projets de développement. Dans cette thèse, nous proposons une approche directe, en utilisant la recherche locale, pour résoudre des problèmes d'optimisation mixte. Notre méthodologie se concentre sur deux points : un vaste ensemble de mouvements et une évaluation incrémentale basée sur des algorithmes approximatifs, travaillant simultanément sur les dimensions combinatoire et continue. Tout d'abord, nous présentons un problème d'optimisation des stocks de banches sur chantiers. Ensuite, nous appliquons cette technique pour optimiser l'ordonnancement des mouvements de terre pour le terrassement d'autoroutes et de voies ferrées. Enfin, nous discutons d'un problème de routage de véhicules avec gestion des stocks. Les coûts logistiques sont optimisés pour livrer un produit fluide par camion dans des zones géographiques d'une centaine de clients, avec la gestion de l'inventaire confiée au fournisseur.

Mots-clefs : Recherche locale, optimisation en variables mixtes, ordonnancement de tâches, tournées de véhicules avec gestion des stocks..

Abstract :

Large mixed-variable optimization problems are often solved by decomposition, with some drawbacks : difficulties to guarantee quality or even feasible solutions and technical complexity of development projects. In this thesis, we propose a direct approach, using local search, for solving mixed-variable optimization problems. Our methodology focuses on two points : a large pool of varied moves and an incremental evaluation based on approximate but highly efficient algorithms, working on combinatorial and continuous dimensions simultaneously. First, we present a formwork stocks optimization problem on construction sites. Then, we rely on this methodology to optimize earthworks scheduling for highway and railway projects. Finally, we solve a vehicle routing problem with inventory management. Inventory routing refers to the optimization of transportation costs for the replenishment of customers' inventories : based on consumption forecasts, the vendor organizes delivery routes.

Title : Local search for solving mixed-variables optimization problems : methodology and industrial applications.

Keywords : Local search, mixed-variable optimization, task scheduling, inventory routing.