POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

PRIMAL METHODS FOR VERY LARGE-SCALE OPTIMIZATION

EL MEHDI ER RAQABI
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Décembre 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

PRIMAL METHODS FOR VERY LARGE-SCALE OPTIMIZATION

présentée par **El Mehdi ER RAQABI**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

DEDICATION

To My Beloved Parents...

ACKNOWLEDGEMENTS

The many years that preceded and enabled this everlasting 4-year journey would not have been what they were without my parents and brother who shaped my personality and character. This work is another milestone added to the years of affection, attention, and rigor from my mother towards me and the obstacle course that my father started in the rural High Atlas Mountains in the Kingdom of The Western Place. Towards the end of my Ph.D. journey, I met my wife, a great human and a magnificent soul. She supported me unconditionally, brought magnificent learning experiences, and reminded me smoothly that self-care should always come before work.

This thesis would not have happened without Prof. Issmaïl El Hallaoui, who supported me continuously and unconditionally with over 200 effective meetings that answered my questions, opened new ones, and shaped my operational, tactical, and strategic research, career, and personal plans. Sensei has given me a level of freedom, autonomy, support, focus, passion, customization, adventure, leadership, and trust I could not have found elsewhere. He significantly boosted my passion for the operations research (OR) discipline and led me to outperform the targets I fixed in the Ph.D. roadmap designed in January 2020. Choosing Sensei as my supervisor was optimal.

While trying to learn as much as possible from Sensei, I was lucky enough to also learn from his former supervisor, the Professor who formed, trained, and supervised several big names in OR. Prof. François Soumis added another layer of OR passion to the layer of Sensei. He was involved in my supervision, training, research, and scholarship applications since the beginning of my Ph.D curriculum. He also allowed me to work closely with him as a teaching assistant for four consecutive years. Having him as a co-supervisor was optimal.

I deeply thank the members of the jury, Prof. Jean-François Cordeau, an OR pioneer who brought his internationally recognized OR expertise, and Prof. Pascal Van Hentenryck, a true OR leader in large-scale optimization and artificial intelligence for optimization. It is intimidating but also intellectually stimulating and rewarding to defend the thesis in front of them. Many thanks go to Prof. Louis-Martin Rousseau, who presided over the jury and gave interesting insights, and Prof. Michel Gamache who provided valuable comments.

To diversify my learning and escape Montreal's harsh winters, I got the chance to visit two great humans and supervisors. During the winter of 2023, I was a visiting scholar at Griffith University in Australia under the supervision of Prof. Yong Wu. Yong Sensei made the Gold Coast trip an everlasting memory and added another layer of OR passion with his rigor, perfection, and excellence. During the winter of 2024, I was a visiting scholar at KU

Leuven in Belgium under the supervision of Prof. Roel Leus. Roel Sensei made the Leuven trip an everlasting memory and added another layer of OR passion with his commitment, consistency, and expertise. Visiting both Yong Sensei and Roel Sensei ignited long-term collaboration and was optimal.

While the supervisors above boosted my OR passion, two great professors ignited my passion for the OR discipline: Prof. Najiba Sbihi and Prof. Wenkai Li. Sbihi Sensei introduced me ten years ago to OR concepts and urged me to pursue a Ph.D. in OR. Li Sensei, my master's supervisor, collaborated with me on my first OR paper. Furthermore, I was inspired by many great Professors who influenced my choice to pursue a Ph.D. program. These Professors include Prof. Abdelaziz Berrado, Prof. Loubna Benabbou, Prof. Nizar El Hachemi, and Prof. Lamia Kerzazi. The list goes up to all my professors who taught me a letter, a word, an insight, or a thought. Thank you so much!

This work would not have been possible without the collaborations I was allowed to start in my lab, the Research Group in Decision Analysis (GERAD), with the fantastic support staff and all the great members. While the list is quite long, I would like to mention Dr. Abderrahman Bani, who supported me unconditionally on the experimental part. I am sure all other great members will pardon not listing them.

Finally, I would like to thank the 40 million Canadian taxpayers who funded my Ph.D. research through the *Fonds de recherche du Québec - Nature et technologies* (FRQNT), the *Institut de Valorisation des Données* (IVADO), and the Mathematics of Information Technology and Complex Systems (MITACS), the 125 million Japanese taxpayers who funded my Japanese academic journey through the African Business Education Initiative (ABE Initiative), and most importantly, the 38 million Moroccan taxpayers who funded my undergraduate university studies in Morocco. All the thanks to the *Office Chérifien des Phosphates* (OCP Group) and Hydro Quebec, the two industrial partners that served as the application frameworks of my research. I am also very grateful to Polytechnique Montreal, KU Leuven, Griffith University, International University of Japan, Mohammadia School of Engineers, and all the universities and schools I attended over my learning curriculum.

As of 06/09/2024, I have completed my worldwide academic tour (Morocco, Japan, Canada, Australia, Chile, and Belgium) with at least one country from each continent. This tour was an objective I wanted to achieve before reaching the highest level of academic studies. Thus, adding to the great people above, I must thank many more people, from family to friends to sports partners to neighbors to colleagues to co-authors to professors. I am deeply grateful to anyone who ϵ-contributed explicitly or implicitly to this modest milestone. Thank you so much :-) !

PREFACE

This thesis presents part of my work performed from January 2020 to June 2024 at the Polytechnique Montreal (Poly) and the Group for Research in Decision Analysis (GERAD), under the supervision of Prof. Issmaïl El Hallaoui. It focuses on primal large-scale optimization. The main results presented in this dissertation appeared in the following publications (* refers to the corresponding author).

[1] Himmich Ilyas, Er Raqabi El Mehdi*, El Hachemi Nizar, El Hallaoui Issmaïl, Metrane Abdelmoutalib and Soumis François (2023). *MPILS: An Automatic Tuner for MILP Solvers.* **Computers & Operations Research**, 159, 106344.

[2] Er Raqabi El Mehdi*, Himmich Ilyas, El Hachemi Nizar, El Hallaoui Issmaïl and Soumis François (2023). *Incremental LNS Framework for Integrated Production, Inventory, and Vessel Scheduling: Application to a Global Supply Chain.* **Omega**, 116, 102821.

[3] Er Raqabi El Mehdi*, El Hallaoui Issmaïl and Soumis François (2024). *The Primal Benders Decomposition.* Submitted to **Operations Research**.

[4] Er Raqabi El Mehdi*, Wu Yong, El Hallaoui Issmaïl and Soumis François (2024). *Towards Resilience: Primal Large-scale Re-optimization.* **Transportation Research Part E: Logistics and Transportation Review**, 192, 103819.

[5] Er Raqabi El Mehdi*, Beljadid Ahmed, Mohammed Ali Bennouna, Rania Bennouna, Latifa Boussaadi, Nizar El Hachemi, Issmaïl El Hallaoui, Michel Fender, Anouar Jamali, Nabil Si Hammou and François Soumis (2024). *OCP Optimizes its Supply Chain for Africa.* To appear in **INFORMS Journal on Applied Analytics**.

During my Ph.D., my colleagues and I also worked on other aspects of primal large-scale optimization, which led to the publications below. Some of these papers (resp., 6, 7, 8) are methodological extensions of some of the publications above (resp., 1, 2, 3) and are not included in this dissertation for conciseness purposes. Some are the result of successful collaborations on a complex petrol station replenishment problem (9, 10). The last publication, written during my Ph.D. curriculum, is the result of my master's thesis in Japan.

[6] Er Raqabi El Mehdi*, Siwane Oussama, El Hallaoui Issmaïl and Beljadid Ahmed (2024). *A Parallel Multi-Purpose Tuner for MILP Solvers.* Soon to be submitted to **INFORMS Journal on Computing**.

[7] Er Raqabi El Mehdi*, Bani Abderrahman, Morabit Mouad, Blondin Massé Alexandre, Besner Alexandre, Fournier Julien and El Hallaoui Issmaïl (2024). *An Efficient Decomposition Matheuristic for the Transient Stability Constrained Unit Commitment at Hydro-Quebec.* Submitted to **IEEE Transactions on Power Systems**.

[8] Er Raqabi El Mehdi*, El Fassi Mohamed, Leus Roel, El Hallaoui Issmaïl and Bani Abderrahman (2024). *The Stochastic Improved Primal Simplex.* Soon to be submitted to **Mathematical Programming**.

[9] Bani Abderrahman, Er Raqabi El Mehdi*, El Hallaoui Issmaïl and Corréa Ayoub Insa (2024). *Combining Benders Decomposition and Column Generation for the Petrol Station Replenishment Problem with Inventory Management.* Submitted to **Management Science**.

[10] Bani Abderrahman, Er Raqabi El Mehdi*, El Hallaoui Issmaïl and Corréa Ayoub Insa (2024). *The Petrol Station Replenishment Problem: A Case Study from West Africa.* Submitted to **African Transport Studies**.

[11] Er Raqabi El Mehdi and Li Wenkai* (2023). *An Electric Vehicle Migration Framework for Public Fleet Planning.* **Transportation Research Part D: Transport and Environment**, 118, 103732.

Note on this Dissertation: The application framework for the main results developed at Poly and GERAD was a complex, large-scale optimization problem provided by the industrial partner, the OCP Group, in collaboration with the Mohammed VI Polytechnic University (UM6P). The methodological contributions, along with the collaboration between Canada and Morocco, led to being a finalist in the IFORS Prize for OR in Development 2023, organized during the 23rd conference of the International Federation of Operational Research Societies (IFORS) from July 10th to 14th, 2023 in Santiago, Chile. It also led to winning the EURO Excellence in Practice Award 2024, organized during the 33rd European Conference on Operational Research (EURO) from June 30th to July 3rd, 2024 in Copenhagen, Denmark. Beyond the OCP Group, the results, with extensions, were also applied to another complex, large-scale optimization problem provided by the Canadian organization Hydro-Québec, where we have achieved a significant reduction from a gap of 3% in 16 hours to a gap of 1.21% in 16 minutes on the provided instances. All the manuscripts under review or revision are available as *Cahiers du GERAD*.

RÉSUMÉ

Cette thèse de doctorat a été motivée par l'intuition suivante: une approche primale est toujours préférée à une approche duale dans l'optimisation des problèmes à très grande échelle. Les approches primales exactes font référence à toute approche permettant de passer d'une solution réalisable entière à une solution réalisable entière jusqu'à convergence à une solution réalisable optimale. Une approche primale converge ainsi rapidement si nous avons une bonne solution initiale. Il est à noter que la plupart des heuristiques connues basées sur la recherche locale sont primales mais non exactes. Ces heuristiques sont très utilisées en pratique grâce entre autres à cet aspect primal. D'autre part, une approche qui n'est pas primale est dite duale. La décomposition de Benders est un example d'approche duale qui montre un comportement en zigzag rendant la convergence lente, ce qui est problématique en pratique. Basée sur cette intuition primale, cette thèse met en évidence les avantages de l'utilisation d'approches primales pour aborder des problèmes d'optimisation à très grande échelle puisqu'elles profitent efficacement de l'information primale disponible (par exemple, l'historique des solutions). Ceci est le cas dans plusieurs contextes réels, où les organisations sont confrontées à des problèmes de grande taille pour lesquels elles disposent généralement de bonnes solutions primales proches de la solution optimale (en termes du support de solution). L'objectif est d'utiliser ces solutions (information primale) pour atteindre rapidement des solutions (presque) optimales. Tous les travaux de cette thèse peuvent être regroupés sous l'égide de l'optimisation primale à grande échelle de différentes perspectives.

Dans le premier essai de cette thèse, nous explorons l'optimisation primale à grande échelle du point de vue *solveur*. En particulier, nous étudions le problème de configuration des paramètres, qui consiste à trouver une configuration de paramètres qui donne à un algorithme particulier les meilleures performances. Nous introduisons un nouveau *tuner* multiphase basé sur la métaheuristique de recherche locale itérée (*iterated local search*). Ce *tuner* résout le problème de configuration des paramètres pour les solveurs déterministes utilisés pour résoudre des problèmes d'optimisation difficiles à très grande échelle. De plus, le *tuner* propose une nouvelle stratégie de recherche basée sur trois idées. Premièrement, au lieu d'explorer l'espace exponentiel de configurations induit par l'ensemble de paramètres, le *tuner* multiphase se concentre sur un petit *pool* de paramètres enrichi de manière dynamique avec de nouveaux paramètres prometteurs. Deuxièmement, il exploite les connaissances acquises au cours de la recherche en utilisant l'apprentissage statistique pour interdire les combinaisons de paramètres moins prometteuses. Troisièmement, il s'entraîne sur une seule instance fournie par un *clustering* antérieur des instances considérées. Le *tuner* est primal car l'heuristique

utilisée est primale. Il permet l'obtention de plusieurs solutions (quasi-)optimales en quelques minutes sur plusieurs instances.

Ensuite, nous explorons l'optimisation primale à grande échelle d'un point de vue *méta-heuristique*. Nous étudions un modèle linéaire mixte en nombres entiers qui intègre la planification de la production, la gestion des stocks et l'affectation des navires pour une chaîne d'approvisionnement globale. Étant donné que de tels problèmes à grande échelle sont NP-difficiles et souffrent généralement de la symétrie, nous effectuons une analyse exploratoire pour identifier les sources de complexité. Suite à cela, nous concevons une nouvelle variante de la métaheuristique de recherche par voisinnage pour résoudre le problème efficacement. Cette variante profite de l'information primale et converge de manière incrémentale vers des solutions (quasi-)optimales, ce qui est très pratique pour les problèmes de grande taille. En outre, bien que la symétrie soit considérée comme un problème dans la littérature, l'algorithme mis en œuvre fournit un moyen pratique de profiter de la symétrie au lieu de la briser. Sur plusieurs instances réelles du problème considéré, des solutions (quasi-)optimales sont obtenues en moins de 10 minutes.

Dans le troisième essai de cette thèse, nous explorons l'optimisation primale à grande échelle d'un point de vue *exact*. Parmi plusieurs décompositions, la décomposition de Benders a été appliquée de manière significative pour résoudre des problèmes à très grande échelle avec des variables complexes qui, une fois temporairement fixées, donnent lieu à des problèmes faciles à résoudre. Pourtant, dans sa forme standard, la décomposition de Benders ne profite pas de l'information primale et montre un comportement en *zigzag*, rendant la convergence très lente, ce qui est problématique en pratique. Sur la base d'observations issues de la pratique, nous proposons la *primal Benders decomposition* (PBD) pour des problèmes peu denses de grande taille, pour lesquels les variables les plus compliquées sont égales à zéro dans la solution optimale. Cette méthode, un changement de paradigme, utilise le problème maître de PBD pour sélectionner les variables compliquantes à insérer dans le sous-problème PBD, qui constitue une restriction du problème d'origine, au lieu de les fixer (source de zigzag). La PBD évite le zigzagging et converge vers l'optimum d'une façon monotone et ainsi améliore la solution primale à chaque itération. C'est un comportement primal très pratique (au lieu d'un comportement dual qui handicapait la méthode Benders depuis sa naissance il y a plus de 50 ans). Les coupes générées sont de meilleure qualité car les solutions obtenues sont de meilleure qualité, ce qui implique la génération de moins de coupes pour converger. Les résultats de la PBD sur des instances du *facility location problem* et d'autres réelles du problème qui a motivé cette essai démontrent les avantages de la méthode.

Dans l'essai suivant, nous explorons l'optimisation primale à grande échelle dans une perspec-

tive d'*apprentissage*. En particulier, nous étudions le rôle de l'apprentissage sur l'information primale pour ré-optimiser après des perturbations. En effet, les perturbations sont universelles pour les problèmes à très grande échelle et leur apparition est devenue plus fréquente ces dernières années en raison des événements globaux. Dans un tel cas, la réoptimisation peut aider les entreprises à atteindre leur résilience en leur permettant de simuler plusieurs scénarios de simulation et de s'adapter aux circonstances et aux défis changeants en temps réel. Nous concevons un cadre de réoptimisation pour la résilience. Nous modélisons les perturbations, les décisions de réparation et le problème de réoptimisation qui en résulte avec le but de maximiser la résilience. Nous exploitons l'information primale grâce à la correction, au *warmstart* et à l'apprentissage automatique. Les résultats numériques démontrent que, dans plusieurs cas, l'optimisation locale (sur une partie de la *supply chain*) est suffisante pour réoptimiser rapidement après des perturbations.

Enfin, nous fusionnons, exploitons et implémentons les différents outils heuristiques et exacts de recherche opérationnelle ci-dessus dans un seul système, ce qui a permis aux spécialistes de la recherche opérationnelle du Groupe OCP, de l'Université Polytechnique Mohammed VI et de Polytechnique Montréal d'opérationnaliser un système qui optimise la chaîne d'approvisionnement du Groupe OCP. De plus, inspirée par la pratique, l'équipe a implémenté la *hybrid Benders decomposition* (HBD), qui consiste à fixer certaines variables compliquées liées aux commandes confirmées (comme dans la décomposition de Benders) et à garder libre les autres liées aux commandes non confirmées dans le sous-problème de Benders (comme dans PBD). La direction d'OCP attribue désormais à l'opérationnalisation du système des avantages opérationnels, contribuant à une augmentation de plus de 240 millions de dollars du chiffre d'affaires annuel, soit l'équivalent de suffisamment d'engrais pour nourrir 30 millions d'êtres humains.

Mots-clés. *Optimisation à grande échelle*, *Décomposition de Benders*, *Méthode en forme de L*, *Décomposition de Dantzig-Wolfe*, *Programmation en nombres entiers*, *Problème de configuration des paramètres*, *Solveurs MILP*, *Métaheuristiques*, *Apprentissage Automatique*, *Réoptimisation*, *Résilience*, *Perturbation*, *Gestion de la chaîne d'approvisionnement*, *OR@AFRICA*.

ABSTRACT

My Ph.D. research was motivated by the following intuition: a primal approach is always preferred to a dual approach in very large-scale optimization problems.

Exact primal approaches refer to any approach that allows moving from a feasible integer solution to a feasible integer one until reaching the optimal solution. Thus, a primal approach quickly converges if we have a good initial solution. It is worth mentioning that most known heuristics based on local search are primal but not exact. These heuristics are widely used in practice thanks, among other things, to this primal aspect. On the other hand, an approach that is not primal is called dual. Benders decomposition is an example of a dual approach that shows zigzag behavior making convergence slow, which is problematic in practice. Based on this primal intuition, this thesis highlights the benefits of using primal approaches to tackle very large-scale optimization problems since they effectively profit from the available primal information (e.g., history of solutions). This is the case in real-life contexts, where organizations face very large-scale problems for which they usually have good primal solutions close to the optimal solution (in terms of solution support). The goal is to use these solutions (primal information) to reach (near-)optimal solution(s) quickly. All the essays in this dissertation can be put under the umbrella of primal large-scale optimization from different perspectives.

In the first essay of this dissertation, I explore the primal large-scale optimization from a *solver* perspective. In particular, I study the parameter configuration problem, which consists of finding a parameter configuration that gives a particular algorithm the best performance. I introduce a new multi-phase tuner based on the iterated local search (ILS) meta-heuristic: the multi-phase iterated local search (MPILS) tuner. This tuner addresses the parameter configuration problem for deterministic mixed-integer linear programming (MILP) solvers that are used to solve challenging very large-scale optimization problems. Further, the proposed tuner offers a new search strategy based on three ideas. First, instead of tuning in the entire configuration space induced by the parameter set, the multi-phase tuner focuses on a small parameter pool that is dynamically enriched with new promising parameters. Second, it leverages the gathered knowledge during the search using statistical learning to forbid less promising parameter combinations. Third, it tunes on a single instance provided by earlier clustering of MILP instances. The MPILS tuner is primal since the ILS metaheuristic is primal. The tuner obtains near-optimal solutions in a few minutes on several NP-hard realistic instances provided by the industrial partner.

Next, I explore the primal large-scale optimization from a *metaheuristic* perspective. I study a multiobjective, MILP model that integrates production scheduling, inventory management, and vessel assignment for a global supply chain. Given that such large-scale problems are NP-hard and usually suffer from symmetry, I conduct an exploratory analysis to identify complexity sources. Following this, I design a novel variant of the large neighborhood search metaheuristic to tackle the problem efficiently. This variant takes advantage of primal information and converges incrementally towards (quasi-)optimal solutions, which is very practical for large-scale optimization problems. Furthermore, while symmetry is considered an issue in the literature, the implemented algorithm provides a practical way of profiting from symmetry instead of breaking it. On several real instances (which typically take several hours) of the problem considered, (quasi-)optimal solutions are obtained in less than 10 minutes.

In the third essay of this dissertation, I explore the primal large-scale optimization from an *exact* perspective. Among several decompositions, the Benders decomposition has been significantly applied to tackle very large-scale problems with complicating variables, which, when temporarily fixed, yield problems easy to solve. Still, in its standard form, the Benders decomposition does not profit from the primal information and shows a zigzagging behavior, making convergence very slow, which is problematic in practice. Driven by observations from the practice, I propose the primal Benders decomposition (PBD) for sparse very large-scale problems, for which most complicating variables are equal to zero in the optimal solution. This method, a paradigm shift, uses the PBD master problem to select the complicating variables to insert in the PBD subproblem, which is a restriction of the original problem, instead of fixing them (source of zigzag). The PBD avoids zigzagging and converges towards the optimum in a monotonous way and thus improves the primal solution at each iteration. It is a very practical primal behavior (instead of a dual behavior which has handicapped the Benders decomposition since its birth more than 50 years). The generated cuts are of better quality because the solutions obtained are of better quality, which implies the generation of fewer cuts to converge. The results of the PBD on instances of the facility location problem and others from the problem that motivated this research demonstrate the method's potential.

After perturbation, the re-optimized solution remains close to the perturbed solution and is therefore rich in primal information. For this reason, a primal approach would be most efficient. In the next essay, I explore the primal large-scale optimization from an *learning* perspective. In particular, I explore the role of learning from the available history in re-optimizing after perturbations. Indeed, perturbations are universal in large-scale optimization, and their appearance has become more frequent in the past few years due to global events. In such a case, re-optimization can support companies in achieving resilience by enabling them to simulate several what-if scenarios and adapt to changing circumstances and challenges in real-time. I design a generic and scalable resilience re-optimization framework. I model perturbations, recovery decisions, and the resulting re-optimization problem, which maximizes resilience. I leverage the primal information through fixing, warm-start, and machine learning.

Finally, I merge, leverage, and implement the various heuristic and exact operations research tools above in one system, which allowed operations research specialists at the OCP Group, the Mohammed VI Polytechnic University, and the Polytechnique Montreal to operationalize a system that optimizes the OCP downstream supply chain operations. Furthermore, inspired by the practice, the team implemented a novel hybrid variant of Benders decomposition, which consists of fixing some complicating variables related to confirmed orders and freeing others related to unconfirmed orders in the Benders subproblem. OCP management now credits the system operationalization with providing operational benefits, contributing to over a $240 million increase in annual turnover, or equivalently enough fertilizers to feed 30 million humans.

Keywords. *Large-Scale Optimization*, *Benders Decomposition*, *L-shaped Method*, *Dantzig-Wolfe Decomposition*, *Mixed-Integer Programming*, *Parameter Configuration Problem*, *MILP Solvers*, *Metaheuristics*, *Machine Learning*, *Re-optimization*, *Resilience*, *Perturbation*, *Supply Chain Management*, *OR@AFRICA*.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

PREFACE vi

RÉSUMÉ viii

ABSTRACT xi

TABLE OF CONTENTS xiv

LIST OF TABLES xix

LIST OF FIGURES xxi

LIST OF SYMBOLS AND ACRONYMS xxiv

LIST OF APPENDICES xxv

CHAPTER 1 INTRODUCTION 1
- 1.1 The Four Perspectives of OR 1
- 1.2 Thesis Contribution 4
- 1.3 Thesis Organization 6

CHAPTER 2 LITERATURE REVIEW 8
- 2.1 Solver Perspective 8
- 2.2 Metaheuristic Perspective 10
  - 2.2.1 The LNS Metaheuristic 10
  - 2.2.2 The ILS Metaheuristic 11
- 2.3 Exact Perspective 12
  - 2.3.1 Dantzig-Wolfe Decomposition 12
  - 2.3.2 Benders Decomposition 13
- 2.4 Learning Perspective 14
- 2.5 Thesis Positioning 15

CHAPTER 3 ARTICLE 1: MPILS: AN AUTOMATIC TUNER FOR MILP SOLVERS 16

3.1 INTRODUCTION . . . 16
3.2 LITERATURE REVIEW . . . 18
3.2.1 Parameter Configuration for MILP Problems . . . 18
3.2.2 Paper Positioning . . . 20
3.3 MULTI-PHASE ITERATED LOCAL SEARCH TUNER . . . 22
3.3.1 Motivation . . . 22
3.3.2 MPILS Tuner Overview . . . 23
3.3.3 Tuning Step . . . 25
3.3.4 Learning Step . . . 27
3.3.5 Evaluation Step . . . 30
3.4 COMPUTATIONAL STUDY . . . 33
3.4.1 Experimental Setting . . . 33
3.4.2 PSIMVA Test Bed . . . 35
3.4.3 MIPLIB Test Bed . . . 46
3.5 CONCLUSION . . . 50

CHAPTER 4 ARTICLE 2: INCREMENTAL LNS FRAMEWORK FOR INTEGRATED PRODUCTION, INVENTORY, AND VESSEL SCHEDULING: APPLICATION TO A GLOBAL SUPPLY CHAIN . . . 51
4.1 INTRODUCTION . . . 51
4.2 LITERATURE REVIEW . . . 53
4.2.1 Deterministic Optimization . . . 53
4.2.2 Multiobjective Optimization . . . 54
4.2.3 Our Research . . . 56
4.3 PROBLEM DESCRIPTION . . . 57
4.4 PROBLEM FORMULATION . . . 61
4.4.1 Decision Variables . . . 61
4.4.2 Objective Function . . . 61
4.4.3 Constraints . . . 62
4.4.4 Models . . . 64
4.5 EXPLORATORY ANALYSIS . . . 66
4.5.1 Problematic . . . 66
4.5.2 Model Analysis . . . 66
4.6 SOLUTION METHODOLOGY . . . 71
4.6.1 Prescriptions against Complexity . . . 72
4.6.2 ILNS Framework . . . 72

4.6.3 Time-based ILNS 75
4.6.4 Reference Algorithm 76
4.7 COMPUTATIONAL STUDY 78
4.7.1 Test Plan 78
4.7.2 Computational Results 79
4.7.3 Managerial Insights 85
4.8 CONCLUSION 88

CHAPTER 5 ARTICLE 3: THE PRIMAL BENDERS DECOMPOSITION 89
5.1 INTRODUCTION 89
5.2 THE PRIMAL BENDERS DECOMPOSITION 93
5.2.1 Motivation 93
5.2.2 The PBD Framework 95
5.2.3 The PBD Convergence 97
5.3 THE ACCELERATED PBD 100
5.3.1 Acceleration Strategies 100
5.3.2 The Accelerated PBD Algorithm 104
5.3.3 The Accelerated PBD Convergence 105
5.4 EXAMPLE 107
5.5 COMPUTATIONAL RESULTS ON THE FACILITY LOCATION PROBLEM 109
5.5.1 Experimental Design 110
5.5.2 Computational Results 111
5.5.3 Computational Insights 115
5.6 COMPUTATIONAL RESULTS ON THE OCP GROUP CASE STUDY . . 117
5.6.1 Problem Description 117
5.6.2 Experimental Design 118
5.6.3 Computational Results 119
5.6.4 Managerial Insights 123
5.7 CONCLUSION 125

CHAPTER 6 ARTICLE 4: TOWARDS RESILIENCE: PRIMAL LARGE-SCALE RE-OPTIMIZATION 126
6.1 INTRODUCTION 126
6.2 LITERATURE REVIEW 129
6.2.1 Resilience 129
6.2.2 Re-optimization 130
6.2.3 Primal Information 132

6.2.4 Our Research 133
6.3 RESILIENCE/RE-OPTIMIZATION FRAMEWORK 134
6.3.1 Framework 134
6.3.2 Managerial Discussion 135
6.4 CASE PROBLEM DEFINITION 136
6.4.1 Context 137
6.4.2 Resilience 141
6.4.3 Re-optimization Problem 142
6.5 CASE SOLUTION METHODOLOGY 143
6.5.1 Primal Information 143
6.5.2 Re-optimization Approach 147
6.6 CASE EXPERIMENTAL DESIGN 149
6.6.1 Instances 149
6.6.2 Machine Learning Model 150
6.6.3 Computational Setting and Implementation Details 152
6.7 CASE COMPUTATIONAL RESULTS 152
6.7.1 Weights Tuning 152
6.7.2 Performance Comparison 153
6.7.3 Sensitivity Analysis 158
6.7.4 Managerial Insights 159
6.8 CONCLUSION 161

CHAPTER 7 ARTICLE 5: OCP OPTIMIZES ITS SUPPLY CHAIN FOR AFRICA 162
7.1 Introduction 162
7.1.1 Need for Fertilizers in Africa 162
7.1.2 The Basic Challenge at OCP 163
7.1.3 The Specifics and the Need for a New Solution 164
7.1.4 Contribution 165
7.1.5 Organization 165
7.2 Context 166
7.2.1 Physical Flow & Storage 166
7.2.2 Products 166
7.2.3 Orders 166
7.2.4 Decisions 167
7.2.5 Solver 167
7.2.6 Exploration & Kernel 168

7.3 Analytics Approach . . . 169
7.3.1 Key Components . . . 169
7.3.2 The Improved DLP . . . 174
7.4 Impact . . . 178
7.4.1 Quantifiable Benefits . . . 178
7.4.2 Planning Process . . . 181
7.4.3 Involvement of Local Researchers . . . 182
7.4.4 Theory Inspired by Practice . . . 182
7.5 Conclusion . . . 183

CHAPTER 8 GENERAL DISCUSSION AND CONCLUSION . . . 185
8.1 Summary of Works . . . 185
8.2 Limitations . . . 185
8.3 Future Research . . . 186

REFERENCES . . . 210

APPENDICES . . . 211

LIST OF TABLES

2.1 Summary of MILP Configuration Algorithms 10
3.1 PSIMVA Instances 37
3.2 Gap within 280s obtained on N Instances using Best Configuration(s) returned by MPILS Tuner (%) 38
3.3 Default CPLEX, MPILS Tuner, ParamILS Tuner, and irace Tuner Performance Comparison on N Instances 39
3.4 Default CPLEX, MPILS Tuner, ParamILS Tuner, and irace Tuner Performance Comparison on P Instances 46
3.5 Default CPLEX, MPILS Tuner, ParamILS Tuner, and irace Tuner Performance Comparison on $MIPLIB$ Instances 48
4.1 Summary of Deterministic Optimization 54
4.2 Summary of Multiobjective Optimization 55
4.3 PSIMVA Notation 60
4.4 Binary Classes 69
4.5 Investigation Strategy Results 70
4.6 q Values for Shipment 4586 71
4.7 Instances 79
4.8 $DPSIMVA^1_{lexi}$ Results 80
4.9 $RPSIMVA^1_{lexi}$ Results 81
4.10 $TPSIMVA^1_{lexi}$ Results 82
4.11 Number of Product Changeovers 85
4.12 Re-optimization Example when considering Confirmed and Unconfirmed Shipments 87
5.1 Performance Comparison for Cap Instances 112
5.2 Performance Comparison for SCAP Instances 113
5.3 Performance Comparison for M Instances 113
5.4 Impact of Acceleration Strategies 114
5.5 Performance on the LBs 115
5.6 Tight Cuts between BD and $\mathcal{PBD}$ 117
5.7 Instances 118
5.8 $\mathcal{PBD}$ Peformance 119
5.9 Impact of Acceleration Strategies 120
5.10 Comparison with Other Methods 121

6.1 Summary of Relevant Resilience Literature 129
6.2 Summary of Relevant Re-optimization Literature 131
6.3 Instances 149
6.4 ML Model Hyperparameters 151
6.5 Average TOP_κ for the Test Pairs 151
6.6 Weight α_1 Sensitivity Analysis 153
6.7 Performance Comparison 154
6.8 Number of Delayed, Advanced, and Other Vessels rescheduled in the Final Schedule 157
6.9 Perturbation Percentage impact on Instance I_1 158
6.10 Primal Information impact on $Time$ for Instance I_1 159
7.1 Results obtained by the $\mathcal{DLP}$, the $\mathcal{MPILS}$ tuner, the $\mathcal{ILNS}$ metaheuristic, and the $\mathcal{ILNS}^{++}$ method on J Instances for the TF KPI . 179
7.2 Results obtained by the $\mathcal{HBD}$ method on hard J Instances for the TF KPI 179
7.3 Optimal TF for J Instances 180
B.1 MPILS Tuner and ParamILS$_1$ Tuner Comparison 213
B.2 MPILS Tuner and ParamILS$_1$ Tuner Deteriorating Configurations % over Total Time for N Instances 214
D.1 Notation 217
D.2 Complicating Binary Classes in the PSIMVA Representative Instance 220
F.1 PSIMVA Notation 224
F.2 Decision Variables 225
H.1 PSIMVA Instances Description 232
I.1 Results obtained on J Instances using the $\mathcal{DLP}$ and the $\mathcal{IDLP}$ for the PC KPI 233

LIST OF FIGURES

1.1 Optimization Age Four Perspectives 3
3.1 Literature Review Summary 19
3.2 MPILS Tuner in the blue frame with three steps: *Tuning* step, *Learning* step, and *Evaluation* step 24
3.3 Gap Evolution over Total Time on Instance $N1$ for MPILS Tuner, ParamILS Tuner, and irace Tuner 40
3.4 *Learning* Step Impact for Instance $N1$: with *Learning* on the left and without *Learning* on the right 41
3.5 *Evaluation* Step Impact for Instance $N1$ 43
3.6 Evolution of Satisfactory Configurations % through MPILS Tuner Phases for Instance $N1$ 44
3.7 Gap Evolution over Total Time on Instance $P1$ for MPILS Tuner, ParamILS Tuner, and irace Tuner 45
3.8 Runtime to Optimality Evolution over Total Time for MPILS Tuner, ParamILS Tuner, and irace Tuner 49
4.1 The Phosphate Supply Chain 57
4.2 A Miniature Zoom on the Framed Part 58
4.3 Objective Function Analysis 67
4.4 Partial Loading Example 71
4.5 $\mathcal{ILNS}$ Framework 73
4.6 $\mathcal{TILNS}$ Example 76
4.7 $\mathcal{RA}$ Example 78
4.8 Comparison of Gap Improvement along Execution Time 84
4.9 Confirmed and Unconfirmed Shipments 86
5.1 A spectrum with two extremes: BD and MILP 91
5.2 Standard View of Benders Decomposition 94
5.3 BD as the dual of DWD 95
5.4 The PBD Framework 96
5.5 $\mathcal{PBD}$ as a compromise between BD and MILP 97
5.6 le titre 108
5.7 le titre 109
5.8 le titre 116
5.9 The Phosphate Supply Chain (reproduced from Figure 1 in [2]) . . . 118

5.10 Bounds Improvement for along Execution Time the Five Methods for Instance I_{10} . . . 122
5.11 Schedule obtained by $\mathcal{PBD}$ for Instance I_{12} . . . 123
5.12 Schedule obtained by HBD for Instance I_{12} . . . 123
5.13 HBD as a compromise between $\mathcal{PBD}$ and BD . . . 124
6.1 RRO Framework . . . 134
6.2 SCOR Model from AIMS . . . 135
6.3 The Phosphate Supply Chain (reproduced from Figure 1 in [2]) . . . 137
6.4 Singapore Port Queue . . . 140
6.5 Valid Inequalities Illustration for Vessel v . . . 145
6.6 2-stage $\mathcal{ILNS}$ . . . 148
6.7 Example of a perturbed schedule . . . 150
6.8 TF and $|\Delta D|$ Evolution based on α_1 for Instance I_1 . . . 153
6.9 Perturbed Instance I_1 under scenario (10%,10%) . . . 155
6.10 Final Schedule obtained by $\mathcal{ILNS}$ for Perturbed Instance I_1 under scenario (10%,10%) . . . 156
6.11 Final Schedule obtained by 2-stage $\mathcal{ILNS}$ for Instance I_1 under scenario (10%,10%) . . . 156
6.12 Average Percentage of Vessels Delayed, Advanced, and Others rescheduled in the Final Schedule(s). Maintained Vessels are Vessels kept as per the Previously Optimal Solution(s). . . . 157
6.13 Two Types of Schedules . . . 160
7.1 The OCP Supply Chain - From Extraction to the Jorf Site . . . 166
7.2 Integrality Gap Evolution with Execution Time . . . 169
7.3 *$\mathcal{MPILS}$ Tuner in the Blue Frame with Three Steps:* Tuning *Step,* Learning *Step, and* Evaluation *Step* . . . 170
7.4 $\mathcal{ILNS}$ Metaheuristic in the Blue Frame with the Four Steps: *Vessel Assignment* Step, *Problem Reduction* Step, *Solving* Step, and *Wrap-up* Step . . . 171
7.5 $\mathcal{PBD}$ Method in the Blue Frame with the Two Steps: *RPSP* Step and *PBD RMP* Step. . . . 172
7.6 $\mathcal{RRO}$ Approach in the Blue Frames . . . 174
7.7 Overview of the $\mathcal{IDLP}$ System . . . 175
7.8 $\mathcal{HBD}$ Method in the Blue Frame . . . 175
7.9 The $\mathcal{IDLP}$ System . . . 177
7.10 Rice Crops Ghana . . . 181

A.1 Summary of MILP Performance Loss 211

LIST OF SYMBOLS AND ACRONYMS

AI	Artificial Intelligence
B&B	Branch-and-Bound
B&C	Branch-and-Cut
B&P	Branch-and-Price
BD	Benders Decomposition
CG	Column Generation
CO	Combinatorial Optimization
DWD	Dantzig-Wolfe Decomposition
ILNS	Incremental Large Neighborhood Search
ILS	Iterated Local Search
KPI	Key Performance Indicator
LNS	Large Neighborhood Search
LP	Linear Programming
MIP	Mixed Integer Programming
MILP	Mixed-integer Linear Programming
ML	Machine Learning
MPILS	Multi-Phase Iterated Local Search
NLP	Non-linear Programming
NP	Non-deterministic Polynomial-time
NPV	Net Present Value
LR	Linear Relaxation
OR	Operations Research
OCP	Office Chérifien des Phosphates
PCP	Parameter Configuration Problem
PBD	Primal Benders Decomposition
PSIMVA	Production Scheduling, Inventory Management, and Vessel Assignment
RG	Raw Generation
RMP	Restricted Master Problem
RRO	Resilience/Re-Optimization
SIP	Stochastic Integer Programming
SP	Subproblem

LIST OF APPENDICES

Appendix A	Troubleshooting of MILP Solving	211
Appendix B	Additional Experiments	213
Appendix C	The stochastic capacitated facility location problem	215
Appendix D	The PSIMVA Formulation and PBD Decomposition	216
Appendix E	Mathematical Proofs	223
Appendix F	The Optimization Model	224
Appendix G	The HBD Method	228
Appendix H	PSIMVA Instances Description	232
Appendix I	Results on the Second KPI	233

CHAPTER 1 INTRODUCTION

By 2100, the world's population is expected to reach around 10.9 billion [12]. With such an increase, our population has growing needs such as food, transportation, and healthcare. A great example is the coronavirus pandemic during which humans dealt with several large-scale optimization problems [13, 14]. These needs are fulfilled by various organizations that face the increasing complexity of operations. To survive, these organizations have no choice but to reduce the cost of these operations through automation and optimization of large-scale problems like operations scheduling and resource management. Given the size and the complexity of these problems, manual solutions are no longer efficient. In this chapter, I highlight the four perspectives of operations research (OR), the thesis contribution, and the thesis organization.

1.1 The Four Perspectives of OR

Shortly after World War II, a new field called *Mathematical Optimization*, or shortly optimization, was born. Among the pioneers, George Dantzig is considered the godfather of the large-scale optimization subfield, especially for his contributions to linear programming. In one of his papers [15], he stated:

> *"Linear programming can be viewed as part of a great revolutionary development which has given mankind the ability to state general goals and to lay out a path of detailed decisions to take in order to "best" achieve its goals when faced with practical situations of great complexity. Our tools for doing this are ways to formulate real-world problems in detailed mathematical terms (models), techniques for solving the models (algorithms), and engines for executing the steps of algorithms (computers and software)."*

The optimization field is huge and includes several branches, e.g., linear, non-linear, deterministic, stochastic, online, and offline programming, just to name a few. In this research, we limit ourselves to the two main branches that have known rapid growth, i.e., linear programming (LP) and mixed-integer programming (MIP). When combined, we obtain the mixed-integer linear programming (MILP). The beauty and superiority of MILP models lie in their ability to model large-scale real-world problems. A canonical formulation of a MILP is the following:

$$\begin{aligned} \min\ & c^T x + d^T y, && \text{(MILP)} \\ s.t.:\ & Ax + Ty \geq b, && (1.1) \\ & x \in \mathbb{R}^m_+,\ \ y \in \mathbb{Z}^n_+, && (1.2) \end{aligned}$$

where $c \in \mathbb{R}^m$, $d \in \mathbb{R}^n$, $A \in \mathbb{R}^{p\times m}$, $T \in \mathbb{R}^{p\times n}$, and $b \in \mathbb{R}^p$. *Mixed 0-1 linear programming* and *pure integer programming* are special cases of the above problem where variables y are restricted to take binary values and $m = 0$, respectively.

Except for trivial cases, MILPs belong to the NP-hard class of optimization problems, i.e., there is no polynomial algorithm known to solve them. Also, the complexity of solving them increases exponentially with the problem size [16]. Practical solution approaches to tackle the (MILP) rely on relaxing the integrality conditions for y variables. The obtained model is called the LP *relaxation.* It is easier to solve and provides a lower bound on the optimal value of the (MILP) problem. Developed by Dantzig [17], the *simplex method* is among the most common solution approaches to solve linear relaxations.

Once the linear relaxation solution is obtained, the *branch-and-bound* (B&B) method [18, 19] can be used. It is based on an implicit enumeration of the integer solutions. It creates a *tree*, where the *root node* is the LP relaxation of the problem. It then explores the tree *branches*, where each *node* is a subproblem corresponding to a subset of the feasible region. After solving the subproblem at the current node, i.e., *bounding*, the node solution is compared to the upper and lower estimated bounds on the optimal solution. A node is discarded if it cannot produce a better solution than the best one reached by the algorithm so far, i.e. *pruning.* If the node cannot be pruned, a *branching* is performed by creating two (in some cases more) new disjoint subsets of the feasible region by restricting the domain of integer variables. Two key decisions should be made: *variable selection* and *node selection.* The first one addresses the selection of a fractional variable in the branching step. The latter addresses the order in which the nodes are explored. Combined, these decisions define the search strategy used to explore the tree. Various strategies can be considered for each decision [20, 21]. If an integer node is found, its value provides an upper bound on the optimal solution. Also, the minimum value among all active nodes gives a lower bound on the optimal value. The algorithm stops when all nodes are explored or when the *optimality gap* is within the desired threshold. In a minimization context, the optimality gap measures the deviation of the best-known upper bound from the optimal solution of the given problem [22].

While the described approach has been very successful, it becomes impractical when applied solely in large-scale contexts. Thus, throughout the *Optimization Age* (a term introduced by Nemhauser [23]), several researchers worldwide explored the optimization field looking for tools that will allow solving more efficiently large-scale real-world problems. As stated by Dantzig [15], once the modeling tool has been determined, i.e., MILP, the remaining tools are algorithms and software. Algorithms can be classified into approximate and exact. The software includes the solvers used for optimization. This trio has evolved significantly over the *Optimization Age*. Also, with the boosting of learning capabilities via the machine learning (ML) and artificial intelligence (AI) tool, optimization has become even more powerful. Considering all these tools, I obtain the four perspectives from which a large-scale optimization problem can be tackled. We refer to the first one inferred from the software tool as the *Solver Perspective*, the second one inferred from the approximate algorithms tool as the *Metaheuristic Perspective*, the third one inferred from the exact algorithms tool as the *Exact Perspective*, and the last one inferred from the ML and AI tool as the *Learning Perspective*. These perspectives in Figure 1.1 can be either combined or used separately to tackle large-scale problems.

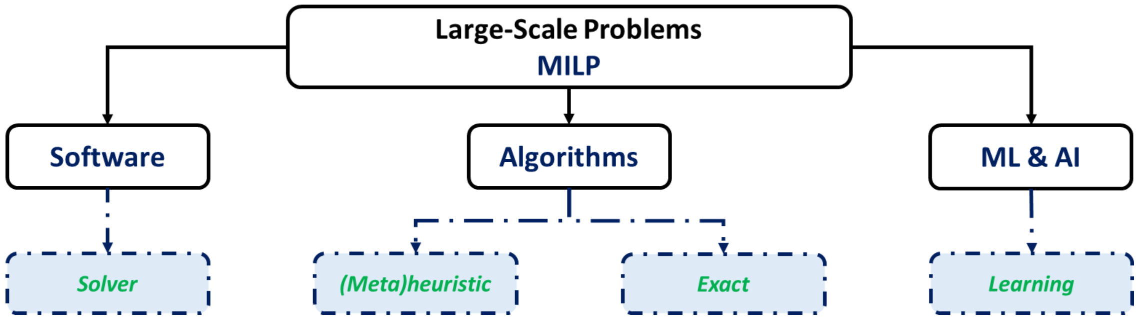


Figure 1.1 Optimization Age Four Perspectives

The tools highlighted in Figure 1.1 can be seen as a classification of methods for large-scale optimization. Indeed, when OR researchers face a large-scale optimization problem, they might give it to a commercial or open solver (*Solver Perspective*). They might also design a (meta)heuristic to solve it approximately (*Metaheuristic Perspective*). They might design an exact method to solve it exactly (*Exact Perspective*). More recently, they might either combine ML with OR or use ML as a standalone to solve it quickly using the available primal information (*Learning Perspective*). Despite the efforts that have been made to push further the boundaries in large-scale optimization, there is still significant room for improvement and innovation. Indeed, the existing tools can still be leveraged to reach high-performing mathematical optimization solutions. Thus, in this thesis, I explore the four perspectives in

Figure 1.1 to push the boundaries in each of them and highlight their importance in primal large-scale optimization contexts.

1.2 Thesis Contribution

In this thesis, I explore primal approaches within the four perspectives of OR to tackle very large-scale optimization problems. By doing so, I hope to push further the boundaries in one of the main branches of mathematical optimization in the context of large-scale problems, i.e., MILP. The most notable contributions of this dissertation are as follows:

- *Solver* Perspective:
  - Introducing a new primal multi-phase tuner based on the iterated local search meta-heuristic. This tuner addresses the parameter configuration problem for MILP solvers that are used to solve challenging large-scale optimization problems, which state-of-the-art tuners cannot tackle.
  - Highlighting that, instead of tuning in the entire configuration space induced by the parameter set, the multi-phase tuner focuses on a small parameter pool that is dynamically enriched with new promising parameters.
  - Leveraging the gathered knowledge during the search using statistical learning to forbid less promising parameter combinations.
  - Reporting that the new primal tuner outperforms state-of-the-art tuners and yields multiple (quasi-)optimal solutions in a few minutes.
- *Metaheuristic* Perspective:
  - Presenting a multiobjective, mixed-integer linear programming (MILP) model that integrates production scheduling, inventory management, and vessel assignment for a global supply chain.
  - Designing and conducting an exploratory analysis to identify the MILP complexity sources.
  - Designing a novel primal variant of the large neighborhood search (LNS) meta-heuristic to tackle the problem efficiently, which the standard LNS cannot solve in a reasonable amount of time. This metaheuristic leverages symmetry, considered an issue in the literature, as a practical benefit from which we can profit instead of breaking.

  - Showing that the metaheuristic reaches near-optimal solutions in less than 10 minutes in real-world instances.

- *Exact* Perspective:
  - Proposing the primal Benders decomposition (PBD) for sparse very large-scale problems, for which most complicating variables are equal to zero in the optimal solution.
  - Proving that this method converges theoretically to optimality and tackles several shortcomings of Benders decomposition, including the zigzagging behavior and the slow convergence.
  - Highlighting that this method, a paradigm shift, uses the PBD master problem to select the complicating variables to insert in the PBD subproblem, which is a restriction of the original problem.
  - Reporting the PBD potential on instances of the facility location problem, for which the number of cuts is reduced drastically for instances of up to 2000 instances. The method also achieves promising results on a very large-scale problem that motivated this research.

- *Learning* Perspective:
  - Designing a generic and scalable primal resilience re-optimization framework. This framework tackles the costly process of re-optimization from scratch, especially in very large-scale contexts.
  - Modeling perturbations, recovery decisions, and the resulting re-optimization problem, which maximizes resilience.
  - Leveraging the primal information through fixing, warm-start, valid inequalities, and machine learning.
  - Showing that, in several cases, local optimization is enough to recover after perturbations and that the re-optimized solution maintains 75% of the perturbed solution.

- The *All-In-One* Perspective:
  - Operationalizing the downstream logistics planner system at OCP Group. The OR team equipped the system with various heuristic and exact operations research tools from the four perspectives.

- Implementing a novel hybrid variant of Benders decomposition inspired by the practice, which consists of fixing some complicating variables related to confirmed orders and freeing others related to unconfirmed orders in the Benders subproblem.
- Highlighting that the system has become central to the OCP planning process. Planners use the optimizer's solutions and insights to improve plans in different OCP sites.
- Reporting that the OCP management now credits the system operationalization with providing operational benefits, contributing to over a $240 million increase in annual turnover.

Most of our developments are generic, scalable, and applicable to various large-scale optimization problems. Indeed, the insights presented can be easily transferred, adapted, and leveraged to tackle a variety of difficult large-scale problems that can be modeled as MILP. To be both theoretically and practically relevant, our research is designed and implemented while trying to solve very large-scale real-world problems. In particular, to analyze the computational performance of our developments, I consider a difficult large-scale optimization problem provided by *OCP Group* [24], one of the largest mining companies in the world. It integrates production scheduling, inventory management, and vessel assignment (PSIMVA), and it is described in detail in Chapter 4. Furthermore, several experiments are conducted on instances from Hydro-Quebec, the mixed integer programming library (MIPLIB) [25], the OR library [26], and others [27].

1.3 Thesis Organization

The studies presented in this thesis have yielded five scientific articles as follows (* refers to the corresponding author).

1. Himmich Ilyas, Er Raqabi El Mehdi*, El Hachemi Nizar, El Hallaoui Issmaïl, Metrane Abdelmoutalib and Soumis François (2023). *MPILS: An Automatic Tuner for MILP Solvers.* **Computers & Operations Research**, 159, 106344.

2. Er Raqabi El Mehdi*, Himmich Ilyas, El Hachemi Nizar, El Hallaoui Issmaïl and Soumis François (2023). *Incremental LNS Framework for Integrated Production, Inventory, and Vessel Scheduling: Application to a Global Supply Chain.* **Omega**, 116, 102821.

3. Er Raqabi El Mehdi*, El Hallaoui Issmaïl and Soumis François (2024). *The Primal Benders Decomposition.* Submitted to **Operations Research**.

4. Er Raqabi El Mehdi*, Wu Yong, El Hallaoui Issmaïl and Soumis François (2024). *Towards Resilience: Primal Large-scale Re-optimization.* **Transportation Research Part E: Logistics and Transportation Review**, 192, 103819.

5. Er Raqabi El Mehdi*, Beljadid Ahmed, Mohammed Ali Bennouna, Rania Bennouna, Latifa Boussaadi, Nizar El Hachemi, Issmaïl El Hallaoui, Michel Fender, Anouar Jamali, Nabil Si Hammou and François Soumis (2024). *OCP Optimizes its Supply Chain for Africa.* To appear in **INFORMS Journal on Applied Analytics**.

The remainder of this thesis is organized as follows. Chapter 2 presents the relevant literature review with regard to the four perspectives. Chapter 3 constitutes the first essay of this dissertation. It addresses the first perspective, i.e., the *solver* perspective. The second essay of this thesis is presented in 4. It addresses the second perspective, i.e., the *metaheuristic* perspective. Chapter 5, forming the third essay of this dissertation addresses the third perspective, i.e., the *exact* perspective. The fourth essay of this research is highlighted in 6 and explores the fourth perspective, i.e., the *learning* perspective. Chapter 7, the fifth essay of this thesis, groups all developed heuristic and exact approaches in an all-in-one perspective system implemented at OCP Group. Chapter 8 summarizes the thesis results, discusses the limitations, and opens windows for future research.

CHAPTER 2 LITERATURE REVIEW

This chapter provides an overview of primal large-scale MILP optimization from the four OR perspectives introduced in Chapter 1. I present the relevant literature to the four perspectives before positioning the thesis.

2.1 Solver Perspective

Over the last years, commercial and open-source solvers have made huge progress. Many OR researchers rely on these solvers to tackle MILP problems. One of the prominent ways to boost a solver's performance is by tuning its parameters. In our context, we limit ourselves to this important niche within the solver perspective and present in what follows an overview of MILP solver tuners.

Many tuners are problem-dependent algorithms, i.e., benefiting from the specific knowledge about an algorithm to tune and its application areas. Some examples include the performance analysis of the Max-Min Ant System (MMAS) heuristic with an application to the traveling salesman problem [28] and the automatic tuning of the Greedy Randomized Adaptive Search Procedure (GRASP) [29]. In this study, we are more interested in problem-independent algorithms; therefore, we must first distinguish between model-based and model-free algorithms.

Model-based algorithms use explicit statistical models for studying the dependence of the target algorithms on their parameters and include the Sequential Model-Based Optimization (SMBO) and Sequential Parameter Optimization (SPO) approaches.

The SMBO approach consists of iterating between the construction of statistical models using the available data and their utilization to inspect the promising regions of the search configuration space through the quantification of important parameter interactions. The SMBO paradigm has undergone several advances in recent years [30, 31, 32] with the Sequential Model-based Algorithm Configuration (SMAC) being the state-of-art version [33, 34]. The latter is the first extension of the SMBO to general algorithm configuration problems, including categorical parameters and several test instances. This generalization was tested on the CPLEX MILP solver and has significantly improved on the speed of the default configuration and the configuration returned by the CPLEX tuning tool. The most recent version is SMAC3, which offers a robust and flexible framework for Bayesian Optimization and can improve performance within a few evaluations. Another method is the Selection Tool for Optimization Parameters (STOP) developed by [35], which uses software testing and machine learning ideas and relies on observing that the MILP solver performance might improve if a few parameters take specific values. It starts by generating an initial set of configurations using different techniques such as random generation, greedy heuristics, and pairwise coverage. The quality of these configurations is evaluated using sequential runs of the algorithm to tune. A study by [36] shows empirically that the tuning of parameters offers opportunities to improve the performance of CPLEX on a large set of MIPLIB instances.

The SPO approach starts with constructing the initial configuration using Latin Hypercube Sampling (LHS). The parameter value interval is divided into equal intervals. Then, a random number is chosen from each interval to generate configurations. The performance of each generated configuration is measured after some executions, and the best-performing configuration is selected as the initial configuration. Then, a stochastic Gaussian model is run to estimate the algorithm performance. This model is updated after each iteration based on the best-found configuration, which is then used in the subsequent iteration. This approach incorporates several variants, such as SPO+ [31], which is fully automated and more robust compared to the original SPO [30]; and Time-Bounded SPO (TB-SPO) [32], which reduces computational overheads.

Model-free algorithms, on the other hand, incorporate several techniques that belong to four classes: Design of Experiments (DoE), Racing, Iterated Local Search (ILS), and Genetic Algorithms (GA).

The DoE approach is a well-established method for planning experiments. It consists of collecting data before analyzing it by statistical methods to draw valid conclusions. In our context, DoE decomposes parameter space to apply automated tuning procedures efficiently. It is widely used to determine the best parameter setting of metaheuristics. CALIBRA is an example of a DoE tuner [37].

The Racing approach sequentially evaluates candidate configurations and discards poor ones as soon as statistically sufficient evidence is gathered against them. The inferior candidates' elimination speeds up the procedure and allows for the evaluation of promising configurations in more instances and the obtaining of more reliable estimates of their behavior [38]. This approach includes variants such as F-Race, Iterated F-Race (I/F-Race) [39], and the Elitist Iterated Racing developed within the irace package [40]. The race tuner has been used to tune optimization solvers such as CPLEX and SCIP [41, 42, 43, 44].

The ILS approach consists of an iterative call of local searches starting from new solutions obtained using a perturbation of a previously found local optimum. Within this approach, ParamILS [45] is state-of-art and has proven its ability to configure CPLEX effectively for solving a variety of MILP instances [45]. The ParamILS tuner is suitable for tuning any algorithm regardless of the number of parameters, and, according to the official website, it performs well in various academic and industrial applications. In particular, it substantially speeds up the resolution of complex combinatorial problems. These problems include propositional satisfiability (SAT), answer set programming (ASP), and timetabling. Furthermore, the ParamILS tuner has helped several systems win solver competitions [46, 47, 48].

The GA approach relies on the GA metaheuristic, commonly used to reach high-quality solutions to optimization problems by relying on biologically inspired operators such as mutation, crossover, and selection. Within this approach, the Gender-based Genetic Algorithm (GGA) is a powerful method that uses the concept of competitive and noncompetitive genders in generating candidate configurations [49].

Table 2.1 Summary of MILP Configuration Algorithms

Method	Type	References	Approach
SMBO	Model-based	[30, 50, 31, 32, 33]	Statistical models
STOP	Model-based	[35, 36]	Software testing and ML
(Iterated) F-Race	Model-free	[39]	Statistical selection
GBGA	Model-free	[49]	Parallel genetic algorithm
ParamILS	Model-free	[46, 47]	Iterated local search
irace	Model-free	[40]	Iterated racing procedure

I summarize the main automatic parameter tuning algorithms in Table 2.1. Among all the highlighted methods, SMAC, irace, ParamILS, and GGA are the most popular [51]. Further details are available in surveys [52, 53, 54].

2.2 Metaheuristic Perspective

When dealing with large-scale problems, which are usually NP-hard, (meta)heuristics are used extensively to either generate initial feasible solutions or reach near-optimal (if not optimal) solutions. Many surveys are presenting these techniques in different contexts [55, 56, 57, 58]. In our context, we limit ourselves to two famous metaheuristics I used in some of the thesis essays: the large neighborhood search (LNS) and the iterated local search (ILS) metaheuristics.

2.2.1 The LNS Metaheuristic

In the last 15 years, heuristics based on LNS and the variant adaptive large neighborhood search (ALNS) have become some of the most successful paradigms for solving various large-scale optimization problems. Large neighborhood search methods explore a complex neighborhood through the use of heuristics. Using large neighborhoods makes it possible to find better candidate solutions in each iteration and hence follow a more promising search path. LNS was introduced by [59] for solving the capacitated vehicle routing problem with time windows. Its mechanism is to iteratively destroy and repair a solution in order to improve it by alternating between an infeasible solution and a feasible solution. The destroy operation creates an infeasible solution which is brought back into a feasible form by the repair heuristic. LNS is suitable for problems that can be decomposed into smaller sub-problems that can be destroyed and then repaired. I refer to [60] and [61] for a thorough description of the method.

2.2.2 The ILS Metaheuristic

The essence of the ILS metaheuristic is to build a chain of solutions generated by the embedded heuristic instead of executing repetitive random calls of it. This insight [62] is quite old and was rediscovered and given different names than ILS by several researchers such as chained local optimization [63], large-step Markov chains [64], iterated descent [65], and iterated Lin-Kernighan [66]. Further historical insights are provided in the comprehensive review by Johnson and McGeoch [67]. In addition to its theoretical simplicity, ILS is malleable and very successful in practice. Belonging to the local search category, it has been widely used if not the most frequently used embedded heuristic in several applications including the traveling salesman problem (TSP) [68, 69], scheduling problems [70, 71], and graph bipartitioning [72]. It supported the achievement of many state-of-the-art results without requiring too much problem-specific knowledge. This explains why ParamILS developers opted for it as well as its potential to tackle the PCP.

Algorithm 1: ILS Procedure

```
1 $\theta_0$ = GenerateInitialSolution
2 $\theta^*$ = LocalSearch($\theta_0$)
3 repeat
4     $\theta'$ = Perturbation($\theta^*$, history)
5     $\theta^{*'}$ = LocalSearch($\theta'$)
6     $\theta^*$ = AcceptanceCriterion($\theta^*$, $\theta^{*'}$, history)
7 until termination condition met;
```

The idea of ILS consists of an iterative call of local searches starting from new solutions obtained using a perturbation of the previously found local optimum. As presented in the high-level Algorithm 1, it starts with the *GenerateInitialSolution* function, which initializes the procedure with an initial solution that can be the default or random configuration in the case of the PCP. Then, it iterates over a combination of three elements. First, a local search rule (*LocalSearch* function), which explores the neighborhood of a given solution θ. Second, a perturbation mechanism (*Perturbation* function), which allows the escape from local optima through a predefined number of random moves. Third, an acceptance criterion (*AcceptanceCriterion* function) based on which a solution θ is accepted such as the cost, the integrality gap, etc. Furthermore, through iterations, a history is built and can be used in various ways including rejecting solutions, defining directions, etc. This is done using the *history* that keeps track of the accumulated knowledge through iterations. Once a termination condition is met, the procedure ends. Throughout iterations, only better or equally good parameter solutions are accepted. Further details are provided in Lourenço et al. [73].

2.3 Exact Perspective

To reach optimality on very large-scale optimization problems, exact methods are employed, especially decomposition techniques. The most famous decompositions are DWD [74], BD [75], and Lagrangian decomposition [76]. More details can be found in the book of Conejo et al. [77]. In our context, we limit ourselves to the first two decompositions. While I did not use DWD directly, it inspired several developments in this thesis.

2.3.1 Dantzig-Wolfe Decomposition

Theoretically, the need for decomposition techniques, felt in the early 1960s, evolved when column generation (CG), an algorithm constructed to solve LP problems having a huge number of variables and having a structure suitable for the DWD, was invented [74]. Still, the technique did not receive enough attention for more than 20 years until Desrosiers et al. [78] contributed to making it applicable. It was a breakthrough since he successfully applied it to tackle practical problems with millions of variables by solving sequences of smaller problems and tuning them until reaching the best possible solution. While CG succeeded, it was very sensitive to degeneracy on large problems, i.e., it often cycles indefinitely without improving the objective. To eliminate this phenomenon, increasing the combinatorial solution space was the dominating idea. Against this trend, Elhallaoui et al. [79] found it is possible to benefit from degeneracy. They suggested using it to reduce the combinatorics and thus accelerate the resolution. In another breakthrough, they contributed to making CG more practical. In the DWD, the main problem is decomposed into two problems. First, the master problem is also called the restricted master problem (RMP) since it contains fewer variables compared to the main one. Second, the subproblem(s) (SPs) help in identifying the interesting columns among others. Similarly to the simplex method, if a column with a negative reduced cost is provided by the SPs, it is added to the RMP and this process is repeated until no more columns can be added to the RMP making DWD an exact method. The method also takes benefit of the structure of the problem in order to decompose it into smaller problems. While CG is very efficient in many contexts, many other problems have different structures or no structure at all [80]. For instance, MILP is a well-known modeling structure for many real-life large-scale problems, which is not suited for CG.

2.3.2 Benders Decomposition

An efficient alternative to DWD is BD, which is another dual method (improves the dual bound) that partitions the problem into multiple smaller problems [75]. Since 1962, BD algorithm has been applied enormously to tackle complex and large scale MILP. It is based on a sequence of projection, outer linearization, and relaxation. First, the model is projected onto the subspace defined by the set of complicating variables. Second, the resulting formulation is then dualized, and the associated extreme rays and points define the feasibility cuts and the optimality cuts respectively. An equivalent formulation can be built by enumerating all the extreme points and rays. However, performing this enumeration is generally computationally untractable given the possible exponential number or extreme rays and points. Hence, one solves the equivalent model by applying a relaxation strategy to the feasibility and optimality cuts, yielding a MP and a subproblem, which are iteratively solved to respectively guide the search process and generate the violated cuts. By doing so, the method smartly exploits the constraint matrix structure of the problem and decomposes it into smaller problems.

Given its practical relevance, BD has been applied in many fields, including production routing [81], electric vehicles [82], airline scheduling [83, 84], water resource management [85], on-demand delivery [86], hub location [87], locomotive assignment [88], traveling salesman [89], vessel service planning [90], capacity expansion [91], and budgeting [92].

Despite its successes, BD is time-consuming, has a zigzagging behavior, and converges slowly. Intense research has been conducted to accelerate BD's convergence. These efforts can, intuitively, be classified into two sides. The first side deals with improving the LBs provided by the RMP, while the second seeks to improve the UBs obtained from the BD PSP. On the first side, intense research has been developed to select *good* or strengthen Benders cuts [93, 94, 95, 27, 96, 97] leading to better LBs. Other acceleration techniques include valid inequalities, warm-starting, managing the branch-and-bound (B&B) tree, and solving in two phases, *i.e.*, generating first cuts from relaxed (BD MP) and then cuts from integer (BD MP). On the second side, as far as I acknowledge, there is no systematic way to generate high-quality UBs. So far, problem-specific heuristics have been used to improve the UBs [98]. Accelerating BD convergence requires a double effort to get the lower and upper bounds close to each other as *fast* as possible. Degeneracy and symmetry amplify this double effort, and the BD might get lost and not converge for sparse very large-scale problems, i.e., large-scale problems for which most complicating variables are equal to zero in the optimal solution (implying high degeneracy). An exhaustive literature review by Rahmaniani et al. [98] highlights the state of the art of BD application, challenges, and improvement strategies, especially for combinatorial optimization (CO).

2.4 Learning Perspective

Over the last decade, applying ML to accelerate (re-)optimization processes gained more attention and focus from both operations researchers and ML practitioners. More recently, the trend has been to revolutionize decision-making at massive scales by fusing AI and OR to deliver scientific breakthroughs that the two fields cannot achieve independently [99].

Bengio et al. [100] survey the recent attempts, both from the ML and operations research (OR) communities, at leveraging ML to solve CO problems. They support pushing further the integration of ML and CO and detail a methodology to do so. The main point of the paper is to see generic optimization problems as data points and inquire what is the relevant distribution of problems to use for learning on a given task. With the current trend, several works on learning for MILPs are emerging. Learning can be applied in several ways. For conciseness purposes, we restrict ourselves to four types of learning. First, it can be applied to learn primal heuristics as in [101, 102, 103, 104]. Second, it can be applied to learn branching policies as in [105, 106, 107, 108]. Third, it can be used to learn to cut as in [109]. Fourth, learning can support MILP solvers configuration as in [110].

In the context of large-scale optimization, we distinguish several papers that rely on ML to boost optimization. Morabit et al. [111] develop a column selection approach and apply it to select a subset of the columns generated at each iteration of CG. The goal is to reduce the computing time spent re-optimizing the RMP at each iteration by selecting the most promising columns. Tahir et al. [112] propose a new integral column generation algorithm for the crew pairing problem, which leaps from one integer solution to another until a near-optimal solution is found. They rely on reduced SPs containing only flight connections that have a high probability of being selected in a near-optimal solution. They predict these probabilities using a deep neural network trained in a supervised framework. Quesnel et al. [113] propose a new B&P algorithm for the aircrew rostering problem. In their pricing SPs, they only include pairings likely to be in a near-optimal solution. They design a neural network to predict the probability that a pairing is assigned to a crew member. Then, they use those predictions to select which pairings to include in each subproblem. Morabit et al. [114] propose a new ML-based pricing heuristic. By taking advantage of the data collected during previous executions, the objective is to reduce the size of the network and accelerate the SPs. Computationally, they achieve up to 40% reductions in computational time. All these papers highlight that using ML enhance significantly the ability to optimize large-scale problems efficiently.

2.5 Thesis Positioning

This thesis comes as an attempt to highlight the benefits and advantages of the primal large-scale optimization from the four OR perspectives. I position the thesis within the four perspectives as follows.

In the *Solver* perspective, the state-of-the-art tuners do not profit from the primal information. As a result, they do not perform well on very large-scale optimization problems. Thus, I fill this gap by developing a new primal tuner that profits from primal information and performs more efficiently on these problems.

In the *Metaheuristic* perspective, the same observation holds since many metaheuristics partially benefit from the available information and require a lot of time to converge towards (near-)optimal solutions. Thus, I fill this gap by developing a new primal variant of the LNS metaheuristic.

In the *Exact* perspective, many decomposition methods are dual. In particular, BD is a dual method that does not profit from the primal information, and consequently shows a zigzagging behavior and converges slowly. Inspired by the DWD, I develop the primal Benders decomposition. Contrary to BD, PBD profits from the available information, does not zigzag, and converges more rapidly.

In the *Learning* perspective, I leverage the primal information to quickly re-optimize after perturbations. This primal view on re-optimization comes as a paradigm shift from the re-optimization from scratch, which is dominant in several contexts.

CHAPTER 3 ARTICLE 1: MPILS: AN AUTOMATIC TUNER FOR MILP SOLVERS

Authors: Ilyas Himmich, El Mehdi Er Raqabi, Nizar El Hachemi, Issmaïl El Hallaoui, Abdelmoutalib Metrane, François Soumis

Submitted on February 16, 2022 and published on June 30, 2023 in Computers & Operations Research, Volume 159, 106344, DOI: 10.1016/j.cor.2023.106344

Abstract. The parameter configuration problem consists of finding a parameter configuration that gives a particular algorithm the best performance. This paper introduces a new multi-phase tuner based on the iterated local search meta-heuristic. This tuner addresses the parameter configuration problem for deterministic MILP solvers that are used to solve challenging industrial optimization problems. Further, the proposed tuner offers a new search strategy based on three ideas. First, instead of tuning in the entire configuration space induced by the parameter set, the multi-phase tuner focuses on a small parameter pool that is dynamically enriched with new promising parameters. Second, it leverages the gathered knowledge during the search using statistical learning to forbid less promising parameter combinations. Third, it tunes on a single instance provided by earlier clustering of MILP instances. A computational study on the widely-used commercial solver CPLEX with instances from the MIPLIB library and a real large-scale optimization problem highlights the promising potential of the tuner.

Keywords. *Parameter Configuration Problem*, *Automatic Algorithm Configuration*, *MILP Solvers*, *Metaheuristics*, *Machine Learning*, *CPLEX*.

History. This article is published in *Computers & Operations Research*.

3.1 INTRODUCTION

The mathematical modeling of real-life optimization problems gives rise to complex, large-scale mixed-integer linear programs (MILP) with conflicting integer and binary variables. To address these MILP problems, MILP solvers require long computational times to return satisfactorily feasible solutions. Furthermore, given that these algorithms are highly parameterized, manual tuning becomes inconvenient. For instance, IBM ILOG CPLEX, the most widely used commercial optimization tool for solving MILP problems, offers a total of 159 user-specifiable parameters which induce a space of up to 4.75×10^{46} possible parameter combinations [45]. Therefore, automatic parameter tuning can be seen as one of the most practical remedies to reduce the time needed and improve the solution quality returned by MILP algorithms, for which the performance depends on the parameter combination used.

The choice of the best parameter combination is an NP-hard optimization problem known in the literature as the parameter configuration problem (PCP). Given a possibly infinite set of instances from which we know only a small subset of instances, the PCP consists of finding a parameter combination that optimizes the expected algorithm (MILP solver in our case) performance in the set of all instances [38]. Formally, let us consider a possibly infinite set $\mathcal{I}$ of instances, a highly parameterized algorithm, an index set $\mathcal{P}$ of parameters and their possible values $\mathcal{O}_p = \{o_p^1, o_p^2, ..., o_p^{|\mathcal{O}_p|}\}$, o_p^1 being the default value of a parameter with index $p \in \mathcal{P}$. The goal is to find the best combination of these possible parameter values in which the algorithm achieves its maximal performance on the given instance space based on a cost criterion. A combination of parameter values is called a *configuration* (or *setting*) and is denoted $\theta = (o_1, o_2, ..., o_p, ..., o_{|\mathcal{P}|})$, such that $\theta_p = o_p \in \mathcal{O}_p$, $\forall p \in \mathcal{P}$. We omit the superscript to alleviate the notation. In particular, we use θ^0, $\theta^d = (o_1^1, o_2^1, ..., o_{|\mathcal{P}|}^1)$, and θ^* to refer to the initial, default, and best configurations, respectively. The space of all possible configurations denoted Ω corresponds to the solution space of the tuning problem. Therefore, a parameter tuner aims at finding $\theta^* = \arg\min_{\theta \in \Omega} c(\theta)$, which is the best configuration given a certain cost function $c(.)$.

To tackle efficiently large-scale MILP problems, we propose a new automatic tuner for MILP solvers. Our tuner iterates over three steps: a *Tuning* step, a *Learning* step, and an *Evaluation* step. Instead of considering the whole combinatorial space Ω, the proposed tuner starts with an initial pool of parameters and tunes over them in the *Tuning* step. It then explores Ω through a dynamic pruning/insertion process. Pruning consists of discarding deteriorating parameter values via the *Learning* step, where the tuner extracts useful information from history using statistical learning techniques. Insertion consists of inserting new parameters via the *Evaluation* step, where the tuner identifies promising parameters to tune. The cycle continues until no promising parameters remain for insertion.

The present article has a fourfold contribution: (1) We develop a model-free multi-phase iterated local search (MPILS) tuner for deterministic MILP solvers where an initial pool of parameters is selected a priori based on MILP problem analysis, (2) We highlight that learning accelerates the tuning process and the PCP resolution as a whole with an accumulation of "expertise" (in the sense of expert systems), (3) Under the MILP solvers' niche, we show that MPILS is competitive to widely used general tuners such as irace and ParamILS, and (4) We evaluate the performance of our proposed tuner on a complex real-life problem and several instances from the MIPLIB library.

The following section gives an overview of the relevant literature. Section 3 presents the proposed MPILS tuner. In Section 4, we highlight the computational study, and we provide conclusions in Section 5.

3.2 LITERATURE REVIEW

In this section, we first review the relevant PCP literature before positioning our paper.

3.2.1 Parameter Configuration for MILP Problems

Many tuners are problem-dependent algorithms, i.e., benefiting from the specific knowledge about an algorithm to tune and its application areas. Some examples include the performance analysis of the Max-Min Ant System (MMAS) heuristic with an application to the traveling salesman problem [28] and the automatic tuning of the Greedy Randomized Adaptive Search Procedure (GRASP) [29]. In this study we are more interested in problem-independent algorithms; therefore, we must first distinguish between model-based and model-free algorithms.

Model-based algorithms use explicit statistical models for studying the dependence of the target algorithms on their parameters, and include the Sequential Model-Based Optimization (SMBO) and Sequential Parameter Optimization (SPO) approaches.

The SMBO approach consists of iterating between the construction of statistical models using the available data and their utilization to inspect the promising regions of the search configuration space through the quantification of important parameter interactions. The SMBO paradigm has undergone several advances in recent years [30, 31, 32] with the Sequential Model-based Algorithm Configuration (SMAC) being the state-of-art version [33, 34]. The latter is the first extension of the SMBO to general algorithm configuration problems, including categorical parameters and several test instances. This generalization was tested on the CPLEX MILP solver and has significantly improved on the speed of the default configuration and the configuration returned by the CPLEX tuning tool. The most recent version is SMAC3, which offers a robust and flexible framework for Bayesian Optimization and can improve performance within a few evaluations. It also provides facades and pre-sets for typical use cases, such as optimizing hyperparameters, solving low dimensional continuous (artificial) global optimization problems, and configuring algorithms to perform well across multiple problem instances [115]. Another method is the Selection Tool for Optimization Parameters (STOP) developed by [35], which uses software testing and machine learning ideas and relies on observing that the MILP solver performance might improve if a few parameters take specific values. It starts by generating an initial set of configurations using different techniques such as random generation, greedy heuristics, and pairwise coverage. The quality of these configurations is evaluated using sequential runs of the algorithm to tune. A study by [36] shows empirically that the tuning of parameters offers opportunities to improve the performance of CPLEX on a large set of MIPLIB instances.

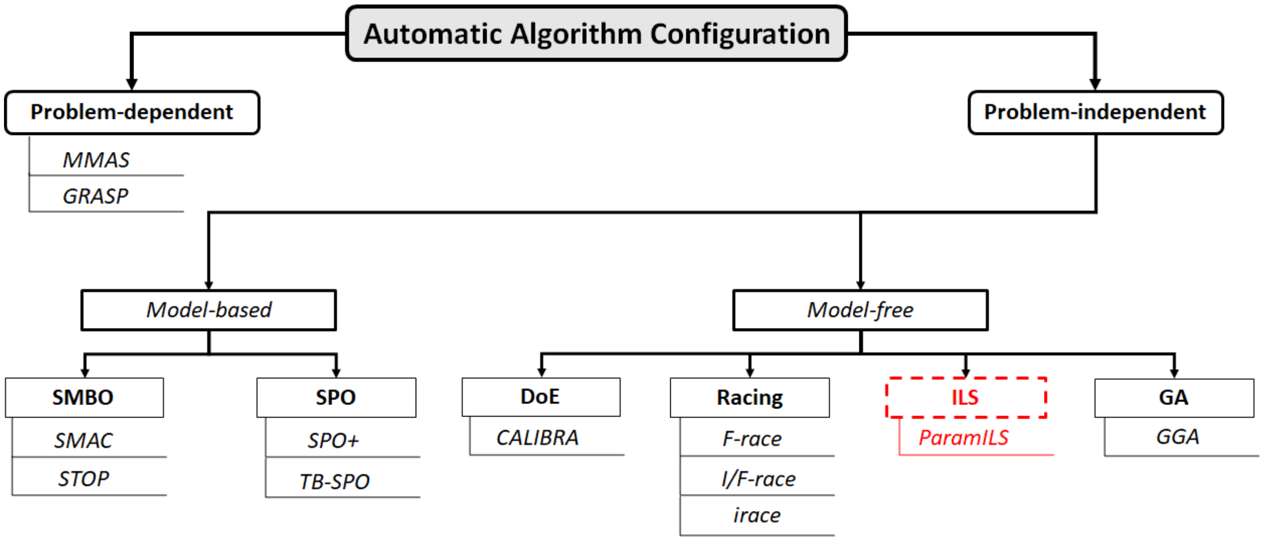


Figure 3.1 Literature Review Summary

The SPO approach starts with constructing the initial configuration using Latin Hypercube Sampling (LHS). The parameter value interval is divided into equal intervals. Then, a random number is chosen from each interval to generate configurations. The performance of each generated configuration is measured after some executions, and the best-performing configuration is selected as the initial configuration. Then, a stochastic Gaussian model is run to estimate the algorithm performance. This model is updated after each iteration based on the best-found configuration, which is then used in the subsequent iteration. This approach incorporates several variants, such as SPO+ [31], which is fully automated and more robust compared to the original SPO [30]; and Time-Bounded SPO (TB-SPO) [32], which reduces computational overheads.

Model-free algorithms, on the other hand, incorporate several techniques that belong to four classes: Design of Experiments (DoE), Racing, Iterated Local Search (ILS), and Genetic Algorithms (GA).

The DoE approach is a well-established method for planning experiments. It consists of collecting data before analyzing it by statistical methods to draw valid conclusions. In our context, DoE decomposes parameter space to apply automated tuning procedures efficiently.

It is widely used to determine the best parameter setting of metaheuristics. CALIBRA is an example of a DoE tuner [37].

The Racing approach sequentially evaluates candidate configurations and discards poor ones as soon as statistically sufficient evidence is gathered against them. The inferior candidates' elimination speeds up the procedure and allows for the evaluation of promising configurations in more instances and the obtaining of more reliable estimates of their behavior [38]. This approach includes variants such as F-Race, Iterated F-Race (I/F-Race) [39], and the Elitist Iterated Racing developed within the irace package [40]. irace has been used to tune optimization solvers such as CPLEX and SCIP [41, 42, 43, 44].

The ILS approach consists of an iterative call of local searches starting from new solutions obtained using a perturbation of a previously found local optimum. Within this approach, ParamILS [45] is state-of-art and has proven its ability to configure CPLEX effectively for solving a variety of MILP instances [45]. The ParamILS tuner is suitable for tuning any algorithm regardless of the number of parameters, and, according to the official website, it performs well in various academic and industrial applications. In particular, it substantially speeds up the resolution of complex combinatorial problems. These problems include: propositional satisfiability (SAT), artificial intelligence (AI) planning, answer set programming (ASP), and timetabling. Furthermore, the ParamILS tuner has helped several systems win solver competitions [46, 47, 48].

The GA approach relies on the GA metaheuristic, commonly used to reach high-quality solutions to optimization problems by relying on biologically inspired operators such as mutation, crossover, and selection. Within this approach, the Gender-based Genetic Algorithm (GGA) is a powerful method that uses the concept of competitive and noncompetitive genders in generating candidate configurations [49].

We summarize the presented automatic parameter tuning algorithms in Figure 3.1. Among all the highlighted methods, SMAC, irace, ParamILS, and GGA are the most popular [51]. Further details are available in surveys [52, 53, 54].

3.2.2 Paper Positioning

We position our paper within the ILS category, highlighted in red in Figure 3.1, in which ParamILS is state-of-art. The essence of the ILS metaheuristic is to build a sequence of solutions generated by the embedded heuristic instead of executing repetitive random calls. This concept [62] has been independently developed by several researchers over the years, who have labeled it as iterated descent [65], iterated Lin-Kernighan [66], and chained local optimization [63]. Further historical insights are provided in a comprehensive review [67]. In addition to its theoretical simplicity, ILS is malleable and very successful in practice. As a result, it has been widely used in several applications, including the traveling salesman problem (TSP) [116], scheduling problems [70, 71], and graph bipartitioning [72]. It has also supported many state-of-art results without requiring too much problem-specific knowledge.

The ILS algorithm consists of an iterative call of local searches starting from new solutions obtained using a perturbation of the previously found local optimum. As presented in the high-level algorithm 2, ILS requires an initial solution θ^0, which can be the default or a randomly generated configuration. The main loop of ILS iterates over three functions. First, a local search (*LocalSearch* function), which explores the neighborhood of a given configuration, is conducted. Second, a perturbation mechanism (*Perturbation* function) allows the escape from local optima through a predefined number of random moves. Third, the algorithm uses an acceptance criterion (*AcceptanceCriterion* function) based on which a solution θ is retained. Furthermore, the information collected during phases can be used in various ways, including rejecting solutions and defining directions. The *history* allows the user to keep track of the accumulated knowledge. Once a termination condition is met, the procedure ends, and the best configuration is θ^*. Further details regarding the ILS mechanism are provided in [73].

Algorithm 2: ILS(θ^0)

```
  Input: $\theta^0$
  Output: $\theta^*$
1 $\theta^* = \text{LocalSearch}(\theta^0)$
2 repeat
3     $\theta^{'} = \text{Perturbation}(\theta^*, \text{history})$
4     $\theta^{*'} = \text{LocalSearch}(\theta^{'})$
5     $\theta^* = \text{AcceptanceCriterion}(\theta^*, \theta^{*'}, \text{history})$
6 until termination condition met;
7 return $\theta^*$
```

While the ILS procedure can escape local optima and converge to satisfactory configurations for one or more instances, it requires a significant amount of time to achieve convergence for massively parametrized algorithms. This is also the case for the ParamILS tuner, which considers the whole configuration space Ω simultaneously, leading to high computational time to identify near-optimal configurations, if not the best ones. One of the possible remedies allowed in the ParamILS is to restrict the tuning to a small subset of parameters, which seems to be the most preferred for improving solver performance. However, this solution is often based only on the user's intuition, which can lead to risks related to the irreversible empirical selection of parameters and, in turn, significant losses in tuner improvement opportunities that utilize possible configurations of rejected parameters. Another possible remedy is allowing the user to disable, following an analysis of the results obtained, a subset of non-influential parameters or reduce the number of values associated with these parameters. Unfortunately, for MILP solvers, the ParamILS tuner does not allow this handling automatically and requires the users to perform the analysis themselves. To address these issues, we designed the MPILS tuner, an attractive alternative because of its simplicity and competitiveness with other state-of-the-art tuners. We recall that our tuner is designed for deterministic MILP Solvers. Under this niche, it is competitive to general tuners such as ParamILS and irace.

3.3 MULTI-PHASE ITERATED LOCAL SEARCH TUNER

In this section, we discuss our motivation for designing the MPILS. Then, we present an overview of the algorithm before providing a detailed description of its steps.

3.3.1 Motivation

The MPILS tuner is motivated by two main observations, presented in the Assumption below, which result from preliminary tests conducted on different MILP instances and the PCP literature [117, 35].

Assumption 1. *For a given MILP solver and a given problem instance, the following two key observations are made:*
(1) $\exists\, \mathcal{P}^{signi} \subset \mathcal{P}$ a small index subset of parameters that significantly impact the solver performance. The behavior of the solver does not change significantly $\forall p \in \mathcal{P} \setminus \mathcal{P}^{signi}$.
(2) $\forall p \in \mathcal{P}^{signi}$, $\exists\, \mathcal{O}_p^{signi} \subset \mathcal{O}_p$ a small subset of values that have in most cases a positive significant effect on the overall performance of the solver.

Based on the assumption, we can conclude the following: It is possible to implicitly explore the combinatorial space without considering it as a whole from the beginning, such as in the ParamILS tuner. This may allow a faster exploration of the search space. To achieve this, one may consider starting with a small number of parameters, tuning over them, discarding their deteriorating values and values combinations, inserting promising parameters, and reiterating until one is satisfied with the results. Such a procedure allows a dynamic pruning/insertion process, which maintains a reduced size of the tuner search space during the exploration process.

Another motivation is that automatic tuning has practical applications in many real-life large-scale MILP problems, whose solving requires many hours of configuration testing. For these problems, one should expect several days/weeks of tuning without interruption in cases of multiple instances. Thus, the idea was to consider clustering available instances from space $\mathcal{I}$ based on the industrial (demand volume, the season of the year, etc.) and mathematical (variables, constraints, etc.) structures, select one instance from each cluster, we hereafter call representative instance, and separately tune on each of these representative instances as a single training instance. This should result in a time reduction compared to tuning on all instances that are generated or will be generated in the future, especially considering that MILP solvers are exact algorithms known to have stable behavior in MILP instances with the same mathematical and industrial structures.

Tuning on a single instance is not appropriate for all cases. It is known, in general, that the better a meta-heuristic performs over the tuning instances, the worse it might do over a different instance of the same class, and therefore, on average, over the whole class [118]. This is also the case when instances change significantly from one year to another, as in the forestry context for instance, where we do not cut the same trees each year. Still, as initial experiments on the considered real-life large-scale problem have shown, there is a high chance that the best configuration obtained while tuning on a given instance performs well on same-cluster instances. We also expect this is the case for large-scale industrial optimization problems with fixed installations, standard processes, and repetitive trends. For such MILP problems, instances show periodicity, similarity, and repetitiveness from one year to another. Therefore, we designed the MPILS tuner for this kind of situation where the instances available from space $\mathcal{I}$ are clustered into m clusters such that $\mathcal{I} = \cup_{i=1}^{m} Cl_i$ before tuning on a single instance from each cluster.

3.3.2 MPILS Tuner Overview

The MPILS tuner is highlighted in blue in Figure 3.2. In this section, we describe its inputs, flows, role, and outputs. We also present briefly its steps, i.e., the *Tuning*, *Learning*, and *Evaluation* steps. We detail each step in the sections that follow.

Besides the preliminary notation presented in Section 3.1, we must provide additional notation to introduce the MPILS tuner. Given an index set of parameters $\mathcal{P}$, we denote by $\mathcal{T}$, $\mathcal{S}$, $\mathcal{R}$, and $\mathcal{D}$ the index set of parameters that are considered in the *Tuning* step, the index set of selected parameters to be added to $\mathcal{T}$, the index set of residual parameters, and the index set of discarded parameters, respectively. At each phase, we have $\mathcal{P} = \mathcal{T} \cup \mathcal{S} \cup \mathcal{R} \cup \mathcal{D}$. For each $p \in \mathcal{P}$, we define a set of active values $\mathcal{A}_p \subseteq \mathcal{O}_p$ that contains the values to tune on for p, $\mathcal{A} = \cup_{p \in \mathcal{P}} \mathcal{A}_p$. We denote by Θ and $\bar{\Theta}$ the set of tested configurations and the set of forbidden ones, respectively. We also introduce $\Omega^r = \{\theta \,:\, \theta_p \in \mathcal{A}_p, \forall p \in \{1, 2, ..., |\mathcal{P}|\}\}$, the reduced space of configurations, which corresponds to the set of all configurations that can be obtained from the set of active values $\mathcal{A}$, and $\mathcal{N}^r(\theta) = \{\theta' \in \Omega^r \setminus \bar{\Theta} \;:\; \exists! p \in \mathcal{T},\ \theta'_p \neq \theta_p\}$, the restricted neighborhood of a configuration θ, which is the set of all unforbidden configurations that differ from θ in a unique active value of a $p \in \mathcal{T}$. Any parameter with index $p \in \mathcal{R}$ is set to its default value o_p^1.

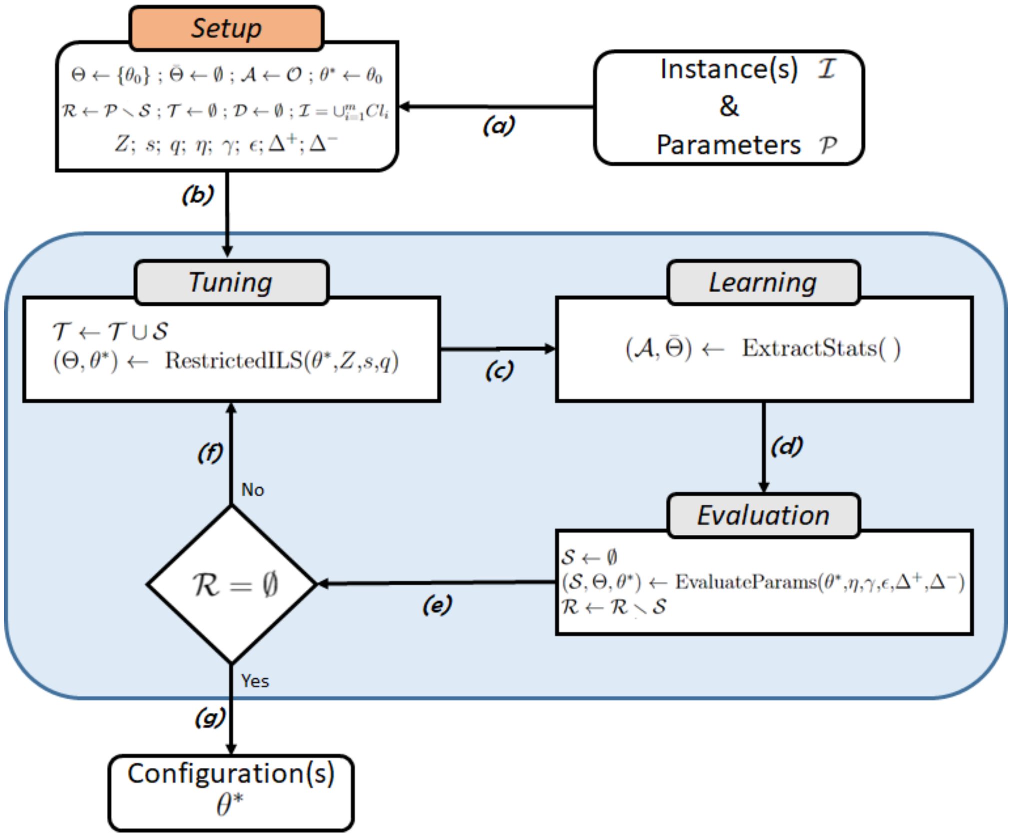

Setup
$\Theta \leftarrow \{\theta_0\}$; $\bar{\Theta} \leftarrow \emptyset$; $\mathcal{A} \leftarrow \mathcal{O}$; $\theta^* \leftarrow \theta_0$
$\mathcal{R} \leftarrow \mathcal{P} \setminus \mathcal{S}$; $\mathcal{T} \leftarrow \emptyset$; $\mathcal{D} \leftarrow \emptyset$; $\mathcal{I} = \cup_{i=1}^{m} Cl_i$
$Z;\ s;\ q;\ \eta;\ \gamma;\ \epsilon; \Delta^+; \Delta^-$
Instance(s) $\mathcal{I}$
&
Parameters $\mathcal{P}$
(a)
(b)
Tuning
$\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{S}$
$(\Theta, \theta^*) \leftarrow$ RestrictedILS(θ^*,Z,s,q)
(c)
Learning
$(\mathcal{A}, \bar{\Theta}) \leftarrow$ ExtractStats( )
(d)
Evaluation
$\mathcal{S} \leftarrow \emptyset$
$(\mathcal{S}, \Theta, \theta^*) \leftarrow$ EvaluateParams(θ^*,η,γ,ϵ,Δ^+,Δ^-)
$\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{S}$
(e)
$\mathcal{R} = \emptyset$
(f)
No
Yes
(g)
Configuration(s)
θ^*

Figure 3.2 MPILS Tuner in the blue frame with three steps: *Tuning* step, *Learning* step, and *Evaluation* step

Before going into the blue-framed MPILS tuner in Figure 3.2, we assume that the *Setup* for running the tuner has been completed. Given sets $\mathcal{I}$ and $\mathcal{P}$ (flow [a]), the *Setup* selects an initial pool of parameters added to set $\mathcal{S}$, partitions instances into m clusters ($\mathcal{I} = \cup_{i=1}^{m} Cl_i$), and initializes sets and parameters. The initial pool of parameters can be identified using manual or automatic troubleshooting, which benefits from the expertise developed within the OR community (see [119] and Appendix A). Troubleshooting, which is beyond the scope of this paper, can be done by running initial tests on representative instances and analyzing the obtained log files. From the latter analysis, we can infer an initial pool of parameters that can be used to construct an initial configuration θ^0. To the extent of our knowledge, there is no automatic troubleshooter for MILP Solvers. The MPILS tuner takes as input (flow [b]) one representative instance from each cluster (Cl_i), several parameters (Z, s, q, η, γ, ϵ, Δ^+, and Δ^-) to be presented in the relevant steps, and all the global sets (Θ, $\bar{\Theta}$, $\mathcal{S}$, $\mathcal{R}$, $\mathcal{T}$, $\mathcal{D}$, and $\mathcal{A}$). Then, for each representative instance, it iterates through three steps: the *Tuning* step, the *Learning* step, and the *Evaluation* step before outputting one or several satisfactory (if not the optimal) configurations (flow [g]). Each iteration is referred to as a phase in the paper.

The *Tuning* step tunes over all parameters with index $p \in \mathcal{T}$. This is done in a restricted search space using the *RestrictedILS* procedure presented in detail in Section 3.3.3. The history accumulated from all the tested configurations is provided (flow [c]) to the *Learning* step, which learns from its deteriorating values and values combinations. This learning is achieved through the *ExtractStats* procedure presented in detail in Section 3.3.4. The deteriorating values and values combinations are provided (flow [d]) to the *Evaluation* step, which evaluates the residual non-tested parameters using two metrics and selects the most promising ones, i.e., the ones that could improve the best configuration θ^*. The less promising parameters in the same step are discarded from the parameters set. The *EvaluateParams* procedure in charge of the evaluation is detailed in Section 3.3.5. As long as there are still residual parameters, i.e., $\mathcal{R} \neq \emptyset$, the most promising parameters, as well as the values and values combinations to forbid, are added (flow [e]) to the tuning pool for the next phase (flow [f]). Once $\mathcal{R} = \emptyset$, the process ends and the best configuration(s) found are returned (flow [g]).

3.3.3 Tuning Step

Instead of considering a huge combinatorial search space (including all the possible configurations) where the tuner is likely to become lost, the *Tuning* step consists of searching for satisfactory configuration(s) in a reduced search space induced by a small subset of parameters and their corresponding active values. As mentioned above, this search is conducted using the *RestrictedILS* procedure described in Algorithm 3. It takes as input the best configuration θ^*, an upper bound Z on the number of configurations to be tested at each *Tuning* step, an integer s, and a probability q. We recall that the global sets and parameters are initialized in the *Setup* (see Figure 3.2).

At each phase, the *RestrictedILS* procedure starts with a *LocalSearch* function that explores intensively $\mathcal{N}^r(\theta^*)$, the neighborhood of the current best configuration θ^* (θ^0 in the initial phase). It recursively updates θ^*, the set Θ of tested configurations, and the counter z, which counts the number of tested configurations in the current *Tuning* step (line 2 in Algorithm 3). We note that, in the case of large-scale optimization problems, measuring $c(\theta)$ with a

Algorithm 3: RestrictedILS(θ^*,Z,s,q)

```
   Input: $\theta^*$, $Z$, $s$, $q$
1  Initialization: $\theta^{'} \leftarrow \theta^*, z = 0$
2  $(\theta^*, z, \Theta) \leftarrow LocalSearch(\mathcal{N}^r(\theta^*), z, \Theta)$
3  $\theta^{'} \leftarrow \theta^*$
4  while $z \leq Z$ do
5      for $i \in 1..s$ do
6          $\theta^{'} \leftarrow Random(\mathcal{N}^r(\theta^{'}))$          /* Successive Mutations */
7      end
8      $(\theta^{'}, z, \Theta) \leftarrow LocalSearch(\mathcal{N}^r(\theta^{'}), z, \Theta)$   /* Intensification */
9      if $c(\theta^{'}) \leq c(\theta^*)$ then
10         $\theta^* \leftarrow \theta^{'}$                          /* Acceptance Criterion */
11     end
12     if $Random(0,1) < q$ then
13         $\theta^{'} \leftarrow Random(\Omega^r \setminus \{\theta^*\})$    /* Conditional Diversification */
14     end
15 end
```

MILP Solver is costly in terms of execution time. Thus, the first improving configuration found in $\mathcal{N}^r(\theta^*)$ becomes θ^*. The neighborhood of this first improving configuration found is explored again, and the same process continues until no further improvement is possible. This idea is in line with the fact that the initial and best configuration(s) differ in only a few parameters. For this reason, we do not need too many changes; an improvement will suffice. Once the initial *LocalSearch* is completed, a loop takes place to improve θ^* further (line 4 in Algorithm 3). We note $\theta^{'}$ the current configuration within the loop.

Within the loop, the *Random* mechanism (line 6 in Algorithm 3) helps in escaping local optima by locally diversifying the search: it simulates the change of some values in the current configuration through the execution of s successive mutations (a term borrowed from genetic algorithms). The number of mutations s is an upper bound regarding the number of parameters to change. The insight behind these mutations is the distance between the initial configuration and the best one(s), i.e., the number of values they differ from each other. Indeed, according to Assumption 1, they differ in a small number of parameters and, consequently, a small number of values. Once completed, the search is intensified in the neighborhood of the current configuration $\theta^{'}$ (line 8 in Algorithm 3) using the same *LocalSearch* function presented above. The cost $c(\theta^{'})$ is compared to the cost $c(\theta^*)$ (line 9 in Algorithm 3). From time to time, the best configuration is subject to a "conditional" diversification, which allows escaping valleys to be searched exhaustively and the exploration of new regions within the reduced search space Ω^r. Diversification takes place if the probability generated using $Random(0, 1)$ is lower than input q (line 12 in Algorithm 3). A q closer to 1 implies more diversification. The *RestrictedILS* procedure stops when the number of configurations tested at the current *Tuning* step reaches the predefined upper bound, i.e., $z = Z$. It is worth mentioning that the upper bound Z is proportional to the cardinal $|\mathcal{T}|$, i.e., $Z = \delta|\mathcal{T}|$.

We note that in this paper, the *RestrictedILS* procedure differs from the ParamILS tuner implementation of the ILS metaheuristic [73] in three main aspects. First, the MPILS tuner proposes to start the search at each tuning step with configuration θ^*, which has proven to be the most effective in the previous phases. Such a choice guides the tuner to search in the most prominent regions of the search space. Second, the tuning is performed on a restricted subset of parameters with indexes $p \in \mathcal{T}$ at each phase. We recall that set $\mathcal{T}$ is dynamically updated at the beginning of each *Tuning* step with all $p \in \mathcal{S}$, as presented in Figure 3.2. Third, the *RestrictedILS* procedure avoids testing configurations with deteriorating values and values combinations identified in the previous phases as follows: The deteriorating values are removed from the set $\mathcal{A}$ and as a result of both Ω^r and $\mathcal{N}^r$. Furthermore, any configuration that can be formed from the deteriorating values is added to set $\bar{\Theta}$. All the configurations in $\bar{\Theta}$ are discarded as per the definition of $\mathcal{N}^r$. This helps tighten the search space and is likely to save substantial computational effort.

3.3.4 Learning Step

At each phase, dozens of configurations are tested during the *Tuning* step. This provides the tuner with a considerable amount of data that can be explored to extract the effect of the tuned parameters and their interactions on the solver performance, which is achieved in the *Learning* step represented by the *ExtractStats*() procedure in Figure 3.2. This step is a statistical study that learns from the history of tested configurations the deteriorating values and values combinations to forbid. The aim is to reduce the search space size in the subsequent phases and to guide the tuner to regions of Ω where it is worth increasing the tuning effort. To do so, we remove, via the *Learning* step, deteriorating values and combinations of two values, referred to hereafter as *1-value* and *2-values*, respectively.

We describe the statistical study briefly before detailing each component in the next paragraphs. The history is organized in a configuration table with $|\mathcal{P}| + 1$ columns and $|\Theta|$ lines. Each column represents a parameter except the final one, which represents the cost. Each line is a tested configuration θ with its $c(\theta)$. The configuration table first goes through the *Data Preparation*. In this statistical study, the dependent variable, also called the target, is the cost $c(\theta)$, while the independent variables are parameters $p \in \mathcal{P}$. These parameters are independent since each one can be set to any of its values independently of the other parameters' values. The prepared table is provided to the *Statistical Test* to identify the independent variables that have the most significant impact on the cost. The resulting cost distribution allows for the construction of cost categories. The independent variables that have the greatest effect on the cost, as well as the cost categories, are provided as input to the *Statistical Model*. The latter allows for the identification of the *1-value* and *2-values* to forbid in subsequent phases. Formally, if $o_p \in \mathcal{O}_p$ for a given $p \in \mathcal{P}$ is a *1-value*, then all configurations of the form $(*, *, ..., o_p, ..., *, *)$ are forbidden, i.e., $\prod_{i \in \mathcal{P} \setminus \{p\}} |\mathcal{O}_i|$ configurations. Similarly, if $(o_p, o_{p'}) \in \mathcal{O}_p \times \mathcal{O}_{p'}$ is a *2-values* for a given pair $(p, p^{'}) \in \mathcal{P}^2$, then all configurations of the form $(*, *, ..., o_p, ..., o_{p'}, ..., *, *)$ are forbidden, i.e., $\prod_{i \in \mathcal{P} \setminus \{p,p'\}} |\mathcal{O}_i|$ configurations.

Data Preparation. For each $p \in \mathcal{P}$, each value $o_p \in \mathcal{O}_p$ represents a category. Consequently, data preparation consists of transforming the independent variables into the categorical type. Furthermore, given the large number of parameters and the implied large number of values, several parameters remain set to the default value, as is the case for parameters with $p \in \mathcal{R}$ during a given phase. Thus, given that each column corresponds to a parameter, the configuration table may contain fix columns, which can be removed since they do not have any impact on the learning. We recall that for $p \in \mathcal{P}$, the default value is o_p^1.

Statistical Test. For each representative instance of each cluster (Cl_i), there are parameters that have the greatest impact on the cost. To identify them, we conduct the widely used analysis of variance (ANOVA) [120], and specifically, a fractional factorial ANOVA [121], which is a statistical test that measures the effect of two or more categorical independent variables (parameters) on a single dependent variable (cost) based on two statistics, the *F-statistic* and the *p-value*. The first statistic assesses the equality of means when we have two or more groups. More details about the *F-statistic* can be found in [122]. It is fractional since the fully-factorial design is expensive computationally. In our context, the number of groups for $p \in \mathcal{P}$ is equal to the number of its active values, i.e., $|\mathcal{A}_p|$. For each value $o_p \in \mathcal{A}_p$, the cost mean is computed from all configurations with value $o_p \in \mathcal{A}_p$. Then, to identify the effect of a parameter with index $p \in \mathcal{P}$, we compare the computed means of its active values $o_p \in \mathcal{A}_p$. This comparison is ensured through the *F-statistic*, which measures the equality of means as follows: *F-statistic*$= \frac{between-groups\ variance}{within-group\ variance}$. A high *F-statistic* value implies that the means are not equal, i.e., changing the active values of the corresponding parameter affects the cost. The second statistic, which is widely used in the literature, is the probability that an observed difference could have occurred just by random chance. The lower its value, the greater the statistical significance of the observed difference. More details about the *p-value* can be found in [123]. In our context, the second statistic checks the statistical significance of the computed *F-statistic* for each $p \in \mathcal{P}$. After conducting the fractional factorial ANOVA, we rank the independent variables in decreasing order of the *F-statistic*. The statistically significant (*p-value* $\leq$ *threshold* where the commonly used significance *thresholds* are 1% or 5%) ones with the largest *F-statistic* values are shortlisted for the multinomial logistic regression. The indexes of the shortlisted parameters (independent variables) are added to set $\mathcal{P}^{signi}$ introduced in Assumption 1.

Cost Distribution. Since our objective is identifying *1-value* and *2-values*, i.e., the ones leading to a *high* cost rather than a specific cost value, it is more practical to transform the cost, originally a quantitative variable, into a categorical variable having $n \in \mathbb{N}^*$ categories. Category 1 corresponds to the lowest cost values, while category n corresponds to the highest cost values. Intermediate cost values lie within intermediary categories ranging from 2 to $n-1$. This transformation can be achieved using rank-transformation or data classification techniques such as quantiles or equal intervals [124]. It can also be done based on the context specifications.

Statistical Model. Since it is more likely to have more than two cost categories, the suitable statistical model is the multinomial logistic regression [125]. This regression is a natural extension of the binary logistic regression widely used to predict the probability of category membership of a dependent variable based on multiple independent variables and their two-by-two interactions [126]. To design this regression, for each statistically significant independent variable with index $p \in \mathcal{P}^{signi}$, we introduce $|\mathcal{O}_p| - 1$ binary variables X_{pk}, $k \in \{2, 3, ..., |\mathcal{O}_p|\}$, and for each pair of statistically significant independent variables with indexes $(p, p^{'}) \in \mathcal{P}^{signi^2}$, we introduce $(|\mathcal{O}_p| - 1) \times (|\mathcal{O}_{p'}| - 1)$ variables $W_{pkp'k'}$, $k \in \{2, 3, ..., |\mathcal{O}_p|\}$ and $k^{'} \in \{2, 3, ..., |\mathcal{O}_{p'}|\}$. The values of these binary variables depend on the considered configuration. For a given configuration θ, we have the following:

$$X_{pk} = \begin{cases} 1, & \text{if } \theta_p = o_p^k \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$W_{pkp'k'} = \begin{cases} 1, & \text{if } \theta_p = o_p^k \text{ and } \theta_{p'} = o_{p'}^{k'} \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

It is worth mentioning that for $p \in \mathcal{P}^{signi}$, if $X_p^k = 0$, $\forall k \in \{2, 3, ..., |\mathcal{O}_p|\}$, then $\theta_p = o_p^1$. Similarly, $(p, p^{'}) \in \mathcal{P}^{signi^2}$, if $W_{pkp'k'} = 0$, $\forall k \in \{2, 3, ..., |\mathcal{O}_p|\}$ and $\forall k^{'} \in \{2, 3, ..., |\mathcal{O}_{p'}|\}$, then $\theta_p = o_p^1$ and $\theta_{p'} = o_{p'}^1$.

After that, we model the log odds of the cost categories as a linear combination of the binary variables above. To define the log odds, we must also choose the cost category we wish to use as the baseline. In our case, we choose the first cost category as the baseline. With Y

being the cost category placement to predict for configuration θ, the first log-odds model is then:

$$\ln \frac{Pr(Y=i)}{Pr(Y=1)} = \alpha_i^0 + \sum_{p \in \mathcal{P}_{signi}} \sum_{k=2}^{|\mathcal{O}_p|} \alpha_i^{pk} X_{pk} \quad \forall i \in \{2,3,...,n\} \tag{3.3}$$

Based on the definition, n cost categories imply $n-1$ linear equations modeling the log odds. The coefficients in (3.3) are interpreted as follows. A one-unit increase in the variable X_{pk} is associated with a α_i^{pk} increase (resp. decrease) in the relative log odds of belonging to category i vs. category 1 if $\alpha_i^{pk} > 0$ (resp. < 0). In particular, for $i = n$, a high value of α_n^{pk} leads to a high probability of belonging to category n, i.e., the category with the largest cost values. Based on this interpretation, we rank the α_n^{pk} coefficients in decreasing order and shortlist the statistically significant ones with the highest values, e.g., the top 10 coefficients. A shortlisted α_n^{pk} implies that the value o_p^k of parameter index $p \in \mathcal{P}_{signi}$ is deteriorating, i.e., a *1-value.*

Based on Assumption 1, to identify *2-values*, we use the parameters selected in the first-order interactions. We consider a second log-odds model as follows:

$$\ln \frac{Pr(Y=i)}{Pr(Y=1)} = \beta_i^0 + \sum_{(p,p^{'}) \in \mathcal{P}_{signi}^2} \sum_{k=2}^{|\mathcal{O}_p|} \sum_{k^{'}=2}^{|\mathcal{O}_{p^{'}}|} \beta_i^{pkp^{'}k^{'}} W_{pkp^{'}k^{'}} \quad \forall i \in \{2,3,...,n\} \tag{3.4}$$

Similarly to the first log odds model in (3.3), in the second log odds model in (3.4), a one-unit increase in the variable $W_{pkp^{'}k^{'}}$ is associated with a β_i^{pk} increase (resp. decrease) in the relative log odds of belonging to category i vs. category 1 if $\beta_i^{pkp^{'}k^{'}} > 0$ (resp. < 0). In particular, for $i = n$, a high value of $\beta_n^{pkp^{'}k^{'}}$ leads to a high probability of belonging to category n, i.e., the category with the largest cost values. Based on this interpretation, we rank the $\beta_n^{pkp^{'}k^{'}}$ coefficients in decreasing order and shortlist the statistically significant ones with the highest values. A shortlisted $\beta_n^{pkp^{'}k^{'}}$ implies that the pair of values $(o_p^k, o_{p^{'}}^{k^{'}})$, $(p, p^{'}) \in \mathcal{P}_{signi}^2$ is deteriorating, i.e., a *2-values.*

3.3.5 Evaluation Step

The *Evaluation* step allows us to dynamically evaluate parameters with $p \in \mathcal{R}$ and select the most promising ones. To do so, we propose to study the effect of changing the values of each residual parameter one by one on the quality of θ^*. The evaluation is conducted using the *EvaluateParams* procedure presented in Algorithm 4.

Formally, we first consider $\theta^* = (o_1^*, o_2^*, ..., o_{p-1}^*, o_p^1, o_{p+1}^1, ..., o_{|\mathcal{P}|}^1)$ as a reference point where the first $(p-1)$ parameters are tuned and the remaining residual. Then, we change the value of exactly one residual parameter at a time to its different values. We note that there is no need to reevaluate the default values given that they are already tested ($\theta_k^* = o_k^1$, $k \in \{p, p+1, ..., |\mathcal{P}|\}$). For each $p \in \mathcal{R}$, this gives rise to a pool of $|\mathcal{O}_p| - 1$ configurations $\theta^k = (o_1^*, o_2^*, ..., o_{p-1}^*, o_p^k, o_{p+1}^1 ..., o_{|\mathcal{P}|}^1)$ such that $k \in \{2, 3, ..., |\mathcal{O}_p|\}$. These configurations are evaluated one by one using the considered MILP solver on the representative instance of the corresponding cluster Cl_i (line 5 in Algorithm 4). The aim is to find the improving configuration(s). We recall that an improving configuration is a configuration for which the cost is better than the current best one. A residual parameter is called improving if it provides improving configuration(s) after its values have been evaluated.

It is worth mentioning that if the current best configuration θ^* is not a suitable initial point for the *Evaluation* step, the MPILS tuner risks evaluating all residual parameters without finding an improving configuration, which can be time-consuming. To avoid such a situation, we introduce set $\mathcal{E} \subset \mathcal{R}$, which contains the first γ indexes of $\mathcal{R}$. This set is obtained using the *First* function (line 1 in Algorithm 4). The *Evaluation* step stops if either an improving configuration is found (line 13 in Algorithm 4) or $|\mathcal{E}| = \gamma$ parameters have been evaluated (line 2 in Algorithm 4). In the former case, the improving configuration becomes θ^*.

The *Evaluation* step provides the tuning pool $\mathcal{T}$ with one or several new parameters using a bi-criteria selection process. In addition to the first criterion, which is the cost, we are also interested in measuring the sensitivity of the MILP solver to the change in a given parameter's values. The solver is more sensitive to parameters with the most dispersed values of $c(\theta^k)$, $k \in \{2, 3, ..., |\mathcal{O}_p|\}$, $p \in \mathcal{R}$. In order to take this aspect into consideration, we introduce s_p as the root-mean-square deviation with regard to $c(\theta^*)$ as a measure of dispersion for $p \in \mathcal{R}$. Thus, each $p \in \mathcal{R}$ has a pair (c_p, s_p) where c_p is the minimal cost obtained. Formally:

$$c_p = \min_{k \in \{2,3,...,|\mathcal{O}_p|\}} c(\theta^k) \tag{3.5}$$

$$s_p = \sqrt{\frac{1}{|\mathcal{O}_p - 1|} \sum_{k=2}^{|\mathcal{O}_p|} (c(\theta^k) - c(\theta^*))^2} \tag{3.6}$$

The most promising parameters are the ones with very *low* values of c_p and very *large* values of s_p. To deal with this bi-objective situation, we make use of the *Pareto* optimality principle using a *Dominance* function (line 18 in Algorithm 4). This function compares each pair of evaluated parameters, selects the promising ones, and discards the unpromising ones according to the dominance rule presented in Definition 1.

Algorithm 4: EvaluateParams(θ^*,γ,η,ϵ,Δ^+,Δ^-)

```
   Input: $\theta^*$, $\gamma$, $\eta$, $\epsilon$,$\Delta^+$,$\Delta^-$
 1 Initialization: $\theta^{'} \leftarrow \theta^*, \mathcal{E} \leftarrow First(\mathcal{R}, \gamma)$
 2 forall $p \in \mathcal{E}$ do
 3     $\theta^{'} \leftarrow \theta^*$
 4     forall $\theta^{'}_p = o \in \mathcal{O}_p \setminus \{o^1_p\}$ do
 5         Evaluate $c(\theta^{'})$  /* Solve Repr. Inst. of $Cl_i$ using MILP Solver */
 6         $\Theta \leftarrow \Theta \cup \{\theta^{'}\}$
 7         if $c(\theta^{'}) \leq c(\theta^*)$ then
 8             $\theta^* \leftarrow \theta^{'}$
 9         end
10     end
11     if $c(\theta^*)$ is improved then
12         $\epsilon_p \leftarrow \epsilon_p + \Delta^+$
13         Break;
14     else
15         $\epsilon_p \leftarrow \epsilon_p + \Delta^-$
16     end
17 end
18 $\mathcal{S} \leftarrow Dominance(\mathcal{E}, \epsilon, \Delta^-)$
19 forall $p \in \mathcal{E}$ do
20     if $\epsilon_p \geq \eta$ then
21         $\mathcal{R} \leftarrow \mathcal{R} \setminus \{p\}$
22         $\mathcal{D} \leftarrow \mathcal{D} \cup \{p\}$
23     end
24 end
25 $\mathcal{R} \leftarrow Sort(\mathcal{R}, \epsilon)$
```

Definition 1. *Let $(p, p^{'}) \in \mathcal{R}^2$ for which the corresponding metrics values are (c_p, s_p) and $(c_{p^{'}}, s_{p^{'}})$, respectively. We say that $p^{'}$ is dominated by p if and only if $c_p \leq c_{p'}$ and $s_p \geq s_{p'}$ and at least one inequality is strict.*

A parameter is said to be *efficient* if it is not dominated by any other parameter. The *Dominance* function identifies the indexes of Pareto-optimal parameters. These indexes are then added to subset $\mathcal{S}$ and are considered for tuning in the subsequent *Tuning* step. While the *Dominance* function is heuristic, initial tests support its efficiency in the promising parameters identification.

Finally, to diversify the set of parameters to evaluate from one phase to another, we associate to each $p \in \mathcal{R}$ a counter ϵ_p. This counter penalizes the rejected parameter either by dominance (line 18 in Algorithm 4) or because it did not provide an improving configuration (line 15 in Algorithm 4). In both cases, the counter ϵ_p is incremented by Δ^-. This counter also rewards the corresponding parameter each time it provides an improving configuration (line 12 in Algorithm 4). It is worth mentioning that all the ϵ_p counters, $p \in \mathcal{R}$ are grouped in the vector ϵ. A residual parameter with ϵ_p ($p \in \mathcal{R}$) larger than η has its index discarded, i.e., removed from set $\mathcal{R}$ and added to set $\mathcal{D}$ (lines 21-22 in Algorithm 4). The remaining residual parameters are then dynamically sorted at the end of each *Evaluation* step in increasing order of their counters using the *Sort* function (line 25 in Algorithm 4). This sorting helps prioritize the evaluation of non-evaluated residual parameters in the subsequent phases.

Remark 1. *In a given Evaluation step, an improving parameter, if found, cannot be dominated by the Dominance function in line 18 of Algorithm 4.*

The evidence for Remark 1 is quite straightforward. Thus, finding an improving parameter in a given phase implies that set $\mathcal{S} \neq \emptyset$.

3.4 COMPUTATIONAL STUDY

We have performed extensive computational experiments to confirm the effectiveness of the MPILS tuner. We first introduce the experimental settings in Section 3.4.1. We then compare the effectiveness of the MPILS tuner to state-of-art tuners using a large-scale real-life problem in Section 3.4.2 and MIPLIB instances in Section 3.4.3.

3.4.1 Experimental Setting

The considered MILP solver is IBM ILOG CPLEX, the most widely used commercial optimization tool for solving MILP problems. We particularly tune the 12.9 release of CPLEX over its 72 non-conditional parameters. The MPILS tuner is developed using *C++* and makes use of the open-source software R embedded in *C++* for the *Learning* step. All experiments were carried out on a 3.20GHz Intel(R) Core(TM) i7-8700 processor, with 64GiB System memory, running on Oracle Linux Server release 7.7.

We compare MPILS to ParamILS and irace tuners. This choice is motivated by the following reasons. First, the MPILS tuner is based on ILS, which justifies its comparison with ParamILS, the state-of-the-art tuner among the family of tuners using the ILS metaheuristic. Second, according to a recent comparison [43] with ParamILS and SMAC, irace is the state-of-the-art tuner for the CPLEX MILP solver.

We report the following results for the four tuners:

- **CPLEX**: Solve directly using the default configuration of CPLEX without any tuning.
- **MPILS**: Solve with MPILS tuner. MPILS parameters are initialized as follows: $Z = \delta|\mathcal{T}| = 7|\mathcal{T}|$, $s = 10$, $q = 0.1$, $\eta = 15$, $\gamma = 20$, $\epsilon = 0$, $\Delta^+ = -1$, and $\Delta^- = 2$. We tune

on a single instance for each class.

- **ParamILS**: Solve with version FocusedILS of ParamILS tuner, and all other settings set to default [46]. We implement two strategies: First, the standard strategy referred to as *ParamILS*, which tunes on several training instances. In particular, we consider 60% of instances for training and 40% for testing. Second, a representative instance strategy we refer to as *ParamILS*$_1$, where we tune on a single instance.

- **irace**: Solve with Elitist Iterated Racing, all other settings set to default [127]. Similarly to ParamILS, we use the standard strategy, denoted *irace*, with the same partition of training and test instances, and the representative instance strategy, denoted *irace*$_1$, where we tune on a single instance.

We refer to the cut-off time of CPLEX as *runtime* and the CPU-time limit of the tuners as *total time*. Our comparisons are based on two metrics: quality within the runtime ($c(\theta)$ is the relative gap, referred to as gap) and runtime to optimality. If optimality is reached, i.e., $c(\theta) = 0$, the runtime to optimality is reported as well. For a fair comparison, we ensure the following. All tuners use adaptive capping and the same set of parameters, i.e., the same space of possible configurations. We first run the MPILS tuner and note the total time needed to complete the execution of the tuner. Then, we run the ParamILS and irace tuners for the same amount of time. We also note that for the *Tuning* step, into which the *RestrictedILS* procedure incorporates randomness, any experience involving the MPILS tuner is conducted five times, and the average is reported. This applies to the ParamILS and irace tuners as well. While MPILS tuner takes an initial pool of parameters, ParamILS and irace tuners take initial configurations generated from changing the parameters' values of that initial pool.

Our experiments are organized as follows: Section 3.4.2 is dedicated to real-world PSIMVA problems, where we compare MPILS to ParamILS and irace according to the first metric (quality within a time frame). After a clustering step, we consider one representative instance as a training instance for each cluster, and the others as test instances. We validate this tuning strategy (tuning on a representative instance) by testing the generated configurations on test instances of the same cluster as the chosen training representative instance, then we compare the performance of the three tuners on the speed of convergence and the quality of the configurations returned. Section 3.4.3 presents the results on a MIPLIB test bed according to the second metric (Runtime to optimality). In this section, we solely evaluate the training performance, i.e., how well the three tuners perform when only one instance is provided, within the same tuning time. Since MIPLIB instances relate to a variety of MILP problems and do not share the same models, mathematical features, or sizes, they are not tested on additional test instances.

3.4.2 PSIMVA Test Bed

In this section, we test the MPILS tuner performances. We focus on a real-life problem that prompted this research. It is a challenge faced by our industrial partner, one of the largest mining companies in the world. Because of performance issues, the company was close to abandoning the acquired optimizer since it failed to reach satisfactory and feasible solutions within a reasonable computational time. Seeking to tackle this challenge, we designed the MPILS tuner. The large-scale optimization problem incorporates production scheduling, inventory management, and vessel assignment (PSIMVA). In this section, we first describe the PSIMVA. Then, we present the instances before highlighting the computational results.

Problem Description

Given an industrial plant with several production units, stocking facilities, and loading quays, the aim is to find the most efficient schedule for production, inventory, and vessels that satisfies demand while meeting due dates and accounting for resource availability. A schedule is said to be efficient if it is both commercially and industrially efficient. Commercial efficiency is measured through the number of fulfilled shipments, while industrial efficiency is measured according to the utilization of machines. Finished products are manufactured through a succession of physical and chemical transformation processes. Each process requires a production unit and several stocking facilities used to supply the production units with the raw materials and store the intermediate and finished products. The movement of these flows of products is ensured by a network of belt conveyors linking the production units to the stocking facilities.

The supply chain network includes a port from which the finished products are delivered to the appropriate destinations using a fleet of ships. The port is composed of several quays with a set of loading points. All resources, including production units, stocking facilities, and loading quays, have limited capacities. To achieve the highest efficiency, an integrated optimization of production scheduling, inventory management, and vessel assignment is strongly needed. More details about the PSIMVA problem can be found in [2].

Instances and Setup Step

The *Setup* step consists of classifying problem instances and selecting an initial pool of parameters to initialize the tuner. MILP instances have necessarily a specific and previously known mathematical structure. They also have a given size in terms of the number of variables and an activated subset of constraints. Besides the mathematical structure, these instances model different industrial structures (demand volume, the season of the year, etc.). Therefore, it is possible to cluster instances based on several features extracted from these structures.

In the PSIMVA case, the company distinguishes between two periods during a given year: the normal and the peak demand periods. This gives rise to two classes of instances, called hereafter N (for normal) and P (for peak). We consider for our tests 15 instances from each class. For each instance, we report in Table 3.1 the total demand (*Total Dem.*), the planning horizon in days (*Horizon*), the number of vessels (*#Vessel*), the number of variables (*#Var*), the number of integer variables (*#Integer*), the number of binary variables (*#Binary*), and the number of constraints (*#Constr*). While both classes N and P have, on average, a one-month horizon, the latter has a 42% higher total demand and 22 more vessels. Solving instances P is as a result more difficult than solving instances N since they require the fulfillment of higher demand and the loading of more vessels in the same time horizon. In addition to the industrial structure, the mathematical structure further highlights the PSIMVA complexity, which is an NP-hard problem.

Based on the classification, we first run the MILP solver on a representative test instance of each class using the default setting θ^d. We analyze the produced log file to understand the different sources of complexity and determine the parameters with the most potential for remedying the weaknesses of the default setting. It is not practical to identify all the influencing parameters; two or three parameters are sufficient. In this aspect, we note that the mathematical modeling of most real-life problems gives rise to multiple symmetries and redundancies that are efficiently and quickly removed using the preprocessing tool of the MILP solver. In such situations, it is preferable to build the clustering on the preprocessed instances obtained after running the MILP solver preprocessing rather than the original test instances.

The manual troubleshooting highlighted that the CPLEX solver spends a lot of time adding cuts without any benefit, struggles to reach feasible solutions, and seems to branch on the wrong variables. Thus, we identified the following parameters to start within the initial pool of MPILS tuner: *CutsFactor*, *Emphasis*, and *VariableSelect* [128]. We recall that we provide ParamILS and irace with initial configurations generated from changing these parameters' values. It is worth mentioning that it may be relevant, in some cases, to run a generated instance before assigning it to a given class. For example, this may be necessary if demand fluctuates significantly because of a special event such as the 2021 Suez Canal Obstruction

Table 3.1 PSIMVA Instances

Class	Inst	*Total Dem.*	*Horizon*	*# Vessel*	*# Var*	*# Constr*	*# Integer*	*# Binary*
N	N1	806360	32	54	944328	3655379	360419	18257
	N2	826460	32	58	944371	3662118	360402	18354
	N3	1066290	32	61	936657	3610913	359570	11155
	N4	1091440	24	40	298693	1488475	142362	33287
	N5	1043330	25	40	298693	1489253	142362	33284
	N6	1044550	32	39	402212	1872088	193602	41136
	N7	806441	32	52	944528	3655379	360419	18357
	N8	828602	30	58	944371	3662118	360402	18354
	N9	1092721	24	40	298693	1488475	142362	33287
	N10	1044548	24	40	298698	1489253	142362	33284
	N11	1046074	31	39	402212	1872088	193602	41136
	N12	809814	31	55	944328	3655379	360419	18257
	N13	1093619	25	40	298693	1488475	142362	33287
	N14	1045118	25	40	298693	1489253	142362	33284
	N15	1052400	31	39	402525	1872088	193649	41157
	Avg	**979851**	**29**	**46**	**577166**	**2430048**	**239777**	**28392**
P	P1	1797910	30	58	450773	1964909	192276	15237
	P2	1797910	30	61	450772	1964864	192276	15122
	P3	1740100	30	58	450789	1957865	192292	15237
	P4	1304370	31	91	947598	3506473	366330	17966
	P5	2031400	32	62	947598	3506473	366330	16746
	P6	1305053	31	88	947598	3506473	366330	17966
	P7	1800079	30	58	450773	1964909	192276	15237
	P8	1804839	30	61	450852	1964908	192276	15237
	P9	1743118	30	58	450789	1957865	192292	15237
	P10	1308440	31	94	947727	3506473	366330	18211
	P11	2031960	32	62	947598	3506473	366330	17966
	P12	1801546	30	58	450773	1964909	192276	15237
	P13	1806820	30	61	450772	1964908	192276	15237
	P14	1744123	30	58	450789	1957865	192292	15237
	P15	1307549	31	89	947598	3506473	366330	17986
	Avg	**1688348**	**31**	**68**	**649520**	**2580123**	**261901**	**16257**

[129]. In what follows, we present and analyze results obtained on clusters N and P of PSIMVA.

Results on *N* Instances

In this section, we first check the validity of the representative instance tuning strategy based on tuning on one instance. Then, we select a representative instance at random among the N instances and provide it to MPILS, ParamILS, and irace tuners as a training instance for the representative instance strategy. We also compare MPILS to ParamILS and irace tuners using the standard strategy. In the end, we analyze the features of the MPILS tuner.

Validity of tuning on a randomly selected training instance. Given that N instances are relatively easier to solve than P instances, we propose to tune on each instance of class N and use the best-found configuration(s) to solve other instances from the same class. This experimental design is not a requirement for the selection of the representative instance. It is presented only for the purpose of showing that the performance of MPILS does not depend on the choice of the representative instance. In other words, we seek to show that once the clustering is done, the choice of the representative instance is arbitrary within the same class, since any instance of a class is enough to find the best configuration for all instances of the same class.

For that, we report in Table 3.2 the gaps obtained within 280s when solving each N instance using the best configurations obtained when tuning on the other instances using the MPILS tuner. In this table, we use C_{Ni}, $i = \{1, 2, ..., 15\}$ to denote the best configuration obtained when using Ni as a representative training instance. Thus, each column, entitled C_{Ni}, shows the gaps obtained when solving each of the N instances using the configuration C_{Ni}.

Table 3.2 Gap within 280s obtained on N Instances using Best Configuration(s) returned by MPILS Tuner (%)

Inst	C_{N1}	C_{N2}	C_{N3}	C_{N4}	C_{N5}	C_{N6}	C_{N7}	C_{N8}	C_{N9}	C_{N10}	C_{N11}	C_{N12}	C_{N13}	C_{N14}	C_{N15}
N1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N2	0.0	0.0	**1.1**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N3	**1.2**	**0.9**	**0.4**	**2.9**	**3.9**	**0.9**	**1.2**	**4.7**	**0.6**	**2.1**	**0.4**	**0.2**	**4.6**	**2.2**	**1.6**
N4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N5	0.0	0.0	**0.9**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N14	0.0	0.0	**4.9**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N15	0.0	0.0	**2.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

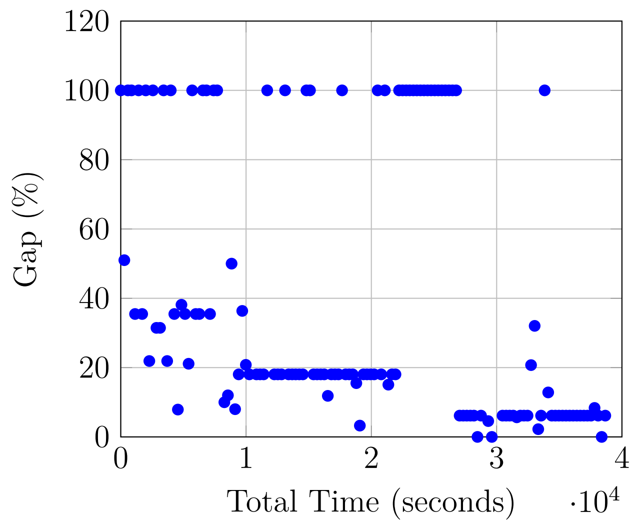
We observe that optimality is the dominating result except, for example, $N3$, for which the gap is below 5%. Given that we could not reach optimality when tuning on $N3$, tuning on other instances also does not allow optimality to be reached on $N3$. Still, the best-found configuration for this instance optimally solves 10 out of the remaining 14 instances. Overall, the percentages in Table 3.2 validate the tuning on a single instance strategy. Using the latter, we do not expect to find the configuration(s) leading to optimality. Reducing the gap is also an interesting result for the MPILS tuner. Without loss of generality, we choose henceforth instance $N1$ as the representative instance for class N.

Performance Comparison between Tuners. We compare the performance of the CPLEX, MPILS, ParamILS, and irace tuners. To do so, we take the best configurations obtained by the tuners when tuning on $N1$ and use them to solve other N instances. For CPLEX, we use the default configuration. In Table 3.3, we report the gap obtained within 280s (Gap_{280}) as well as the runtime to optimality ($Time$). For visualization purposes, any $Gap \geq 100\%$ is reported as 100%. For our experiments, we limit the runtime to 5200s since Default CPLEX requires at most 5127s to optimally solve all instances. Thus, we report the runtime to optimality when the optimal is found within 5200s.

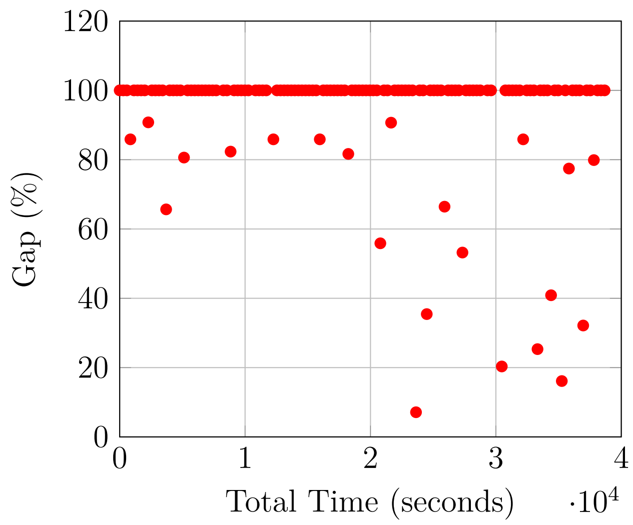
Table 3.3 Default CPLEX, MPILS Tuner, ParamILS Tuner, and irace Tuner Performance Comparison on N Instances

Inst	*CPLEX*		*MPILS*		*ParamILS*		*ParamILS*$_1$		*irace*		*irace*$_1$	
	Gap_{280}	$Time$	Gap_{280}	$Time$	Gap_{280}	$Time$	Gap_{280}	$Time$	Gap_{280}	$Time$	Gap_{280}	$Time$
N1	100.0	912	0.0	280	13.6	1065	7.1	982	3.9	717	1.7	514
N2	100.0	836	0.0	198	7.9	1592	5.5	870	3.0	583	1.4	418
N3	37.2	5127	1.2	-	10.3	-	9.6	-	1.8	-	1.7	-
N4	6.4	1748	0.0	133	10.6	942	7.3	740	0.0	279	0.0	213
N5	15.8	1179	0.0	140	12.4	2157	7.3	1154	0.0	178	0.0	154
N6	27.5	976	0.0	97	15.7	1564	13.9	783	1.1	396	1.0	320
N7	0.0	254	0.0	225	8.4	657	4.6	419	2.5	581	1.3	359
N8	9.0	1031	0.0	261	3.6	475	3.1	393	2.0	437	2.7	378
N9	2.0	419	0.0	84	5.6	698	4.7	468	0.0	276	0.0	223
N10	7.3	838	0.0	106	20.3	2831	17.07	1670	0.0	264	0.0	214
N11	82.1	883	0.0	172	10.0	1421	5.2	1226	0.0	650	0.0	207
N12	27.6	996	0.0	224	22.0	1448	11.1	767	3.0	639	2.6	561
N13	12.2	662	0.0	218	14.8	805	8.8	503	0.0	221	0.0	170
N14	15.9	803	0.0	87	2.8	442	2.4	311	1.4	323	1.3	271
N15	99.8	395	0.0	198	56.6	2273	48.37	2045	3.3	424	1.7	311
Avg	**36.2**	**852**	**< 0.1**	**161**	**14.3**	**1312**	**10.4**	**822**	**1.5**	**424**	**1.0**	**308**

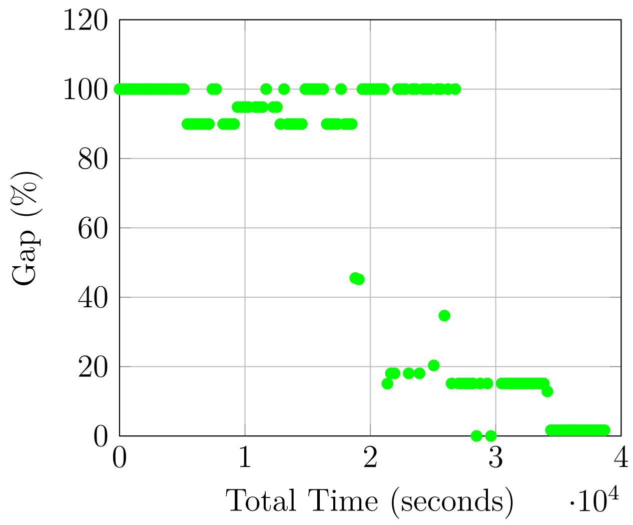
While the MPILS tuner requires, on average, 161s to reach optimality, the Default CPLEX requires, on average, 852s, i.e., a reduction factor of 6. Also, the average gap within 280s is reduced from around 36% to less than 0.1%. ParamILS$_1$ performs better than ParamILS, while irace$_1$ performs better than irace. The MPILS tuner surpasses ParamILS and irace tuners as well in all instances. On average, the time reduction factor of MPILS is around 5 and 2 when compared to ParamILS$_1$ and irace$_1$, respectively. The gap is reduced from around 10% for ParamILS$_1$ and 1% for irace$_1$ to less than 0.1% for MPILS. ParamILS and irace surpass CPLEX on runtime to optimality and reduce the gap within 280s, from around 36% to around 10% and 1%, respectively. For PSIMVA, focusing the tuning effort on a representative instance is more efficient than tuning on several instances. It is also worth mentioning that CPLEX outperforms all the tuners on instance $N3$ since all of them fail to reach optimality within 5200s. Instance $N3$ is therefore excluded from average runtime to optimality computations.



(a) MPILS Tuner



(b) ParamILS Tuner



(c) irace Tuner

Figure 3.3 Gap Evolution over Total Time on Instance $N1$ for MPILS Tuner, ParamILS Tuner, and irace Tuner

Figure 3.3 shows the gap improvement for $N1$ over the total time: the MPILS tuner scatter plot (in blue) appears in Subfigure 3.3a, the ParamILS tuner scatter plot (in red) appears in Subfigure 3.3b, and the irace tuner scatter plot (in green) appears in Subfigure 3.3c. For both ParamILS and irace, we consider strategies ParamILS$_1$ and irace$_1$, respectively. The ParamILS tuner fails to reach optimality and reaches only a limited number of satisfactory configurations. Indeed, among all the tested configurations, 89% have a gap above 80%. On the other hand, the MPILS tuner converges to optimal or near-optimal configurations over time. The described behavior is typical among all N instances. The irace tuner converges over time without reaching optimality with a behavior similar to the MPILS tuner. This can be explained by the fact that irace keeps only elitist configurations over time, and therefore significantly reduces the number of deteriorating configurations. The superior performance of the MPILS tuner can be explained by the fact that the ParamILS and irace tuners consider all CPLEX 72 parameters at once, which lengthens the time required to obtain satisfactory configurations. In contrast, the MPILS tuner considers an initial pool of parameters and keeps adding promising parameters through the *Evaluation* step while removing deteriorating values (*1-value*) and combination of values (*2-values*) identified in the *Learning* step by the $ExtractStats()$ procedure. Indeed, as observed in Subfigure 3.3a, the number of deteriorating configurations is reduced significantly towards the end of the total time TT. At the same time, the insertion of promising parameters allows more satisfactory configurations to be reached. We recall that a deteriorating configuration is any configuration with *1-value* or *2-values*. Further comparisons between the MPILS and ParamILS tuners are presented in Appendix B.

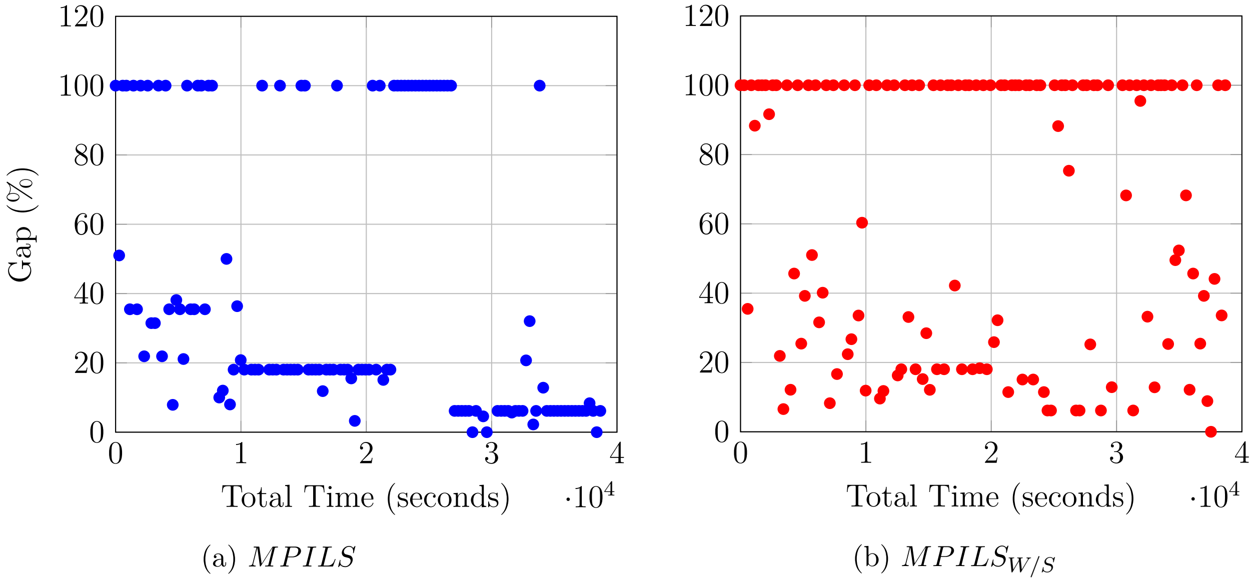
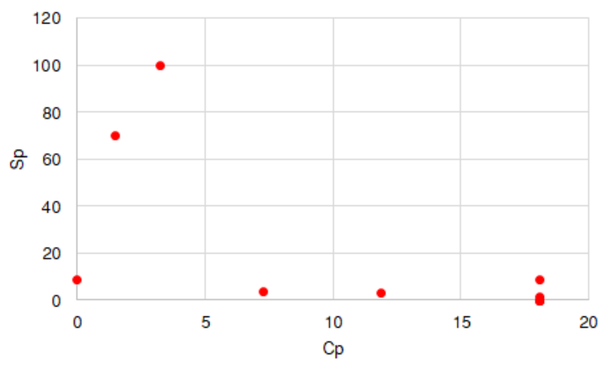


Figure 3.4 *Learning* Step Impact for Instance $N1$: with *Learning* on the left and without *Learning* on the right

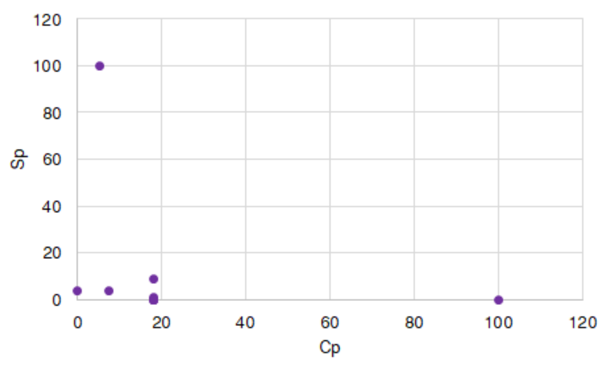
MPILS Tuner Analysis. To highlight the statistical learning impact, Figure 3.4 compares the MPILS tuner with and without statistical learning ($MPILS_{W/S}$), i.e., without the *Learning* step. While the MPILS tuner (on the left) converges over time towards promising regions where the gap is below 5%, $MPILS_{W/S}$ (on the right) suffers from deteriorating configurations. Without statistical learning, *1-value* and *2-values* persist and do not allow convergence towards promising regions. It is worth mentioning that before running the ANOVA in the *Learning* step, we ensure that the following assumptions are met: First, observations are independent, i.e., each subject should belong to only one group without repetition, and there is no relationship between the observations in each group. This is the case since the 72 CPLEX parameters are independent, and no conditional parameter is considered. Second, there are no significant outliers in any cell of the design. Third, normality exists, i.e., the data for each design cell should be approximately normally distributed. Fourth, variances are homogeneous, i.e., the variance of the outcome variable should be equal in every design cell. Assumptions 2 to 4 are controlled automatically within the R software. If these assumptions are not met, the *Learning* step is skipped. It is worth mentioning that slight departures from these assumptions usually create no serious problems. Furthermore, proper randomization procedures should be used to eliminate systematic error and assure the validity of appropriate statistical tests and inferences. In our context, randomization is ensured via the *RestrictedILS* procedure.

The *Evaluation* step is also crucial in the MPILS tuner since it allows the insertion of promising parameters that can further improve the best configuration(s) found. The more promising parameters are identified by the *EvaluateParams* procedure and inserted in the subsequent *Tuning* steps, the more efficient configuration(s) are found. Figure 3.5 highlights the *Evaluation* step for instance $N1$. Figures 3.5a to 3.5e represent the scatter plots of the pair (c_p, s_p) (see. Section 3.3.5) for the evaluated parameters in phases 1 to 5, respectively. The non-dominated parameters identified by the *Dominance* function in the *EvaluateParams* procedure are 3, 2, 2, 3, and 2 for phases 1 to 5, respectively. Figure 3.5f represents the Pareto-optimal fronts for the five phases. For a given phase, one may observe that its corresponding front dominates all the fronts of the subsequent phases. This suggests that a non-dominated parameter ($p \in \mathcal{R}$) in a given phase is never dominated in subsequent phases, as highlighted in Remark 1. This also explains why the best configuration is found, on average, within 50% of the total time. In fact, if the most promising non-dominated parameters are identified from the early phases, the best configuration is more likely to be reached earlier, as in Table B.1. On average, two promising parameters are identified at each *Evaluation* step. This is equivalent to 10 parameters added to set $\mathcal{T}$ in 5 phases. In addition to the three initial parameters, the total number of parameters that impact PSIMVA instances is around 13, i.e., 18% of all parameters considered. This confirms Assumption 1.

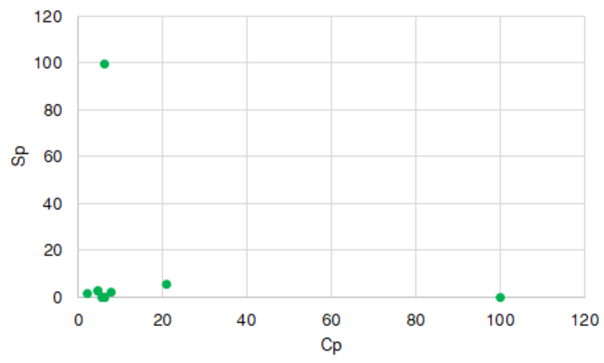
The strength of the MPILS tuner is highlighted further in Figure 3.6, which shows the percentage evolution of satisfactory configurations found throughout 5 phases for instance



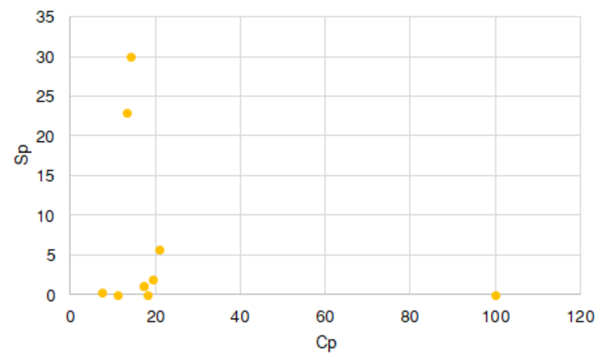
(a) Phase 1



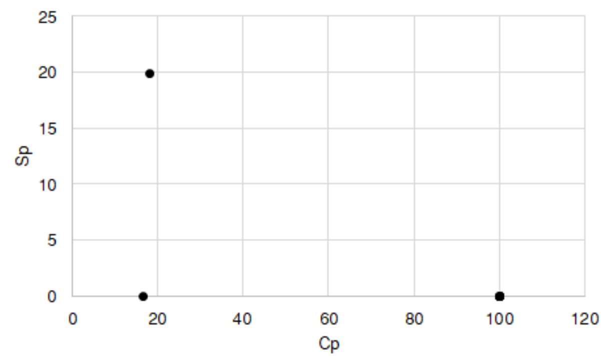
(b) Phase 2



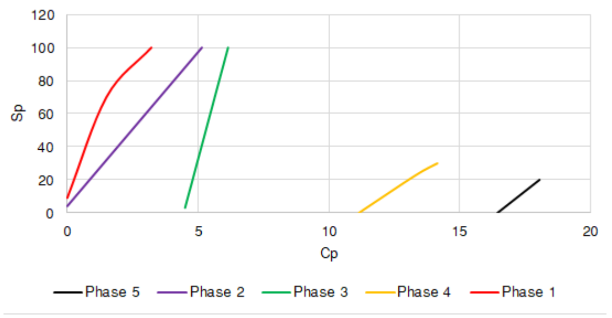
(c) Phase 3



(d) Phase 4



(e) Phase 5



(f) Pareto-Optimal Curves

Figure 3.5 *Evaluation* Step Impact for Instance $N1$

$N1$. Indeed, from phase 1 to phase 5, the percentage of satisfactory configurations rises from around 10% to around 95%. This implies that the MPILS tuner gradually converges towards promising regions.

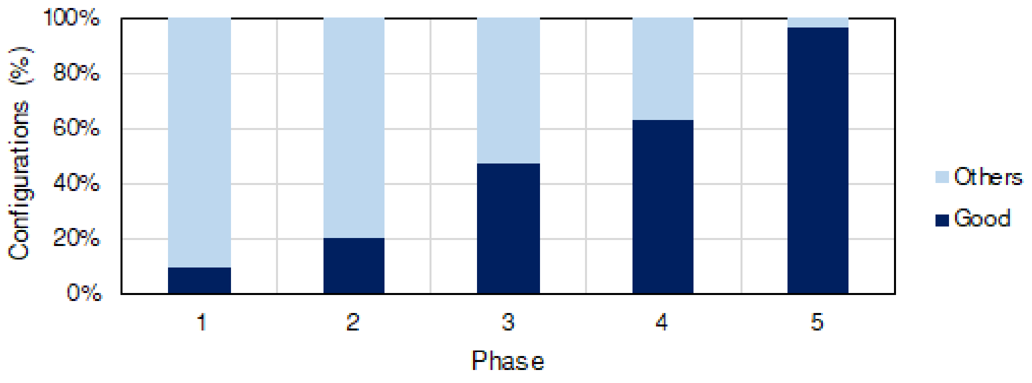


Figure 3.6 Evolution of Satisfactory Configurations % through MPILS Tuner Phases for Instance $N1$

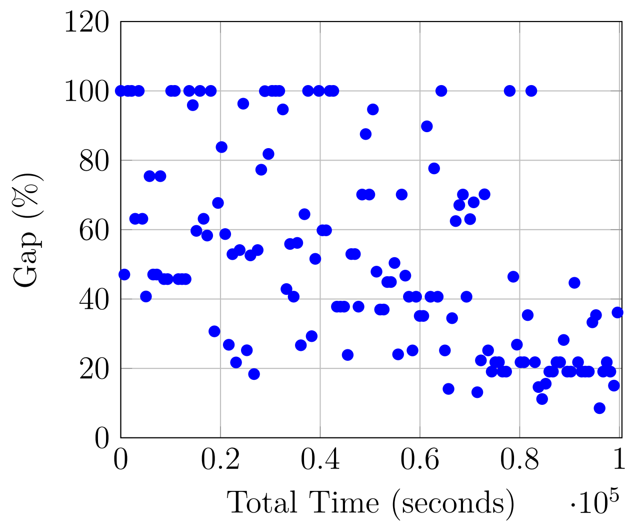
Results on P Instances

For P instances, we picked one instance randomly from the class and used it for tuning. Then, we used the best configuration found on other P instances to further check the single instance tuning strategy. Without loss of generality, the representative instance considered is $P1$. Given that this class is more complex, we provide a higher total time and runtime. We run each configuration for 720s.

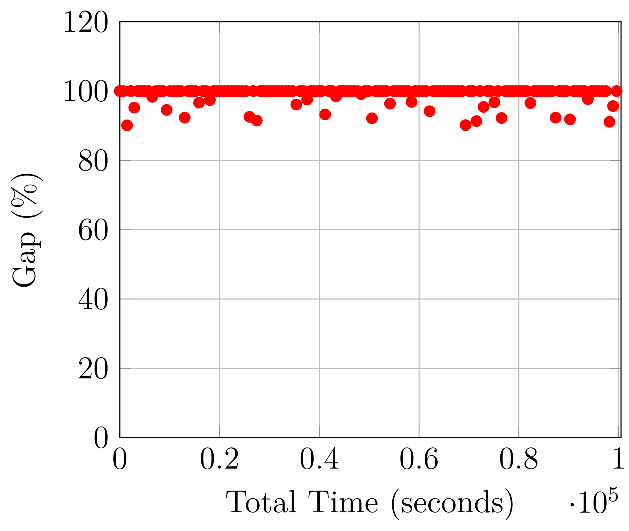
Figure 3.7 compares the MPILS tuner to the ParamILS and irace tuners. In this hard class, both the ParamILS (Subfigure 3.7b) and irace (Subfigure 3.7c) tuners become lost and cannot improve the gap as time progresses. However, as can be seen in Subfigure 3.7a, the gap improves significantly as time progresses for the MPILS tuner.

We run P instances for 720s using the best configuration obtained while tuning on instance $P1$, and compare the results with CPLEX run for both 720s and 1800s, while the ParamILS and irace tuners run for 720s. We report the gaps within 720s (Gap_{720}) and 1800s (Gap_{1800}) in Table 3.4. The best configuration obtained through the MPILS tuner ensures an average gap of 5% for the P instances, which validates the single instance tuning. Such an average is acceptable to our industrial partner given that the problem data (demand, vessel arrival, machine breakdown, etc.) have an expected margin of error of this order. On the other hand, CPLEX, the ParamILS tuner, and the irace tuner all fail within the same amount of time to bring the gap below 5%. Even with more time (1800s), CPLEX fails to reach a gap below 10% on 12 of the 15 considered instances. It is also worth mentioning that CPLEX surpasses the performance of both the ParamILS and irace tuners.

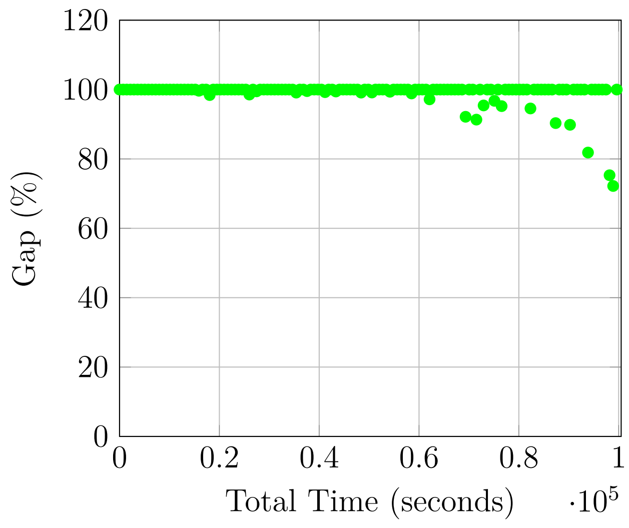
To sum up this section, the MPILS tuner outperforms CPLEX, the ParamILS tuner, and the



(a) MPILS Tuner



(b) ParamILS Tuner



(c) irace Tuner

Figure 3.7 Gap Evolution over Total Time on Instance $P1$ for MPILS Tuner, ParamILS Tuner, and irace Tuner

irace tuner on $PSIMVA$ instances. For N instances, on average, the MPILS tuner reduces the gap reached in 280s by CPLEX from around 36% to less than 0.1% and reaches optimality six times faster. For P instances, on average, the MPILS tuner reduces the gaps reached by CPLEX in 720s and 1800s from around 59% and 25% to 5%, respectively. Furthermore, the MPILS tuner surpasses the ParamILS and irace tuners in terms of both quality and runtime to optimality. For N instances, on average, the MPILS tuner reduces the gap reached in 280s

Table 3.4 Default CPLEX, MPILS Tuner, ParamILS Tuner, and irace Tuner Performance Comparison on P Instances

Inst	*CPLEX*		*MPILS*	*ParamILS*	*ParamILS*$_1$	*irace*	*irace*$_1$
	Gap_{720}	Gap_{1800}	Gap_{720}	Gap_{720}	Gap_{720}	Gap_{720}	Gap_{720}
P1	62.2	26.7	8.5	100.0	88.1	100.0	72.2
P2	56.4	52.8	4.7	100.0	79.3	100.0	65.0
P3	58.3	50.4	6.4	100.0	83.7	100.0	68.6
P4	15.7	11.9	3.7	100.0	78.4	100.0	66.7
P5	31.5	7.5	6.9	100.0	85.5	100.0	74.4
P6	92.8	18.3	0.2	100.0	70.5	100.0	62.7
P7	85.9	50.7	5.0	100.0	82.8	100.0	67.1
P8	87.7	15.7	4.4	100.0	81.1	100.0	67.3
P9	100.0	64.1	4.2	100.0	80.9	100.0	67.9
P10	32.9	5.9	0.7	100.0	61.7	100.0	53.0
P11	52.4	16.1	8.0	100.0	87.2	100.0	76.8
P12	86.3	12.1	2.3	100.0	80.2	100.0	71.4
P13	34.2	16.3	9.9	100.0	89.0	100.0	71.2
P14	70.2	27.8	9.8	100.0	89.4	100.0	73.3
P15	3.20	0.6	0.4	100.0	68.7	100.0	61.8
Avg	**58.9**	**25.1**	**5.0**	**100.0**	**80.4**	**100.0**	**68.0**

from around 10% and 1% to less than 0.1%, and reaches optimality 5 and 2 times faster. For P instances, the ParamILS and irace tuners fail to reach satisfactory configuration(s). Lastly, the single instance tuning strategy has been validated empirically, thus leveraging its implementability in practice. In fact, when considering several instances, double the effort is required since both the configuration and the training instances spaces must be explored. However, when considering a representative instance, the effort is focused on exploring just the configuration space. Overall, the MPILS tuner proves its efficiency in finding satisfactory configurations for PSIMVA.

3.4.3 MIPLIB Test Bed

To check further the MPILS tuner performance, we conducted additional experiments tests on several instances from the MIPLIB library [130] using the same experimental setting as the PSIMVA instances. Given that the MIPLIB instances are different, each class has a unique instance, i.e., *ParamILS* and *ParamILS*$_1$ coincide. The same goes for the *irace* and *irace*$_1$ strategies. Thus, we tune each instance independently.

The purpose of these MIPLIB experiments is threefold. First, we compare the performance of the three tuners during the training time, and their ability to fastly return good configurations over time. We do not evaluate the efficiency of these configurations on additional test instances as we did for PSIMVA Test Bed. Second, we evaluate the performance of our tuner using the second metric (Runtime to optimality), since we previously know that the chosen MIPLIB instances converge to optimal solutions. Third, we demonstrate that the MPILS tuner can produce excellent configurations for a wide range of MILP problems, not just PSIMVA, as MIPLIB offers MILP problems with a variety of models and mathematical features.

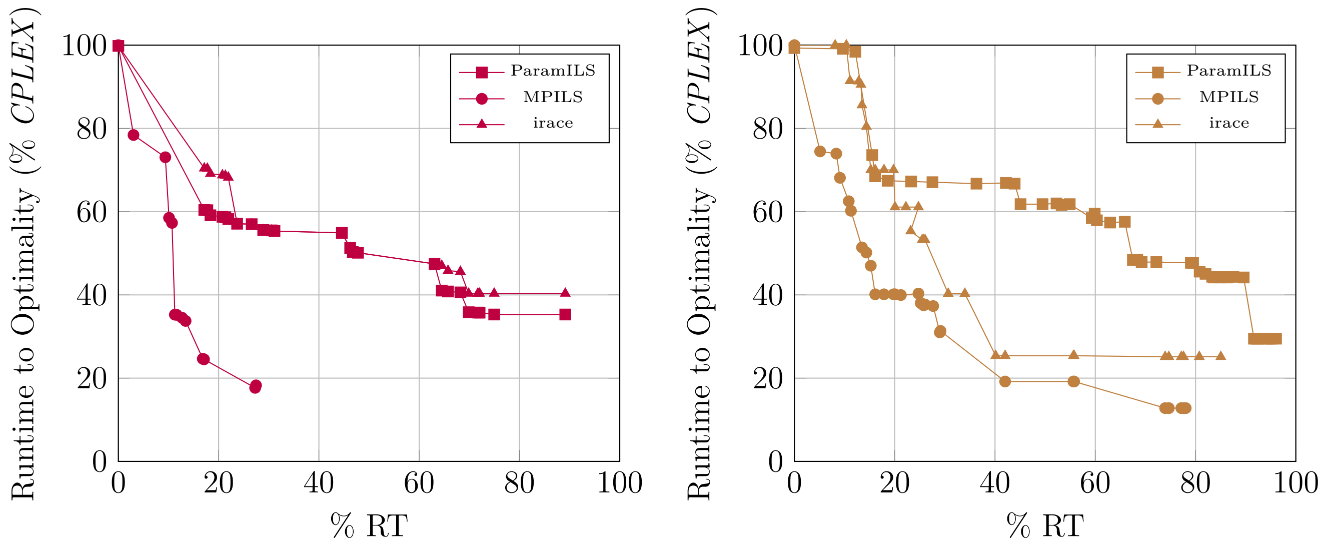
The results on the MIPLIB instances for the MPILS, ParamILS, and irace tuners are reported in Table 3.5. All the considered instances can be solved to optimality. Thus, we do not report the gap, and instead, report the runtime to optimality after 10% of the total time ($Time_{\frac{TT}{10}}$), i.e., the runtime returned by the best configuration reached within 10% of the total time. We also report the runtime to optimality at the end of the total time ($Time_{TT}$) and the time when the best configuration was found (TT^*). The default CPLEX runtime to optimality is reported for information purposes only.

We observe that the MPILS tuner outperforms the ParamILS and irace tuners. Since the ParamILS and irace tuners fail to reach optimality before 10% of the total time on instances *trento1* and *neos-3004026-krka*, we exclude them from the average $Time_{\frac{TT}{10}}$ computations. After 10% of the total time, the MPILS tuner reduces the runtime to optimality by factors of up to 7.34 with an average of 1.65 compared to the ParamILS tuner, and up to 5.87 with an average of 1.29 compared to the irace tuner. These results reflect the performance of the MPILS tuner in returning improving configurations within a very limited time. The runtime to optimality reached by the final configurations is, on average, 1.56 times shorter than that of the ParamILS tuner and 1.78 times shorter than that of the irace tuner. The results also show that the MPILS tuner is very efficient from the beginning, with convergence to satisfactory configurations becoming more significant towards the end. On the contrary, the ParamILS and irace tuners search far from satisfactory configurations at the beginning and catch up with slowness towards the end. On average, while the irace tuner surpasses the ParamILS tuner in the beginning, the latter outperforms the former by the end of the total time.

Table 3.5 Default CPLEX, MPILS Tuner, ParamILS Tuner, and irace Tuner Performance Comparison on $MIPLIB$ Instances

$Inst$	$CPLEX$	TT	$MPILS$			$ParamILS$			$irace$		
	$Time$		$Time_{\frac{TT}{10}}$	$Time_{TT}$	TT^*	$Time_{\frac{TT}{10}}$	$Time_{TT}$	TT^*	$Time_{\frac{TT}{10}}$	$Time_{TT}$	TT^*
brasil	4.07	1050	2.10	2.01	153	2.15	2.04	947	4.22	3.56	788
comp08-2idx	4.16	808	3.55	2.18	554	6.33	2.40	468	5.02	2.64	606
mzzv11	6.52	1458	4.21	2.11	1018	8.47	2.78	1083	4.69	3.66	1094
pk1	7.80	1118	3.72	2.10	951	5.89	2.50	1056	5.18	4.18	839
mas76	8.51	683	3.87	1.09	287	5.63	2.51	571	5.22	2.14	512
swath3	3.81	1173	2.84	2.15	947	3.62	3.14	514	3.48	1.74	880
fjspeasy01i	12.02	773	1.31	1.01	728	1.61	0.89	459	1.20	1.08	580
supportcase7	13.24	1913	5.18	2.42	1620	28.69	2.59	1790	5.16	2.93	1435
app1-2	19.07	2177	10.47	6.17	1163	9.88	5.87	1934	7.90	6.71	1633
fiball	15.82	896	4.21	0.89	806	4.79	2.57	847	3.83	2.94	672
mod011	16.02	2150	5.09	2.66	1796	6.61	3.15	1937	5.29	3.60	1613
gasprod2-2	23.06	5195	13.54	11.16	4168	17.11	13.25	1643	13.69	15.14	3896
fastxgemm-n2r6s0t2	31.15	2521	20.26	3.05	2471	49.02	8.66	2022	39.22	9.90	1891
supportcase18	28.96	2753	10.75	3.69	1309	78.86	9.51	2318	63.09	10.87	2065
triptim1	29.74	3450	21.36	12.87	3000	32.08	20.87	3115	25.66	23.85	2588
neos-4722843-widden	39.40	4342	10.88	5.77	4248	20.30	8.48	3935	16.24	9.69	3257
supportcase33	19.71	3452	10.22	8.52	1288	21.61	12.40	3509	17.29	14.17	2589
fast0507	14.15	4589	21.12	7.64	1724	20.56	9.52	3852	16.45	10.88	3442
glass4	192.34	3786	12.95	11.41	2685	70.17	24.51	3044	56.14	28.01	2840
aflow40b	18.47	11285	22.38	21.72	1225	32.92	18.59	9178	26.34	21.25	8464
neos-4413714-turia	69.79	4388	14.33	6.21	3583	16.47	8.45	3481	13.18	9.66	3291
fastxgemm-n2r7s4t1	65.71	2506	49.19	4.79	796	71.48	12.58	2354	57.18	14.38	1880
neos-1456979	49.41	9178	31.25	28.27	3906	80.73	42.37	5515	64.58	48.42	6884
trento1	200.05	29147	62.61	55.11	6630	—	52.20	26031	—	59.66	21860
neos-3004026-krka	18.61	1547	20.55	3.34	1124	—	53.22	708	—	60.82	1160
neos-2746589-doon	116.48	10792	34.78	24.77	4513	41.29	32.03	5862	33.03	36.61	8094
mas74	84.60	8695	69.93	15.46	2372	87.33	49.86	6518	85.86	34.13	6521
Avg	**41.21**	**4512**	**17.51**	**9.21**	**2039**	**28.95**	**14.33**	**3507**	**23.17**	**16.39**	**3384**

The runtime to optimality evolution, presented as a percentage of the default CPLEX, is illustrated in Figure 3.8 for two different instances. For *mas74* instance (purple), the MPILS tuner reduces runtime to optimality to less than 35% of the default CPLEX in less than 12% of the total time. At the same time, i.e., less than 12% of the total time, the ParamILS and irace tuners do not return any configuration better than the default CPLEX. After 30% of the total time, the runtime to optimality is further reduced by the MPILS tuner to less than 20% of the default CPLEX, achieving a reduction factor of five. On the other hand, the best configuration reaches optimality in 36% of the default CPLEX after 70% of the total time for the ParamILS tuner and 40% of the default CPLEX after 70% of the total time for irace tuner. For *mas76* instance (brown), at less than 30% of the total time, the best configurations returned by MPILS, ParamILS, and irace tuners succeed in reaching optimality in 30%, 68%, and 56% of the default CPLEX, respectively. The best MPILS, ParamILS, and irace tuners configurations can reach optimality in around 15%, 30%, and 27% of the default CPLEX, respectively.



(a) Instance *mas74* (b) Instance *mas76*

Figure 3.8 Runtime to Optimality Evolution over Total Time for MPILS Tuner, ParamILS Tuner, and irace Tuner

To sum up, while the MPILS tuner is more suitable for hard large-scale optimization problems like the PSIMVA, it is still more efficient when compared to the ParamILS and irace tuners on the various MIPLIB instances considered, even if the latter can be solved to optimality quickly.

3.5 CONCLUSION

The present work introduces a new MPILS tuner for deterministic MILP solvers. The MPILS tuner iteratively explores the configuration space over a sequence of dynamically changing subspaces. Benefiting from the knowledge of the MILP problem, it exploits efficiently the strengths of the iterated local search metaheuristic. Moreover, it uses statistical learning to remove non-promising parameter values and combinations of parameter values. The MPILS tuner returns configurations that reach optimality six times faster than the default CPLEX for a large-scale complex optimization problem and multiple MIPLIB instances. Additionally, a comparison with more general state-of-the-art ParamILS and irace tuners showed that the MPILS tuner converges faster to near-optimal configurations within the MILP solvers' niche. In particular, it quickly generates satisfying configurations.

The MPILS tuner is a deterministic specialized tuner designed specifically for tuning MILP solvers, which are used to solve a variety of real-world optimization problems. Within this niche, although the MPILS shows significant performance compared to other general state-of-the-art tuners, it has some limitations. First, the MPILS tuner cannot tune more than one instance at a time. Second, the tuner design is specific to the MILP solvers' niche where we cluster the instances and select only one for tuning from each cluster. Third, it requires prior identification of an initial set of parameters based on problem knowledge. To tackle the above limitations, future research includes the implementation of the multi-instance (as a training set) option, automatic clustering of instances, and automatic troubleshooting. We will also explore the design and implementation of more advanced deep learning techniques to strengthen the identification and removal of non-promising combinations of parameter values.

Acknowledgements

This work was generously supported by the Fonds de Recherche du Québec - Nature et Technologies (FRQNT) through a Ph.D. Excellence Scholarship, the Institute for Data Valorisation (IVADO) through another Ph.D. Excellence Scholarship, and the Group for Research in Decision Analysis (GERAD) through the provision of computing facilities for conducting experiments. All authors gratefully acknowledge the valuable comments received from the editors and three anonymous referees on an earlier version of this article.

CHAPTER 4 ARTICLE 2: INCREMENTAL LNS FRAMEWORK FOR INTEGRATED PRODUCTION, INVENTORY, AND VESSEL SCHEDULING: APPLICATION TO A GLOBAL SUPPLY CHAIN

Authors: El Mehdi Er Raqabi, Ilyas Himmich, Nizar El Hachemi, Issmaïl El Hallaoui, François Soumis

Submitted on April 5, 2022 and published on December 14, 2022 in Omega, Volume 116, 102821, DOI: 10.1016/j.omega.2022.102821

Abstract. This paper presents a multiobjective, mixed-integer linear programming (MILP) model that integrates production scheduling, inventory management, and vessel assignment for a global supply chain. Given that such large-scale problems are NP-hard and usually suffer from symmetry, we conduct an exploratory analysis to identify complexity sources. Following this, we design a novel variant of the large neighborhood search metaheuristic to tackle the problem efficiently. While symmetry is considered an issue in the literature, the implemented algorithm provides a practical way of profiting from instead of breaking it. Computationally, we reach near-optimal solutions in real-world instances. Compared to the default CPLEX and a reference algorithm that mimics real life, we gain significantly in terms of time, quality, and the number of feasible integer solutions found during the solving process. In addition to efficiency, integrated optimization enhances operations management capabilities and supply chain resilience.

Keywords. *Large-Scale Optimization*, *MILP*, *Multiobjective Optimization*, *Metaheuristics*, *Symmetry*, *Global Supply Chain.*

History. This article is published in *Omega: The International Journal of Management Science.*

4.1 INTRODUCTION

A supply chain can be defined as a set of three or more entities (organizations or individuals) directly involved in the upstream and downstream flows of products, services, finances, and/or information from a source to a customer [131]. It follows that a global supply chain is a supply chain with cross-border entities. Global supply chains have become increasingly prevalent in the last decades of the twentieth century, especially in the mining, apparel, automobile, and aerospace industries [132, 133, 134]. Globalization, along with the additional management challenges it brings, has prompted companies to look for better ways of coordinating the flow of materials throughout the supply chain. In addition, companies now compete based more on speed and quality than ever before. Faster and more reliable delivery of a defect-free product is no longer seen as a competitive advantage, but simply as a necessity. Customers demand personalized products that are consistently delivered on time with the exact features they requested. Each of these calls for improved global supply chain coordination [135].

Although supply chain entities can optimize their operations separately, the inherent strategic nature of coordination renders global supply chain integration paramount. Integration refers to the extent to which separate entities work together cooperatively to arrive at mutually acceptable outcomes. In this sense, this definition covers measures of coordination, interaction, cooperation, and collaboration [136]. The literature acknowledges that the higher the degree of integration across the supply chain, the better a firm performs [137, 138].

In order to reach the highest degree of integration and, taking into account the complexity of global supply chains, companies opt for gradual integration of supply chain components [139]. Although a detailed top-down approach to developing an integrated supply chain strategy is essential, its achievement is likely to be bottom-up [140]. Thus, starting the integration downstream is a promising strategy. In this paper, we consider a global supply chain for which we integrate the downstream operations. These operations include production scheduling, inventory management, and vessel assignment, resulting in the integrated production scheduling, inventory management, and vessel assignment (PSIMVA) problem.

Global supply chain integration generates large and complex scheduling problems such as the PSIMVA. Given their size, it is not possible to tackle these problems manually. Furthermore, even using naive scheduling software, no feasible schedules are found rapidly. Thus, companies invest intensively in more sophisticated systems that rely on operations research (OR) tools. Once modeled mathematically, there is an opportunity to tackle these problems rapidly using the vast choice of OR algorithms available in the literature. These algorithms become more relevant when the mathematical models involve millions of variables, constraints, and complex multiobjective functions.

The contribution of the paper is fourfold: (1) To conduct an exploratory analysis to identify the complexity sources of the PSIMVA mixed-integer linear programming (MILP) model with two multiobjective optimization methods, (2) To solve the PSIMVA, a novel variant of the large neighborhood search (LNS) metaheuristic benefiting from symmetry is designed, (3) To carry out a computational study on real-world instances showing significantly better solutions, resulting in an average time reduction factor of 12 compared to the default CPLEX and an expected increase of 5% in the global supply chain's annual turnover, and (4) To provide a post-computational analysis that highlights several managerial insights.

The rest of the paper is organized as follows: An overview of the relevant literature is presented in Section 4.2. Section 4.3 is devoted to a detailed description of the considered problem while the problem formulation is given in Section 4.4. Section 4.5 and Section 4.6 present the exploratory analysis and the solution methodology, respectively. The computational study and managerial insights are highlighted in Section 4.7. The following sections provide the conclusion and the references.

4.2 LITERATURE REVIEW

Integrating global supply chains efficiently requires tackling several problems simultaneously. Among these problems, production scheduling [141, 142, 143, 144, 145], inventory management [146], and vessel assignment [147, 148, 90], also known as berth allocation, are intensively studied in the literature. Given a processing factory with one or several transformation units, stocking points, and loading quays, the goal is to formulate the most effective plan to ensure efficient management of all these operations while satisfying demand within deadlines and respecting the capacities of the available resources.

4.2.1 Deterministic Optimization

In the literature, tackling the three problems relies on exact and heuristic techniques. Production scheduling is covered in many papers. [149] decomposes the problem into a master problem, which incorporates linking constraints, and a subproblem, which incorporates the sequence of extraction. [150] model scheduling as a MILP model and solve it using a branch and cut ($B\&C$) algorithm. [151] propose a two-step approach. First, they use a decomposition method to solve the linear relaxation (LR). Then, they designed a local search-based heuristic to reach near-optimal solutions for the integer problem. Berth allocation is also covered in the literature. [152] designed a lagrangian-based heuristic to reduce the computational assignment efforts significantly and demonstrated its applications to real-life problems. [153] present a heuristic for the berth allocation problem in continuous locations. [154] integrate the quay crane assignment problem and the quay crane scheduling problem before solving the integrated problem using a genetic algorithm. [155] integrate berth allocation, quay crane assignment, and quay scheduling problems. They use a rolling horizon heuristic to tackle the integrated model. Inventory management is usually incorporated within production scheduling as in the analysis provided by [156], and the book of [157], where quality and maintenance are also incorporated.

Table 4.1 Summary of Deterministic Optimization

Reference	Problem	Type	Objective	Design
[155]	Vessel	Heuristic	Min Time	MILP
[158]	Inventory & Vessel	Branch&Cut	Min Cost	MILP
[150]	Production	Branch&Cut	Max Block	MILP
[151]	Production	Decomposition	Max Block	LP
[154]	Vessel	Genetic Algorithm	Min Time	MILP
[153]	Vessel	Heuristic	Min Time	MILP
[152]	Vessel	Heuristic	Min Time	MILP
[149]	Production	Decomposition	Max Flow	LP
[159]	Inventory & Vessel	Column Generation	Min Cost	MILP
[160]	Production & Inventory	Heuristic	Min Cost	MILP

More recently, [158] address an annual delivery program for liquefaction supply chain planning. It includes inventory and berth management at the producer, routing and scheduling of a fleet of ships, and contract management between the producer and its customers. They used a $B\&C$ algorithm with four families of valid inequalities to produce better lower bounds and accelerate convergence. [159] integrate planning and scheduling decisions to achieve efficient operation and use of port terminal facilities. Their problem consists of defining the amount and destination of orders in a bulk cargo terminal, establishing a set of feasible routes to guarantee that products are stored and shipped on schedule, and minimizing operational costs, which they solve using a branch-and-price algorithm. [160] consider practical multi-factory job allocation and scheduling problems with maritime transport limits and design a new heuristic to improve the performance of both exact and genetic algorithms. In their model, vessel schedules are provided as input data, and no decisions are required concerning vessels' assignment to quays. The deterministic literature is summarized in Table 4.1.

4.2.2 Multiobjective Optimization

Another important aspect of supply chain management is multiobjective optimization [161]. Indeed, optimizing the integrated production scheduling, inventory management, and vessel assignment implies the consideration of several key performance indicators (KPIs), i.e., a multiobjective function. We distinguish two main multiobjective optimization approaches: the classical approach and the metaheuristics approach. The classical approach consists of aggregating the objective function, i.e., optimizing the most important KPIs while considering others as constraints. Within this approach, many classical methods have been developed including the weighted sum, the ϵ-constraint, and the lexicographic methods. Without loss of generality, let us consider a minimization case. The weighted sum method is the most common and consists of selecting scalar weights w_i and minimizing the objective function $\sum_{i=1}^{k} w_i f_i(x)$ where $f_i(x)$ is the i^{th} objective function. The ϵ-constraint minimizes the most important objective function $f_s(x)$ and adds the constraints $f_i(x) \leq \epsilon_i$, $i \neq s \in \{1, 2, ..., k\}$ to the mathematical model to bound the other objectives. In the lexicographic method, objective functions are ranked in order of importance. Then, the optimization process is conducted individually with each objective following the ranking order. After an objective is optimized, optimization continues to the subsequent objective after bounding the previously-optimized ones using the ϵ-constraint mechanism. This cycle continues until the last objective has been optimized. Further details can be found in the survey of [162].

In the context of supply chain management, [163] address production, distribution, and capacity planning as a MILP considering simultaneously cost, responsiveness, and customer service level. They employ lexicographic and ϵ-constraints methods to tackle MILP. [164] presents a network design model to evaluate the trade-off between total network cost minimization and overall supply chain network connectivity maximization. To solve their multiobjective MILP, they use the ϵ-constraint method by minimizing the cost in the objective and constraining the connectivity. [165] develop an integrated multiobjective model for medium-term tactical decision-making for an oil and gas supply chain downstream segment. The objectives include minimizing the cost and maximizing the revenue and service level (SL). They also use an ϵ-constraint algorithm to generate Pareto optimal solutions. [166] propose a MILP to optimize the design of a sustainable supply chain in terms of the total cost, time, and sustainability using an ϵ-constraint method.

Table 4.2 Summary of Multiobjective Optimization

Reference	Approach	Type	Objective	Design
[167]	Metaheuristics	Genetic Algorithm	Min Cost & Equity / Max Customer Demand	MILP
[165]	Classical	ϵ-constraint	Min Cost / Max Revenue & SL	LP
[166]	Classical	ϵ-constraint	Min Cost & Time / Max Sustainability	MILP
[168]	Metaheuristics	Evolutionarys Algorithm	Min Centralization & Emission / Max Profit	MILP
[163]	Classical	ϵ-cosntraint & Lexicographic	Min Cost & Time & Lost Sales	MILP
[164]	Classical	ϵ-constraint	Min Cost / Max Connectivity	MILP
[169]	Metaheuristics	Bees Algorithm	Min Cost & Time	MILP

The metaheuristics approach relies on approximate methods, including evolutionary algorithms, ant colony optimization, and bee algorithm. While treating each objective function separately and by evolving a population of solutions, evolutionary algorithms can approximate the Pareto optimal front of solutions in a single run [170]. Inspired by real life, ant colony optimization has been used to tackle multiobjective problems. The basic idea is to model these problems as the search for a minimum cost path in a graph and to use artificial ants to search for efficient paths [171]. Similar to ant colony optimization, the bee algorithm mimics the foraging behavior of honey bees. It has two balanced searches: The first is a local search that explores a neighborhood surrounding some available solutions, and the second is a random global search that explores the set of feasible solutions [172].

In the supply chain management context, [167] propose a novel procedure based on genetic algorithms to find the set of Pareto-optimal solutions for a multiobjective supply chain network design problem. [169] tackle a supply chain problem, which minimizes both the total cost and the total lead time using the bees algorithm. [168] present a robust non-linear multiobjective optimization model to configure a green global supply chain network structure under disruption and resolve the disruptions using a novel hybrid heuristic based on evolutionary algorithms. The relevant multiobjective literature is summarized in Table 4.2. Further details about multiobjective optimization for supply chain management are provided in the review of [173].

It is worth mentioning that inverse optimization is emerging as a promising field in multi-objective optimization, especially when some of the parameters may not be precisely known, such as the KPIs weights in the objective function. Using available information including feasible, near-optimal, or optimal solutions, the aim is to determine these parameter values. Relevant papers on this topic are the ones by [174] and [175].

4.2.3 Our Research

To our best knowledge, there is no paper tackling the PSIMVA in the literature. To fill this gap, this paper proposes tackling the PSIMVA considering several products, up to 91 vessels, and a maximum 32-day time horizon. To do so, we explore two multiobjective PSIMVA formulations based on two classical multiobjective methods: the weighted sum and the lexicographic methods. Furthermore, given that the simultaneous mathematical modeling of production scheduling, inventory management, and vessel assignment gives rise to large NP-hard MILP problems that are complex, if not impossible, to solve optimally in reasonable computational time, this study introduces a novel variant of the LNS algorithm that has been widely successful in handling large-scale optimization problems [176]. LNS was introduced by [59] for solving the capacitated vehicle routing problem with time windows. Its mechanism is to iteratively destroy and repair a solution in order to improve it by alternating between an infeasible solution and a feasible solution. The destroy operation creates an infeasible solution which is brought back into a feasible form by the repair heuristic. LNS is suitable for problems that can be decomposed into smaller sub-problems that can be destroyed and then repaired.

We refer to [60] and [61] for a thorough description of the method. The novel variant of the LNS implemented in this paper, called hereafter incremental LNS ($\mathcal{ILNS}$), incrementally constructs near-optimal, if not optimal, solution(s). It differs from the standard LNS [59] in the following ways: First, the standard LNS requires an initial solution to improve; in contrast, the $\mathcal{ILNS}$ constructs an initial solution before improving it. Second, compared to the standard LNS that may go through infeasible solution(s), the $\mathcal{ILNS}$ maintains feasibility. Third, while the standard LNS may destroy any fixed part of the solution, the $\mathcal{ILNS}$ destroys only part(s) of the solution that may be improved.

4.3 PROBLEM DESCRIPTION

We consider the global supply chain of [24], one of the largest phosphate companies worldwide, holding 70% of the world's phosphate rock reserves [177]. It has branches in Morocco, Brazil, India, and other countries, and specializes in phosphate mining, production, and exportation. Phosphate products include raw phosphate, phosphoric acid, and phosphate fertilizers.

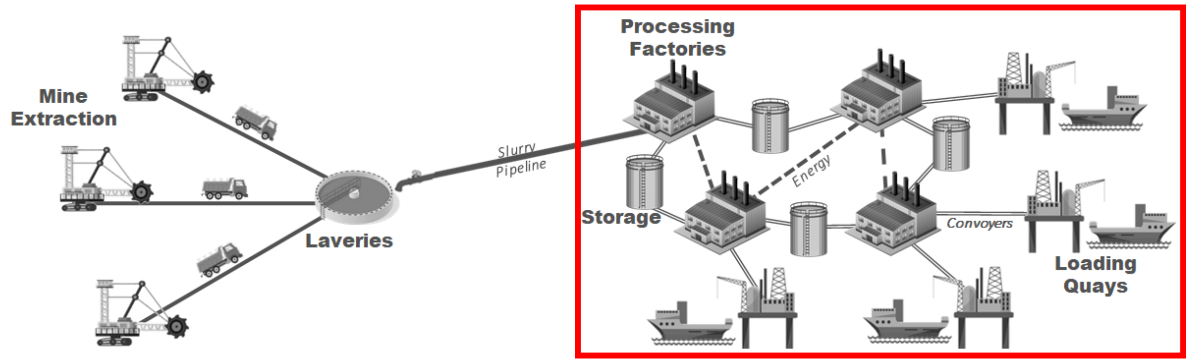


Figure 4.1 The Phosphate Supply Chain

The company promotes *precision farming*, i.e., utilizing a unique fertilizer for a specific type of soil [178], and has increased its number of products from 3 to more than 30. Its global supply chain, highlighted in Figure 4.1, is made up of four main components, through which 45 raw, semi-finished, and finished products flow. The phosphate rocks are extracted from the mine; then, these rocks are transported using trucks to a physical treatment facility where they undergo the washing and floating processes. The washed rocks are transported by a 187 km slurry pipeline to the coastal processing plant for chemical treatment. Several derivative products are processed through 32 various chemical processes. The final products are then stored in 29 large tanks before being supplied through conveyors to 6 quays, where vessels of clients worldwide are loaded. The coastal processing factory, as well as the loading port, spreads over an area of $5 \times 10^6\ m^2$. On average, 37.6 million metric tons of phosphate rock are processed each year, accounting for 31% of the phosphate world market share. The supply chain is connected through 102 conveyors and pipelines [24]. The complexity of OCP's supply chain makes it a representative example of a global supply chain. While the work presented in this paper is inspired by a large-scale mining company, its findings can be transferred, adapted, and applied to tackle similar difficult and large-scale problems in any process industry relying on vessels and other means of transportation for product delivery.

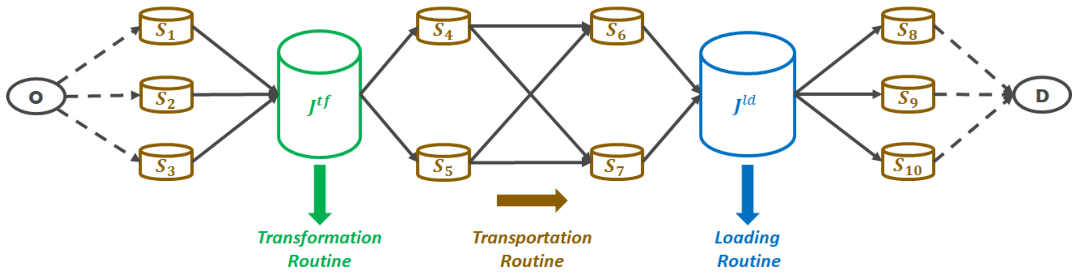


Figure 4.2 A Miniature Zoom on the Framed Part

The PSIMVA groups several components of the downstream supply chain, making it quite complex. Thus, to facilitate its description, a simplified zoom on the red-framed part of Figure 4.1 is presented in Figure 4.2. After describing the demand, we highlight the production scheduling, inventory management, and vessel assignment components. For the PSIMVA notation, indices are in lower-case, sets are in calligraphic style, and parameters are in bold style. The notation is summarized in Table 4.3.

Let $\mathcal{T} = \{1, ..., \overline{T}\}$ be the set of $\overline{T}$ periods (days in our context). Over a given time horizon, *OCP Group* receives worldwide sale requests, called hereafter shipments. A shipment $h \in \mathcal{H} = \{1, ..., \overline{H}\}$ is the package of requested products with their corresponding quantities, and has a corresponding vessel. Each vessel can be loaded in a quay $k \in \mathcal{K} = \{1, ..., \overline{K}\}$. The company produces several products $p \in \mathcal{P} = \{1, ..., \overline{P}\}$, which can be raw, semi-finished, or finished. We use the terms *produce* and *transform* interchangeably.

Raw materials are provided through several origins $o \in \mathcal{O} = \{1, ..., \overline{O}\}$ such as the physical treatment facility in Figure 4.1. The plant has several units $j \in \mathcal{J} = \{1, ..., \overline{J}\}$. We distinguish transformation $\mathcal{J}^{tf} \subset \mathcal{J}$ and loading units $\mathcal{J}^{ld} \subset \mathcal{J}$. The former represents production lines, and the latter refers to cranes used for loading shipments. We define a routine $r \in \mathcal{R} = \{1, ..., \overline{R}\}$ as a regular process followed by a product at an entity of the supply chain. For instance, a transformation unit can produce several products. Thus, it has a specific set of routines and operates according to one such routine at a time. Activating a routine allows the production of a single product. The same is valid for transportation and loading units. In $j \in \mathcal{J}^{tf}$ (e.g. the green cylinder in Figure 4.2), transformation routines transform several products (e.g. input products in S_1, S_2, and S_3 in Figure 4.2) to other products (e.g. output products in S_4, S_5 in Figure 4.2). Transportation routines transport products using a network of belt conveyors and pipelines (e.g. conveyor between S_5 and S_7 in Figure 4.2). In $j \in \mathcal{J}^{ld}$ (e.g. blue cylinder in Figure 4.2), loading routines load the ordered products onto corresponding vessels. To shift production from one product to another, a decision to change the active routine on a transformation unit is necessary. Such a decision is called a *changeover*. The setup time to execute a changeover leading to a new routine $r \in \mathcal{R}_j$ with $j \in \mathcal{J}^{tf}$ is $\mathbf{G}_r$. The ratio of product $p \in \mathcal{P}$ to produce, transport, or load one tonne using routine $r \in \mathcal{R}$ is $\mathbf{D}_r^p$. In particular, it is equal to 1 for transportation and loading routines.

In our context, inventory management ensures the capacity restrictions at the stocking points where one or several products are stored and the flow conserved along the downstream supply chain. We distinguish three types of stocking points $s \in \mathcal{S} = \{1, ..., \overline{S}\}$: upstream points (e.g. S_1 in Figure 4.2), intermediate points (e.g. S_4 in Figure 4.2), and downstream points (e.g. S_8 in Figure 4.2). The upstream stocking points ensure the supply of the raw phosphate slurry and other raw products. Intermediate stocking points facilitate the supply of the necessary semi-finished products to the transformation units and the finished products to the loading units. The downstream stocking points control the vessels' inventory levels over the loading time interval.

Vessels must be assigned to quays for loading within a time interval (a few consecutive periods) before sailing to the destination. The loading cannot be partial, i.e., either a shipment is fulfilled or not fulfilled. To accomplish this, a decision must be made on the assignment of vessels to quays based on a set of possible assignments $i \in \mathcal{I} = \{1, ..., \overline{I}\}$. Each $i \in \mathcal{I}$ is a quadruplet $(\underline{t}, \overline{t}, k, h)$ where $i_1 = \underline{t}$ is the starting period, $i_2 = \overline{t}$ is the ending period, $i_3 = k$ is the quay, and $i_4 = h$ is the shipment. It is worth mentioning that these decisions are incorporated into the model for two reasons. First, the part of the port dedicated to shipping belongs and is private to *OCP Group*. The latter manages customer orders and loading planning to meet deadlines and ensure responsiveness. Second, the coastal processing factory is located in the port. Thus, there is a significant gain in integrating vessel assignment in the model since it allows for the consideration of vessels as stock, and permits the direct loading of some products onto vessels without stocking them in between.

It should also be mentioned that there are several stages in the production process with intermediate stocks and several non-identical parallel transformation units. The same product can be produced by different transformation units. Still, products supplied by different units usually have minor differences in some characteristics, such as color. Unfortunately,

Table 4.3 PSIMVA Notation

	Notation	Definition
Indices	h	shipment
	i	possible assignment
	j	unit
	k	quay
	o	origin
	p	product
	r	routine
	s	stocking point
	t	time period
Sets	$\mathcal{H}$	set of shipments
	$\mathcal{H}^m$	set of mono-source shipments
	$\mathcal{I}$	set of possible assignments
	$\mathcal{I}_h$	set of possible assignments for shipment h
	$\mathcal{I}_{hk}$	set of possible assignments for shipment h on quay k
	$\mathcal{J}$	set of units
	$\mathcal{J}^{ld}$	set of loading units
	$\mathcal{J}^{tf}$	set of transformation units
	$\mathcal{K}$	set of quays
	$\mathcal{O}$	set of origins
	$\mathcal{P}$	set of products
	$\mathcal{P}_h$	set of products required by shipment h
	$\mathcal{R}$	set of routines
	$\mathcal{R}_h$	set of shipment h loading routines
	$\mathcal{R}_j$	set of unit j routines
	$\mathcal{R}_k$	set of routines flowing into quay k
	$\mathcal{R}_s^{in}$	set of routines flowing into stocking point s
	$\mathcal{R}_s^{out}$	set of routines flowing from stocking point s
	$\mathcal{S}$	set of stocking points
	$\mathcal{T}$	set of time periods
Parameters	$\mathbf{A}_j$	daily availability of unit j (hours)
	$\mathbf{B}_{rt}$	production capacity of routine r during period t (tonne)
	$\mathbf{C}_j$	daily nominal capacity of unit j (tonne)
	$\mathbf{D}_r^p$	ratio of product p to produce, transport, or load one tonne using routine r
	$\mathbf{E}_r$	daily loading capacity of routine r (tonne)
	$\mathbf{F}_r^p$	ratio of product p in one tonne produced using routine r
	$\mathbf{G}_r$	required time to activate routine r (hours)
	$\mathbf{L}_k$	length of quay k (meter)
	$\mathbf{MAX}_j$	weekly loading capacity of unit j (tonne)
	$\mathbf{MIN}_{s0}^p$	initial safety stock of product p at stocking point s (tonne)
	$\mathbf{MIN}_{st}^p$	safety stock of product p at stocking point s during period t (tonne)
	$\mathbf{Q}_h^p$	quantity of product p ordered by shipment h (tonne)
	$\mathbf{T}_r$	time to transform, transport, or load one tonne using routine r (hours)
	$\mathbf{V}_h$	length of vessel h (meter)
	$\boldsymbol{\tau}_j$	binary parameter equal to 1 if $j \in \mathcal{J}^{tf}$, 0 otherwise.

this is not acceptable to some customers. To remedy this situation, each product supplied to customers requesting the same color must follow a unique path from the origin to the vessel. This is ensured by an additional requirement, called the mono-source.

4.4 PROBLEM FORMULATION

Following the problem description, we introduce the decision variables, the objective function, and the constraints before presenting the PSIMVA MILP formulations.

4.4.1 Decision Variables

The problem variables are defined below:

$a_{rt} \in \{0,1\}$	: binary variable equal to 1 if the routine $r \in \mathcal{R}$ is active during period $t \in \mathcal{T}$, 0 otherwise.
$d_{rt} \geq 0$	: real variable indicating the total quantity produced, transported, or loaded by routine $r \in \mathcal{R}$ over period $t \in \mathcal{T}$.
$q_i \in \{0,1\}$	: binary variable equal to 1 if the possible assignment $i \in \mathcal{I}$ is selected, 0 otherwise.
$u^p_{ho} \in \{0,1\}$	: binary variable equal to 1 if product $p \in \mathcal{P}_h$ of shipment $h \in \mathcal{H}$ is supplied from origin $o \in \mathcal{O}$, 0 otherwise.
$v^p_{ht} \geq 0$	: real variable indicating the total volume of product $p \in \mathcal{P}$ loaded onto the vessel associated with shipment $h \in \mathcal{H}$ at the end of period $t \in \mathcal{T}$.
$v^p_{st} \geq 0$	: real variable indicating the total volume of product $p \in \mathcal{P}$ stored at stocking point $s \in \mathcal{S}$ at the end of period $t \in \mathcal{T}$.
$y_{rt} \in \{0,1\}$	: binary variable equal to 1 if a changeover allowing to activate routine $r \in \mathcal{R}_j$ with $j \in \mathcal{J}^{tf}$ is executed during period $t \in \mathcal{T}$, 0 otherwise.
$z_h \in \{0,1\}$	: binary variable equal to 1 if the demand of shipment $h \in \mathcal{H}$ is fully satisfied, 0 otherwise.

4.4.2 Objective Function

The quality of a solution of the PSIMVA is evaluated using KPIs. These KPIs measure the overall commercial and industrial effectiveness of the manufacturing system. At the commercial level, the aim is to maximize shipments' total fulfillment (TF). A shipment is said to be fulfilled if the whole quantities requested are delivered by the due date. At the industrial level, we seek the lowest possible number of product changeovers (PC). The two KPIs are defined as follows:

- **TotalFulfillment** (TF). This variable computes the percentage of fulfilled shipments (in terms of quantities): $TF(z) = 100 \times \frac{\sum_{p \in \mathcal{P}_h, h \in \mathcal{H}} \mathbf{Q}^p_h z_h}{\sum_{p \in \mathcal{P}_h, h \in \mathcal{H}} \mathbf{Q}^p_h}$

- **ProductChangeovers** (PC). This variable counts the number of changeovers to be minimized: $PC(y) = \sum_{r \in \mathcal{R}_{tf}, t \in \mathcal{T}} y_{rt}$

4.4.3 Constraints

The sets, parameters, and decision variables being introduced, the MILP constraints, are as follows:

$$\textit{Prod. cstr.:}\quad d_{rt} \leq \mathbf{B}_{rt} a_{rt} \qquad \forall r \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \tag{4.1}$$

$$\textit{Capa. cstr.:}\quad \sum_{r \in \mathcal{R}_j} d_{rt} \leq \mathbf{C}_j \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{4.2}$$

$$\sum_{r \in \mathcal{R}_j} \mathbf{T}_r d_{rt} + \boldsymbol{\tau}_j \sum_{r \in \mathcal{R}_j} \mathbf{G}_r y_{rt} \leq \mathbf{A}_j \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{4.3}$$

$$\textit{Inv. cstr.:}\quad v^p_{s0} = \mathbf{MIN}^p_{s0} \qquad \forall p \in \mathcal{P}, s \in \mathcal{S} \tag{4.4}$$

$$\sum_{r \in \mathcal{R}^{in}_s} \mathbf{F}^p_r d_{rt} + v^p_{s,t-1} = \sum_{r \in \mathcal{R}^{out}_s} \mathbf{D}^p_r d_{rt} + v^p_{st} \qquad \forall p \in \mathcal{P}, s \in \mathcal{S}, t \in \mathcal{T} \tag{4.5}$$

$$v^p_{st} \geq \mathbf{MIN}^p_{st} \qquad \forall p \in \mathcal{P}, s \in \mathcal{S}, t \in \mathcal{T} \tag{4.6}$$

$$v^p_{h0} = 0 \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H} \tag{4.7}$$

$$\sum_{r \in \mathcal{R}_h} \mathbf{F}^p_r d_{rt} + v^p_{h,t-1} = \begin{cases} v^p_{ht} & t \neq \overline{T} \in \mathcal{T} \\ \mathbf{Q}^p_h z_h & t = \overline{T} \end{cases} \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H} \tag{4.8}$$

$$\textit{Chg. cstr.:}\quad \sum_{r \in \mathcal{R}_j} a_{rt} \leq 1 \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{4.9}$$

$$\sum_{\substack{r \in \mathcal{R}_j \\ r \neq r'}} a_{r,t-1} + a_{r't} \leq 1 + y_{r't} \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \setminus \{1\} \tag{4.10}$$

$$y_{r't} \leq \frac{1}{2}\Big(\sum_{\substack{r \in \mathcal{R}_j \\ r \neq r'}} a_{r,t-1} + a_{r't}\Big) \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \setminus \{1\} \tag{4.11}$$

$$\textit{Ld. cstr.:}\quad \sum_{\substack{i \in \mathcal{I}_{hk} \\ i_1 \leq t \leq i_2}} \mathbf{E}_r q_i \geq d_{rt} \qquad \forall r \in \mathcal{R}_k, k \in \mathcal{K}, t \in \mathcal{T}, h \in \mathcal{H} \tag{4.12}$$

$$\sum_{i \in \mathcal{I}_h} q_i \leq z_h \qquad \forall h \in \mathcal{H} \tag{4.13}$$

$$\sum_{h \in \mathcal{H}} \sum_{\substack{i \in \mathcal{I}_{hk} \\ i_1 \leq t \leq i_2}} \mathbf{V}_h q_i \leq \mathbf{L}_k \qquad \forall k \in \mathcal{K}, t \in \mathcal{T} \tag{4.14}$$

$$\sum_{r \in \mathcal{R}_j} \sum_{t'=t}^{t+6} d_{rt'} \leq \mathbf{MAX}_j \qquad \forall j \in \mathcal{J}^{ld}, t \in \mathcal{T} \setminus \{\overline{T}-5, ..., \overline{T}\} \tag{4.15}$$

Mono. cstr.: $$\sum_{o \in \mathcal{O}} u_{ho}^p \leq z_h \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H}^m \tag{4.16}$$

$$\sum_{t \in \mathcal{T}} d_{rt} \geq \mathbf{Q}_h^p u_{ho}^p \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H}^m, r \in \mathcal{R}_o, o \in \mathcal{O} \tag{4.17}$$

Non-negativity and binary conditions in 4.4.1 (4.18)

The model constraints can be categorized into six groups, which are production (Prod. cstr.), capacity (Capa. cstr.), inventory (Inv. cstr.), changeovers (Chg. cstr.), loading (Ld. cstr.), and mono-source (Mono. cstr.) constraints. If a routine is active during a specific period, constraints (4.1) ensure that the quantity produced by this routine is bounded; otherwise, it is equal to zero. Constraints (4.2) impose the respect of production capacities in terms of quantities, and constraints (4.3) impose the respect of production capacities in terms of time. In the case of transformation units, the time consumed by changeovers is added. Constraints (4.4) correspond to the initial stocks at stocking points. Constraints (4.5) ensure inflow and outflow conservation for each stocking point. Constraints (4.6) control safety stock respect. Constraints (4.7) assume initial stocks in vessels are equal to zero while Constraints (4.8) track flow conservation on the vessels. For $t \neq \overline{T} \in \mathcal{T}$, these constraints update the stock level within the vessel while for $t = \overline{T}$, they ensure that the stock level of product $p \in \mathcal{P}_h$ corresponding to shipment $h \in \mathcal{H}$ is equal to the shipment request for the same product. Constraints (4.9) guarantee that at most one routine can be active on each unit during a specific period. If constraints (4.10) and (4.11) count the shift from one routine to another one on the same unit, a new changeover must then be activated. Constraints (4.12) guarantee the assignment of vessels to quays for loading. Constraints (4.13) ensure that each shipment is loaded with one assignment of its corresponding vessel. These constraints do not allow partial loading on different quays and time intervals. Constraints (4.14) check the respect of quays' length while assigning vessels. Constraints (4.15) make sure that the sum of the loaded quantities by the loading routines in the interval $[t, t+6]$ (over a week) should not exceed a maximal capacity. This capacity can change based on periods because of meteorological conditions. Constraints (4.16) ensure the satisfaction of shipments $h \in \mathcal{H}^m$ from at most one origin. If an origin is selected, constraints (4.17) ensure that its corresponding routines can be activated.

It is worth mentioning that the integration of production scheduling, inventory management, and vessel assignment is ensured through the variables d_{rt}, $r \in \mathcal{R}$, $t \in \mathcal{T}$, which control the quantities produced in the units, transported through conveyors, or loaded in stocking points or vessels. All the constraints, except the mono-source constraints (4.16) and (4.17) that are specific to this context, are generic and can be adapted to any process industry.

4.4.4 Models

Let Ω be the domain defined by Constraints (4.1) to (4.18) and x be the vector of the decision variables with $x_z = z$ and $x_y = y$ being the z and y variables, respectively. Let $f(x) = (f_1(x), f_2(x))$ be the vector function where $f_1(x) = TF(x_z)$ and $f_2(x) = -PC(x_y)$. The PSIMVA is then:

$$\begin{aligned} \textbf{Max} \;\; & f(x) && \text{(PSIMVA)} \\ s.t.: \;\; & x \in \Omega && (4.19) \end{aligned}$$

The solutions of the PSIMVA problem are called Pareto-optimal (efficient or non-dominated) solutions [179]. To tackle the PSIMVA, we consider two classical multiobjective methods: weighted sum and lexicographic. In the weighted sum method, the objective function maximizes a weighted function of the two KPIs, leading to the following model:

$$\begin{aligned} \textbf{Max} \;\; & w_1 f_1(x) + w_2 f_2(x) && (\text{PSIMVA}_{wgt}) \\ s.t.: \;\; & x \in \Omega && (4.20) \end{aligned}$$

In the lexicographic method, we first maximize TF in the following model:

$$\begin{aligned} \textbf{Max} \;\; & f_1(x) && (\text{PSIMVA}^1_{lexi}) \\ s.t.: \;\; & x \in \Omega && (4.21) \end{aligned}$$

Then, we minimize PC in the following model:

$$\begin{aligned} \textbf{Max} \;\; & f_2(x) && (\text{PSIMVA}^2_{lexi}) \\ s.t.: \;\; & x \in \Omega && (4.22) \\ & f_1(x) \geq \underline{TF} && (4.23) \end{aligned}$$

Constraint (4.23) restricts the $f_1(x)$ lower bound using $\underline{TF}$, the optimal value returned by model PSIMVA$^1_{lexi}$. The lexicographic order reflects decision makers' subjective preference with regard to the importance of the KPIs.

The Pareto optimality of model PSIMVA$_{wgt}$ solution(s) can be derived from the following theorem [180]:

Theorem 1. *If for any positive weights w_1 and w_2, there exists some $\bar{x} \in \Omega$ with the property:*

$$w_1 f_1(x) + w_2 f_2(x) \leq w_1 f_1(\bar{x}) + w_2 f_2(\bar{x}) \;\; \forall x \in \Omega$$

Then $\bar{x}$ is a Pareto-optimal solution for model PSIMVA.

The Pareto optimality of model PSIMVA$^2_{lexi}$ solution(s) are derived from the following theorem [181]:

Theorem 2. *$\bar{x} \in \Omega$ is Pareto-optimal for model PSIMVA if and only if it is the optimal solution of model PSIMVA$^2_{lexi}$ with $f_1(\bar{x}) = \underline{TF}$.*

Since PSIMVA is convex (f_1, f_2, and Ω with relaxed Constraints (4.18) are convex), the Pareto-optimal solution $\bar{x} \in \Omega$ returned by model PSIMVA$^2_{lexi}$ can also be obtained using model PSIMVA$_{wgt}$, as suggested by the following theorem [182]:

Theorem 3. *If $f(\Omega)$ is convex and $\bar{x}$ is Pareto-optimal for model PSIMVA, then there exist positive weights w_1 and w_2 with the property:*

$$w_1 f_1(x) + w_2 f_2(x) \leq w_1 f_1(\bar{x}) + w_2 f_2(\bar{x}) \;\; \forall x \in \Omega$$

Still, finding these weights for a given Pareto-optimal solution obtained by model PSIMVA$^2_{lexi}$ is complicated. Practically, to obtain a close $\underline{TF}$ value by PSIMVA$_{wgt}$, we choose *large* w_1 weights. Such a choice comes with a price, as will be discussed in the next section.

4.5 EXPLORATORY ANALYSIS

In the literature, the optimal allocation of resources to tasks over time is known to be a very complex combinatorial problem. This complexity grows significantly when considering this problem as a part of a large context of simultaneous optimization of production scheduling, inventory management, and vessel assignment. This section aims to explore the root causes of PSIMVA complexity. For this purpose, we first present the problematic followed by a detailed PSIMVA model analysis with the relevant findings.

4.5.1 Problematic

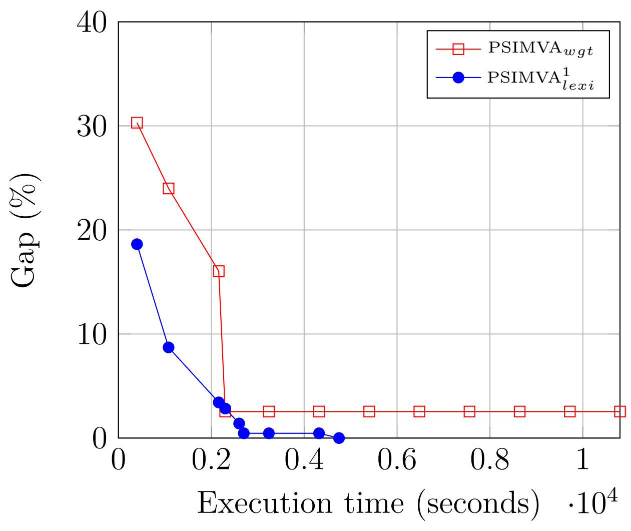
Over several PSIMVA instances, *OCP Group* operators used to rely on manual solutions. While these solutions are acceptable from a practical perspective, they do not provide any guarantee regarding the quality of the solutions obtained. Another weakness of the manual methods lies in their inability to take into consideration all the constraints imposed by the working rules of the industry. Furthermore, the mathematical model needs to be solved regularly at the end of each period to take into consideration the changes that can impact the input data. For instance, demand is subject to market fluctuations and is likely to change frequently. In particular, new shipments can be suddenly requested, while others can be subject to cancellation. Also, the capabilities of the manufacturing system can be disrupted due to maintenance requirements, machine failures, or delays in vessels' arrivals. Therefore, new tailored solutions must be provided in short intervals of time.

4.5.2 Model Analysis

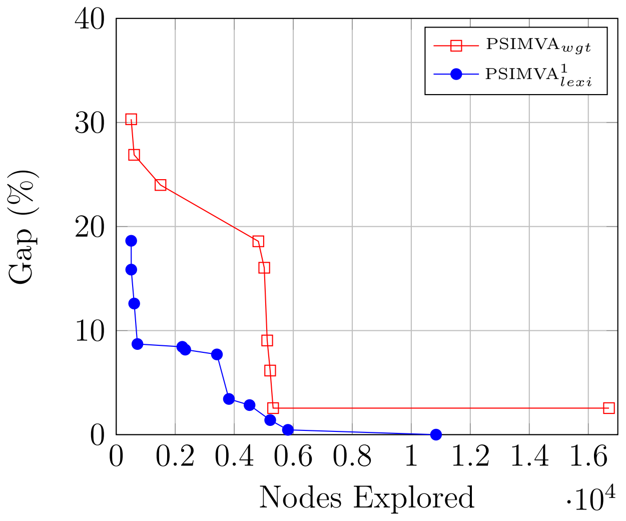
To identify the complexity sources, the investigation path consists of exploring each component of the PSIMVA model. All experiments are conducted using the default CPLEX version 12.9.0 on a representative instance of the PSIMVA.

Objective Function

The weighted-sum in model (PSIMVA$_{wgt}$) shows two main drawbacks. First, given that the chosen weights are not mathematically founded, the modeler has no control over the quality of the returned solution. The objective function is neither significant nor interpretable. TF is a percentage to be maximized, while PC is a counter to be minimized. Therefore, the objective value obtained cannot be analyzed correctly. Second, such a weighted objective function enhances the risk that the solver wastes significant computational time exploring unpromising regions of the polyhedron, i.e., regions with few good integer solutions. Indeed, different weights in the objective function induce different hyperplanes, and thus a focus on different regions of the polyhedron.



(a) Gap Evolution with Execution Time



(b) Gap Evolution with the # of Nodes Explored

Figure 4.3 Objective Function Analysis

To explore further the objective function, we compare, using the representative instance, the gap evolution utilizing the weighted bi-objective function, i.e., model (PSIMVA$_{wgt}$), and the gap evolution when maximizing TF only, i.e., model (PSIMVA$^1_{lexi}$). For the former, we use the weights $w_1 = 100$ and $w_2 = 1$. We run all experiments with a cutoff of three hours. Figure 4.3a highlights the integrality gap (henceforth referred to as "gap") evolution along the execution time, i.e., the gap between the objective value of the best integer solution and one of the optimal solution of the LR of the MILP. One can observe that default CPLEX reduces the gap below 2.55% within 2000s in both cases. However, while it succeeds in reaching optimality when maximizing only TF, it fails to accomplish the same using the weighted-sum objective function. This is supported further by Figure 4.3b, which highlights the gap evolution along with the number of explored nodes. Even when exploring significantly more nodes, default CPLEX fails to reach optimality for the weighted-sum optimization. One possible explanation is inefficient branching decisions, especially since they are based on arbitrary weighted-sum values that are neither significant nor interpretable. As a consequence, in terms of the dual simplex iterations per node, the same experiments show that, on average, default CPLEX executes 830 iterations per explored node during (PSIMVA$^1_{lexi}$) model optimization compared to 1155 iterations per explored node during (PSIMVA$_{wgt}$) model optimization. We use model (PSIMVA$^1_{lexi}$) for the exploration of constraints and decision variables.

Constraints

Throughout the exploration of constraints in the representative instance, we found that when the mono-source constraints (4.17) are removed from the model (PSIMVA$^1_{lexi}$), instances become relatively easier to solve. Contrariwise, with these constraints, the model is more difficult and requires huge computational time to find sufficiently good solutions. This suggests that the MILP without these constraints becomes significantly more loosely coupled compared to one with these constraints [183]. Indeed, from the industrial perspective, while a product $p \in \mathcal{P}$ can be produced on different units $j \in \mathcal{J}^{tf}$, small fluctuations in input doses affect the chemical transformation processes, leading to the same product $p \in \mathcal{P}$, but with different colors. From the client's perspective, the color change from the process is significant and not acceptable. The role of constraints (4.17) is ensuring, for each mono-source shipment, the uniqueness of the path linking the selected origin $o \in \mathcal{O}$ to the corresponding vessel. Thus, they have a significant impact on the coupling of production, inventory, and loading. For the remaining constraints, tests show that they do not have a significant impact on the solution time and do not significantly affect the gap, which usually slows down the solution process. Based on these findings, we conclude that in terms of constraints, the main source of complexity is the mono-source constraints.

Decision Variables

The solution of the (PSIMVA$^1_{lexi}$)'s LR, which is very fractional, is obtained rapidly. Then, a significant amount of time is consumed during the branching process both to find a feasible solution and to improve it. Based on this remark, it is necessary to check the effect of the decision variables, especially the binary variables of the model. We distinguish five classes of binary variables $\mathcal{B}_c, c \in \mathcal{C}$ where $\mathcal{C}$ denotes the set of classes. To alleviate notations, we remove the indexes. These classes are namely: the set of binary variables q assigning vessels to quays, the set of production variables a ruling the production inside the transformation units, the set of fulfillment variables z which decide whether a given shipment is satisfied or not, the set of mono-source variables u, and, finally, the set of changeover variables y. For the sake of identifying which classes of binary variables are the origin of the slowness of the solving process within the branch and bound ($B\&B$), we consider two metrics: the percentage of each class among all the binary variables (% Class), and the percentage of non-zero variables per class in the solution of the representative instance with mono-source

(% Non-zeros). Table 4.4 shows the statistics related to each class of binary variables.

Table 4.4 Binary Classes

Class	Index	# Variables	# Non-zeros	% Non-zeros	% Class
q	1	13402	64	**0.48**	**50.38**
a	2	5832	1870	32.06	21.92
z	3	64	61	95.31	0.24
u	4	1434	83	5.79	5.39
y	5	5868	714	12.17	22.06

We first observe that variables q dominate the whole set of the binary variables, consisting of 50.38 %. Second, only 0.48 % of these variables take non-zero values in the considered typical solution. These variables are more likely to be very influential in the solution process. To validate this hypothesis, we designed an investigation strategy, for which the mechanism is as follows. We solve (PSIMVA$^1_{lexi}$) using default CPLEX for a predefined time limit. Then, we identify the optimal solution found. Based on the latter and considering one class of binary variables at a time, we fix the binary variables worth 1 in the solution and solve (PSIMVA$^1_{lexi}$) again by using the optimal solution found as a warm-start. By doing this, we aim to check whether the problem becomes easier when fixing some binary variables from each class. The investigation strategy is described in Algorithm 4.4. Formally, we first solve the (PSIMVA$^1_{lexi}$) for a predefined time limit θ using function $solveMILP(x, \theta)$, where x is the vector of variables. The optimal integer solution found is saved in a vector x^{init}. We denote by x_n and x_n^{init} the elements of vectors x and x^{init}, respectively. For each class of binary variables, we fix to 1 the variables of class c that worth 1 in x^{init} using function $setBounds(x_n, 1, 1)$ where $x_n \in \mathcal{B}_c$. For a given $l, u \in \mathbb{R}$ and a variable $x \in \mathbb{R}^n$ where $n \in \mathbb{N}$, function $setBounds(x, \underline{b}, \bar{b})$ set the variable's lower bound to $\underline{b}$ and its upper bound to $\bar{b}$. In addition to the fixation process, the optimal solution found x^{init} is provided to the solver as a warm-start solution using the function $addWarmStart(x^{init})$. This gives rise to a restricted sub-problem that is solved again in the same amount of time θ. We denote x^* the returned solution and x_n^* its elements.

Five variable fixation scenarios are then considered. The corresponding results on the representative instance are given in Table 4.5. For each scenario, we report the number of binary variables in the model after fixation (# Binaries). The q variables proved to be the ones having a significant impact on the computational time. While the original model requires 801 seconds to reach the first integer solution, both the first integer (1st TF) and the optimal (Opt TF) solutions for the restricted q scenario problem are found rapidly, within 8s (1st

Algorithm 5: Investigation Strategy

```
Input: $x$, $\theta$
$x^{init} = solveMILP(x, \theta)$
for $c \in \{1, 2, 3, 4, 5\}$ do
    forall $x_n \in \mathcal{B}_c$ do
        if $x_n^{init} = 1$ then
            $setBounds(x_n, 1, 1)$
        end
    end
    $addWarmStart(x^{init})$
    $x^* = solveMILP(x, \theta)$
    forall $x_n \in \mathcal{B}_c$ do
        $setBounds(x_n, 0, 1)$
    end
end
```

Time) and 17s (Opt Time), respectively. For fixation scenarios related to the other classes of binary variables, the times required to reach the first integer and the optimal solutions are significantly higher. In some cases, no integer solution is found.

Table 4.5 Investigation Strategy Results

Scenario	# Binaries	1^{st} TF	1^{st} Time (s)	Opt TF	Opt Time (s)
Original	20538	92.98%	801	–	–
q	8662	88.52%	**8**	100%	**17**
a	13036	93.04%	447	100%	1510
z	20508	–	–	–	–
u	19873	90.77%	850	100%	1435
y	20382	–	–	–	–

These findings confirm the hypothesis, i.e., the variables q are very influential on the solution process. This is further reaffirmed by the following observation. For every single shipment, there are hundreds of possible combinations of periods during which and quays to which the corresponding vessel can be assigned. This gives rise to high symmetry within the model, increasing the time required by CPLEX to reach satisfactory solutions in a reasonable amount of time, making the commercial solver impractical. Furthermore, from all these possible combinations, at most one assignment should be selected. This makes the q variables highly fractional in the LR.

Example 1. *Considering the representative instance, Table 4.6 provides part of the solution of the LR for a given vessel. Except for the eight q variables provided, all other q variables are equal to zero. Each q has a starting period (e.g. 0227 equivalent to February 27th), an* *ending period (e.g. 0301 equivalent to March 01st), a quay (e.g. 1), and a shipment ID (e.g. 4586). From an industrial perspective, this is equivalent to a partial loading throughout the month until completion, either on the same quay or on different ones, as visualized in Figure 4.4.*

Table 4.6 q Values for Shipment 4586

q	Start	End	Quay	ID	Value
q_1	0227	0301	1	4586	0.3070
q_2	0226	0229	2	4586	0.1620
q_3	0224	0227	3	4586	0.1500
q_4	0210	0213	1	4586	0.1500
q_5	0209	0212	3	4586	0.0945
q_6	0208	0211	2	4586	0.0850
q_7	0213	0215	2	4586	0.0450
q_8	0225	0228	1	4586	0.0065

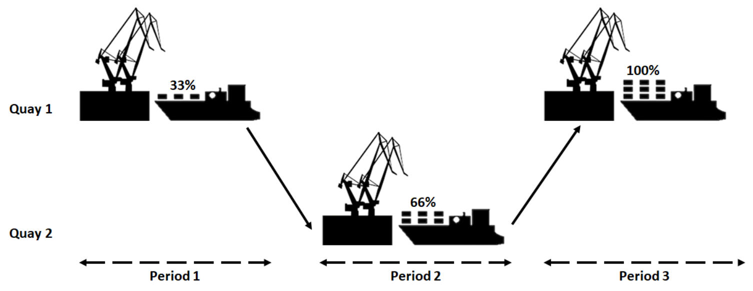


Figure 4.4 Partial Loading Example

To sum up, the complexity lies in the bi-objective weighted function, the mono-source constraints, and the large combinatorial space induced by the *q* variables. Given the large choice of the latter, the permutations of vessels among quays and periods induce several feasible solutions with the same objective value, making the execution time within the B&B lengthy. The exploratory analysis highlights the importance of tackling symmetry efficiently to reach satisfactory feasible solutions in a reasonable amount of time.

4.6 SOLUTION METHODOLOGY

In this section, we first highlight the actions taken to tackle the different sources of complexity identified previously. Then, we present the general decomposition framework. After that, we implement a practical variant for which the decomposition criterion considered is time. For comparison purposes, we finally mimic the greedy manual method used by *OCP Group* operators.

4.6.1 Prescriptions against Complexity

From the exploratory analysis, it is more practically relevant to define a KPIs hierarchy. Thus, to handle the bi-objective function, we rely on the lexicographic optimization method. We first maximize TF in the first stage (model (PSIMVA^1_{lexi})) and save the solution as a warm-start for the second stage, where we minimize PC with a lower bound constraint on TF (model (PSIMVA^2_{lexi})). For the constraints and variables, the mono-source constraints permit the classification of the provided instances into easy and hard instances while the q variables allow identifying from which side the hard ones can be tackled. The relaxed maximization of the first KPI provides a good upper bound. Thus, designing an efficient metaheuristic that will rapidly provide good lower bounds and tackle effectively the aforementioned complexities is the strategy we followed. Such a strategy will allow the elimination of unpromising branches in the B&B tree, making the solving process faster. We present next the designed metaheuristic framework, called hereafter the $\mathcal{ILNS}$ framework.

4.6.2 ILNS Framework

The $\mathcal{ILNS}$ framework, appearing inside the blue frame in Figure 4.5, relies on an iterative exploration of the search space to reduce the combinatorial complexity induced by the huge number of q variables. We denote by $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1$ the index set of q variables to be fixed, where $\mathcal{F}_0$ and $\mathcal{F}_1$ (initially empty) are the subsets of $\mathcal{F}$ containing the indexes of variables q to be fixed to 0 and 1, respectively. For a given instance, the framework takes as input the pool of q variables (flow [a]) and selects from it by filling $\mathcal{F}_0$ based on a predefined decomposition criterion in the *Vessel Assignment* step (see Sections 4.6.3 and 4.6.4). The subsets $\mathcal{F}_0$ and $\mathcal{F}_1$ (only $\mathcal{F}_0$ initially) are provided (flow [b]) as input to the *Problem Reduction* step, where all the q variables having their corresponding indexes $i \in \mathcal{F}_0$ are removed from model (PSIMVA^1_{lexi}) using preprocessing. Following the latter, some variables are removed from the model and others are fixed, leading to useless and redundant constraints. An example of a useless constraint is $0 \leq \mathbf{L}_k$, which can be obtained from constraints (4.14) for given quay $k \in \mathcal{K}$ and period $t \in \mathcal{T}$ when the *Problem Reduction* step removes all corresponding q variables. An example of a redundant constraint is $a_{rt} \geq 0$, which can be obtained from constraints (4.1) for routine $r \in \mathcal{R}_j$, unit $j \in \mathcal{J}^{tf}$, and period $t \in \mathcal{T}$ when d_{rt} is fixed to 0. Similarly, the latter occurs in constraints (4.12) when the *Problem Reduction* step removes all corresponding q variables. The constraint is redundant because a_{rt} is already defined as a binary variable. The reduced problem is then provided as an input (flow [c]) to the *Solving* step, where TF is maximized for a predefined time limit. In the solution obtained from the third step, the indexes of the q variables equal to 1 are added to $\mathcal{F}_1$. If the stopping condition is not met, the current solution is provided (flow [d]) as an input to the *Vessel Assignment* step, and the same process takes place again. The q variables having their corresponding indexes $i \in \mathcal{F}_1$ are fixed definitively to 1 subsequently. Once the stopping condition is satisfied, the current solution is provided as a warm-start to the *Wrap-up* step, where a restricted problem with all q variables corresponding to unfulfilled shipments (flow [a]) is solved. The final solution (flow [e]) is saved as an input for the second stage, where the model (PSIMVA$^2_{lexi}$) is solved.

Proposition 1. *Let $TF(itr)$ be the objective value corresponding to iteration $itr \in \mathcal{ITR} = \{1, 2, ..., \overline{ITR}\}$, which is a single repetition of the process that starts from step 1 and ends in step 3, and let TF_{corr} be the objective value after the* Wrap-up *step within the $\mathcal{ILNS}$ framework. Then:*
(i) $TF(itr)$ is non-decreasing in itr.
(ii) $TF_{corr} \geq TF(\overline{ITR})$.

The proof is straightforward since the solution of the previous iteration remains feasible for the current iteration, and the fixing mechanism does not alter it. Therefore, the non-decrease is guaranteed.

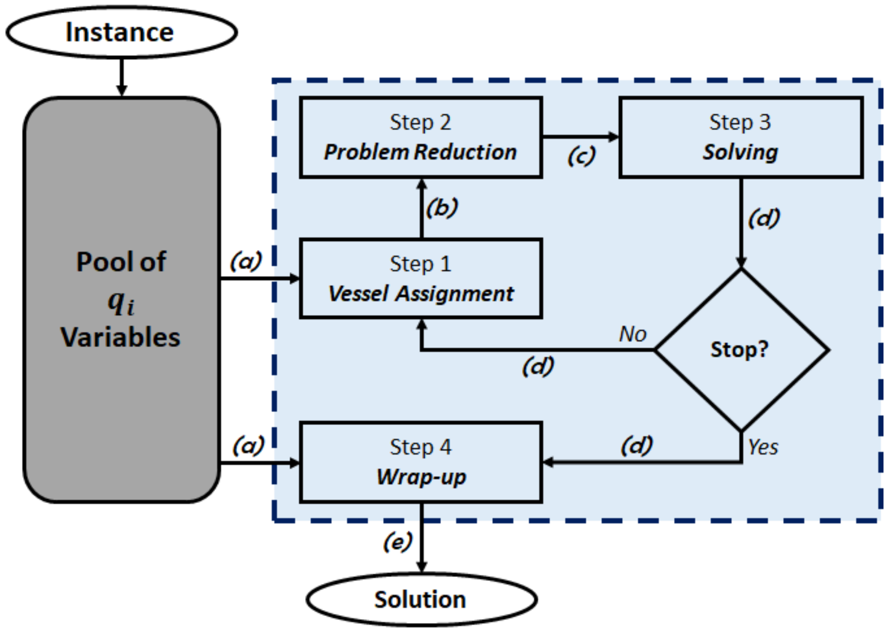


Figure 4.5 $\mathcal{ILNS}$ Framework

Proposition 1 highlights the relevance of the framework in maximizing TF. Compared to the standard LNS [61], which takes an initial solution as input, the $\mathcal{ILNS}$ framework constructs an initial solution in the first iteration before improving it in subsequent ones. Furthermore, at each iteration, part of the solution (fulfilled shipments) is fixed while the remaining (unfulfilled shipments) is destroyed to search for better solutions in the neighborhood of the current solution; this also maintains feasibility. In the proposed framework, once an initial solution is constructed, the *Vessel Assignment* and *Problem Reduction* steps act like the destroy operator while the *Solving* step acts like the repair operator. The *Wrap-up* step groups both operators. The destroy operator destructs part of the current solution while the repair operator rebuilds the destroyed solution. In contrast with the standard LNS, where the destroy operator usually contains an element of stochasticity such that different parts of the solution are destroyed in every invocation of the method [184], in the $\mathcal{ILNS}$, only the part of the solution we expect to improve is destroyed. Thus, the destroy operator destructs part of the current solution corresponding to unfulfilled shipments while the repair operator rebuilds part of the destroyed solution to fulfill the previously unfulfilled shipments.

Large-scale optimization problems suffer from symmetry. For PSIMVA, given the huge combinatorial space induced by the q variables, each shipment $h \in \mathcal{H}$ has hundreds of possible assignments $i \in \mathcal{I}_h$. This leads to several equivalent optimal solutions that can be obtained through permutations among quays and periods, consequently increasing execution time in the B&B tree. In the literature, the main trend is breaking symmetry, as in [185, 186]. However, several symmetry-breaking suggestions are costly in practice [187, 188]. A different perception consists of viewing symmetry as an asset instead of an issue for the following reasons. First, the more symmetry there is, the greater the possibility of finding high-quality solutions with a well-designed heuristic. The impact of wrong heuristic decisions is lessened because there are more possibilities. Second, the better the heuristic solution, the more chances there are of eliminating branches in the B&B tree and, therefore of converging more quickly. Third, we can begin with heuristics taking advantage of symmetry, and then apply mathematical programming at the end that exploits the heuristic solution to reduce the effects of symmetry. To do so in our context, we may for instance rank shipments in an increasing order $|\mathcal{I}_h|$, $h \in \mathcal{H}$ and prioritize them based on this order. Such a view is leveraged in the next sections.

The remaining aspect of the $\mathcal{ILNS}$ framework is the decomposition criterion. While several criteria are possible, we implement $\mathcal{ILNS}$ using a practically relevant criterion, which is time.

4.6.3 Time-based ILNS

The implemented variant called Time-based ILNS ($\mathcal{TILNS}$) decomposes the time horizon into smaller time intervals, called hereafter windows. Let $\mathcal{N} = \{1, 2, ..., \overline{N}\}$ be the set of windows. Each window $n \in \mathcal{N}$ is formed of several consecutive periods. Within the framework, before reaching the *Wrap-up* step, each iteration corresponds to a window solving, and the stopping condition corresponds to the exploration of all windows, i.e., $\overline{N}$ iterations. In step 1, the time decomposition criterion ensures that only the q variables belonging to the considered window are kept. Formally, let TI_n be the set of periods belonging to window $n \in \mathcal{N}$ and TI be the set of periods of the time horizon. A variable q is said to belong to window $n \in \mathcal{N}$ if its starting period i_1 belongs to TI_n. It follows that a shipment $h \in \mathcal{H}$ belongs to window $n \in \mathcal{N}$ if it has at least one q variable in that window. All q variables, for which $i_1 \notin TI_n$, are added to $\mathcal{F}_0$. While constructed windows can be a partition of the planning horizon, we consider overlapping windows to make better assignment decisions in the early periods and to benefit from improvement opportunities. To create windows, an intuitive way is to divide the number of periods of the time horizon by the number of windows, i.e., $|TI|/\overline{N}$. Each window $n \in \mathcal{N} \setminus \{\overline{N}\}$ is formed of $\lceil |TI|/\overline{N} \rceil$ periods in increasing order. The last window is formed of the remaining periods, i.e., $|TI| - (\overline{N} - 1)\lceil |TI|/\overline{N} \rceil$. To incorporate overlapping, we add to each window $n \in \mathcal{N} \setminus \{1\}$ the last δ periods of window $n - 1$.

In the $\mathcal{TILNS}$, the destroy operator is a variant of the related or Shaw-destroy strategy [189], often referred to as time-oriented destroy [190]. The repair operator is a greedy operator [60] since any satisfied shipment in a given window is fixed to that window.

After forming windows, we observe that there is a higher chance of satisfying more shipments when processing windows in non-decreasing order of the number of shipments. The intuition behind our observation is that windows with fewer shipments are easily solved due to the small combinatorial space, thus allowing the lower bound to be improved more quickly. We recall that this lower bound is used in subsequent iterations to reduce the execution time through branch pruning in the B&B, as well as variables fixing by reduced cost. Also, the prioritization of windows with fewer shipments frees space in subsequent windows. This offers shipments that can only be fulfilled in a limited number of windows more opportunities of being fulfilled. Such a strategy is a practical way to benefit from symmetry. Indeed, knowing that symmetry is high, prioritizing shipments with few possibilities by freeing space for them increases the chances of finding high-quality solutions. This is because other shipments with many possibilities can be assigned elsewhere. Example 2 illustrates $\mathcal{TILNS}$. We recall that since each shipment corresponds to a vessel, so we use the two terms interchangeably.

Example 2. *Consider a 30 days time horizon with 19 shipments. With* $\overline{N} = 4$ *and* $\delta = 2$*,* *periods 1-8 form window 1, periods 7-16 form window 2, periods 15-24 form window 3, and periods 23-30 form window 4. We consider that the number of shipments that can be fulfilled within each constructed window is 10, 13, 17, and 19, respectively. As for our observation, we process the windows in order 1, 2, 3, and 4.*

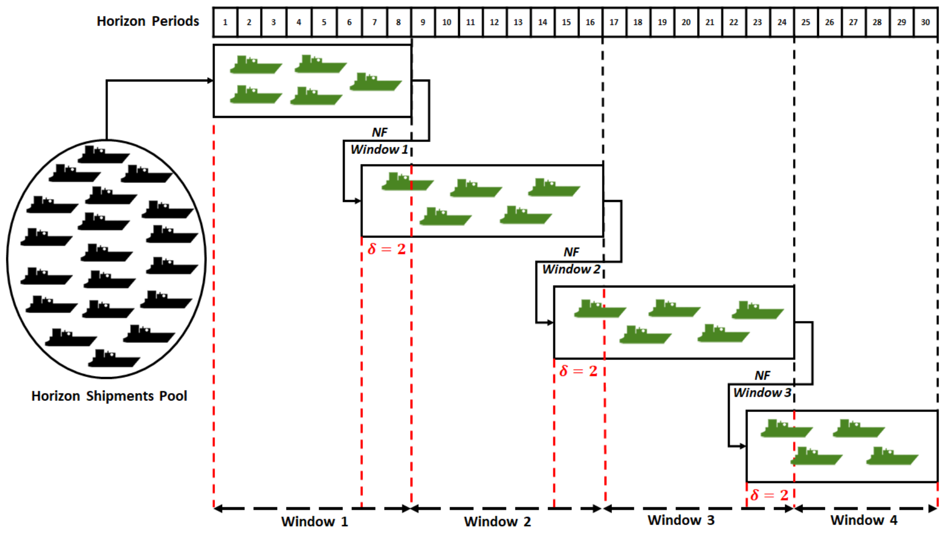


Figure 4.6 $\mathcal{TILNS}$ Example

We first solve ($PSIMVA^1_{lexi}$) restricted to the first window, where 5 among 10 possible shipments are fulfilled and assigned to it. We recall that capacity restrictions do not allow the fulfillment of all shipments belonging to a given window. Then, we solve ($PSIMVA^1_{lexi}$) restricted to the second window, where another 5 shipments (different from the previously fulfilled ones) among 13 possible are fulfilled and assigned to it. Similarly, after solving ($PSIMVA^1_{lexi}$) restricted to the third window, 5 among 17 possible shipments are fulfilled and assigned to it. The remaining shipments are fulfilled and assigned to window four after solving ($PSIMVA^1_{lexi}$) restricted to the last window. From a pool of initially unfulfilled shipments (in black), we fulfill all shipments (in green) after assigning each one to a single window. In such a case, the Wrap-up step is not necessary. The example is visualized in Figure 4.6.

4.6.4 Reference Algorithm

A natural way to tackle the PSIMVA is partitioning the planning horizon into several windows and then balancing the total demand and the number of vessels among them. This is the essence of the manual method followed by *OCP Group* experienced operators. For instance, a planning horizon of one month can be partitioned into four windows (weeks). Then, a quarter of the total demand, as well as a quarter of the number of vessels, can be assigned to each window. This ensures a balance in scheduling over the month. We refer to the algorithm we designed and implemented to mimic the manual method as the reference algorithm ($\mathcal{RA}$). $\mathcal{RA}$ can be viewed as another implementation of the $\mathcal{ILNS}$ framework, where the decomposition criteria are total demand and the number of vessels.

To implement $\mathcal{RA}$, we first partition ($\delta = 0$) the planning horizon into $\overline{N}$ windows like $\mathcal{TILNS}$. We maintain TI_n as the set of periods belonging to window $n \in \mathcal{N} = \{1, 2, ..., \overline{N}\}$. In addition, we denote by TP_h the index set of the possible windows for shipment $h \in \mathcal{H}$ and by $|TP_h|$ the number of these windows. We seek to assign each shipment to exactly one window among all its possible windows, i.e., windows during which the corresponding vessel can be loaded. Let TD be the total demand. For all windows, we note UB_{qty} as the upper bound of the total quantity to be assigned to each window based on the demand balance and UB_{nbr} as the upper bound of the number of shipments to be assigned to each window based on the balance of shipments. To balance demand and the number of shipments, it may be natural to compute UB_{qty} and UB_{nbr} as $\lceil TD/N \rceil$ and $\lceil (|\mathcal{H}|/N) \rceil$, respectively. However, such thresholds may not be practical as shown in the illustrative example 3 below.

Example 3. *Consider four windows and four shipments for which the demand is 40, 40, 40, and 10, respectively. It follows that* $TD = 130$*,* $UB_{qty} = 33$*, and* $UB_{nbr} = 1$*. Given these values, while the fourth shipment can be assigned to any window, all other shipments cannot be assigned to any window (*$40 > 33$*).*

To deal with the observation in example 3, the bounds are computed as $\alpha * (\lceil TD/N \rceil)$ and $\lceil \beta * (|\mathcal{H}|/N) \rceil$, respectively. The parameters α and β are chosen such that each shipment is assigned to a specific window. They are tuned based on the PSIMVA instances. For instance, in example 3, $\alpha = 40/33$ and $\beta = 1$ will ensure the assignment of the three shipments since $UB_{qty} = 40$ and $UB_{nbr} = 1$.

We order shipments in an increasing order of $|TP_h|$, $h \in \mathcal{H}$. Then, we start with shipments with only one possible window, i.e., $|TP_h| = 1$, $h \in \mathcal{H}$. These shipments are assigned first, then we move in ascending order of the number of windows to shipments that can be assigned to two, three, four, or more. All shipments are assigned based on bounds satisfaction. If a shipment can be assigned to more than one window, it is assigned in a greedy way to the earliest one as long as the two bounds are not violated. We recall that a variable q fits with window $n \in \mathcal{N}$ if its starting period i_1 belongs to TI_n. Thus, once a shipment is assigned to $n \in \mathcal{N}$, all q variables, for which $i_1 \notin TI_n$, are added to $\mathcal{F}_0$. This leads to a significant reduction in the number of q, and sometimes more than 85% of these binary variables are eliminated over the considered horizon. We solve all windows in parallel, i.e., $\mathcal{RA}$ requires a single iteration. In the obtained solution, the indexes of the q variables equal to 1 are added to $\mathcal{F}_1$ and are fixed definitively before the *Wrap-up* step. Example 4 illustrates $\mathcal{RA}$.

Example 4. *Consider a 30 days time horizon, 20 shipments, and four windows ($\overline{N} = 4$). The demand from each shipment is 20, i.e., $TD = 400$. Parameters α and β can both be set to 1. It follows that $UB_{qty} = 100$ and $UB_{nbr} = 5$. Except the blue shipments for which $|TP_h| = 4$, all other shipments have $|TP_h| = 1$. In particular, orange, yellow, grey, and black shipments can only be assigned to windows 1, 2, 3, and 4, respectively.*

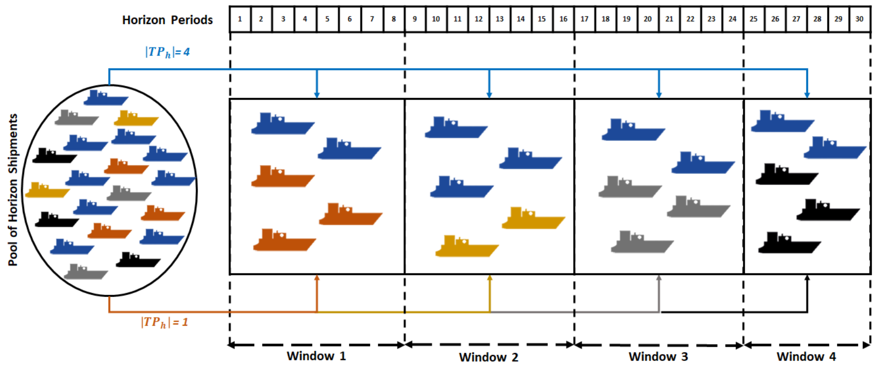


Figure 4.7 $\mathcal{RA}$ Example

We first assign these shipments to their windows and then we complete the assignment with the blue shipments. For the first window, the orange shipments have a total demand of 60. Thus, we can insert two blue shipments to reach $UB_{qty} = 100$, $UB_{nbr} = 5$. We move then to other windows, where we assign the remaining blue shipments similarly as above. A feasible assignment is then obtained for each window. The example is visualized in Figure 4.7. Once the Solving step is completed, if all shipments are fulfilled, $\mathcal{RA}$ is completed. Otherwise, we undergo the Wrap-up step as described in Section 4.6.2.

4.7 COMPUTATIONAL STUDY

Given that the PSIMVA is not covered in the literature, we compare the $\mathcal{TILNS}$ with both the Default CPLEX and the $\mathcal{RA}$ inspired by real life.

4.7.1 Test Plan

Instances are split into two classes, $L1$ and $L2$, without and with the mono-source constraints, respectively. The features of these instances including the number of shipments, demand (in tonne), and the number of variables, binaries, and constraints are presented in Table 4.7.

Table 4.7 Instances

Level	Name	Horizon	Shipments	Demand	Variables	Binaries	Constraints
L1	L1-1	32	54	806360	944328	18257	3655379
	L1-2	32	58	826460	944371	18354	3662118
	L1-3	32	61	1066290	936657	11155	3610913
	Avg	32	58	899703	941785	15922	3642803
L2	L2-1	30	58	1797910	450772	15237	1964908
	L2-2	30	58	1566120	450773	15237	1964909
	L2-3	30	58	1852340	450772	15144	1964908
	L2-4	30	58	1433630	450773	15144	1964909
	L2-5	30	58	1904100	450789	15237	1957865
	L2-6	30	58	1740780	450789	15144	1966865
	L2-7	32	61	955738	948009	18264	3679588
	L2-8	32	39	1044550	402212	41136	1872088
	L2-9	32	62	2031400	947598	17966	3506473
	L2-10	31	91	1304370	947598	17966	3506473
	L2-11	24	40	1043330	298693	33284	1488475
	Avg	30	58	1515842	568070	19978	2348860

The coding language is C++, and tests are conducted using version 12.9.0 of the IBM ILOG CPLEX solver. All experiments were carried out on a 3.20GHz Intel(R) Core(TM) i7-8700 processor, with 64GiB System memory, running on Oracle Linux Server release 7.7. We use real time to measure runtime.

4.7.2 Computational Results

In what follows, we first present the results obtained by the default CPLEX on model (PSIMVA$^1_{lexi}$) ($DPSIMVA^1_{lexi}$) and compare them to those of $\mathcal{RA}$ ($RPSIMVA^1_{lexi}$) and $\mathcal{TILNS}$ ($TPSIMVA^1_{lexi}$), respectively. We use the following metrics: the TF value, the relative gap, the number of integer solutions, and the time reduction factor. The latter is computed using the formula $\frac{\theta^{Best}_{Default}}{\theta^{Best}_{Algorithm}}$, i.e., the ratio of the time required to reach the best TF using the default CPLEX and the time required to reach the best TF using either $\mathcal{RA}$ or $\mathcal{TILNS}$ algorithms.

Table 4.8 describes the $DPSIMVA^1_{lexi}$ results over $\theta_{Default} = 180$ minutes. The TF progress is recorded after 10 (TF_{10}), 30 (TF_{30}), 60 (TF_{60}), and 180 (TF_{180}) minutes. We also report the value of TF for the best solution found (TF_{Best}), its relative gap (G_{Best}), time to find it (T_{Best}), and the number of integer solutions (IS) found. The relative gap is the integrality gap, computed with regards to the optimal solution of the problem's LR. For complex, large-scale, real-world problems, *good* feasible solutions are also desirable when optimality cannot be reached. Thus, we report the metric IS, i.e., the number of integer solutions found during

Table 4.8 $DPSIMVA_{lexi}^{1}$ Results

Name	UB	$DPSIMVA_{lexi}^{1}$							
		TF_{10}	TF_{30}	TF_{60}	TF_{180}	TF_{Best}	G_{Best}	T_{Best}	IS
L1-1	92.52%	70.98%	92.52%	-	-	92.52%	0.00%	15.20	14
L1-2	92.65%	71.11%	92.65%	-	-	92.65%	0.00%	13.93	6
L1-3	93.10%	83.69%	83.72%	92.98%	93.07%	93.07%	0.03%	85.45	22
Avg	92.76%	75.26%	83.72%	92.98%	93.07%	92.75%	**0.01%**	**38.19**	**14**
L2-1	97.30%	45.89%	45.89%	45.89%	77.90%	77.90%	19.94%	113.72	18
L2-2	99.22%	72.65%	72.65%	77.60%	87.25%	87.25%	12.06%	123.72	17
L2-3	97.30%	45.89%	45.89%	45.88%	77.90%	77.90%	19.94%	116.38	18
L2-4	99.22%	72.65%	72.65%	77.60%	87.25%	89.25%	12.06%	124.33	17
L2-5	99.03%	49.04%	49.04%	49.04%	89.71%	89.71%	9.41%	159.83	22
L2-6	99.03%	49.04%	49.04%	49.04%	89.71%	89.71%	9.41%	159.83	22
L2-7	93.43%	0.00%	80.81%	85.17%	88.06%	88.06%	5.75%	179.07	12
L2-8	93.34%	40.70%	93.34%	-	-	93.34%	0.00%	16.27	9
L2-9	99.08%	1.70%	91.60%	98.62%	99.08%	99.08%	0.00%	80.45	22
L2-10	93.74%	0.00%	82.52%	92.94%	92.94%	93.71%	0.03%	168.00	9
L2-11	95.68%	83.14%	95.68%	-	-	95.68%	0.00%	19.65	19
Avg	96.94%	40.29%	65.57%	69.09%	86.34%	89.05%	**8.06%**	**114.66**	**17**

the solving process. For $L1$ instances, the default CPLEX reaches optimality except for $L1-3$, where it is near-optimal. The average time needed to find the best solution is around 38 minutes. For $L2$, optimality is reached for three instances only ($L2-8$, $L2-9$, and $L2-11$), and the average gap within $\theta_{Default}$ is around 8%.

Table 4.9 presents the $RPSIMVA_{lexi}^{1}$ results over $\theta_{\mathcal{RA}} = 10$ minutes, partitioned into 8 minutes for the first iteration (TF_8) and 2 minutes for the *Wrap-up* step (TF_{8+2}). Since the average time horizon is 30 days, we fix parameter $\overline{N} = 4$, i.e., four windows. Parameters α and β are set to 1.02 and 1.5, respectively. In this table, the section Gain to Default measures the gain of $\mathcal{RA}$ to default CPLEX using: the difference between the TF found by $\mathcal{RA}$ in 10 (8+2) minutes and its counterpart found by the default CPLEX in 10 minutes (ΔTF_{10}), the difference between the TF found by $\mathcal{RA}$ in 10 (8+2) and the best TF value reached by default CPLEX within $\theta_{Default}$ (ΔTF_{Best}), the time reduction factor (TRF) as a ratio of default CPLEX T_{Best} to 10 minutes (even if $\mathcal{RA}$ may require less), and the difference in terms of the number of IS regardless of the execution time (ΔIS).

For $L1$ instances, $\mathcal{RA}$ achieves optimality while reducing the execution time with an average

Table 4.9 $RPSIMVA^1_{lexi}$ Results

Name	UB	$RPSIMVA^1_{lexi}$				$\Delta DPSIMVA^1_{lexi}$			
		TF_8	TF_{8+2}	Gap	IS	ΔTF_{10}	ΔTF_{Best}	TRF	ΔIS
L1-1	92.52%	90.80%	92.52%	0.00%	14	21.54%	0.00%	1.52	0
L1-2	92.65%	92.65%	92.65%	0.00%	14	21.54%	0.00%	1.39	8
L1-3	93.10%	87.95%	93.10%	0.00%	14	9.41%	0.03%	8.55	-8
Avg	92.76%	89.47%	92.75%	**0.00%**	14	**17.50%**	**0.01%**	**3.82**	**0**
L2-1	97.30%	81.12%	89.39%	8.13%	20	43.50%	11.49%	11.37	2
L2-2	99.20%	84.63%	90.17%	9.12%	16	17.52%	2.92%	12.37	-1
L2-3	97.30%	81.12%	91.51%	5.95%	20	45.62%	13.61%	11.64	2
L2-4	99.22%	84.63%	87.80%	11.51%	16	32.64%	0.55%	12.43	-1
L2-5	99.03%	68.75%	90.10%	9.02%	19	41.06 %	0.39%	15.98	-3
L2-6	99.03%	69.40%	90.20%	8.92%	19	41.16%	0.49%	15.98	-3
L2-7	93.43%	75.77%	88.46%	5.32%	14	88.46%	0.40%	17.91	2
L2-8	93.34%	88.64%	93.34%	0.00%	8	52.64%	0.00%	1.63	-1
L2-9	99.08%	80.29%	99.08%	0.00%	20	97.38%	0.00%	8.05	-2
L2-10	93.74%	91.10%	92.70%	1.11%	18	92.70%	-1.01%	16.80	9
L2-11	95.68%	78.31%	95.68%	0.00%	9	12.54%	0.00%	1.97	-10
Avg	96.94%	80.34%	91.68%	**5.37%**	16.27	**51.38%**	**2.62%**	**11.47**	**0**

TRF of 3.82. For $L2$ instances, TF_{10} values are improved by 51.38 % on average, while slight improvements are reported for TF_{best}. This shows that $\mathcal{RA}$ can quickly converge to the best solution compared to default CPLEX. Indeed, $\mathcal{RA}$ improves TF by 2.62% while being 11.47 times faster on average for $L2$ instances compared to default CPLEX. Another important aspect of $\mathcal{RA}$ is that the second solving TF_{8+2} can significantly improve the feasible solutions found in the first solving. Overall in L2 instances, the $\mathcal{RA}$ average gap in 10 minutes is about 5.37 %, while this gap is about 8.06 % after more than 114 minutes on average for the default CPLEX. On average, default CPLEX and $\mathcal{RA}$ find the same number of integer solutions.

Table 4.10 presents the $TPSIMVA^1_{lexi}$ results within $\theta_{\mathcal{TILNS}} = 10$ minutes and its comparison with both $DPSIMVA^1_{lexi}$ and $RPSIMVA^1_{lexi}$. Parameter δ is set to 2, i.e., two periods overlap. For a fair comparison with $\mathcal{RA}$, we allocated two minutes for each of the four windows considered $W_i, i \in \{1,2,3,4\}$. Then, we run the *Wrap-up* step ($St4$) for two additional minutes.

As one can observe, $\mathcal{TILNS}$ brings the gap below 5% in all instances. It beats default CPLEX, and also outperforms $\mathcal{RA}$ in terms of the solution quality in all the instances (except

Table 4.10 $TPSIMVA^1_{lexi}$ Results

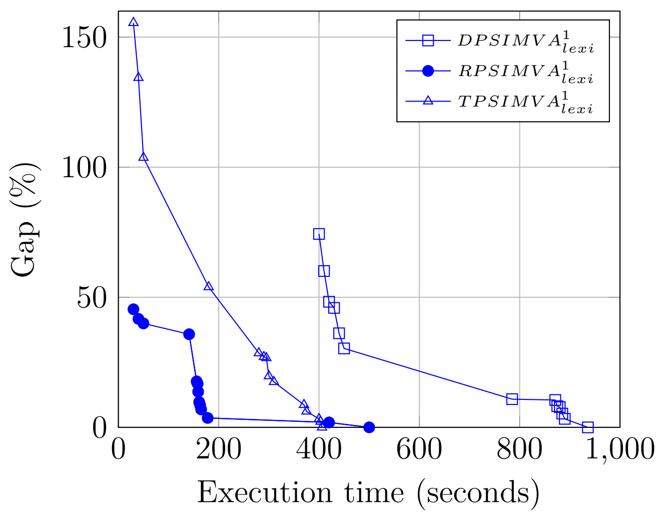
Name	UB	$TPSIMVA^1_{lexi}$							$\Delta DPSIMVA^1_{lexi}$				$\Delta RPSIMVA^1_{lexi}$	
		$W1$	$W2$	$W3$	$W4$	$St4$	Gap	IS	ΔTF_{10}	ΔTF_{Best}	TRF	ΔIS	ΔTF_{Best}	ΔIS
L1-1	92.52%	45.42%	60.10%	78.75%	92.52%	92.52%	0.00%	16	21.54%	0.00%	1.52	2	0.00%	2
L1-2	92.65%	45.61%	62.85%	82.95%	89.55%	92.65%	0.00%	25	21.54%	0.00%	1.39	19	0.00%	11
L1-3	93.10%	0.00%	0.00%	63.48%	83.03%	93.10%	0.00%	60	9.41%	0.03%	8.55	38	0.00%	46
Avg	92.76%	30.34%	40.98%	75.06%	88.37%	92.75%	**0.00%**	34	**17.50%**	**0.01%**	**3.82**	**20**	**0.00%**	**20**
L2-1	97.30%	58.20%	81.71%	88.00%	92.18%	93.48%	3.93%	27	47.59%	15.58%	11.37	9	4.09%	7
L2-2	99.22%	59.84%	72.64%	83.48%	94.44%	94.44%	4.82%	38	21.79%	9.71%	12.37	21	4.27%	22
L2-3	97.30%	57.43%	82.25%	87.30%	91.78%	92.92%	4.50%	25	47.03%	15.02%	11.64	7	1.41%	5
L2-4	99.22%	60.10%	71.34%	84.20%	94.72%	96.96%	2.28%	36	41.80%	9.71%	12.43	19	9.16%	20
L2-5	99.03%	47.24%	59.46%	78.63%	88.70%	95.10%	3.97%	25	46.06%	5.39%	15.98	3	5.00%	6
L2-6	99.03%	47.24%	59.46%	71.43%	92.70%	94.96%	4.11%	22	45.92%	5.25%	15.98	0	4.76%	3
L2-7	93.43%	47.30%	76.25%	87.20%	88.69%	89.84%	3.84%	18	89.84%	1.78%	17.91	6	1.38%	4
L2-8	93.34%	49.66%	57.98%	60.52%	91.68%	91.68%	1.78%	15	50.98%	-1.66%	1.63	6	-1.66%	7
L2-9	99.08%	43.04%	53.53%	53.53%	53.53%	95.06%	4.06%	30	93.36%	-4.08%	8.05	8	-4.02%	10
L2-10	93.74%	50.30%	65.43%	77.72%	84.43%	92.84%	0.96%	21	92.84%	-0.87%	16.80	12	0.14%	3
L2-11	95.68%	32.23%	70.47%	81.38%	90.52%	95.68%	0.00%	23	12.54%	0.00%	1.97	4	0.00%	14
Avg	96.94%	50.23%	68.23%	77.61%	87.58%	93.91%	**3.11%**	25	**53.61%**	**5.08%**	**11.47**	**9**	**2.23%**	**9**

$L2-8$, $L2-9$, and $L2-11$) and the number of integer solutions found during the solution process. While $\mathcal{RA}$ finds rapidly the first good integer solution, $\mathcal{TILNS}$ finds more integer solutions over time and achieves a better gap after the *Wrap-up* step. More integer solutions are very important for such large-scale problems. It implies that promising regions of the polyhedron are being explored. This gives the user the possibility to stop before optimality, i.e., once a satisfactory solution is obtained.

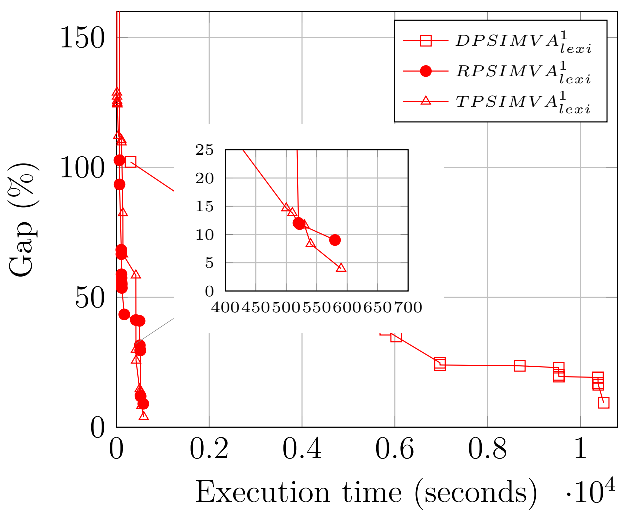
The results in Table 4.10 highlight the incremental feature of $\mathcal{TILNS}$. Indeed, from one iteration to the subsequent one, TF improves strictly in almost all instances. It also improves strictly in the *Wrap-up* step. This confirms Proposition 1. In addition, the significant improvements within 10 minutes suggest that combining the good upper bound from the LR with the good lower bounds from each iteration permits the pruning of several branches in the B&B tree and fixes several variables using reduced costs. Another explanation is that $\mathcal{TILNS}$ benefits from symmetry, as mentioned in our observation. It is worth mentioning that the effect of the mono-source constraints is reduced through the reduction of the number of q variables in the model. The mono-source constraints are linking u variables to d variables, and the latter is linked to q variables. Hence, once the number of q variables is reduced, several u variables are fixed either to 0 or 1, leading finally to a dampening of the effect of these constraints.

The improvement of TF over time is visualized in Figure 4.8 for one instance from each class. For $L2-5$, given that $\mathcal{TILNS}$ and $\mathcal{RA}$ outperform the default CPLEX, key data pertaining to time and the gap axes is highlighted in the zoom-in box.

The (PSIMVA^1_{lexi}) results highlight three main aspects. First, the most important result of our solution methodology is the shift towards an execution time that is less than or equal to 10 minutes. While the default CPLEX takes hours to reach satisfactory feasible solutions, both $\mathcal{RA}$ and $\mathcal{TILNS}$ provide mostly better solutions in less than 10 minutes. Comparing $L1$ and $L2$ instances, we can observe that, for the first class, both $\mathcal{TILNS}$ and $\mathcal{RA}$ reduce the time to optimality compared to default CPLEX. For the second class, both methods reduce significantly the time required to reach significantly better solutions compared to default CPLEX. This shift is very practical, as will be discussed in Section 4.7.3. Second, $\mathcal{TILNS}$ provides significantly more integer solutions during the solving process. Third, the designed heuristics ensure a significant relative gap gain, especially for $L2$ instances. On average, while the default CPLEX gap is around 50% within 10 minutes, $\mathcal{RA}$ reaches a gap below 10% and $\mathcal{TILNS}$ reaches a gap below 5%. It is worth mentioning that the results (gap $<$ 5%) achieved considering *OCP Group*'s main coastal processing factory can be easily extended to other coastal processing factories of *OCP Group* located in Morocco



(a) L1-1 Instance Gap Evolution



(b) L2-5 Instance Gap Evolution

Figure 4.8 Comparison of Gap Improvement along Execution Time

and worldwide. From a financial perspective, and compared to the manual solutions of *OCP Group* operators, the $\mathcal{TILNS}$ average gap below 5% on $L2$ instances is equivalent to an expected increase of about 5% in the corporation's annual turnover, i.e., hundreds of millions of dollars.

From an industrial perspective, there are often low tides in the port or shipment delays. This makes constraints (4.3) slack. Indeed, the company cannot produce indefinitely because of limited storage capacity. CPLEX does not incorporate this, and it allows more changeovers as long as it does not deteriorate the TF value. Following this observation, we want to minimize the number of changeovers and the time associated with them using the model (PSIMVA$^2_{lexi}$). We also compare the obtained results with the weighted-sum model (PSIMVA$_{wgt}$).

Table 4.11 presents the results of the weighted-sum model (PSIMVA$_{wgt}$), the lexicographic model (PSIMVA$^2_{lexi}$) where $\underline{TF}$ is obtained from $DPSIMVA^1_{lexi}$ ($DPSIMVA^2_{lexi}$), the lexicographic model PSIMVA$^2_{lexi}$ where $\underline{TF}$ is obtained from $RPSIMVA^1_{lexi}$ ($RPSIMVA^2_{lexi}$), and the lexicographic model PSIMVA$^2_{lexi}$ where $\underline{TF}$ is obtained from $TPSIMVA^1_{lexi}$ ($TPSIMVA^2_{lexi}$). We report the best PC (PC_{Best}), the time when it is found (T_{Best}), and TF_{Best}. The cutoff time for the (PSIMVA$_{wgt}$) model optimization with default CPLEX is 3 hours. In the lexicographic models, the cutoff time for PC minimization is 15 minutes. For TF, we report TF_{Best} achieved by default CPLEX within 180 minutes, $\mathcal{RA}$ within 10 minutes, and $\mathcal{TILNS}$ within 10 minutes. For (PSIMVA$_{wgt}$), we use the weights $w_1 = 100$ and $w_2 = 1$, which reflect decision makers' subjective preference with regard to the importance of the KPIs. They were

Table 4.11 Number of Product Changeovers

Name	LB	$PSIMVA_{wgt}$		$DPSIMVA^2_{lexi}$			$RPSIMVA^2_{lexi}$			$TPSIMVA^2_{lexi}$		
		PC_{Best}	TF_{Best}	PC_{Best}	T_{Best}	TF_{Best}	PC_{Best}	T_{Best}	TF_{Best}	PC_{Best}	T_{Best}	TF_{Best}
L1-1	0	0	92.52%	0	7.10	92.52%	0	19.42	92.52%	0	80.31	92.52%
L1-2	0	0	92.65%	0	0.45	92.65%	0	0.43	92.65%	0	0.42	92.65%
L1-3	0	0	93.10%	0	0.44	93.07%	0	0.43	93.10%	0	0.41	93.10%
Avg	0	**0**	**92.75%**	**0**	2.67	**92.75%**	**0**	6.76	**92.75%**	**0**	27.05	**92.75%**
L2-1	2	2	87.24%	2	12.51	77.90%	2	11.91	89.39%	2	171.56	93.48%
L2-2	2	2	81.87%	2	11.68	87.25%	2	12.00	90.17%	2	409.79	94.44%
L2-3	2	2	85.27%	2	5.08	77.90%	2	16.64	91.51%	2	222.11	92.92%
L2-4	2	2	80.01%	2	8.33	89.25%	2	11.91	87.80%	2	688.09	96.96%
L2-5	2	**4**	84.63%	2	41.00	89.71%	2	29.09	90.10%	2	70.89	95.10%
L2-6	2	**12**	73.43%	2	31.27	89.71%	2	27.75	90.20%	2	200.81	94.96%
L2-7	0	0	86.05%	0	200.14	88.06%	0	162.53	88.46%	0	70.73	89.84%
L2-8	0	0	93.34%	0	1.62	93.34%	0	5.55	93.34%	0	1.61	91.68%
L2-9	0	0	96.92%	0	2.16	99.08%	0	7.07	99.08%	0	2.21	95.06%
L2-10	0	0	93.74%	0	1.81	93.71%	0	2.39	92.70%	0	1.77	92.54%
L2-11	0	0	95.68%	0	1.46	95.68%	0	1.89	95.68%	0	1.39	95.68%
Avg	1	**2**	**87.11%**	**1**	28.82	**89.05%**	**1**	26.25	**91.68%**	**1**	167.36	**93.91%**

obtained from simulations conducted by the company. From a profit perspective, the company prefers fulfilling a higher demand than minimizing changeovers. As one can observe, the lexicographic method requires less time compared to the weighted sum method. Also, when using $\mathcal{RA}$ and $\mathcal{TILNS}$ to maximize TF, we reach optimality on the second KPI. This is not always the case using model $PSIMVA_{wgt}$, where in two instances ($L2-5$ and $L2-6$), optimality is not reached for PC even if the optimization is run for 3 hours. Moreover, one could observe that the TF values obtained by the lexicographic method are far better than those of the weighted-sum method, especially in $L2$ instances.

The gained changeover time saves thousands of dollars (e.g. fewer hours worked, less energy consumed), which can be used to perform maintenance or generate thousands of dollars by creating space for more shipments. It is worth mentioning that the significant time reduction allows decision-makers to optimize several order scenarios for KPIs. This supports the construction of an importance order for KPIs.

4.7.3 Managerial Insights

With the quick optimization capability gained from the shift to less than 10 minutes, using the optimizer becomes an efficient decision-making tool to check, control, simulate, and re-optimize schedules. It makes the supply chain more resilient to unexpected events and risks

[191], including the weather in the port, resource management, and the management of shipments. Regarding the weather in the port, the rapid re-optimization allows the company to react to inclement weather conditions that make vessel loading impossible. For resource management, maintenance planning becomes more efficient by incorporating it a priori or a posteriori into the mathematical model. In this section, we focus on events pertaining to the management of shipments, which have been implemented and put into practice.

The planning division usually incorporates two types of shipments: confirmed and unconfirmed ones. Confirmed shipments are those for which the vessels are expected to arrive at a specific interval, while unconfirmed shipments are those for which the arrival schedule has not yet been fixed. They may be delayed or even canceled. Instead of considering the whole set of shipments, the provided solutions allow the iterative insertion of shipments according to the level of confirmation. This gives rise to a layer-based optimization process, where each layer restrictively contains the confirmed shipments. Schematically, we start solving a restricted model focusing first on the confirmed shipments. Then, the unconfirmed ones can be added to the model once they are confirmed, and a new re-optimization solving, using the best available integer solution as warm-start, is carried out. This fits very well with the iterative re-optimization process introduced through $\mathcal{TILNS}$. A visualization with two layers is illustrated in Figure 4.9.

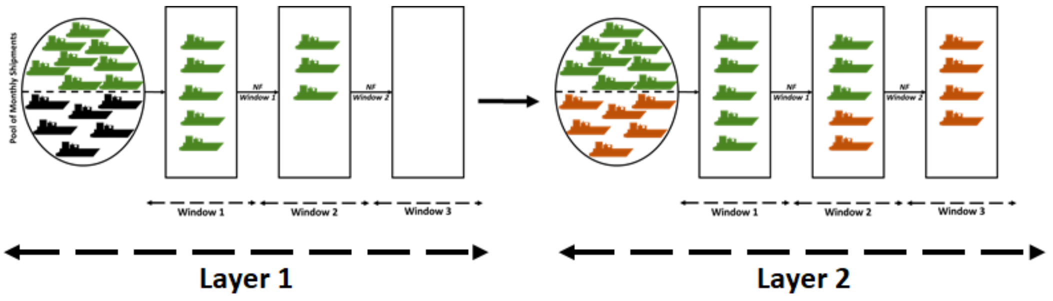


Figure 4.9 Confirmed and Unconfirmed Shipments

The first optimization layer deals with the confirmed shipments (in green), while the second optimization layer deals with the unconfirmed shipments (in black) that have become confirmed (in orange). As shown, the first layer solution is kept as a starting point for the second layer. It is an efficient way to leverage the available information and incorporate new information once it is available. The company can also plan the first time interval(s) deterministically since information is usually available and there is a delay in the planning of the remaining uncertain intervals. This can be done in different ways, including relaxing all binary variables or considering representative aggregated variables for uncertain intervals.

In Table 4.12, we present an example in which we consider both confirmed and unconfirmed shipments in a small set of instances. Using $\mathcal{TILNS}$, we compare the scenario where we conduct a layer-based optimization with the one where we take into consideration all shipments at once (like in Table 4.10). In the former, we consider solely the confirmed shipments in each layer. We report the number of confirmed shipments (# Conf.), the fulfillment percentage of confirmed shipments (% Conf.), the number of unconfirmed shipments (# Unconf.), and the fulfillment percentage of unconfirmed shipments (% Unconf.). In the first layer, we report the number of confirmed shipments, and in the second layer, the subsequently confirmed shipments (initially unconfirmed in the first layer).

Table 4.12 Re-optimization Example when considering Confirmed and Unconfirmed Shipments

Name	Layer 1		Layer 2		Shipments		$TPSIMVA^1_{lexi}$	
	# Conf.	% Conf.	# Conf.	% Conf.	# Conf.	# Unconf.	% Conf.	% Unconf.
L2-1	39	100.00%	17	100.00%	56	2	87.50%	100.00%
L2-3	35	100.00%	21	100.00%	56	2	83.93%	100.00%
L2-5	40	100.00%	16	100.00%	56	2	91.07%	100.00%
L2-7	32	100.00%	26	100.00%	58	3	91.38%	100.00%
L2-9	36	100.00%	23	100.00%	58	3	86.21%	100.00%
Avg	36	100.00%	21	100.00%	57	2	88.02%	100.00%

The main challenge when considering all shipments is that the possibly unconfirmed shipments may be fulfilled instead of confirmed ones. As observed in Table 4.12, when considering all shipments at once, on average 88.02% of confirmed shipments are fulfilled while 100.00% of unconfirmed ones are fulfilled. On the other hand, using layer-based optimization, all of the confirmed shipments are fulfilled, with 36 and 21 shipments fulfilled in the first and second layers, respectively. When optimizing based on the confirmation status, we ensure that the possibly unconfirmed shipments won't be prioritized over confirmed ones, thus allowing more opportunities to fulfill the confirmed shipments. Furthermore, the layer-based optimization schedule is more robust because unconfirmed shipments are subject to cancellations and, therefore, more disruptions.

To sum up, optimizing production scheduling, inventory management, and vessel assignment simultaneously instead of separately significantly supports the company in handling the future more resiliently, allowing it to re-optimize quickly after perturbations, become more competitive, and leverage its bargaining power. The mathematical optimization solution initiated the establishment of new management rules in the company, enhancing consequently the operations planning.

4.8 CONCLUSION

This research represents an efficient approach to tackling a real and complex industrial problem while providing generic insights that can be adapted and scaled to similar large-scale problems. After presenting the PSIMVA model, the exploratory analysis highlights that the complexity of the problem comes from the weighted-sum objective function, the binary variables ruling the vessel assignment to quays, and the mono-source constraints. We design a criterion-based $\mathcal{ILNS}$ metaheuristic and implement a practical variant for which the criterion is time. The latter proves to be very effective in terms of improving the quality and speed of the problem-resolution process compared to both the default CPLEX and the $\mathcal{RA}$ that mimic real life. From an industrial perspective, reaching good (gap $\leq 5\%$) feasible solutions in reasonable solving time is achieved. Furthermore, the mathematical optimization solution significantly impacts the decision rules within *OCP Group* and ensures a 5% increase in the annual turnover. This success story led the group to becoming a finalist in the prestigious *Franz Edelman Award 2021* for outstanding application of management science and advanced analytics in practice worldwide. It also demonstrates the efficacy of the designed solution methodology and its potential utilization by other companies and organizations. In the future, we plan on improving the PSIMVA using more sophisticated mathematical optimization techniques, including Benders decomposition.

Acknowledgements

This work was generously supported by the *OCP Group*, the Fonds de Recherche du Québec - Nature et Technologies (FRQNT) through a Ph.D. Excellence Scholarship, the Institute for Data Valorisation (IVADO) through another Ph.D. Excellence Scholarship, and the Group for Research in Decision Analysis (GERAD) through the provision of computing facilities for conducting experiments. All authors gratefully acknowledge the valuable comments received from the editors and two anonymous referees on an earlier version of this article.

CHAPTER 5 ARTICLE 3: THE PRIMAL BENDERS DECOMPOSITION

Authors: El Mehdi Er Raqabi, Issmaïl El Hallaoui, François Soumis

Submitted on October 30, 2023 to Operations Research

Abstract. In real-life contexts, organizations face very large-scale problems for which they usually have good primal solutions (e.g., expert/heuristic solutions) close to the optimal solution (in terms of solution support). The goal is to use these solutions (primal information) to reach satisfactory solutions quickly. Among several decompositions, the Benders decomposition has been significantly applied to tackle very large-scale problems with complicating variables, which, when temporarily fixed, yield problems easy to solve. Still, in its standard form, the Benders decomposition does not profit from the primal information and shows a zigzagging behavior, making convergence very slow, which is problematic in practice for large-scale problems. Driven by observations from the practice, we propose the primal Benders decomposition (PBD) for sparse very large-scale problems, for which most complicating variables are equal to zero in the optimal solution. This method, a paradigm shift, uses the PBD master problem to select the complicating variables to insert in the PBD subproblem, which is a restriction of the original problem and provides the primal solution implemented in practice. We report promising computational results on deterministic and stochastic facility location instances. We also ran additional experiments on a real-life, very large-scale problem that motivated this research. While companies usually generate enough complicating variables to ensure the feasibility of their mathematical models, the PBD allows for identifying a nearly minimal set of complicating variables, which ensures feasibility and is enough to solve the model optimally and quickly.

Keywords. *Benders Decomposition*, *L-shaped Method*, *Dantzig-Wolfe Decomposition*, *Mixed-Integer Programming*, *Large-Scale Optimization*, *Exact Method*.

History. This article is submitted to *Operations Research*.

5.1 INTRODUCTION

When tackling very large-scale optimization problems, mixed integer linear programming (MILP) has been used intensively as the modeling tool [192]. With such usage, intense research has been conducted to tackle MILP problems efficiently [193]. Let us consider the MILP problem of the following generic form, referred to as the original problem (OP):

$$\begin{aligned} \mathbf{min}\ & f^T y + c^T x && \text{(OP)} \\ s.t.:\ & Ay \geq b && (5.1) \\ & Wy + Tx \geq d && (5.2) \\ & y \in \mathbb{Z}_+^n && (5.3) \\ & x \in \mathbb{R}_+^m && (5.4) \end{aligned}$$

where $f \in \mathbb{R}_+^n$, $c \in \mathbb{R}_+^m$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $W \in \mathbb{R}^{l \times n}$, $T \in \mathbb{R}^{l \times m}$, and $d \in \mathbb{R}^l$. Let us assume without loss of generality that the (OP) is feasible and bounded. Benders decomposition (BD) (Benders, 1962) is a well-known way to tackle the problem (OP) when fixing y implies an easy problem. We refer to y variables as the complicating variables. By projecting (OP) on the space defined by y variables [194], we obtain

$$\begin{aligned} \mathbf{min}\ & f^T y + \mathbf{min}\{c^T x \mid Tx \geq d - Wy,\ x \in \mathbb{R}_+^m\} && (\text{OP}_y) \\ s.t.:\ & Ay \geq b && (5.5) \\ & y \in \mathbb{Z}_+^n && (5.6) \end{aligned}$$

The inner minimization problem is the Benders primal subproblem (BD PSP). Its dual is the Benders dual subproblem (BD DSP):

$$\begin{aligned} \mathbf{max}\ & (d - Wy)^T \lambda && \text{(BD DSP)} \\ s.t.:\ & T^T \lambda \leq c && (5.7) \\ & \lambda \geq 0 && (5.8) \end{aligned}$$

The (BD DSP) is preferred to BD PSP because its polyhedron is independent of the complicating variables y. Solving (BD DSP) with $y = \bar{y}$ yields either an extreme point or an extreme ray. Let P and Q be extreme points and rays sets of the (BD DSP) polyhedron, respectively. The Benders master problem is as follows:

$$\mathbf{min}\ f^T y + z \qquad \text{(BD MP)}$$

$$
\begin{aligned}
s.t.: \ & Ay \geq b && (5.9)\\
& z \geq (d - Wy)^T \lambda^p, \ \ p \in P && (5.10)\\
& 0 \geq (d - Wy)^T \lambda^q, \ \ q \in Q && (5.11)\\
& y \in \mathbb{Z}_+^n && (5.12)
\end{aligned}
$$

Enumerating all extreme points and rays is computationally untractable. Thus, the Benders algorithm starts initially with a subset (or empty set) of extreme points and rays. The restricted BD MP problem (BD RMP) is solved, and its solution $\bar{y}$ is provided to (BD DSP). If the latter is feasible and bounded, an optimality cut (corresponding to solution λ^p with $p \in P$) is obtained. If it is unbounded, a feasibility cut (corresponding to solution λ^q with $q \in Q$) is obtained. These cuts are added to the RMP. Being a relaxation (fewer constraints), the BD RMP provides a lower bound (LB) on the optimal solution of (OP). Also, if feasible, the BD PSP generates a feasible solution to (OP), i.e., an upper bound (UB). The Benders algorithm continues until the difference between the UB and LB is smaller than a selected threshold $\epsilon \geq 0$. Compared to MILP which contains all complicating variables, BD does not contain any complicating variable as shown on the spectrum in Figure 5.1.

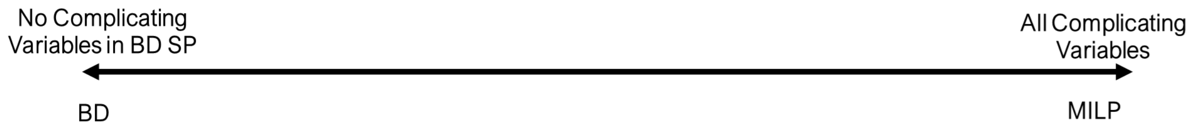

No Complicating Variables in BD SP
All Complicating Variables
BD
MILP

Figure 5.1 A spectrum with two extremes: BD and MILP

Given its practical relevance, BD has been applied in many fields, including production routing [81], electric vehicles [82], airline scheduling [83, 84], water resource management [85], on-demand delivery [86], hub location [87], locomotive assignment [88], traveling salesman [89], vessel service planning [90], capacity expansion [91], and budgeting [92].

Despite its successes, BD is time-consuming, has a zigzagging behavior, and converges slowly. These drawbacks may not be apparent if the problem is small or medium-sized but are very problematic for large-scale problems. The solution of the BD RMP has a good part (with useful primal information) and a bad one. When fixing this solution in the BD SP, the BD SP does not profit from the good part and thus BD shows a zigzagging behavior due to the bad part of the BD RMP solution, which is problematic in practice. Furthermore, the BD SP has a marginal role (this is paradoxical because BD SP is the provider of the primal solution we implement in practice). In contrast, the BD RMP has a primordial role (integrality is handled here). Still, the BD RMP is dual (good for assessing the quality of the primal solution at hand if any). Thus, there is an unbalanced computational load between BD SP and BD RMP.

Intense research has been conducted to accelerate BD convergence. These efforts can, intuitively, be classified into two sides. The first side deals with improving the LBs provided by the RMP, while the second seeks to improve the UBs obtained from the BD PSP. On the first side, intense research has been developed to select *good* or strengthen Benders cuts [93, 94, 95, 27, 96, 97] leading to better LBs. Other acceleration techniques include valid inequalities, warm-starting, managing the branch-and-bound (B&B) tree, and solving in two phases, *i.e.*, generating first cuts from relaxed (BD MP) and then cuts from integer (BD MP). On the second side, as far as we acknowledge, there is no systematic way to generate high-quality UBs. So far, problem-specific heuristics have been used to improve the UBs. An exhaustive literature review by Rahmaniani et al. [98] highlights the state of the art of BD application, challenges, and improvement strategies. Accelerating BD convergence requires a double effort to get the lower and upper bounds close to each other as *fast* as possible. Degeneracy and symmetry amplify this double effort, and the BD might get lost and not converge for *sparse* very large-scale problems. In our context, a sparse large-scale problem is a problem for which the vector of complicating variables is sparse, i.e., most complicating variables are equal to zero in the optimal solution of the (OP) (implying high degeneracy).

For large-scale problems, we often need to decompose the problem, improve a primal solution iteratively until satisfaction or time runs out, and accelerate the solving process starting from a good solution with good primal information. This needs a primal decomposition method, not a dual one like BD. Motivated by observations from the practice, we position the paper in the context of sparse very large-scale problems for which we have *good* solutions (obtained using heuristics or machine learning). For these problems, the *intuition* is that these good solutions are close to the optimal solution in terms of the solution support and not necessarily in terms of cost. For example, in the case of planning re-optimization, a small part of the current planning might no longer be feasible and will therefore be given a large cost, which means that BD, as a dual method, does not benefit much. Thus, using the primal information contained in these solutions, it is possible to reach the optimal solution of (OP) quickly without considering all the complicating variables in the mathematical model, especially given the potentially high degeneracy and symmetry when considering all these variables at once. Driven by this intuition, we first view BD from a different angle than the commonly established perception, which does not profit from the primal information (Benders, 1962). Then, leveraging BD as the dual of Dantzig-Wolfe decomposition (DWD) [195, 80], we propose the Primal Benders Decomposition ($\mathcal{PBD}$) for sparse very large-scale problems. In a nutshell, the $\mathcal{PBD}$ method consists of augmenting the $\mathcal{PBD}$ subproblem with complicating variables from the support of the $\mathcal{PBD}$ restricted master problem solution $\bar{y}$ instead of fixing it as in the standard BD. In such a case, the $\mathcal{PBD}$ subproblem becomes a restriction of the original problem (OP) with few complicating variables $\bar{y}$ (solving the $\mathcal{PBD}$ subproblem remains quick). Using the solutions of the $\mathcal{PBD}$ subproblem, we generate Benders cuts for the $\mathcal{PBD}$ restricted master problem and the process continues until convergence. The main contributions of this paper are the following:

1. Motivating, designing, and proving the convergence of the $\mathcal{PBD}$ for sparse very large-scale problems. The basic form of the $\mathcal{PBD}$ method is easily implementable.

2. Highlighting that the accelerated version of $\mathcal{PBD}$ reaches optimal or near-optimal solutions with decreasing steps and without zigzagging. Furthermore, the accelerated version requires only *interesting* optimality cuts, referred to as Pareto-optimal primal Benders cuts.

3. Testing the proposed method for deterministic and stochastic facility location problems. For the latter, the number of Benders cuts has been reduced drastically on instances of up to 2000 facilities. We conducted additional experiments on a real-world, very large-scale problem that motivated this research. On this problem, we observe that, while companies usually add enough complicating variables into mathematical models to ensure optimality and sometimes feasibility, the proposed $\mathcal{PBD}$ allows for selecting a nearly minimal set of complicating variables, which are enough to solve the model optimally and quickly.

The remainder of this article is as follows. In Section 5.2, we present the $\mathcal{PBD}$ method, and in Section 5.3, we design an accelerated version of it. Section 5.4 provides a facility location example to illustrate the method. The computational results are provided in Sections 5.5 and 5.6, respectively. We conclude in Section 5.7.

5.2 THE PRIMAL BENDERS DECOMPOSITION

In this section, we present the motivation behind the $\mathcal{PBD}$ method development. Then, we highlight the $\mathcal{PBD}$ framework and its convergence.

5.2.1 Motivation

The solutions to several optimization problems, including scheduling [196], facility location [197, 198], and vehicle routing [199] are highly degenerate and symmetrical. Degeneracy and symmetry become more significant in real life, occur often, cause computational difficulties in reaching the optimal solution, and lead to sparse very large-scale problems. Formally, we define, in this context, a sparse very large-scale problem as follows.

Definition 2. *Let y^* be the optimal solution of (OP). We say that (OP) is sparse if $\frac{|supp(y^*)|}{n} \approx 0$, where $supp(y^*)$ is the support of the complicating variables in the optimal solution.*

In the literature, the main trend for tackling the (OP) problem (including the sparse case) is the dual view of BD (visualized in Figure 5.2), where the BD RMP provides part of the solution (primal information) to the BD SP (i.e., BD PSP). The latter completes the primal information and uses its dual solution to generate Benders cuts (rows) for the BD RMP. Such a view has driven research in BD since early developments of the method [75, 98].

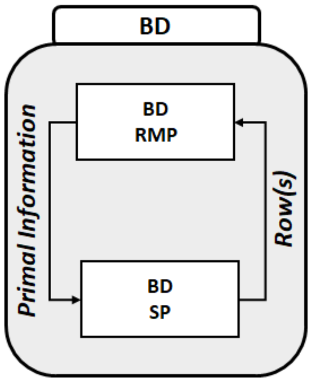


Figure 5.2 Standard View of Benders Decomposition

Another interesting view consists of perceiving the BD as the dual of the DWD. From such a perspective, the BD RMP corresponds to the DWD SP, while the BD SP corresponds to the DWD RMP. In the DWD, the DWD RMP is used to provide the DWD SP with the necessary dual information (dual solutions) from which the DWD SP generates improving (if they exist) columns for the DWD RMP. While the latter accumulates the columns (the UB decreases because the domain grows), the BD SP uses only the last column generated, leading to the UB zigzagging behavior. Also, while the BD RMP accumulates the dual information (the LB increases because the domain is reduced), the DWD SP uses only the dual information of the last DWD RMP solution, leading to the LB zigzagging behavior [200]. The LB of BD RMP is dual and resembles, in this sense, the lower bound obtained with the reduced cost in the SP of DWD. Conversely, the BD SP and DWD RMP give an UB (primal). In practice, we prefer more primal solutions since they inform the decisions to be implemented. Thus, the problem (BD SP in our context) that provides primal solutions should receive more attention, in our opinion. We highlight this view in Figure 5.3.

The presented insight allows the design of the $\mathcal{PBD}$. In the latter, the BD RMP inserts

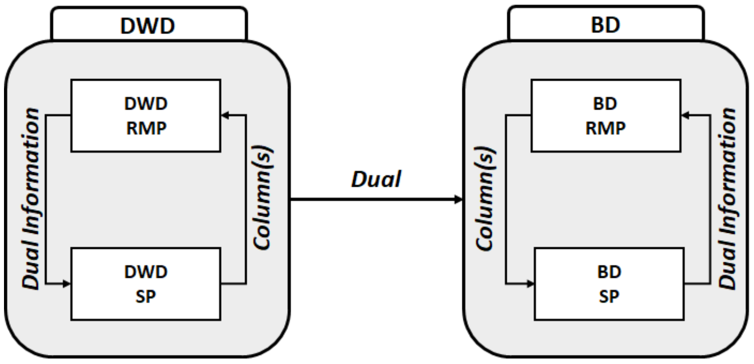


Figure 5.3 BD as the dual of DWD

improving column(s) into the BD PSP. Then, the BD PSP generates Benders cut(s) for the BD RMP. In such a way, the $\mathcal{PBD}$ accumulates information in both the BD RMP and BD PSP and improves monotonically the lower and upper bounds. This insight aligns with the intuition behind solving sparse very large-scale problems, i.e., identifying and inserting as few as possible complicating variables into the BD PSP to reach the optimal solution. Using $\mathcal{PBD}$, we seek to profit from the good part of the BD RMP solution, avoid the BD zigzagging, and ensure a load balance between BD SP and BD RMP. Based on the motivation, we present next the $\mathcal{PBD}$ framework.

5.2.2 The PBD Framework

Given a pool of complicating variables y, we generate an initial solution y^{init}. In practice, we may already have a good initial solution either from the company's history of solutions using machine learning or in case of re-optimization after perturbation [4]. The support of the initial solution, referred to as $supp(y^{init})$, is added to a set $\mathcal{S}$ (initially empty), and the corresponding complicating variables (y_j with $j \in \mathcal{S}$) are inserted into the reduced primal subproblem (RPSP$_\mathcal{S}$). Since we do not consider all the y variables, we use the qualification *reduced*.

Formally, let $y_\mathcal{S} \in \mathbb{Z}^n_+$ be the vector of complicating variables and let $y_j, j \in J = \{1, 2, ..., n\}$ be an element of it such that $y_j = 0$ if $j \notin \mathcal{S}$. We formulate the RPSP$_\mathcal{S}$ as follows:

$$\mathbf{min} \;\; f^T y_\mathcal{S} + c^T x \qquad (\text{RPSP}_\mathcal{S})$$

$$s.t.: \;\; Ay_\mathcal{S} \geq b \qquad (5.13)$$

$$Wy_\mathcal{S} + Tx \geq d \;\; [\lambda] \qquad (5.14)$$

$$y_{\mathcal{S}} \in \mathbb{Z}^n_+ \tag{5.15}$$

$$x \geq 0 \tag{5.16}$$

By fixing $y_{\mathcal{S}}$ to $\bar{y}_{\mathcal{S}}$, a feasible solution of (RPSP$_{\mathcal{S}}$) and by denoting λ the vector of dual variables corresponding to the second set of constraints, we obtain the Reduced Dual Subproblem (RDSP$_{\mathcal{S}}$):

$$\mathbf{max}\ (d - W\bar{y}_{\mathcal{S}})^T\lambda \tag{RDSP$_{\mathcal{S}}$}$$

$$s.t.:\ T^T\lambda \leq c \tag{5.17}$$

$$\lambda \geq 0 \tag{5.18}$$

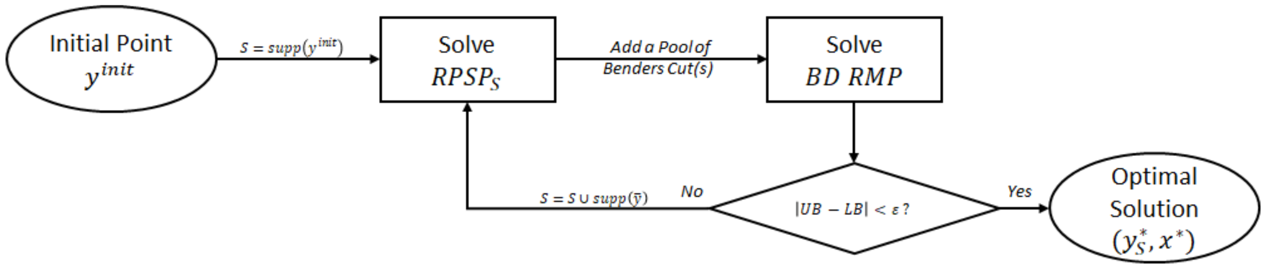

Initial Point y^{init}
$S = supp(y^{init})$
Solve $RPSP_S$
Add a Pool of Benders Cut(s)
Solve $BD\ RMP$
$|UB - LB| < \varepsilon$?
No
$S = S \cup supp(\bar{y})$
Yes
Optimal Solution (y^*_S, x^*)

Figure 5.4 The PBD Framework

The $\mathcal{PBD}$ framework is highlighted in Figure 5.4. Using y^{init} as a warm-start, we first solve RPSP$_{\mathcal{S}}$ to optimality. From the explored leaf nodes of the Branch & Bound (B&B), we obtain a pool of Benders cuts. We add these cuts to the BD RMP, which is solved to optimality. Let $\bar{y}$ be its solution. The BD RMP provides a LB on the optimal value of (OP). Also, the RPSP$_{\mathcal{S}}$ provides a feasible solution to the (OP), i.e., an UB on the optimal value of (OP). If $|UB - LB| \geq \epsilon$, the new complicating variables in $supp(\bar{y})$ (i.e., $supp(\bar{y}) \nsubseteq \mathcal{S}$ with $\bar{y}$ being the BD RMP's solution) are inserted into the RPSP$_{\mathcal{S}}$ ($S = S \cup supp(\bar{y})$). The algorithm continues until the difference between the UB and LB is smaller than a selected threshold $\epsilon \geq 0$. In such a case, we return the optimal solution $(y^*_{\mathcal{S}}, x^*)$ of RPSP$_{\mathcal{S}}$, which is the optimal solution of (OP). From one iteration to another, the UB decreases because the domain grows, and the LB increases because the domain shrinks. The solving of the RPSP$_{\mathcal{S}}$ should be *fast* if we warm-start using its previous solution and $\bar{y}$.

The complicating variables added to the RPSP$_{\mathcal{S}}$ are not fixed, as in BD SP. Thus, the RPSP$_{\mathcal{S}}$ is a restriction of (OP), and this is why $\mathcal{PBD}$ is primal because we improve the current integer solution at each iteration. On the spectrum of Figure 5.1, the $\mathcal{PBD}$ comes between BD and MILP as highlighted in Figure 5.5.

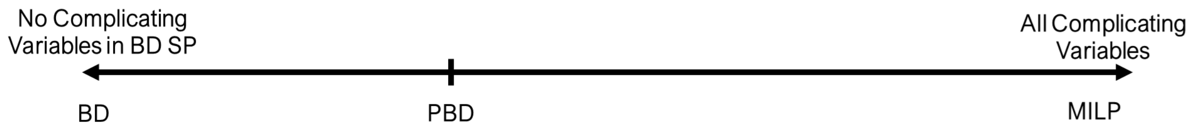

No Complicating Variables in BD SP
All Complicating Variables
BD
PBD
MILP

Figure 5.5 $\mathcal{PBD}$ as a compromise between BD and MILP

5.2.3 The PBD Convergence

This section discusses the $\mathcal{PBD}$ method convergence. We start with an observation related to the cuts obtained by the standard BD.

Observation 1. *Let $y \in \mathbb{N}^n$ be a vector of complicating integer variables. From this vector, there are $\aleph_0$ potential solutions $\bar{y}$ to explore during the standard BD iterations.*

As per Observation 1, the huge combinatorial space of $\bar{y}$ solutions is the main insight behind BD's slow convergence and zigzagging behavior, especially in very large-scale contexts. This is similar to the DWD case where *oscillations* are observed [201]. In the BD case, the BD RMP solution space is huge and it is possible to move from a good BD RMP solution to a much worse one. This affects the quality of the UB obtained by the RPSP$_{\mathcal{S}}$ in the following iteration. Thus, we may wonder whether all the BD RMP solutions are relevant to reach the optimal solution. If this is not the case, then we may seek to identify just the relevant solutions. One way to confirm is finding the *best* $\bar{y}$ as sketched in the example below.

Example 5. *Given a vector $y \in \mathbb{B}^2$, instead of providing $\bar{y} = (0,0)$, $\bar{y} = (1,0)$, $\bar{y} = (0,1)$, and $\bar{y} = (1,1)$ to the Benders subproblem as in BD SP, we provide the variables $y = (y_1, y_2)$ (i.e., $\mathcal{S} = \{1,2\}$). Then, solving the RPSP$_{\mathcal{S}}$ via B&B will provide the optimal (the best) $\bar{y}$.*

The Observation 1 and the Example 5 above align with the $\mathcal{PBD}$ motivation and design. Viewing the RPSP$_{\mathcal{S}}$ from a B&B perspective leads us to generate the following result.

Proposition 2. *On the B&B tree of RPSP$_{\mathcal{S}}$, each integer feasible node allows generating a Benders' optimality cut, while each infeasible node allows generating a Benders' feasibility cut. Furthermore, this cut can be obtained using the node's dual solution.*

Proof. Within the B&B tree, we distinguish feasible and infeasible nodes. Within the feasible nodes, we consider integer nodes. Assuming that no cuts are added and that the branching is standard on each variable alone (not on a subset of variables), the (RPSP$_{\mathcal{S}}^{Node}$) at an integer node is written as:

$$\begin{aligned}
\mathbf{min}\ & f^T y_{\mathcal{S}} + c^T x && (\text{RPSP}_{\mathcal{S}}^{Node})\\
s.t.:\ & A y_{\mathcal{S}} \geq b && (5.19)\\
& W y_{\mathcal{S}} + T x \geq d\ [\lambda] && (5.20)\\
& y_{\mathcal{S}} = \bar{y}_{\mathcal{S}} && (5.21)\\
& x \geq 0 && (5.22)
\end{aligned}$$

Constraints $y_{\mathcal{S}} = \bar{y}_{\mathcal{S}}$ follow from the branching constraints. By fixing $y_{\mathcal{S}} = \bar{y}_{\mathcal{S}}$ in (RPSP$_{\mathcal{S}}^{Node}$) and removing constraints $y_{\mathcal{S}} = \bar{y}_{\mathcal{S}}$, we obtain BD PSP. Thus, we generate the same Benders optimality cut using the integer node's dual solution. In a similar way, we prove that we generate a Benders feasibility cut using an infeasible node's dual solution. □

We refer to the Benders cuts obtained from the B&B leaf nodes' dual solutions as the primal Benders cuts. From the B&B perspective, an interesting observation follows.

Observation 2. *In the B&B process, some of the nodes will be pruned, thus eliminating several irrelevant solutions $\bar{y}_{\mathcal{S}}$ in the corresponding leaves, and consequently their corresponding Benders cuts.*

The pruned nodes are *dominated* by the unpruned ones. It implies that the *pruned* Benders cuts (corresponding to pruned nodes) are *dominated* by the *unpruned* Benders cuts (corresponding to unpruned nodes). Thus, pruning allows reducing the number of Benders cuts obtained throughout the $\mathcal{PBD}$ framework. It is not the case for BD, which may generate all Benders cuts, thus significantly increasing the number of iterations and the size of the BD RMP. In the next lemma, we show that the primal Benders cuts are valid for the BD RMP.

Lemma 1. *The primal Benders cut corresponding to an integer feasible or an infeasible node of RPSP$_{\mathcal{S}}$ is valid for the BD RMP.*

The proof of Lemma 1 is straightforward (see Proposition 2). Lemma 1 implies that no cut lifting is needed. Another strength of the $\mathcal{PBD}$ is highlighted in the observation below.

Observation 3. *Once the RPSP$_{\mathcal{S}}$ feasibility and boundness are ensured from the initial iteration, it remains feasible and bounded (we enrich the subproblem with new variables) throughout iterations.*

The next proposition shows that when there are no more complicating variables to insert in the RPSP$_{\mathcal{S}}$, the UB and LB coincide.

Proposition 3. *Let ($\bar{y}$,$\bar{z}$) be the solution of BD RMP at iteration $t \in \mathbb{N}^*$. If supp($\bar{y}$) $\subseteq \mathcal{S}$, then $UB = LB$.*

Proof. Suppose that $UB > LB$. Since *supp*($\bar{y}$) $\subseteq \mathcal{S}$, we distinguish two cases: optimality or feasibility cut. Let us discuss the optimality cut case (the same reasoning applies to the feasibility cut case). If $\bar{y}$ is feasible for RPSP$_{\mathcal{S}}$, the optimality cut corresponding to $\bar{y}$ (valid for the BD RMP as per Lemma 1) at iteration $t-1$ with $\bar{\lambda}$ the dual solution of (BD DSP) is:

$$z \geq (d - Wy)^T \bar{\lambda}$$

With $z = \hat{z}$ being the BD RMP solution at iteration $t-1$, we have:

$$\hat{z} \geq (d - W\bar{y})^T \bar{\lambda}$$

Since $UB > LB$, the optimality cut above is such that at iteration t:

$$\bar{z} = (d - W\bar{y})^T \bar{\lambda} > \hat{z}$$

Contradiction. □

An interesting result follows from Proposition 3.

Corollary 1. *The $\mathcal{PBD}$ does not generate the same pool of Benders cut(s) twice.*

The proof is straightforward since generating the same pool of Benders cut(s) implies obtaining the same $\bar{y}$ for the RMP, i.e., *supp*($\bar{y}$) $\subseteq \mathcal{S}$. Next, we prove the $\mathcal{PBD}$ convergence.

Theorem 4. *The $\mathcal{PBD}$ method converges.*

Proof. As per Lemma 1, the Benders cuts computed in the B&B tree of RPSP$_{\mathcal{S}}$ are valid for the BD RMP. As per Corollary 1, the $\mathcal{PBD}$ does not generate the same pool of Benders cut(s) twice. Given that the number of Benders cuts is finite (because the number of extreme points and rays is finite), the $\mathcal{PBD}$ converges. □

Under its basic form, the $\mathcal{PBD}$ method may insert unpromising complicating variables. This implies solving potentially large MILP subproblems and master problems at each iteration, and consequently a large execution time. The latter increases further in the context of large-scale optimization. The larger the master problem and the subproblem(s), the larger the execution time. Thus, to make it efficient, we design an accelerated $\mathcal{PBD}$ version, which is presented next.

5.3 THE ACCELERATED PBD

In this section, taking into consideration the sparse very large-scale context, we present the acceleration strategies, the accelerated $\mathcal{PBD}$ algorithm, and its convergence. We highlight in this section that optimality cuts are sufficient to reach optimal or near-optimal (primal) solution(s).

5.3.1 Acceleration Strategies

In this section, we discuss some strategies to efficiently implement the $\mathcal{PBD}$ method. These acceleration strategies are classified into master problem and subproblem acceleration strategies.

Master Problem Acceleration Strategies

Solving both the integer master problem and the integer subproblem is very costly. To tackle such a burden, we shift integrality to the subproblem and tackle the master problem in its relaxed (integrality) form. For the master problem, we distinguish the following acceleration strategy: selection strategy.

Selection Strategy. To alleviate the BD RMP, it is not necessary to consider all y variables. We may, similarly to the RPSP$_{\mathcal{S}}$, insert the promising complicating variables gradually. In such a case, the reduced restricted master problem (RRMP$_{\mathcal{T}}$) can be written as:

$$\begin{aligned}
\mathbf{min}\ \ & f^T y_{\mathcal{T}} + z && && \text{(RRMP}_{\mathcal{T}}\text{)}\\
s.t.:\ \ & A y_{\mathcal{T}} \geq b && [\alpha] && (5.23)\\
& z \geq (d - W y_{\mathcal{T}})^T \lambda^p,\ \ p \in P && [\beta] && (5.24)\\
& y_{\mathcal{T}} \in \mathbb{Z}_+^n && && (5.25)
\end{aligned}$$

Similarly to RPSP$_{\mathcal{S}}$, $y_{\mathcal{T}} \in \mathbb{Z}^n_+$ is the vector of $y_j, j \in J = \{1, 2, ..., n\}$ such that $y_j = 0$ if $j \notin \mathcal{T}$. The motivation behind the consideration of extreme points without extreme rays follows from the feasibility of RPSP$_{\mathcal{S}}$ (Observation 3). When the (RRMP$_{\mathcal{T}}$) is relaxed (integrality), let α and β_p, $p \in P$ be the dual solutions vectors corresponding to its constraints. Also, let a_j be column j of A and w_j be column j of W where $j \in \{1, 2, ..., n\}$. The reduced cost formula corresponding to variable y_j with $j \in J \setminus \mathcal{T}$ is:

$$\bar{f}_j = f_j - a_j^T \alpha - \sum_{p \in P} COEF(j,p)\beta_p, \;\; j \in J \setminus \mathcal{T} \tag{5.26}$$

The goal is to find promising complicating variables, i.e., y_j, $j \in J \setminus \mathcal{T}$ such that $\bar{f}_j < 0$, and select a few to be added to the (RRMP$_{\mathcal{T}}$). For each variable y_j, $j \in J \setminus \mathcal{T}$ (not present in the (RRMP$_{\mathcal{T}}$) problem), the computation of reduced cost $\bar{f}_j$ requires the computation of this variable's coefficients $COEF(j,p)$, $p \in P$ in the already existing primal Benders cuts in the (RRMP$_{\mathcal{T}}$) problem. The next proposition shows that the computation of reduced costs is quick since λ^p, $p \in P$ are already computed.

Proposition 4. *Let $p \in P$. The coefficient of a potential variable $\phi \in J \setminus \mathcal{T}$ in the Benders cut corresponding to extreme point p is $COEF(\phi, p) = w_\phi^T \lambda^p$.*

Proof. Let $\mathcal{S}^+ = \mathcal{S} \cup \{\phi\}$, $\phi \in J \setminus \mathcal{T}$. In the (BD DSP) problem, we have $\bar{y}_{\mathcal{S}^+} = \bar{y}_{\mathcal{S}}$ because $\bar{y}_\phi = 0$. Thus, the (BD DSP)'s solution remains the same, i.e., λ^p. From $(d - Wy_{\mathcal{S}^+})^T \lambda^p$, we infer that $COEF(\phi, p) = w_\phi^T \lambda^p$. □

It is worth highlighting that the selection strategy is exact for the relaxed (RRMP$_{\mathcal{T}}$) because it is based on the optimal dual values of the current relaxed (RRMP$_{\mathcal{T}}$). Still, it remains heuristic for the integer (RRMP$_{\mathcal{T}}$) problem. Theoretically, there might exist some complicating variables that do not improve the relaxation but improve the current integer solution. Practically, initial experiments showed that this selection strategy remains efficient in the sparse very large-scale context since it supports identifying most of the complicating variables in *supp*(y^*). Another aspect to observe is that the subset of complicating variables in (RRMP$_{\mathcal{T}}$) contribute to only a subset of constraints, implying a subset of x variables. Some constraints become redundant. We can remove them by preprocessing. It is worth mentioning that, to differentiate between the LB obtained by the BD RMP and the LB obtained by the RRMP$_{\mathcal{T}}$, we refer to the latter as reduced LB (RLB). Since the RRMP$_{\mathcal{T}}$ contains fewer variables, its RLB is not necessarily a LB for (OP). This completes the acceleration strategies for the RMP. We can now present the acceleration strategies for the RPSP$_{\mathcal{S}}$.

Subproblem Acceleration Strategies

For the subproblem, the acceleration strategies are classified into local Pareto-optimal cuts, warm-start, and tuning.

Local Pareto-optimal Cuts. When solving RPSP$_{\mathcal{S}}$, several integer solutions $\bar{y}$ explored in the B&B tree can be collected and corresponding Benders optimality cuts can be obtained. Adding several cuts to the RRMP$_{\mathcal{T}}$ may increase execution time. Instead of inserting all the Benders cuts corresponding to the identified $\bar{y}_{\mathcal{S}}$ solutions, we seek the relevant ones. Inspired by the notion of Pareto-optimal cuts (Magnanti and Wong, 1981), we introduce the notion of *local* Pareto-optimal cuts.

Definition 3. *A cut*

$$z \geq (d - Wy_{\mathcal{S}})^T \bar{\lambda}$$

is dominated locally by

$$z \geq (d - Wy_{\mathcal{S}})^T \lambda^*$$

if

$$(d - Wy_{\mathcal{S}})^T \bar{\lambda} \leq (d - Wy_{\mathcal{S}})^T \lambda^* \;\; \forall y_{\mathcal{S}} \in \mathbb{Z}_+^n$$

and there exist a $\bar{y}_{\mathcal{S}} \in \mathbb{Z}_+^n$ *such that*

$$(d - W\bar{y}_{\mathcal{S}})^T \bar{\lambda} < (d - W\bar{y}_{\mathcal{S}})^T \lambda^*$$

A cut is locally Pareto-optimal if it is not locally dominated by any other cut.

Following the definition, we show that the Benders optimality cut(s) obtained from the optimal node(s) in the B&B of (RPSP$_{\mathcal{S}}$) are locally Pareto-optimal.

Proposition 5. *In the B&B Tree of (RPSP$_{\mathcal{S}}$), the Benders cut obtained from the optimal node is locally Pareto-optimal.*

Proof. Let $(x_{\mathcal{S}}^*, y_{\mathcal{S}}^*)$ be the optimal solution of (RPSP$_{\mathcal{S}}$). When fixing $y_{\mathcal{S}} = y_{\mathcal{S}}^*$ in (RPSP$_{\mathcal{S}}$) and moving to the dual, let λ^* be the dual optimal solution corresponding to $(x_{\mathcal{S}}^*, y_{\mathcal{S}}^*)$. Then, the Benders cut corresponding to the optimal node is the following:

$$z \geq (d - Wy_{\mathcal{S}})^T \lambda^*$$

Suppose that this cut is dominated by another cut corresponding to a non-optimal node. Then by Definition 3, there exists $\bar{\lambda}$ such that:

$$(d - Wy_{\mathcal{S}})^T \lambda^* \leq (d - Wy_{\mathcal{S}})^T \bar{\lambda} \;\; \forall y_{\mathcal{S}} \in \mathbb{Z}_+^n$$

For $y_{\mathcal{S}} = y_{\mathcal{S}}^*$, we have:

$$(d - Wy_{\mathcal{S}}^*)^T \lambda^* \leq (d - Wy_{\mathcal{S}}^*)^T \bar{\lambda}$$

Given that λ^* is a dual optimal solution of (RDSP$_{\mathcal{S}}$), we also have:

$$(d - Wy_{\mathcal{S}}^*)^T \bar{\lambda} < (d - Wy_{\mathcal{S}}^*)^T \lambda^*$$

Contradiction. □

Based on Propositions 5, we insert only the primal Benders optimality cut(s) corresponding to the optimal node(s), which are locally Pareto-optimal (when compared to non-optimal nodes). These cuts capture all the necessary information to be provided to the RRMP$_{\mathcal{T}}$ at each iteration. In such a way, the master problem has fewer constraints, and the solving is faster. The next result follows.

Corollary 2. *If $y_{\mathcal{S}}^*$ augmented by zeros is optimal for the (OP), the corresponding Benders cut is Pareto-optimal.*

The proof is straightforward since the local cut is valid for the (OP) and is Pareto-optimal using the same proof as Proposition 5.

Warm-start. RPSP$_{\mathcal{S}}$ is a MILP, implying that solving it from scratch might be costly. Thus, warm-starting is an effective way to tackle it efficiently. To accelerate the (RPSP$_{\mathcal{S}}$) solving, we may warm-start it at each iteration. Let RPSP$_{\mathcal{S}+}$ be RPSP$_{\mathcal{S}}$ augmented with a new complicating variable provided by the selection strategy, we have the following observation.

Observation 4. *Let $(x_{\mathcal{S}}^*, y_{\mathcal{S}}^*)$ be the optimal solution of (RPSP$_{\mathcal{S}}$). When augmenting (RPSP$_{\mathcal{S}}$) with an improving complicating variable y_ϕ, a basic feasible solution for RPSP$_{\mathcal{S}+}$ is obtained when $y_{\mathcal{S}}^*$ is augmented with a zero corresponding to variable y_ϕ and $x_{\mathcal{S}}^*$ is augmented with zero(s) corresponding to the new x variable(s) added to RPSP$_{\mathcal{S}}$. Furthermore, the integrality gap is much smaller than if we consider all the variables y and x of the (OP).*

Warm-starting allows the $\mathcal{PBD}$ to benefit from the primal information to close the gap and reach optimality quickly. Furthermore, we recall that, at each iteration, we insert only the most promising variables identified using Formula 5.26. It allows keeping the RPSP$_{\mathcal{S}}$ as small as possible.

Tuning. To solve (RPSP$_{\mathcal{S}}$), one can use a commercial or open-source solver. When tackling MILP problems, each solver relies on a default configuration (e.g., branching strategy, number of cuts, feasibility versus optimality). For most solvers, the default setting was designed based on a small benchmark of instances from the literature, such as the ones from the MIPLIB library [25]. This default setting does not work well on real-life large-scale instances [1]. Thus, another acceleration strategy for the (RPSP$_{\mathcal{S}}$) is tuning the solver's parameters to speed up the solving process. Since the (RPSP$_{\mathcal{S}}$) is a restriction of the (OP), a configuration that works well for the (RPSP$_{\mathcal{S}}$) works well for the (OP) problem.

Similarly to the (RRMP$_{\mathcal{T}}$), since not all the complicating variables y are present in the RPSP$_{\mathcal{S}}$, some constraints and some x variables become redundant. We may reduce the model size by removing these variables and constraints. For instance, let us consider the constraint of the form $\sum_{i=1}^{m} x_{ij} \leq y_j \ \forall j \in J$, with all variables being positive. For a given $k \in J$, if $y_k = 0$ then $x_{ik} = 0 \ \forall i \in \{1, 2, ..., m\}$ and constraint $\sum_{i=1}^{m} x_{ik} \leq y_k$ is redundant. This completes the acceleration strategies for the RPSP$_{\mathcal{S}}$. We present next the Accelerated $\mathcal{PBD}$ algorithm.

5.3.2 The Accelerated PBD Algorithm

The Accelerated $\mathcal{PBD}$ algorithm is summarized in Algorithm 6. Let y^{init} be an initial point. We set $supp(y^{init})$ to sets $\mathcal{S}$ and $\mathcal{T}$, and the current solution y^{init} to $\bar{y}$. Then, we solve RPSP$_{\mathcal{S}}$ to optimality. We generate the local Pareto-optimal primal Benders cut(s) using the optimal node(s)' dual solution(s). We add these cuts to the RRMP$_{\mathcal{T}}$, which is relaxed (integrality) and solved. Let $\bar{y}$ be its new solution. If the solution changes ($supp(\bar{y})$ changes), we add $supp(y^{*})$ to $\mathcal{S}$ and insert the new complicating variables into RPSP$_{\mathcal{S}}$. We solve the latter, and the process continues. Otherwise (the solution does not change), using RRMP$_{\mathcal{T}}$'s dual solution, we compute the reduced cost for each y_j with $j \in J \setminus \mathcal{T}$. If we identify promising complicating variables, we select a few, add their indexes to set $\mathcal{T}$, and insert them into the (RRMP$_{\mathcal{T}}$) by lifting the existing Benders cuts. We solve the new (RRMP$_{\mathcal{T}}$). The process continues until no improving complicating variable is identified ($\Phi = \emptyset$). In such a case, we run Algorithm 7 to check convergence (detailed in Section 5.3.3). Algorithm 7 returns the optimal solution $(y^{*}_{\mathcal{S}}, x^{*}_{\mathcal{S}})$ of RPSP$_{\mathcal{S}}$, which is augmented with zeros to obtain an optimal or near-optimal solution of (OP). Since obtaining a cut from each node is impractical, we collect only Pareto-optimal cuts.

Algorithm 6: Accelerated PBD

```
1  Generate an initial point $y^{init}$
2  $\mathcal{S} \leftarrow supp(y^{init})$, $\mathcal{T} \leftarrow supp(y^{init})$, $\Phi \leftarrow \emptyset$, $\bar{y} \leftarrow y^{init}$
3  Solve RPSP$_\mathcal{S}$ to optimality, generate primal Benders optimality cut(s) at optimal
     B&B node(s), and insert these cuts into RRMP$_\mathcal{T}$
4  Solve relaxed RRMP$_\mathcal{T}$ to get new solution $\bar{y}$
5  if supp($\bar{y}$) changes then
6      $\mathcal{S} \leftarrow \mathcal{S} \cup supp(\bar{y})$
7      Return to Line 3
8  else
9      Using relaxed RRMP$_\mathcal{T}$'s dual solutions, compute $\bar{f}_j$ for each $y_j$ with $j \in J \setminus \mathcal{T}$ as
       per Formula 5.26
10     $\Phi \leftarrow \{j \in J \setminus \mathcal{T} : \bar{f}_j < 0\}$
11     if $\Phi \neq \emptyset$ then
12         Select few variables $y_j$, $j \in \Phi$ and add their indexes to set $\mathcal{T}$
13         Lift Benders cuts in RRMP$_\mathcal{T}$ using the coefficients $COEF(j, p)$, $p \in P$
14         Return to Line 4
15     end
16 end
17 Run Algorithm 7 to check convergence
```

5.3.3 The Accelerated PBD Convergence

A feasible solution of (OP) can be constructed by taking the (RPSP$_\mathcal{S}$) solution and augmenting it with zeros corresponding to the remaining variables (present in (OP) problem and not in (RPSP$_\mathcal{S}$) problem). The following theorem highlights that the Accelerated $\mathcal{PBD}$ provides a finite decreasing sequence to an optimal or near-optimal solution of (OP).

Theorem 5. *If we start with an initial point (x^1, y^1) and execute Algorithm 6, we generate a sequence $(x^1, y^1), (x^2, y^2), ..., (x^h, y^h)$ of solutions such that:*

1. $f^T y^1 + c^T x^1 \geq f^T y^2 + c^T x^2 \geq ... \geq f^T y^h + c^T x^h$;

2. $(x^1, y^1), (x^2, y^2), ..., (x^h, y^h)$ *are solutions of (OP);*

3. (x^h, y^h) *is an optimal or near-optimal solution of (OP).*

Proof. From any feasible solution of ($\text{RPSP}_{\mathcal{S}}$), we can obtain a feasible solution for (OP). Furthermore, when augmenting $\text{RPSP}_{\mathcal{S}}$ with a variable y_ϕ with $\phi \in J \setminus \mathcal{S}$, we have $f_S^T y_{\mathcal{S}}^* + c^T x_{\mathcal{S}}^* \geq f_{\mathcal{S}+}^T y_{\mathcal{S}+}^* + c_{\mathcal{S}+}^T x_{\mathcal{S}+}^*$. Combining the above, we form a decreasing sequence $f^T y^1 + c^T x^1 \geq f^T y^2 + c^T x^2 \geq ... \geq f^T y^h + c^T x^h$ where $(x^1, y^1), (x^2, y^2), ..., (x^h, y^h)$ are solutions of (OP) and where the last solution (x^h, y^h) is an optimal or near-optimal solution of (OP) because either there are no more complicating variables with a negative reduced cost to insert or all the complicating variables are inserted, i.e., ($\text{RPSP}_{\mathcal{S}}$) is equivalent to (OP). □

To confirm convergence to optimality and since the selection strategy for ($\text{RRMP}_{\mathcal{T}}$) is heuristic, we proceed as follows. Once no complicating variable(s) with a negative reduced cost are identified or there is no change in the optimal solution of ($\text{RRMP}_{\mathcal{T}}$), we insert all remaining complicating variables into $\text{RRMP}_{\mathcal{T}}$ using the previously computed lifting coefficients. Then, we solve the $\text{RRMP}_{\mathcal{T}}$ and compute its objective value, which becomes a LB for the (OP) problem ($\text{RRMP}_{\mathcal{T}}$ contains, at this stage, all complicating variables and becomes RMP). If $|UB - LB| < \epsilon$, we return the optimal solution $(y_{\mathcal{S}}^*, x^*)$ of $\text{RPSP}_{\mathcal{S}}$, which is augmented with zeros to obtain the optimal solution of (OP). Otherwise, we continue the insertion of complicating variables into $\text{RRMP}_{\mathcal{T}}$ as per the framework in Figure 5.4 until satisfying $|UB - LB| < \epsilon$. In such a case, we return the optimal solution $(y_{\mathcal{S}}^*, x_{\mathcal{S}}^*)$ of $\text{RPSP}_{\mathcal{S}}$, which is augmented with zeros to obtain the optimal solution of (OP). The convergence checking is highlighted in Algorithm 7. The augmented $\text{RRMP}_{\mathcal{T}}$ contains fewer cuts (Pareto-optimal primal Cuts). Thus, its solving should be quick. To accelerate $\text{RRMP}_{\mathcal{T}}$ solving further, we can rely on a warm start using the last solution obtained augmented with zeros corresponding to the complicating variables in $\bar{\mathcal{T}}$. To conclude this section, we make the following observation.

Algorithm 7: Convergence Checking

```
1 Insert all remaining complicating variables in $\bar{\mathcal{T}} = J \setminus \mathcal{T}$ into $\text{RRMP}_{\mathcal{T}}$ using the
    previously computed lifting coefficients.
2 if $|UB - LB| < \epsilon$ then
3     Return $(y_{\mathcal{S}}^*, x_{\mathcal{S}}^*)$ optimal solution of $\text{RPSP}_{\mathcal{S}}$
4 else
5     Continue via the framework in Figure 5.4 until convergence.
6 end
```

Observation 5. *The number of iterations might be large if supp(y^*) is large. Still, it is worth highlighting that, since the solution procedure (column generation) of DWD converges* *by adding only a small number of the columns to the DWD RMP, it might be the same (BD is the dual of DWD) behavior for $RPSP_{\mathcal{S}}$.*

With the convergence checking, we complete the discussion on the Accelerated $\mathcal{PBD}$ convergence. Next, we present an illustrative facility location example to visualize the method's benefits.

5.4 EXAMPLE

To highlight the benefits of the proposed method, consider the following facility location problem (FLP) example. A formulation for the capacitated facility location problem as given in Wentges (1996) is:

$$\mathbf{min} \quad \sum_{j=1}^{n} f_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{FLP}$$

$$s.t.: \quad \sum_{j=1}^{n} x_{ij} = d_i \qquad \forall i \in I \tag{5.27}$$

$$x_{ij} \le d_i y_j \qquad \forall i \in I, j \in J \tag{5.28}$$

$$\sum_{i=1}^{m} x_{ij} \le s_j y_j \qquad \forall j \in J \tag{5.29}$$

$$x_{ij} \ge 0, \quad y_j \in \{0,1\} \qquad \forall i \in I, j \in J \tag{5.30}$$

where $n = |J| = 5$ (facilities), $m = |I| = 6$ (clients), $f_j = 1\ \forall j \in J$ (fixed cost), $s_j = 16\ \forall j \in J$ (capacity), $d = \{1,1,2,3,4,5\}$ (demand), and unit cost matrix (variable cost)

$$c = \begin{pmatrix} 60 & 10 & 20 & 30 & 1 \\ 10 & 1 & 20 & 30 & 40 \\ 40 & 20 & 30 & 1 & 20 \\ 40 & 20 & 1 & 10 & 20 \\ 1 & 20 & 40 & 40 & 40 \\ 1 & 20 & 40 & 40 & 40 \end{pmatrix}$$

The binary variable y_j is equal to 1 if facility $j \in J$ is opened and zero otherwise. The variable x_{ij} is the quantity supplied to customer $i \in I$ by facility $j \in J$. For this example, the optimal value is 21, and all 5 depots are opened. We compare the BD and the $\mathcal{PBD}$ using two scenarios. In the first scenario, $\bar{y} = (0,0,0,0,1)$ is provided as an initial point to BD. Equivalently, $\mathcal{S} = \{5\}$ is an initial point for $\mathcal{PBD}$. Figure 5.6 shows the results. For each method, we report Obj_1 (in blue) corresponding to the subproblem objective value and Obj_2 (in red) corresponding to the master problem objective value.

While BD requires 8 iterations (8 Benders cuts) to converge, $\mathcal{PBD}$ requires only 5 iterations (5 primal Benders cuts) to reach the optimal solution, i.e., the remaining 5 depots to open. Also, the objective value of the BD subproblem (Obj_1 in Blue on the left) shows a zigzagging behavior while the objective value of the $\mathcal{PBD}$ subproblem (Obj_1 in Blue on the right) shows a strict decrease. The necessity to close the gap between the UB and LB implies more iterations for BD. In contrast, $\mathcal{PBD}$ reaches optimality when no more depots with negative reduced costs are available, i.e., in five iterations (Obj_1 and Obj_2 are overlapping in Figure 5.6b because they are close at each iteration).

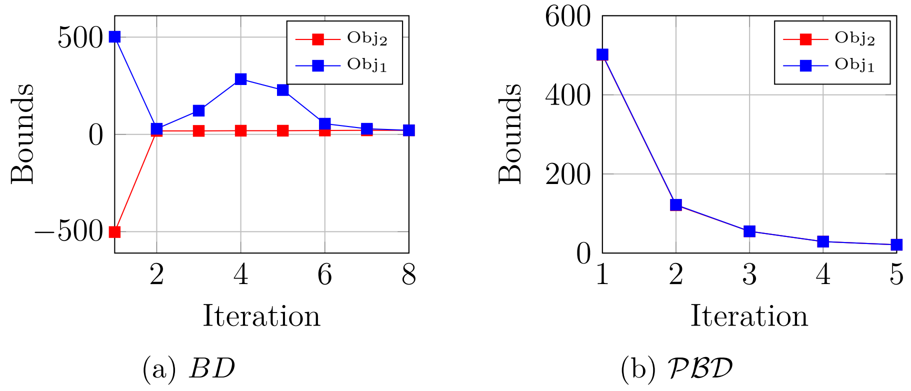


Figure 5.6 Comparison between BD vs PBD for $\bar{y} = (0, 0, 0, 0, 1)$ and $\mathcal{S} = \{5\}$

Another interesting observation is the comparison between the Benders cuts generated. In the first iteration, the Benders cut generated using BD is:

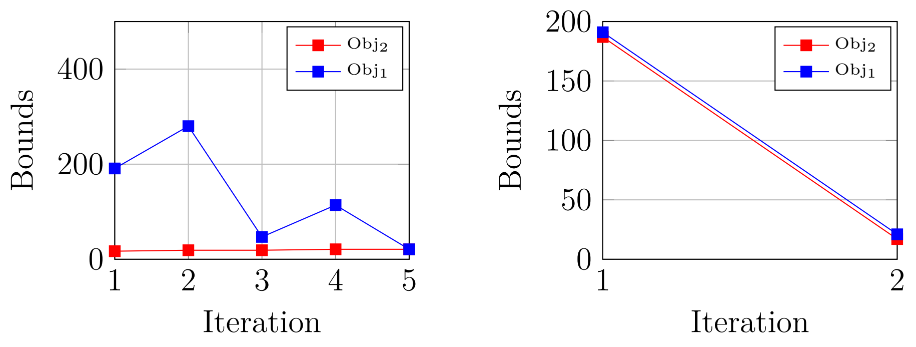
$$z \geq 501 - 381y_2 - 219y_3 - 77y_4 - 78y_5 \tag{5.31}$$

The Benders cut generated using $\mathcal{PBD}$ is:

$$z \geq 501 \tag{5.32}$$

Since y_1 is binary, the second cut dominates the first one as shown in Section 5.3.3.

In the second scenario, $\bar{y} = (0, 1, 1, 1, 1)$ is provided as an initial point to BD. Equivalently, $\mathcal{S} = \{2, 3, 4, 5\}$ is an initial point for $\mathcal{PBD}$. Figure 5.7 highlights the results for the second scenario. $\mathcal{PBD}$ reaches the optimal solution in two iterations while BD requires more. For $\mathcal{PBD}$, the first iteration corresponds to $\mathcal{S} = \{2, 3, 4, 5\}$ while the second iteration corresponds to $\mathcal{S} = \{1, 2, 3, 4, 5\}$. Two Pareto-optimal cuts are required to reach the optimal solution. They are sufficient and provide all the necessary information. It is not the case with BD, which requires 5 Benders cuts to converge while showing a zigzagging behavior.



(a) BD (b) $\mathcal{PBD}$

Figure 5.7 Comparison between BD vs PBD for $\bar{y} = (0, 1, 1, 1, 1)$ and $\mathcal{S} = \{2, 3, 4, 5\}$

For this second scenario, the Benders cut generated using BD in the first iteration is:

$$z \geq 187 - 171y_1 \tag{5.33}$$

The Benders cut generated using $\mathcal{PBD}$ is:

$$z \geq 187 \tag{5.34}$$

Since y variables are binary, the second cut dominates the first one. A last observation is that all the Benders cuts generated by $\mathcal{PBD}$ are tight in the (relaxed) solution of (RRMP$_{\mathcal{T}}$) and in each iteration. This is not the case for BD where a few cuts are tight in each iteration.

Beyond the zigzagging behavior, another important aspect to highlight is that BD does not profit from the primal information. Whether the initial point is close (in terms of the solution support) to the optimal solution (second scenario) or not (first scenario), BD shows the same behavior and requires several iterations to converge. On the other hand, $\mathcal{PBD}$ profits from the primal information and converges more quickly when the initial point is close to the optimal solution.

5.5 COMPUTATIONAL RESULTS ON THE FACILITY LOCATION PROBLEM

To confirm the $\mathcal{PBD}$ method is computationally efficient, we complement the theoretical analysis presented in previous sections with an extensive computational study on deterministic and stochastic facility location instances from the literature. We first present the experimental design. Then, we highlight the computational results. To conclude the section, we discuss the computational insights. In this section, to avoid selecting many complicating variables with a negative reduced cost (without being in the optimal solution) and significantly increasing the size of the (RPSP$_{\mathcal{S}}$), we only select the most promising complicating variable at each iteration, i.e., the variable y_ϕ such that $\phi = \min\limits_{j \in J \setminus S} \{\bar{f}_j : \bar{f}_j < 0\}$.

5.5.1 Experimental Design

In this section, we describe the general characteristics of the test instances, the computational setting, and implementation details.

Instances

We test our method on the FLP, often used as a classical example in the BD context. We consider three different sets of instances from the literature. Here, we provide a high-level summary of these problems and instances. Further details can be found in the provided references.

The first instance set is the deterministic FLP used in Beasley (1988) and available in the OR Library. These benchmarks are probably the most widely used benchmarks when testing algorithm performance for the FLP. We have considered all 52 instances from the 16 classes available. Each class includes at most four instances with varying costs and capacity ratios. Also, these instances include up to 1000 customers with up to 100 potential facilities. A capacitated formulation of the deterministic case is (FLP). We refer to these instances as *CAP* instances.

The second instance set is the stochastic variant of the facility location problem. For this variant, we have used the instances generated by Bodur et al. (2017) with up to 5000 scenarios. We have considered 80 instances from 4 classes and 5 scenarios. Each class includes four instances with varying costs and capacity ratios. Also, these instances include 50 customers with up to 25-50 potential facilities. A stochastic variant formulation is provided in Appendix C. We refer to these instances as *SCAP* instances. The main reason behind the choice of stochastic instances is that BD, also referred to as the L-shaped method within the stochastic optimization community, is significantly used to tackle stochastic optimization problems. Thus, we aim to check the performance of the $\mathcal{PBD}$ in the stochastic case as well.

The third instance set is a large-scale version of the facility location problem available in the Max Planck Institut Informatik. These benchmarks are designed to be similar to real-life problems and have a large number of near-optimal solutions. We have considered all 22 instances from 6 classes. Each class includes at most five instances with varying costs and capacity ratios. Also, these instances have a similar number of customers and facilities with 2000 being the largest size. We refer to these instances as M instances.

Computational Setting and Implementation Details

The coding language is C++, and we conduct tests using version 12.10.0 of IBM ILOG CPLEX solver. All experiments were carried out on a 3.20GHz Intel^{R} Core^{TM} i7-8700 processor, with 64GiB System memory, running on Oracle Linux Server release 7.7. We use real-time to measure runtime. We do not use specialized codes or algorithms to make the implementations simple and easily replicable. We solve all LPs and MILPs using IBM ILOG CPLEX 12.10.0 run on a single thread. The time limit for any execution is three hours. Since these facility location instances are from the literature, the default CPLEX configuration works well and we do not check the effect of the tuning strategy. We compare the $\mathcal{PBD}$ with the following three methods:

- **MILP:** Solve directly with a MIP solver. In our case, we use the default CPLEX.
- **CPLEX BD:** Solve with CPLEX implementation of BD. To do so, we set the CPLEX parameter *Benders Strategy* to option *Full*, i.e., CPLEX automatically decomposes the given model. We keep other CPLEX parameters to their default options.
- **BD:** Solve with the standard BD as it was initially developed [75] without any acceleration strategies. We use BD as a baseline to compare the number of Benders cuts.

There are several (meta)heuristics used to find an initial point for the FLP. While it is not our focus here, to find initial points, we implement the DROP heuristic that starts with all facilities open, keeps dropping (closing) the facility that gives the maximum decrease in the total cost, and stops if dropping any more facilities will no longer reduce the total cost [203, 204]. For a fair comparison, we provide all methods with the same initial point. The time limit for solving any instance by any of the four methods is one hour.

5.5.2 Computational Results

In this section, we quantify the computational benefits of the $\mathcal{PBD}$ when solving the instances considered. We first check the performance of the $\mathcal{PBD}$ on the CAP instances. Then, we evaluate its performance on the SCAP instances. After that, we highlight its performance on the M instances. We then quantify the impact of the acceleration strategies.

Deterministic Facility Location Problem

Table 5.1 presents the performance of the four methods on the CAP instances. We report the optimality gap (*Gap*) and the execution time (*Time*) in seconds. Since we do not know the optimal solution a priori, we approximate the optimality gap as $\frac{UB-LB}{LB}$ where LB is the best LB known, and UB is the best solution found. This formula overestimates the optimality gap formula $\frac{UB-OPT}{OPT}$, where OPT is the optimal value, and UB is the best solution found. For the execution time and when it is the case, we indicate the number of instances that could not be solved optimally between parentheses. We also report the number of Benders cuts (*Cuts*) for the three decompositions. Columns |I| and |J| refer to the number of clients and the number of facilities, respectively. Column |J*| refers to the number of opened facilities in the optimal solution, i.e., $supp(y^*)$. We report all the values as averages.

Table 5.1 Performance Comparison for Cap Instances

Cap #	**|I|**	**|J|**	**|J*|**	**MILP**		**CPLEX BD**			**BD**			**PBD**		
				Gap	**Time**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**
41-44	50	16	9	0.00	0.73	0.00	0.58	4	0.00	24.50	24	0.00	0.62	6
51	50	16	8	0.00	0.78	0.00	0.55	4	0.00	19.33	34	0.00	0.59	6
61-64	50	16	8	0.00	0.71	0.00	0.51	4	0.00	11.42	32	0.00	0.52	6
71-74	50	16	7	0.00	0.74	0.00	0.49	5	0.00	8.15	28	0.00	0.51	5
81-84	50	25	11	0.00	0.98	0.00	0.75	8	4.11	(4)	>5000	0.00	0.77	9
91-94	50	25	11	0.00	1.18	0.00	0.84	8	3.93	(4)	>5000	0.00	0.85	8
101-104	50	25	10	0.00	1.33	0.00	0.95	5	3.78	378.67 (1)	48	0.00	0.96	8
111-114	50	50	11	0.00	1.49	0.00	1.07	10	4.27	(4)	>5000	0.00	1.12	9
121-124	50	50	11	0.00	4.14	0.00	1.20	12	4.77	(4)	>5000	0.00	1.26	8
131-134	50	50	10	0.00	4.64	0.00	1.27	7	3.97	(4)	>5000	0.00	1.36	8
a	1000	100	4	0.00	232.00	0.00	2.00	7	8.21	(4)	>5000	0.00	1.71	2
a1-4	1000	100	6	0.00	72.50	0.00	5.50	15	10.15	(4)	>5000	0.00	3.57	5
b	1000	100	7	0.00	68.00	0.00	2.55	13	8.34	(4)	>5000	0.00	2.31	5
b1-4	1000	100	9	0.00	73.25	0.00	13.25	52	12.33	(4)	>5000	0.00	5.58	7
c	1000	100	9	0.00	104.00	0.00	3.15	15	8.67	(4)	>5000	0.00	2.48	6
c1-4	1000	100	10	0.00	101.25	0.00	10.50	43	12.07	(4)	>5000	0.00	5.86	9
Avg	**406**	**60**	**9**	**0.00**	**41.73**	**0.00**	**2.82**	**13**	**5.29**	**2502.63**	**>5000**	**0.00**	**1.88**	**7**

From Table 5.1, we can infer that CPLEX BD and $\mathcal{PBD}$ outperform both MILP and BD. The latter fails to reach optimality on several CAP instances with the average gap being 5.29%. In many of them, BD generates more than 5000 Benders cuts without converging, highlighting the convergence issues of BD. Also, while CPLEX BD performs better compared to $\mathcal{PBD}$ on instances with 25-50 facilities and 50 customers, $\mathcal{PBD}$ outperforms other methods on larger instances with 100 facilities and 1000 customers. On average, $\mathcal{PBD}$ outperforms all other methods with an average time to optimality equal to 1.88 and an average number of Benders cuts equal to 7.

Stochastic Facility Location Problem

Table 5.2 presents the results obtained on SCAP instances. We consider from 250 to 5000 scenarios. The SCAP instances are more difficult than the CAP instances. Indeed, BD fails to reach optimality on all of them and MILP starts to fail from scenario 500. For CPLEX BD and $\mathcal{PBD}$, they both reach optimality in all instances. CPLEX BD requires many Benders cuts to reach optimality with the lowest number being 549 cuts and the highest number being 29037 cuts. On the other side, $\mathcal{PBD}$ requires far fewer cuts and converges in at most |J*| iterations. For execution time, $\mathcal{PBD}$ significantly outperforms more CPLEX BD on SCAP instances than CAP instances.

Table 5.2 Performance Comparison for SCAP Instances

K	Cap#	|I|	|J|	|J*|	MILP		CPLEX BD			BD			PBD		
					Gap	Time	Gap	Time	Cuts	Gap	Time	Cuts	Gap	Time	Cuts
250	101-104	50	25	9	0.00	18.00	0.00	4.00	601	17.91	(4)	>5000	0.00	4.21	6
	111-114	50	50	16	0.00	364.75	0.00	13.00	624	9.62	(4)	>5000	0.00	12.35	13
	121-124	50	50	10	0.00	557.00	0.00	12.75	894	19.33	(4)	>5000	0.00	12.23	7
	131-134	50	50	9	0.00	168.75	0.00	7.00	549	23.33	(4)	>5000	0.00	4.81	7
	Avg	**50**	**44**	**11**	**0.00**	**341.91**	**0.00**	**9.19**	**667**	**17.55**	**-**	**>5000**	**0.00**	**8.40**	**8**
500	101-104	50	25	9	0.00	64.75	0.00	7.00	1218	19.23	(4)	>5000	0.00	8.82	6
	111-114	50	50	16	0.00	1620.50	0.00	28.25	1736	9.79	(4)	>5000	0.00	27.84	13
	121-124	50	50	10	0.24	1988.25 (1)	0.00	27.25	1956	19.63	(4)	>5000	0.00	26.15	7
	131-134	50	50	9	0.00	760.50	0.00	14.00	1080	23.61	(4)	>5000	0.00	9.86	7
	Avg	**50**	**44**	**11**	**0.08**	**1108.50**	**0.00**	**19.13**	**1497**	**18.07**	**-**	**>5000**	**0.00**	**18.17**	**8**
1500	101-104	50	25	9	0.00	674.25	0.00	27.75	3575	21.01	(4)	>5000	0.00	29.21	6
	111-114	50	50	16	7.69	(4)	0.00	186.00	11044	10.06	(4)	>5000	0.00	138.44	13
	121-124	50	50	10	11.60	(4)	0.00	93.00	6838	20.99	(4)	>5000	0.00	89.87	7
	131-134	50	50	9	3.60	3259.25 (3)	0.00	52.25	3891	26.54	(4)	>5000	0.00	35.63	7
	Avg	**50**	**44**	**11**	**5.72**	**2868.56**	**0.00**	**89.75**	**6337**	**19.65**	**-**	**>5000**	**0.00**	**73.29**	**8**
3000	101-104	50	25	9	4.31	2407.75 (2)	0.00	62.25	5585	23.11	(4)	>5000	0.00	54.05	6
	111-114	50	50	16	13.63	(4)	0.00	309.00	29037	11.07	(4)	>5000	0.00	256.12	13
	121-124	50	50	10	20.58	(4)	0.00	208.00	10913	23.09	(4)	>5000	0.00	166.25	7
	131-134	50	50	9	9.25	(4)	0.00	126.50	7099	29.19	(4)	>5000	0.00	65.91	7
	Avg	**50**	**44**	**11**	**11.94**	**3301.94**	**0.00**	**176.44**	**13158**	**21.62**	**-**	**>5000**	**0.00**	**135.58**	**8**
5000	101-104	50	25	9	7.41	3106.25 (3)	0.00	112.50	11273	25.42	(4)	>5000	0.00	98.83	6
	111-114	50	50	17	16.73	(4)	0.00	528.25	22120	12.17	(4)	>5000	0.00	439.20	13
	121-124	50	50	10	27.13	(4)	0.00	380.75	16213	25.40	(4)	>5000	0.00	305.46	7
	131-134	50	50	9	11.12	(4)	0.00	216.75	11645	32.11	(4)	>5000	0.00	111.75	7
	Avg	**50**	**44**	**11**	**15.60**	**3476.50**	**0.00**	**309.56**	**15313**	**23.78**	**-**	**>5000**	**0.00**	**238.81**	**8**

Large-Scale Facility Location Problem

For the M instances, which mimic real-life cases, we report the results in Table 5.3. These instances are more complicated than CAP and SCAP instances since they have more complicating variables. BD fails in these instances. MILP fails in instances with more than 500 customers and facilities. CPLEX BD also fails on instances with a size larger than 500 except for instance R2. $\mathcal{PBD}$ outperforms all methods and reaches optimality in all instances.

Table 5.3 Performance Comparison for M Instances

M#	|I|=|J|	|J*|	MILP		CPLEX BD			BD			PBD		
			Gap	Time	Gap	Time	Cuts	Gap	Time	Cuts	Gap	Time	Cuts
O$_1$-O$_5$	100	4	0.00	35.20	0.00	2.33	213	5.33	(5)	>5000	0.00	0.21	4
P$_1$-P$_5$	200	5	0.00	139.00	0.00	43.80	1029	6.26	(5)	>5000	0.00	1.11	4
Q$_1$-Q$_5$	300	5	0.00	729.60	0.00	251.20	1766	8.66	(5)	>5000	0.00	3.82	5
R$_1$-R$_5$	500	6	13.01	(5)	2.64	3229.80 (4)	4075	12.12	(5)	>5000	0.00	9.53	6
S$_1$	1000	6	42.11	(1)	2.92	(1)	2562	19.85	(1)	>5000	0.00	24.78	6
T$_1$	2000	6	53.78	(1)	1.96	(1)	627	25.44	(1)	>5000	0.00	53.18	6
Avg	**683**	**5**	**18.15**	**1950.63**	**1.25**	**1787.86**	**1712**	**12.94**	**-**	**>5000**	**0.00**	**15.44**	**5**

For the M instances, |J*| is on average 5. Thus, $\mathcal{PBD}$ reaches the optimal solution in a few iterations despite the large size of facilities. This aligns with Observation 5 since the number of cuts is much less than the number of y variables and is rather of the order of the number of y variables in the optimal solution. On average, within 15.44 seconds, 5 Benders cuts are added to reach the optimal solution. In the remaining sections, we investigate further the methods and consider only the M instances with a large number of complicating variables (≥ 100).

Impact of Acceleration Strategies

In this section, we evaluate the impact of acceleration strategies on the $\mathcal{PBD}$. The abbreviations IP, WS, Cuts, and SS refer to the initial point, warm-start, Benders cuts, and selection strategy, respectively. In the first approach, we evaluate $\mathcal{PBD}$ when no initial point is provided. We recall that no initial point means the usage of an artificial solution with *high* costs as in DWD. In the second approach, we evaluate $\mathcal{PBD}$ when no warm start is applied to both RRMP and RPSP. In the third approach, we evaluate $\mathcal{PBD}$ when several integer optimality cuts are collected from the B&B tree of RPSP in a given iteration. We recall that we consider solely the local Pareto-optimal cut(s), obtained from the optimal node(s), in $\mathcal{PBD}$. Lastly, in the fourth approach, we compare $\mathcal{PBD}$ to $\mathcal{PBD}$ without the selection strategy. In the latter, we solve the RRMP to optimality and insert the support of its solution in the RPSP as described in Section 5.2. Table 5.4 highlights the effect of the acceleration strategies on the $\mathcal{PBD}$.

Table 5.4 Impact of Acceleration Strategies

M#	|I|=|J|	|J*|	No IP			No WS			All Cuts			No SS			PBD		
			Gap	Time	Cuts	Gap	Time	Cuts	Gap	Time	Cuts	Gap	Time	Cuts	Gap	Time	Cuts
O_1-O_5	100	4	0.00	0.53	5	0.00	0.50	4	0.00	0.61	108	0.00	1.12	4	0.00	0.21	4
P_1-P_5	200	5	0.00	2.24	6	0.00	2.20	4	0.00	2.65	168	0.00	7.00	4	0.00	1.11	4
Q_1-Q_5	300	5	0.00	7.14	6	0.00	7.12	5	0.00	8.88	245	0.00	22.80	5	0.00	3.82	5
R_1-R_5	500	6	0.00	18.77	7	0.00	18.40	6	0.00	22.07	571	0.00	110.80	6	0.00	9.53	6
S_1	1000	6	0.00	28.15	7	0.00	48.30	6	0.00	57.96	1266	0.00	883.00	6	0.00	24.78	6
T_1	2000	6	0.00	184.32	7	0.00	316.25	6	0.00	379.49	1872	5.50	(1)	6	0.00	53.18	6
Avg	**683**	**5**	**0.00**	**40.19**	**6**	**0.00**	**65.46**	**5**	**0.00**	**78.61**	**705**	**0.92**	**770.79**	**5**	**0.00**	**15.44**	**5**

On M instances, the initial point contributes to reducing the execution time with a factor of 2.60. Since |J*| is quite small, the $\mathcal{PBD}$ reaches the optimal solution in at most 6 iterations. Warm-starting significantly impacts the $\mathcal{PBD}$ performance, especially for large instances. For the T1 instance, while $\mathcal{PBD}$ reaches optimality in 53.18 seconds, $\mathcal{PBD}$ without warm-start requires 316.25 seconds. This is due to the large size of the subproblem and the time taken to solve it at each iteration from scratch. The same observation happens when all Benders cuts are inserted. The selection strategy also plays a crucial role, especially when J is large.

Indeed, when solving the BD RMP to optimality and inserting the support of its solution in the RPSP$_{\mathcal{S}}$, several complicating variables that do not belong to the optimal solution might be inserted in the RPSP$_{\mathcal{S}}$. The latter becomes quite large and consequently computationally expensive. This is the case for instance T1, which can no longer be solved within one hour when the selection strategy is removed.

5.5.3 Computational Insights

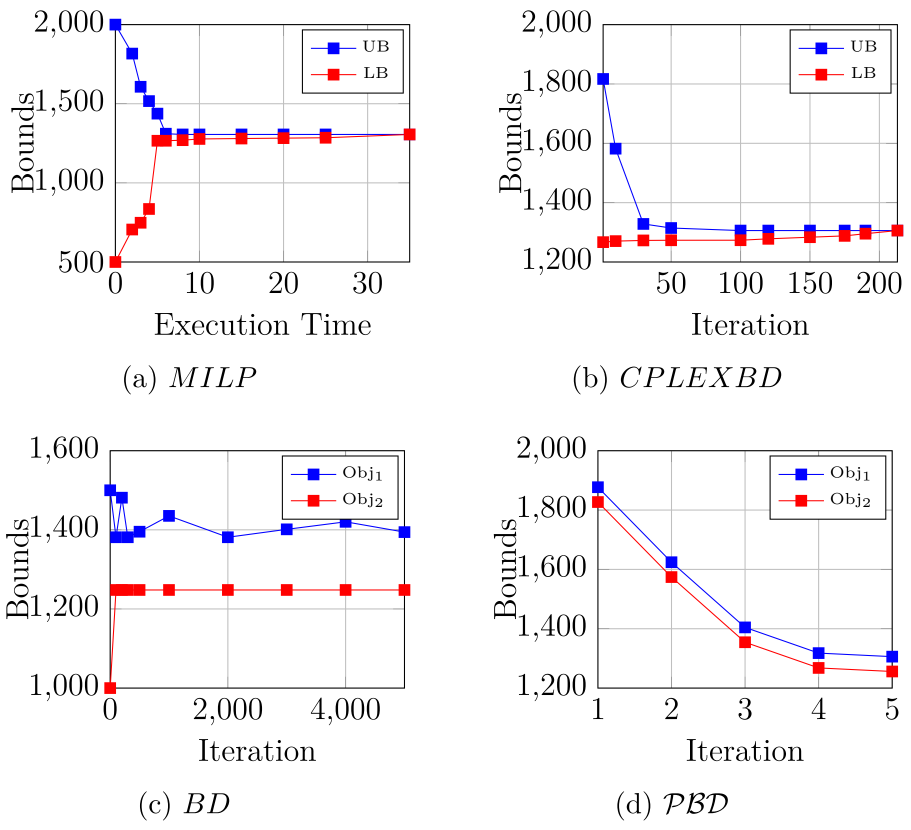
The results above report the optimality gap using the UBs obtained by each method. We check the RLBs as well and report in Table 5.5 the gap (Gap^{-}) between the best UB found and the best LB obtained. It is computed as $\frac{UB-LB}{UB}$. For $\mathcal{PBD}$, we use the RLB returned by the (RRMP$_{\mathcal{T}}$) within the time limit of one hour. The number of cuts and the execution time remain the same as in Table 5.3. In what follows and for the $\mathcal{PBD}$, we report the results before running Algorithm 7.

Table 5.5 Performance on the LBs

M#	|J*|	MILP		CPLEX BD			BD			PBD		
		Gap^{-}	Time	Gap^{-}	Time	Cuts	Gap^{-}	Time	Cuts	Gap^{-}	Time	Cuts
O$_1$-O$_5$	4	0.00	35.20	0.00	2.33	213	0.23	(5)	>5000	0.38	0.15	4
P$_1$-P$_5$	5	0.00	139.00	0.00	43.80	1029	0.03	(5)	>5000	0.02	0.85	4
Q$_1$-Q$_5$	5	0.00	729.60	0.00	251.20	1766	3.76	(5)	>5000	1.41	3.10	5
R$_1$-R$_5$	6	14.95	(5)	3.03	3229.80 (4)	4075	5.26	(5)	>5000	0.49	8.71	6
S$_1$	6	48.39	(1)	5.65	(1)	2562	7.63	(1)	>5000	0.64	22.28	6
T$_1$	6	61.80	(1)	9.15	(1)	627	10.06	(1)	>5000	1.89	50.02	6
Avg	**5**	**20.86**	**1950.63**	**2.97**	**1787.86**	**1712**	**6.35**	-	**>5000**	**2.07**	**14.19**	**5**

Table 5.5 highlights that $\mathcal{PBD}$ reaches the optimal solution without necessarily closing the gap between the UB and the RLB. While BD improves the LB more efficiently than the UB, both MILP and CPLEX BD improve the LB less efficiently than the UB. The performance of MILP, CPLEX BD, and BD is curbed by the necessity of closing the gap between the UB and the LB. In contrast, $\mathcal{PBD}$ does not have such an issue and can be stopped once there are no more promising variables to insert, thus ensuring the practical superiority of the proposed method.

Figure 5.8 shows the behavior of the four methods on instance MO1. For MILP, we report the bounds and the execution time along the x-axis. For CPLEX BD, we report the bounds and the number of iterations on the x-axis. For the BD and $\mathcal{PBD}$ decompositions, we report Obj$_1$ corresponding to the subproblem objective value, Obj$_2$ corresponding to the master problem objective value, and the number of iterations on the x-axis. The CPLEX BD requires 225 Benders iterations to converge. The BD does not converge and stagnates while the $\mathcal{PBD}$ reaches optimality in less than one second. The graphs confirm that MILP and CPLEX BD reach optimality on the UB in 6 and 0.75 seconds, respectively. Still, they do not converge



(a) *MILP* (b) *CPLEXBD* (c) *BD* (d) $\mathcal{PBD}$

Figure 5.8 Comparison between the Four Methods for Instance MO1

because the LB is not tight. On both of the top graphs of Figure 5.8, much of the time is spent bringing the LB to optimality. On the bottom graphs, BD stagnates since the UB and the LB do not change. On the bottom right side, $\mathcal{PBD}$ ensures a strict improvement from one iteration to another until reaching the optimal solution. This feature is one of the main strengths of $\mathcal{PBD}$.

An interesting aspect to check is the number of Benders cuts tight at each iteration, i.e., when solving the (RRMP$_\mathcal{T}$). Table 5.6 highlights such a comparison between BD and $\mathcal{PBD}$. We report the number of cuts as well as the average percentage of cuts tight (%Tight) in the (RRMP$_\mathcal{T}$) solution. For BD, less than half of the cuts are tight at each iteration. For $\mathcal{PBD}$, all cuts added are tight at each iteration. It highlights the significant difference in the quality of Benders cuts added by each method. $\mathcal{PBD}$ provides the best cut(s) given a subset of y variables compared to BD. Thus, it implies a quicker convergence to the optimal solution.

Another interesting point is that while BD relies on the gap between the UB provided by the subproblem and the LB provided by the master problem, $\mathcal{PBD}$ relies solely on the UB of

Table 5.6 Tight Cuts between BD and $\mathcal{PBD}$

M#	|J*|	BD		PBD	
		Cuts	%Tight	Cuts	%Tight
$\mathbf{O}_1$-$\mathbf{O}_5$	4	>5000	50.12	4	100.00
$\mathbf{P}_1$-$\mathbf{P}_5$	5	>5000	47.74	4	100.00
$\mathbf{Q}_1$-$\mathbf{Q}_5$	5	>5000	45.93	5	100.00
$\mathbf{R}_1$-$\mathbf{R}_5$	6	>5000	43.36	6	100.00
$\mathbf{S}_1$	6	>5000	40.12	6	100.00
$\mathbf{T}_1$	6	>5000	38.75	6	100.00
Avg	**5**	**>5000**	**44.37**	**5**	**100.00**

the subproblem. Thus, instead of the double effort required to get both bounds close to each other in BD, a single effort is required to get $\mathcal{PBD}$ to reach the optimal solution. Once no more complicating variables are identified, $\mathcal{PBD}$ stops. Then, Algorithm 7 allows confirming that the optimal solution of the original problem is the optimal solution of the subproblem augmented with zeros. The master problem in $\mathcal{PBD}$ acts like a guide, i.e., it guides the subproblem toward the optimal solution. Indeed, as it is shown on the bottom right graph of Figure 5.8, the master and subproblem bounds behave in a similar and, interestingly, decrease strictly. From the computational insights above, we formulate the following conjecture.

Conjecture 1. *For the FLP, the $\mathcal{PBD}$ converges strictly to the optimal solution in at most n iterations where n is the size of complicating variables.*

The BD does not profit from the primal information. On the other hand, $\mathcal{PBD}$ profits from the primal information and converges more quickly than BD. This is because the master problem provides the subproblem with interesting complicating variables, which improves the Obj_1. We highlight that, for the $\mathcal{PBD}$ method, the RLB (Obj_2) and UB (Obj_1) move in a correlated manner from one iteration to another.

5.6 COMPUTATIONAL RESULTS ON THE OCP GROUP CASE STUDY

We apply the $\mathcal{PBD}$ method to the large-scale problem that motivated this research. We first describe the problem and highlight its decomposition using the $\mathcal{PBD}$ method. Following that, we present the experimental design and computational results. We conclude this section with computational and managerial insights.

5.6.1 Problem Description

We consider the global supply chain of the OCP Group. It is one of the largest phosphate mining companies worldwide. It holds 70% of the world's phosphate rock reserves [177]. The company is promoting *precision farming*, i.e., a unique fertilizer for certain types of soil [178]. It has led to an increase in the number of products from three to more than 30.

The considered large-scale optimization problem involves integrated production scheduling, inventory management, and vessel assignment (PSIMVA), grouping several components of the downstream supply chain (red-framed part in Figure 5.9), making it quite complex. We detail the PSIMVA mathematical formulation and $\mathcal{PBD}$ decomposition in Appendix D.

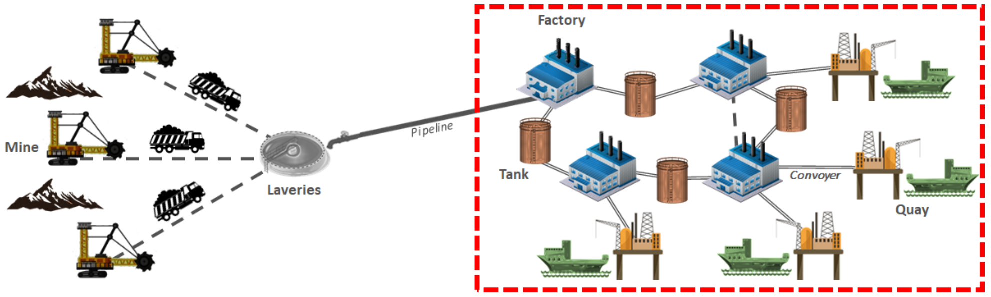


Figure 5.9 The Phosphate Supply Chain (reproduced from Figure 1 in [2])

5.6.2 Experimental Design

We consider a set of realistic PSIMVA instances provided by the OCP Group. The features of these instances, including the season, the horizon (days), the number of shipments (Vessels), the total demand in tonne, and the number of variables, binaries, and constraints, are presented in Table 5.7.

Table 5.7 Instances

ID	Season	Horizon	Vessels	Demand	Variables	Binaries	Constraints
$\mathbf{I}_1$	Winter	30	38	846702	125880	16292	109977
$\mathbf{I}_2$	Winter	30	38	856686	126314	16695	110922
$\mathbf{I}_3$	Autumn	34	34	926476	106020	17289	97896
$\mathbf{I}_4$	Summer	30	46	2340500	51330	4894	43150
$\mathbf{I}_5$	Autumn	31	60	3552590	103946	17506	80677
$\mathbf{I}_6$	Spring	30	62	1328880	183556	36923	160004
$\mathbf{I}_7$	Autumn	32	61	957338	948009	18264	780267
$\mathbf{I}_8$	Autumn	32	62	1328880	118539	13156	112158
$\mathbf{I}_9$	Winter	30	58	1797910	450772	15237	370827
$\mathbf{I}_{10}$	Winter	30	58	1797910	450773	15237	370827
$\mathbf{I}_{11}$	Winter	30	58	1797910	450789	15144	370844
$\mathbf{I}_{12}$	Summer	31	54	1199700	159414	19400	141335
$\mathbf{I}_{13}$	Spring	31	58	1273760	118435	14625	103810
Avg		**31**	**53**	**1538865**	**261060**	**16974**	**219438**

We use the same computational setting as Section 7.5. Furthermore, for the PSIMVA instances, we stop within epsilon of 0.1% between the upper and lower bounds to avoid the tailing effect [98] for all decompositions. Since the ($\mathcal{PBD}$ Master Problem) is a maximization problem, it provides a reduced upper bound (RUB). The ($\mathcal{PBD}$ Master Problem) minimizes the changeovers and provides the changeovers' and vessel assignments' information to the subproblem. The ($\mathcal{PBD}$ Subproblem) provides a LB on the PSIMVA optimal solution. We apply Algorithm 6.

5.6.3 Computational Results

We first present the performance of the $\mathcal{PBD}$ method. Then, we highlight the impact of the acceleration strategies. Lastly, we compare the $\mathcal{PBD}$ with other methods.

The PBD Performance

Table 5.8 presents the performance of the $\mathcal{PBD}$ method. We report the best-known upper bound (UB) which is the total fulfillment percentage, the optimality gap (Gap) in percentage, the execution time (Time) in seconds, and the number of cuts (Cuts). Since we do not know the optimal solution a priori, we approximate the optimality gap as $\frac{UB-LB}{UB}$ where LB is the best solution found, and UB is the best UB known. This formula overestimates the optimality gap formula $\frac{OPT-LB}{OPT}$, where OPT is the optimal value, and LB is the best solution obtained.

Table 5.8 $\mathcal{PBD}$ Peformance

ID	UB	Gap	Time	Cuts
$\mathbf{I}_1$	100.00	0.10	9	2
$\mathbf{I}_2$	100.00	0.10	8	2
$\mathbf{I}_3$	95.68	0.10	657	4
$\mathbf{I}_4$	100.00	0.10	600	5
$\mathbf{I}_5$	99.09	0.10	600	5
$\mathbf{I}_6$	92.02	0.24	10800	7
$\mathbf{I}_7$	93.43	0.32	10800	18
$\mathbf{I}_8$	96.99	0.43	10800	19
$\mathbf{I}_9$	97.30	0.56	10800	27
$\mathbf{I}_{10}$	99.22	0.77	10800	31
$\mathbf{I}_{11}$	99.03	0.79	10800	35
$\mathbf{I}_{12}$	95.29	0.82	10800	39
$\mathbf{I}_{13}$	96.86	0.89	10800	45
Avg	**97.30**	**0.37**	**6790**	**18**

We observe that the $\mathcal{PBD}$ method reaches near-optimal solutions in all instances. In particular, the method requires around 10 minutes to reach the optimal solution for the first five instances. For the remaining instances, the $\mathcal{PBD}$ fails to close the gap within the 3-hour time limit. For the number of cuts, the method requires 18 cuts on average to reach near-optimal solutions. The three-hour execution time provides a one-month plan.

Impact of Acceleration Strategies

We evaluate the impact of the acceleration strategies on the $\mathcal{PBD}$. Again, the abbreviations IP, WS, Cuts, SS, and Tuning refer to the initial point, warm-start, Benders cuts, selection strategy, and tuning, respectively. We evaluate $\mathcal{PBD}$ with the following strategies: (1) no initial point, (2) no warm-start, (3) all Benders cuts, (4) no selection strategy, and (5) no tuning. In the OCP case, no initial point means starting with the null solution, which is feasible. Furthermore, the *All Cuts* strategy corresponds to collecting all the optimality cuts corresponding to all encountered integer solutions from the B&B tree of (RPSP$_{\mathcal{S}}$) in a given iteration (using the *solution pool* procedure of CPLEX). We recall that we consider solely the local Pareto-optimal cut(s), obtained from the optimal node(s), in $\mathcal{PBD}$. Lastly, for the no selection strategy, we solve the (RRMP$_{\mathcal{T}}$) to optimality and insert the support of its solution in the (RPSP$_{\mathcal{S}}$) as described in Section 5.2. This is equivalent to running the basic $\mathcal{PBD}$. It is worth mentioning that, for the tuning, we conduct manual troubleshooting by exploring the log files of the PSIMVA instances, which highlights that few parameters impact the CPLEX's performance [1]. These parameters are *emphasis*, *heuristic effort*, and *CutsFactor*. No tuning strategy is simply running executions with the default CPLEX configuration.

Table 5.9 Impact of Acceleration Strategies

ID	**UB**	**No IP**			**No WS**			**All Cuts**			**No SS**			**No Tuning**			**PBD**		
		Gap	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**
$\mathbf{I}_1$	100.00	0.10	300	4	0.10	690	2	0.10	154	14	0.10	495	4	0.10	483	2	0.10	9	2
$\mathbf{I}_2$	100.00	0.10	330	4	0.10	894	2	0.10	173	14	0.10	612	4	0.10	626	2	0.10	8	2
$\mathbf{I}_3$	95.68	0.10	2715	7	0.10	6272	4	0.10	494	36	0.10	4493	7	0.10	4390	4	0.10	657	4
$\mathbf{I}_4$	100.00	0.10	1948	8	0.10	4500	5	0.10	1658	45	0.10	3224	10	0.10	3150	5	0.10	600	5
$\mathbf{I}_5$	99.09	0.10	2440	8	0.10	5636	5	0.10	1274	50	0.10	4038	10	0.10	3945	5	0.10	600	5
$\mathbf{I}_6$	92.02	1.20	10800	9	1.48	10800	5	0.72	10800	56	0.96	10800	17	1.18	10800	6	0.24	10800	7
$\mathbf{I}_7$	93.43	1.46	10800	20	1.96	10800	15	0.89	10800	90	1.18	10800	43	1.57	10800	16	0.32	10800	18
$\mathbf{I}_8$	96.99	1.69	10800	25	2.26	10800	16	1.06	10800	95	1.38	10800	46	1.81	10800	17	0.43	10800	19
$\mathbf{I}_9$	97.30	2.54	10800	34	3.40	10800	21	1.55	10800	135	2.05	10800	65	2.72	10800	20	0.56	10800	27
$\mathbf{I}_{10}$	99.22	2.98	10800	41	3.99	10800	25	1.88	10800	155	2.43	10800	74	3.19	10800	24	0.77	10800	31
$\mathbf{I}_{11}$	99.03	3.11	10800	57	4.17	10800	29	1.95	10800	175	2.53	10800	84	3.33	10800	28	0.79	10800	35
$\mathbf{I}_{12}$	95.29	3.54	10800	61	4.74	10800	30	2.18	10800	195	2.86	10800	94	3.79	10800	29	0.82	10800	39
$\mathbf{I}_{13}$	96.86	4.62	10800	73	6.19	10800	33	2.76	10800	225	3.69	10800	108	4.95	10800	31	0.89	10800	45
Avg	**97.30**	**1.63**	**7241**	**27**	**2.17**	**8030**	**15**	**1.00**	**6935**	**99**	**1.31**	**7636**	**43**	**1.74**	**7615**	**15**	**0.37**	**6790**	**18**

Table 5.9 highlights the impact of the acceleration strategies on the $\mathcal{PBD}$. When removing the initial point (No IP strategy), the gap deteriorates for the instances we could not solve optimally using the $\mathcal{PBD}$ in the 3-hour time limit. This is due to the information lost when no initial point is provided. The number of cuts increases for this strategy. When removing the warm-start (No WS strategy), we observe a similar trend with the gap deteriorating more than in the No IP strategy. Without warm-starting, the (RPSP$_{\mathcal{S}}$) requires more time. On the other hand, the number of cuts decreases because the higher time required by the (RPSP$_{\mathcal{S}}$) implies lower iterations. Under the All Cuts strategy, the (RPSP$_{\mathcal{S}}$) consumes more time, and the (RRMP$_{\mathcal{T}}$) as well since it contains more cuts. The number of cuts significantly increases. For the No SS strategy, the gap deteriorates, the time increases, and the number of cuts increases. Without the selection strategy, some unpromising complicating variables might be in the *supp*$(\bar{y})$ of the (RRMP$_{\mathcal{T}}$) solution, and thus add into (RPSP$_{\mathcal{S}}$). In such a case, the latter becomes larger, and further iterations are required to insert promising complicating variables. For the No Tuning strategy, the (RPSP$_{\mathcal{S}}$) takes more time, the number of iterations decreases, and the gap deteriorates. The five strategies solve the first five instances optimally but with more time.

Comparison with Other Methods

In the last section of the computational results, we compare $\mathcal{PBD}$ with the following four methods:

- **MILP:** solve directly with a MIP solver using the default CPLEX. The default CPLEX is provided as a baseline to measure the PSIMVA complexity.
- **ILNS:** solve using the $\mathcal{ILNS}$ metaheuristic developed by [2] for the same problem.
- **BD:** solve with the standard BD as it was initially developed [75] without any acceleration strategies. The BD is provided as a baseline to compare the number of cuts.
- **HBD:** solve using the hybrid Benders decomposition ($\mathcal{HBD}$), which is a hybridization of the BD [75, 98] and the $\mathcal{PBD}$. The HBD is provided since it fits with the practice at OCP.

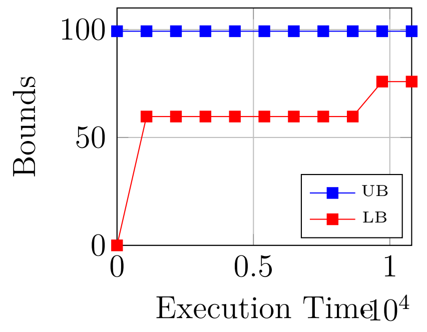
The intuition behind HBD is fixing some complicating variables (like BD) and keeping others free (like $\mathcal{PBD}$) in the Benders subproblem. At OCP, we distinguish confirmed and unconfirmed vessels. The vessel assignment information corresponding to the confirmed vessels is fixed in the subproblem, while the vessel assignment information corresponding to the unconfirmed vessels is kept free. We solve the resulting ($\mathcal{PBD}$ Subproblem) and obtain the Benders cuts. Table 5.10 reports the gap and time for each method. For decompositions, we report the number of cuts as well.

Table 5.10 Comparison with Other Methods

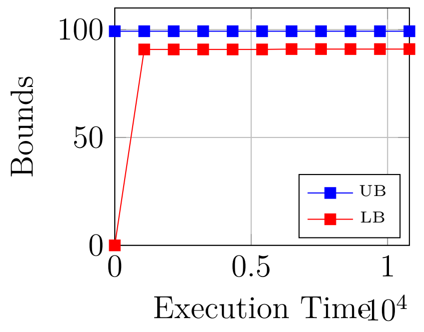
ID	UB	MILP		ILNS		BD			HBD			PBD		
		Gap	**Time**	**Gap**	**Time**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**	**Gap**	**Time**	**Cuts**
$\mathbf{I}_1$	100.00	4.96	10800	0.00	10499	0.10	900	110	0.10	400	44	0.10	9	2
$\mathbf{I}_2$	100.00	3.19	10800	0.00	294	0.10	900	123	0.10	500	49	0.10	8	2
$\mathbf{I}_3$	95.68	10.67	10800	13.29	10800	3.33	10800	348	0.10	4800	138	0.10	657	4
$\mathbf{I}_4$	100.00	100.00	10800	1.08	10800	3.05	10800	352	0.10	5600	140	0.10	600	5
$\mathbf{I}_5$	99.09	20.108	10800	13.11	10800	12.69	10800	379	7.06	10800	151	0.10	600	5
$\mathbf{I}_6$	92.02	51.82	10800	31.82	10800	12.87	10800	391	6.57	10800	155	0.24	10800	7
$\mathbf{I}_7$	93.43	17.78	10800	3.97	10800	8.42	10800	415	4.30	10800	165	0.32	10800	18
$\mathbf{I}_8$	96.99	24.77	10800	13.52	10800	7.25	10800	501	6.94	10800	169	0.43	10800	19
$\mathbf{I}_9$	97.30	67.80	10800	7.13	10800	6.15	10800	522	5.89	10800	176	0.56	10800	27
$\mathbf{I}_{10}$	99.22	30.74	10800	9.08	10800	13.98	10800	591	7.78	10800	186	0.77	10800	31
$\mathbf{I}_{11}$	99.03	46.94	10800	15.43	10800	15.81	10800	656	8.86	10800	181	0.79	10800	35
$\mathbf{I}_{12}$	95.29	100.00	10800	6.84	10800	16.06	10800	795	9.24	10800	193	0.82	10800	39
$\mathbf{I}_{13}$	96.86	49.82	10800	23.04	10800	17.22	10800	832	10.31	10800	215	0.89	10800	45
Avg	**97.30**	**40.67**	**10800**	**10.64**	**9969**	**8.99**	**9277**	**463**	**5.16**	**8346**	**151**	**0.37**	**6790**	**18**

We observe that $\mathcal{PBD}$ significantly outperforms the four methods. It obtains the lowest gap in all instances within less time and using fewer cuts. The MILP method fails to solve any instance optimally. The $\mathcal{ILNS}$ metaheuristic and the BD solve the first two instances to optimality. The $\mathcal{HBD}$ method solves the first four instances to optimality. Figure 5.10

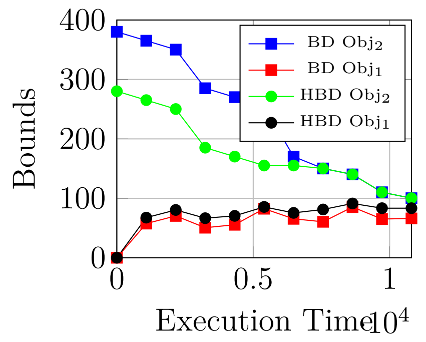
shows the bounds improvement along execution time for the instance I_{10} using the MILP and the $\mathcal{ILNS}$. It also shows, for the decompositions, Obj_1 corresponding to the subproblem objective value and Obj_2 corresponding to the master problem objective value.



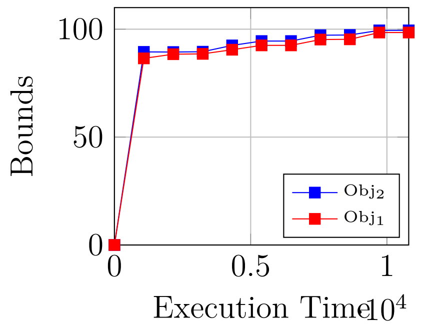
(a) MILP for Instance I_{10}



(b) ILNS for Instance I_{10}



(c) BD and HBD for Instance I_{10}



(d) $\mathcal{PBD}$ for Instance I_{10}

Figure 5.10 Bounds Improvement for along Execution Time the Five Methods for Instance I_{10}

We observe that MILP and ILNS struggle to bring the UB and LB close to each other. The decompositions show better behavior than the MILP and ILNS. In addition to the zigzagging behavior of BD, another important aspect to highlight is that BD does not profit from the primal information. For HBD, the zigzagging is less sharp than BD. It follows that HBD profits partially from the primal information. Still, both BD and HBD require several iterations to converge. On the other hand, $\mathcal{PBD}$ profits from the primal information and converges more quickly than BD and HBD. This is because the master problem provides the subproblem with interesting complicating variables, which improves the Obj_1. We highlight that, for the $\mathcal{PBD}$ method, the RUB (Obj_2) and LB (Obj_1) move in a correlated manner from one iteration to another.

5.6.4 Managerial Insights

The $\mathcal{PBD}$ shows great potential in the OCP case and outperforms several methods. Such a performance leads us to generate the following insights. While companies usually add enough complicating variables into mathematical models to ensure feasibility and optimality, the proposed $\mathcal{PBD}$ allows for selecting a nearly minimal set of complicating variables. This set is enough to solve the model to near-optimality and quickly. It is also practical for problems with high symmetry since the $\mathcal{PBD}$ allows benefiting from symmetry, i.e., fixing without regret, and breaking it, i.e., having fewer complicating variables in the model. In the OCP case, the company created more than enough complicating variables to maximize the fulfillment of all vessels. The same observations hold in practice at OCP. For instance, few changeover variables g are necessary. For many PSIMVA instances, fewer than four variables take non-zero values. Also, a few vessel assignment variables q are needed. For several PSIMVA instances, less than 200 variables take non-zero values.

Another interesting aspect is the schedules obtained using $\mathcal{PBD}$ and HBD methods. For instance I_{12}, Figures 5.11 and 5.12 show the schedules obtained by the $\mathcal{PBD}$ and HBD methods, respectively. The confirmed vessels are in green, and the unconfirmed vessels are in red.

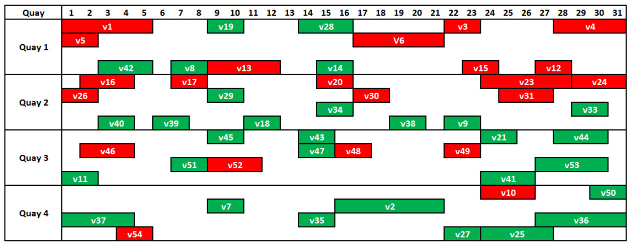


Figure 5.11 Schedule obtained by $\mathcal{PBD}$ for Instance I_{12}

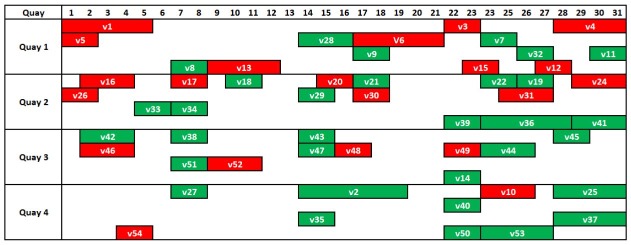


Figure 5.12 Schedule obtained by HBD for Instance I_{12}

Theoretically, the $\mathcal{PBD}$ method fulfills more vessels and ensures a higher total fulfillment than the HBD method. The reason is that $\mathcal{PBD}$ is a relaxation of HBD. However, from a practical perspective, the HBD guarantees better results for OCP since fixing the complicating variables corresponding to confirmed vessels prioritizes these shipments and forces their fulfillment. Figure 5.12 confirms this observation and shows the fulfillment of all the confirmed vessels when using HBD. On the other hand, Figure 5.11 shows that $\mathcal{PBD}$ might select unconfirmed vessels (v23 from period 24 to 28 on Quay 2) that contribute more to the TF maximization over other confirmed vessels (v19 and v22).

It is worth mentioning that, on the spectrum of Figure 5.1, the HBD comes as a compromise between $\mathcal{PBD}$ and BD as highlighted in Figure 5.13.

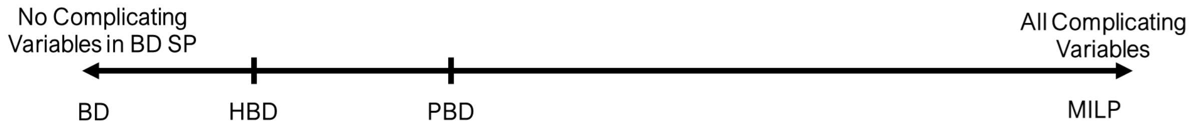

No Complicating Variables in BD SP
All Complicating Variables
BD
HBD
PBD
MILP

Figure 5.13 HBD as a compromise between $\mathcal{PBD}$ and BD

Given that $\frac{supp(y^*)}{n} \approx 0$, the size of (RPSP$_{\mathcal{S}}$) remains moderate, making the B&B procedure inexpensive and the solving efficient using acceleration strategies. The $\mathcal{PBD}$ explores good regions of the problem polyhedron and quickly accumulates, through the initial good solution(s) and the $\mathcal{PBD}$ restricted master problem, the primal information in the $\mathcal{PBD}$ subproblem. It is also practical for problems with high symmetry (e.g., PSIMVA) since the PBD allows benefiting from symmetry (various equivalent vessel assignments) and breaking it (exploring a small space of vessel assignments).

To sum up, the computational results highlight the importance of leveraging the primal information when tackling optimization problems, especially problems relying on decomposition techniques such as BD. Indeed, using the primal information, one can follow improving directions towards the optimal solution. This aspect is lacking in all dual methods, including BD in its standard form. Furthermore, while we use the PSIMVA and FLP to evaluate the $\mathcal{PBD}$, it is worth mentioning that it can be applied to many problems in the literature such as the multicommodity capacitated fixed charge network design problem, the stochastic network interdiction problem, the applications introduced in the Section 5.1, and many others [81]. Indeed, as highlighted with the PSIMVA and FLP instances, it is very promising to apply it to real-world sparse problems where the ratio $\frac{supp(y^*)}{n}$ is *small*.

5.7 CONCLUSION

This paper presents an attempt to view BD from a different perspective. Indeed, when seeing it as the dual of DWD, it is possible to leverage the primal information to converge more quickly than all dual BD methods. As far as we know, there has not been any such work in the literature. We believe this paper will open a new era in BD with many future research directions including other ways for *primalization*. While the basic version of the proposed algorithm, i.e., without acceleration strategies, is easily implementable and can be used for prototyping, there are several improvement opportunities. These improvements include the cut generation process using analytic centers and the exact selection of complicating variables. The main strengths of the $\mathcal{PBD}$ are the generation of solely optimality cuts, the convergence to optimal or near-optimal solutions in at most the size of complicating variables, the improvement of the UB at each iteration, and the scalability to several real-life problems since it relies on an intuitive dynamic insertion of complicating variables.

Acknowledgements

This work was supported by the Fonds de Recherche du Québec - Nature et Technologies (FRQNT) through a Ph.D. Excellence Scholarship, the Institute for Data Valorisation (IVADO) through another Ph.D. Excellence Scholarship, and the Group for Research in Decision Analysis (GERAD) through computing facilities for the experiments. This support is gratefully acknowledged.

CHAPTER 6 ARTICLE 4: TOWARDS RESILIENCE: PRIMAL LARGE-SCALE RE-OPTIMIZATION

Authors: El Mehdi Er Raqabi, Yong Wu, Issmaïl El Hallaoui, François Soumis

Submitted on February 7, 2024 and published on October 23, 2024 in Transportation Research Part E: Logistics and Transportation Review, Volume 192, 103819, DOI: 10.1016/j.tre.2024.103819

Abstract. Perturbations are universal in supply chains, and their appearance has become more frequent in the past few years due to global events. These perturbations affect industries and could significantly impact production, quality, cost/profitability, and consumer satisfaction. In large-scale contexts, companies rely on operations research techniques. In such a case, re-optimization can support companies in achieving resilience by enabling them to simulate several what-if scenarios and adapt to changing circumstances and challenges in real-time. In this paper, we design a generic and scalable resilience re-optimization framework. We model perturbations, recovery decisions, and the resulting re-optimization problem, which maximizes resilience. We leverage the primal information through fixing, warm-start, valid inequalities, and machine learning. We conduct extensive computational experiments on a real-world, large-scale problem. The findings highlight that local optimization is enough to recover after perturbations and demonstrate the power of our proposed framework and solution methodology.

Keywords. *Large-scale Optimization*, *Re-Optimization*, *Resilience*, *Primal Information*, *Machine Learning*, *Perturbation*.

History. This article is published in *Transportation Research Part E: Logistics and Transportation Review.*

6.1 INTRODUCTION

Perturbations are universal in supply chains (SCs), and their appearance has been more frequent in the past few years. These perturbations affect industries and organizations and could significantly impact production, quality, cost/profitability, and consumer satisfaction. They can be caused by several factors, including global events, localized incidents with global impact, and shifting environmental conditions. Global events such as the recent incidents and attacks at commercial vessels around the Red Sea [205], the Ukraine war [206], the COVID-19 pandemic [207, 208], and the food crisis [209, 210] have brought about unprecedented changes, creating uncertainty about what the future will look like. Localized incidents, such as the blockage of the Suez Canal by Ever Given [129], can create a huge challenge to global logistics. Furthermore, environmental conditions are shifting as we become more aware of issues such as climate change [211, 212] and natural disasters [213], which in turn have led to changes in operations and SC management, consumer behavior, and government policies, etc.

The perturbations highlighted above increase the operations management complexity within and among corporations. This increased complexity generates large and complex optimization problems. Given these problems' size, manual solving is intractable. Thus, companies invest heavily in optimization solvers. Still, these large-scale optimization problems involve combinatorial mathematical models with complex multiobjective functions, high symmetry and degeneracy, and millions of constraints and variables, requiring long solution time. In some, no feasible solutions can be rapidly identified, even using state-of-the-art optimizers. In such a case, organizations rely on sophisticated operations research (OR) techniques (e.g., mathematical optimization) to generate good quality or even feasible solutions. Still, the execution time might be relatively *long* if these OR techniques are run repetitively.

Using mathematical optimization for large-scale problems, companies seek to remain resilient to perturbations. Among many definitions [214], resilience can be defined as the ability of a system (e.g., company, organization, SC) to return to its original state or move to a new, more desirable state after being disturbed [215]. To do so, organizations should stay informed and adapt to any changes to sustain their operations and performance in the market. In several contexts, recovering after being perturbed and adapting to changes must be quick. Thus, companies cannot afford to optimize after each change using off-the-shelf optimizers or sophisticated OR techniques because recovery time might be relatively *long*.

Re-optimization can support companies in achieving resilience by enabling them to adapt to changing circumstances and challenges in real-time. It is an effective and efficient way to recover the original state quickly or move to a better one. Compared to optimizing from scratch after each change, re-optimizing from a previous state leverages the existing *primal* information, significantly reducing the recovery time, i.e., solving the updated and refined optimization models that reflect the new data and changing circumstances. Such gain can allow companies to re-optimize several times, i.e., whenever a perturbation affects its system. The key strength is the ability to sustain the primal information, such as the planned decisions, which are not affected by the perturbation and do not need to be changed. Furthermore, re-optimization also supports companies in identifying, simulating, and mitigating risks before they happen. By continuously analyzing data and considering potential trends, these companies can proactively identify and address vulnerabilities in their operations, increasing further resilience.

We position this paper in the context of large-scale organizations, facing frequent perturbations, and relying on mathematical optimization to plan their operations. Furthermore, we assume these companies run the deterministic optimization overnight (high complexity, millions of constraints, and integer variables, thus high execution time) and re-optimize during the following morning under two cases. The first case is when the stakeholders (e.g., managers, operators) from various departments (e.g., commercial, production, logistics) are meeting and like to simulate several what-if scenarios. In such a case, they want to perturb *artificially* the optimal solution computed overnight, re-optimize it, and get the results during the same meeting to support decision-making. The second case is when the company wants to re-optimize when *real* perturbation(s) happen and recover quickly. We merge both cases under the umbrella of *quick* re-optimization after perturbations and note that the execution time and complexity make re-optimization from scratch costly and inappropriate. We note that stochastic optimization is out of this research scope for the following reasons: (1) The mathematical optimization problems are quite large, (2) The solutions are required within very few minutes, and (3) The probability distribution for scenarios is unavailable in practice.

To illustrate our research, we consider a complex, real-world, large-scale optimization problem with frequent perturbations. The goal is to quickly recover after perturbations and reach a *satisfactory* solution as close as possible to the perturbed solution, which was previously the optimal solution. The present article has a threefold contribution: (1) we design a generic and scalable resilience re-optimization framework; (2) we highlight four ways to leverage the primal information, i.e., fixing, warm-start, valid inequalities, and machine learning (ML) techniques, in particular, we design a deep neural network (DNN) to reduce the number of complicating variables of the model; and (3) we conduct extensive computational experiments on a real-world, large-scale optimization problem, which highlight that local optimization is enough to recover quickly after perturbations.

We organize the rest of the paper as follows: We first present an overview of the relevant literature in Section 6.2. Then, we highlight the generic re-optimization framework in Section 6.3. Section 6.4 is devoted to a detailed description of the case and the corresponding problem formulation. Section 6.5 presents the case solution methodology. We highlight the case experimental design in Section 6.6, show the case computational results and managerial insights in Section 6.7, and conclude the paper in Section 6.8.

6.2 LITERATURE REVIEW

This section presents relevant literature on resilience, re-optimization, and primal information before positioning our research.

6.2.1 Resilience

Supply chain resilience (SCR) has been studied from qualitative and quantitative perspectives. The former, which dominated in the past [216], consists of approaching SCR in a rather qualitative manner, providing a set of strategies that can increase SCR without providing performance metrics to quantify the impact of a particular strategy on SC operations. The latter, which is more dominant in recent years [217, 218], consists of mathematically and analytically modeling and measuring SCR.

From a qualitative perspective, [219] study a manufacturer that faces a supplier privileged with private information about supply perturbations. They investigate how the risk-management strategies of the manufacturer change and examine whether risk-management tools are more or less valuable in the presence of such asymmetric information. [220] compares the disruption mitigation effects of three information management strategies using control theory modeling and simulation. They show that SCs with popular information management strategies are not more stable than traditional ones. [221] offer the notion of "commons" at different levels (company, private across the company, and government-sponsored across-industry sectors) and discuss how the creation of such commons enabled firms to be both efficient during normal times and resilient against the disruptions resulting from COVID-19.

Table 6.1 Summary of Relevant Resilience Literature

Author (Year)	Approach	Context	Model	Algorithm	Objective
Yang et al. [219]	Qualitative	Manufacturing	—	—	—
Chen and Miller-Hooks [222]	Quantitative	Transportation	SMIP	BD+CG+MC	Max Demand
An et al. [223]	Quantitative	Location	SMINLP	LR	Min Cost
Khaled et al. [224]	Quantitative	Transportation	MIP	Heuristic	Min Cost
Yang and Fan [220]	Qualitative	SC	—	—	—
[225]	Quantitative	SC	TSMSP	Multi-step	Min Cost
Sahebjamnia et al. [226]	Quantitative	Manufacturing	MIRPP	Two-phase	Min Loss
Elluru et al. [227]	Quantitative	SC	SMIP	LINGO	Min Cost
Hosseini et al. [228]	Quantitative	SC	SMIP	Two-step	Max Distance
Sawik [229]	Quantitative	SC	SMIP	—	Min Cost
Chopra et al. [221]	Qualitative	Industry	—	—	—
This Paper	Quantitative	SC	MILP	Heuristic	Max Resilience

From a quantitative perspective, [222] design an indicator for network resilience that quantifies the ability of an intermodal freight transport network to recover from disruptions due to natural or human-caused disasters. This indicator considers the network's inherent ability to cope with the negative consequences of disruptions as a result of its topological and operational attributes. They propose a stochastic mixed-integer program (SMIP) for quantifying network resilience and identifying an optimal post-event course of action (i.e., set of activities). They solve it using a technique that combines concepts from Benders decomposition (BD), column generation (CG), and Monte Carlo (MC) simulation. [223] present a scenario-based stochastic mixed-integer non-linear program (SMINLP) model that integrates facility disruption risks, en-route traffic congestion, and in-facility queuing delay into an integrated facility location problem. After deriving lower and upper bounds, they tackle it using Lagrangian relaxation (LR). [224] propose a mixed-integer programming (MIP) model for making up and routing trains in a disruptive situation to minimize the system-wide total cost, including classification time at yards and travel time along links. They solve it using an iterative heuristic algorithm. [225] present a two-stage scenario-based mixed stochastic-possibilistic programming (TSMSP) model for the integrated production and distribution planning problem in a two-echelon supply chain over a midterm horizon under risk. They solve it via a multi-step approach. [226] propose an integrated business continuity and disaster recovery planning (IBCDRP) model to build organizational resilience that can respond to multiple disruptive incidents, which may occur simultaneously or sequentially. A multi-objective mixed-integer robust possibilistic programming (MIRPP) model, which accounts for sensitivity and feasibility robustness, is formulated. They tackle it using a two-phase approach. [228] provide a stochastic bi-objective mixed integer programming model to support the decision-making in how and when to use proactive and reactive strategies in supplier selection and order allocation. They solve it using a two-step approach. [227] stipulate that the supply chain distribution network broadly comprises two major decisions: facility location and vehicle routing. Then, they address these distribution decisions jointly as a location-routing problem and solve it using the solver LINGO. [229] proposes a two-period modeling approach for supply chain disruption mitigation and recovery and compares it with a multi-period approach. We summarize the presented literature in Table 6.1.

6.2.2 Re-optimization

Re-optimization is an efficient way to ensure resilience in large-scale contexts. We distinguish two types of re-optimizations: major and minor. Major re-optimizations happen after a disruption and are more strategic/tactical and less frequent (e.g., annually). They are often conducted from scratch and usually require an exact algorithm. On the contrary, minor re-optimizations occur after a perturbation and are more tactical/operational and frequent (e.g., weekly, daily, real-time). They are conducted more from a previous solution than from scratch and usually require a heuristic algorithm.

From a major perspective, [230] rely on re-optimization to re-engineer a system. The latter consists of reorganizing the collection system of an Italian postal service provider. They model the problem as a MIP that identifies the number of postboxes (currently located in an urban area) to be closed. They solve it using a two-phase methodology based on mathematical programming. [222], cited in Section 6.2.1, also rely on major re-optimization. We recall that they propose a SMIP for quantifying network resilience and identifying an optimal post-event course of action (i.e., set of activities).

Table 6.2 Summary of Relevant Re-optimization Literature

Author (Year)	Type	Context	Model	Algorithm	Objective
D'Ariano et al. [231]	Minor	Transportation	Graph	Heuristic	Max Trajectory
Chen and Miller-Hooks [222]	Major	Transportation	SMIP	Exact	Max Demand
Archetti et al. [232]	Minor	Transportation	—	Heuristic	Min Cost
Dong et al. [233]	Minor	Transportation	MIP	CPLEX	Min Cost
Schieber et al. [234]	Minor	CRO	MIP	Heuristic	Min Distance
Doerr et al. [235]	Minor	CRO	MO	Heuristic	Min Distance
Hassani et al. [236]	Minor	Personnel	MIP	Heuristic	Min Cost
Bruno et al. [230]	Major	Location	MIP	Exact	Min Distance
Hasani et al. [168]	Minor	Personnel	MIP	Labeling	Min Cost+Distance
This Paper	Minor	SC	MIP	Heuristic	Max Resilience

From a minor perspective, [231] present a graph formulation for the train running profile problem and develop a conflict solution system that models the train scheduling problem as an alternative graph. From a network point of view, they improve the optimal solution by modifying the speed profiles locally for the individual train routes. They propose a constructive heuristic algorithm for the dynamic modification of running times during operations that satisfies the timetable constraints of train routes and orders and guarantees the feasibility of the running profile while considering the properties of the signaling and train protection systems in use. [232] explore the Rural Postman Problem (RPP) re-optimization given an instance and its optimal solution. They study the problem of finding a satisfactory feasible solution after a perturbation (new edge added or removed) of the instance has occurred and tackle it heuristically. [234] develop a general framework for combinatorial re-optimization (CRO), encompassing classical objective functions to minimize the transition cost from one solution to the other. Using their model, they derive re-optimization and re-approximation algorithms for several combinatorial re-optimization problem classes. [233] study a maritime inventory routing MIP problem in which vessels carry the shipments between production and consumption nodes. In the face of new information and uncertainty, this optimization model has to be resolved as the horizon is rolled forward. They discuss how to account for different sources of uncertainty and present a rolling-horizon re-optimization framework that allows studying various policies that impact the quality of the implemented solution. They use the solver CPLEX for re-optimization. [235] show that evolutionary algorithms can have unexpected difficulties in solving re-optimization problems, which build on a previously good feasible solution. Then, they propose a simple diversity mechanism that works for various mathematical optimization (MO) problems, including the LeadingOnes, linear functions with modified uniform constraints, and the minimum spanning tree problems. [236] develop a fast re-scheduling heuristic to solve the personnel re-scheduling problem in a context where employees can work on various shifts, such as in the retail industry. [237] propose a fast re-scheduling heuristic to correct minor disruptions in the same retail context. This heuristic computes a set of approximate Pareto-optimal solutions that achieve a good compromise between cost and number of shift changes. Theoretically, the authors show, for a shift scheduling problem, that from generating decisions (e.g., advancing an employee, delaying a shift, etc.), it is possible to construct elementary decisions. The latter are equivalent to edges in the convex hull of integer solutions. Thus, they allow moving towards better solutions. It is similar to a labeling algorithm that partially explores the network defined by the edges of the convex hull of the solutions of an integer program. They formalize the neighborhood notion in heuristics, which depends on the problem studied. Computationally, their primal approach is very effective. We summarize the relevant re-optimization literature in Table 6.2.

6.2.3 Primal Information

Practically, primal methods (e.g., Dantzig-Wolfe decomposition) have always been preferred to their counterparts (dual methods like Benders decomposition) since they allow an efficient improvement of the primal solution implemented in practice. Thus, over the last decades, OR practitioners have been using these methods and leveraging the available primal information (e.g., previous solutions) through classical strategies such as fixing, warm-start, and valid inequalities [238, 239].

More recently, leveraging the primal information to accelerate (re-)optimization processes gained more attention and focus from both OR and ML practitioners. In the context of large-scale optimization, we distinguish several papers that leverage primal information to boost optimization. Morabit et al. [111] develop a column selection approach and apply it to select a subset of the columns generated at each iteration of column generation. The goal is to reduce the computing time spent re-optimizing the restricted master problem at each iteration by selecting the most promising columns. Tahir et al. [112] propose a new integral column generation algorithm for the crew pairing problem, which leaps from one integer solution to another until a near-optimal solution is found. They rely on reduced subproblems containing only flight connections that have a high probability of being selected in a near-optimal solution. They predict these probabilities using a deep neural network trained in a supervised framework. They also fix several variables using a diving heuristic. Quesnel et al. [113] propose a new branch-and-price algorithm for the aircrew rostering problem. In their pricing subproblems, they only include pairings likely to be in a near-optimal solution. They design a neural network to predict the probability that a pairing is assigned to a crew member. Then, they use those predictions to select which pairings to include in each subproblem. Morabit et al. [114] propose a new ML-based pricing heuristic. By taking advantage of the data collected during previous executions, the objective is to reduce the size of the network and accelerate the subproblems. Computationally, they achieve up to 40% reductions in computational time. All these papers highlight that using ML enhance significantly the ability to optimize large-scale problems efficiently.

Bengio et al. [100] survey the recent attempts, both from the ML and OR communities, at leveraging the primal information using ML to solve combinatorial optimization (CO) problems. They support pushing further the integration of ML and CO and detail a methodology to do so. More recently, the trend has been to revolutionize decision-making at massive scales by fusing ML and OR to deliver scientific breakthroughs that the two fields cannot achieve independently [99].

6.2.4 Our Research

In our paper, we seek SCR via re-optimization. As far as we acknowledge, the only close research paper to ours is the one in the intersection of Sections 6.2.1 and 6.2.2, which is the work of [222]. While the paper of [222] and this paper are quantitative, the former belongs to the major re-optimization (disruption) case. However, this paper belongs to the minor re-optimization (perturbation) case. Furthermore, we quickly re-optimize from a previous solution while [222] re-optimize from scratch. All other papers belong to either Section 6.2.1 or Section 6.2.2 but not both.

Within the resilience and re-optimization pieces of literature, we distinguish our paper as follows. First, we propose a generic and scalable resilience/re-optimization framework. Second, using a global SC (large-scale problem) case to illustrate, we quantify resilience and build a new MIP model from the model in [2], where we keep the relevant constraints and variables and set resilience as the objective function. Third, we leverage the primal information using ML, valid inequalities, warm-starting, and fixing techniques to reach satisfactory solutions quickly to suit the need for SCR via re-optimization.

6.3 RESILIENCE/RE-OPTIMIZATION FRAMEWORK

In this section, we introduce the Resilience/Re-Optimization (RRO) framework. We then offer discussions from a managerial perspective based on the supply chain operations reference (SCOR) model.

6.3.1 Framework

Let us consider a large-scale optimization problem (*Original Problem*) modeling the SC of a company. Using solvers and OR techniques, the company obtains an optimal solution. Faced with uncertainties (*Perturbation(s)*), this solution may no longer be feasible. In such a case, we note that solution ~~q~~*. Thus, the company wants to be resilient and to quickly re-optimize and reach a near-optimal, if not optimal, solution q* as *satisfactory* as possible as solution ~~q~~*.

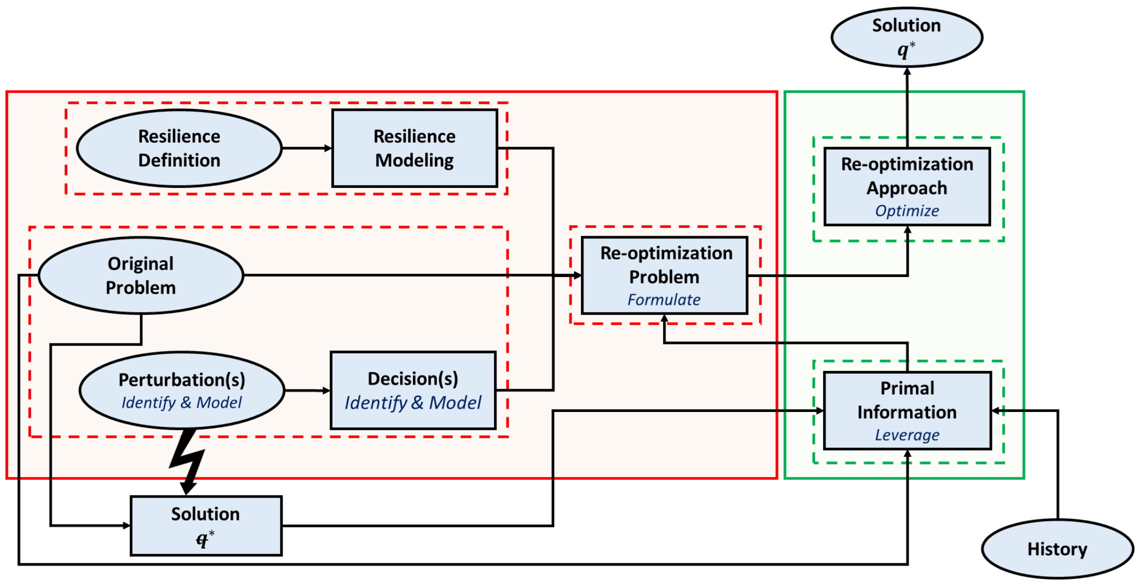


Figure 6.1 RRO Framework

To remain resilient, the company has to define (*Resilience Definition*) and model (*Resilience Modeling*) resilience. The *resilience definition* must be clear to allow the company model accordingly. Furthermore, the company has to identify and model *perturbation(s)*. Then, it has to identify and model the set of actions (*Decision(s)*) to take. Following these aspects, it can formulate the *re-optimization problem* to maximize resilience given the *original problem*. We refer to this first stage (red frame) as the *problem definition* stage.

After formulating the *re-optimization problem*, the company can design its *re-optimization* *approach* while leveraging the *primal information* using the *solution* $\not{\text{q}}^*$, the *original problem*, and the company's *history* (e.g., solutions history, problem knowledge, accumulated expertise). Such information is relevant since we do not want to optimize from scratch. The qualification *primal* is borrowed from the optimization lexicon and used mainly to distinguish between dual and primal methods. The former does not incorporate the available information in the optimization process, while the latter leverages it to reach optimality quickly. The *re-optimization approach* allows reaching a feasible solution q^* as *close* as possible to the no-longer feasible solution $\not{\text{q}}^*$ and hence causes the least amount of changes in response to the *perturbations*. We refer to the second stage (green frame) as the *solution methodology* stage. We reflect these stages in the RRO framework in Figure 6.1.

6.3.2 Managerial Discussion

We discuss the RRO framework from a managerial perspective based on the SCOR model in Figure 6.2. A company with a set of suppliers and customers faces several perturbations. These perturbations can happen in any pillar of the SCOR model. Based on the point of emergence in the supply chain, these perturbations are either internal or external. The internal ones are taking place inside the company. The external ones are either inbound (i.e., from the frontier with the supplier and above) or outbound (i.e., from the frontier with the customer and below). For instance, on the *make* pillar of the company, the company may face a machine breakdown curbing production. On the *deliver* pillar of the supplier, the company may receive raw materials later than planned. On the *source* pillar of the customer, the customers may cancel orders and or change their requirements.

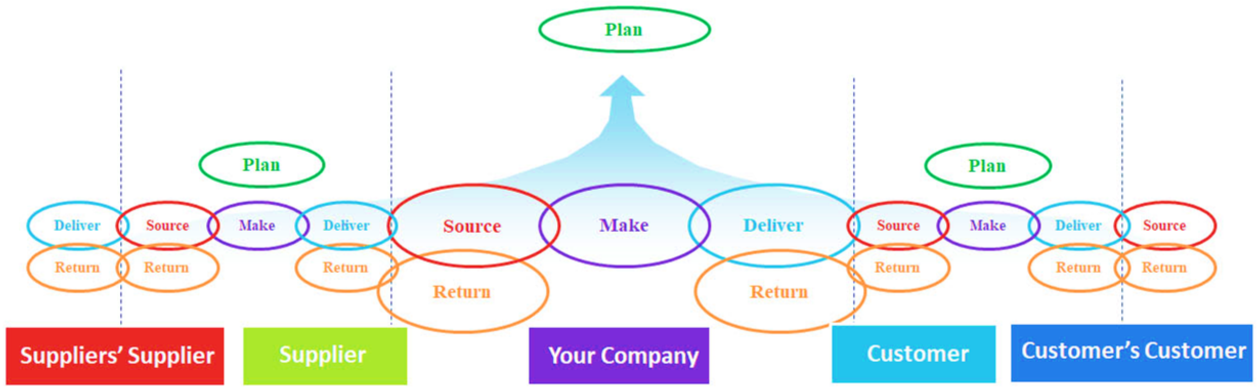


Figure 6.2 SCOR Model from AIMS

These perturbations also vary in terms of impact magnitude. Based on the impact, we distinguish minor and major perturbations. For instance, delayed delivery of raw materials is a minor perturbation when the company has enough safety stock. A machine breakdown curbing the production process is a major perturbation. It is worth mentioning that we focus in this paper on perturbations since it is possible to recover quickly. It is not the case for disruptions. Indeed, a company may need several weeks to recover and resume operations after a natural disaster (e.g., tsunami or earthquake).

To deal with these perturbations, the company has to take a set of actions to deal with them. For instance, on the company's *source* pillar, a company may decide to diversify suppliers or increase the safety stock levels. On the company's *make* pillar, the company may opt for a strict preventive maintenance strategy to diminish the machine breakdown occurrence. On the company's *deliver* pillar, the company may classify customer orders into confirmed and unconfirmed ones. Then, it can focus on fulfilling the confirmed ones and postponing unconfirmed ones until confirmation.

The company must also seek resilience to adapt quickly to changes. To do so, it must establish a clear resilience definition. Then, it can develop an indicator that models resilience. Using these inputs, the company can design a re-optimization problem that can, when solved quickly based on the company's history, support decision-making when faced with perturbations. By merging resilience and re-optimization, the RRO framework ensures the following benefits. First, the re-optimization from a previous solution $ꝗ^*$ (leveraging primal information) is quicker than re-optimizing from scratch. It ensures a quick solution, which implies agile adaptation and operations recovery. Second, the resilience consideration implies few changes, making the company stakeholders (e.g., operators, supervisors, and managers) less worried about the necessary changes. Indeed, operators prefer solutions that require fewer changes. The reason is that they do not have to deviate a lot from the already prepared plans.

In what follows, we illustrate the RRO framework considering the case of a global SC. Section 6.4 highlights the red-framed part (*Problem Definition*) with the dashed red frames being subsections. Section 6.5 presents the green-framed part (*Solution Methodology*) with the dashed green frames being subsections.

6.4 CASE PROBLEM DEFINITION

In this section, we present one implementation of the RRO framework for a complex real-world large-scale optimization problem. We first present the context (*Original Problem*, *Perturbations*, and *Decisions*). Then, we discuss resilience (*Resilience Definition* and *Resilience Modeling*).

6.4.1 Context

To present the context, we first describe briefly the original problem. Then, we highlight the perturbations and the decisions to tackle them. While the work presented for illustration is inspired by a large-scale mining company, its findings can be transferred, adapted, and applied to tackle similar difficult and large-scale problems in other industries.

Original Problem

We consider the global SC of *OCP Group*, one of the largest phosphate companies worldwide, holding 70% of the world's phosphate rock reserves [177]. It specializes in phosphate mining, production, and exportation and has branches in Brazil, India, Morocco, and other countries. Raw Phosphate, Phosphoric Acid, and Phosphorus Fertilisation Products are also included in the product range.

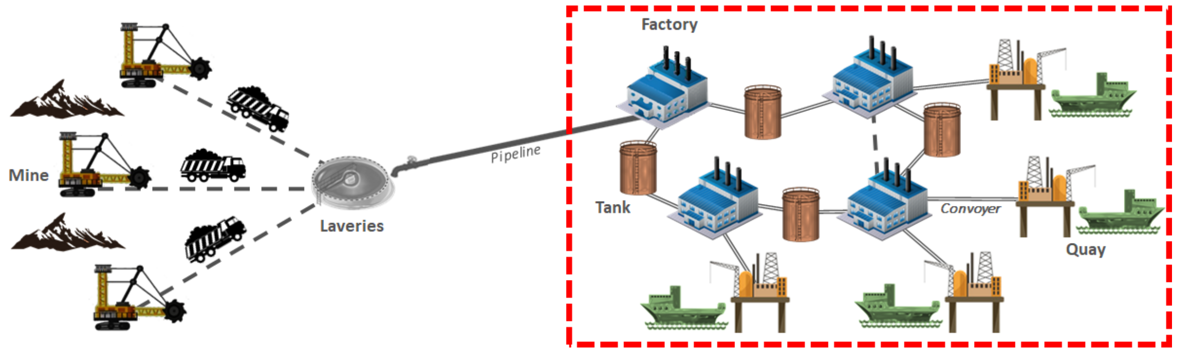


Figure 6.3 The Phosphate Supply Chain (reproduced from Figure 1 in [2])

The company is promoting *precision farming*, i.e., a unique fertilizer for certain types of soil [178]. It has led to an increase in the number of products from three to more than 30. The global SPC has four main components, through which 45 raw, partially finished, and final products flow, as highlighted in Figure 6.3. Phosphate rock is taken from the mine and transported by truck to a physical processing facility, where they are washed and floated. A 187 km slurry pipeline transports washed rocks to a chemical treatment plant on the coast. There are 32 different chemical processes used to process some derivatives. After that, the final products are stored in 29 large tanks before being transported by conveyors to the docks where the ships of clients worldwide are loaded. The coastal processing plant and the loading dock are spread over a 5 km^2 area. The phosphate rock is processed at an annual rate of 37.6 million tonnes, which accounts for 31% of the world's phosphate market. There are 102 conveyors and pipelines to connect the supply chain [24]. In a nutshell, the considered large-scale optimization problem involves integrated production scheduling, inventory management, and vessel assignment (PSIMVA), grouping several components of the downstream supply chain (red-framed part in Figure 6.3), making it quite complex. Information on the PSIMVA is available at [2].

The OCP case is interesting because of the following: (1) OCP faces several frequent *real* perturbations, which makes re-optimization from scratch costly; (2) The company has enough historical data, which allows leveraging the primal information; and (3) Operators run the PSIMVA optimization overnight. Then, during the day, OCP stakeholders in charge of the downstream operations planning perturb *artificially* the available optimal solution (schedule) $\not{\mathrm{q}}^*$ (computed overnight), re-optimize *quickly*, and get the new solutions in the same meeting (e.g., what orders to prioritize/confirm). Such interaction between stakeholders (arbitrage) is necessary because the mathematical model does not capture all the factors, mainly the qualitative ones.

Without loss of generality, we consider a sample of relevant perturbations among all possible ones. In particular, we are interested in perturbations on the port side, i.e., vessel assignment. In what follows, we assume that the company already has an optimal solution (schedule) $\not{\mathrm{q}}^*$ no longer feasible (because of perturbations) and obtained overnight using, for example, solvers and OR techniques [2, 1, 3]. The company wants to use this solution to (1) simulate several what-if scenarios during morning meetings and support decision-making, (2) recover quickly near-optimal, if not optimal, solutions following *real* perturbations.

Perturbations

The OCP downstream operations are located in a coastal area. On the vessel assignment (VA) part of the PSIMVA, each customer vessel $v \in \mathcal{V} = \{1, ..., \overline{V}\}$ must be assigned to a quay $k \in \mathcal{K} = \{1, ..., \overline{K}\}$ for loading within a time interval (a few consecutive periods, with a period equivalent to one day). Let $\mathcal{T} = \{1, ..., \overline{T}\}$ be the set of $\overline{T}$ periods. The loading cannot be partial, i.e., a vessel is either fulfilled or not fulfilled. To accomplish this, a decision must be made on the assignment of each vessel $v \in \mathcal{V}$ to a quay based on a set of possible assignments $i \in \mathcal{I}_v = \{1, ..., \overline{I}_v\}$. We denote $\mathcal{I} = \bigcup\limits_{v \in \mathcal{V}} \mathcal{I}_v$ and $\mathcal{I}_{vk}$ the set of possible assignment of vessel $v \in \mathcal{V}$ restricted to quay $k \in \mathcal{K}$. Each $i \in \mathcal{I}$ is a quadruplet $(\underline{t}, \overline{t}, k, v)$ where $i_1 = \underline{t}$ is the starting period, $i_2 = \overline{t}$ is the ending period, $i_3 = k$ is the quay, and $i_4 = v$ is the vessel. Each vessel has then a set of binary variables q_i with $i \in \mathcal{I}$, from which only one must be selected.

The OCP Group faces several perturbations related to the port, i.e., the VA part. We consider two perturbations as follows. A weather Perturbation p_1 occurs when the weather in the port is bad enough to affect normal operations. A vessel Perturbation p_2 occurs when a vessel's arrival at the port is delayed. If a perturbation p_1 occurs, all the vessels previously scheduled for loading during the perturbation's period(s) can no longer be loaded. Also, if a perturbation p_2 occurs for a given vessel, this vessel cannot be loaded as scheduled. Thus, these perturbations make ꝗ* no longer feasible. Furthermore, while being both minor, it is worth mentioning that the two perturbations considered are different in terms of time. In fact, assuming that the weather is accurately predicted a week before, perturbations p_1 can be tackled on a weekly basis, i.e., we eliminate the case where a vessel $v \in \mathcal{V}$ is in the loading process when a perturbation p_1 happens. Thus, after forecasting a weather perturbation period(s), vessels are rescheduled for loading with no intersection with the perturbation's period(s). On the other hand, perturbations p_2 happen in real-time.

From a modeling perspective, both perturbations p_1 and p_2 imply the removal of vessel assignment variables from the optimization model. For instance, if the weather is bad during a time interval $[t_1, t_2]$, all the assignment variables q_i, $i \in \mathcal{I}$ having a non-empty intersection with this time interval ($[i_1, i_2] \cap [t_1, t_2] \neq \emptyset$) are removed from the optimization model. Similarly, if a vessel $v \in \mathcal{V}$ is delayed, all its variables q_i, $i \in \mathcal{I}_v$ before its new arrival period are removed from the optimization model.

Decisions

When perturbations occur, decisions must be taken to recover quickly. Before introducing decisions, we present the following assumptions: (1) the company has enough stocking space at the port; (2) customer vessels may arrive earlier than scheduled and wait close to the port. The first assumption is realistic in the large-scale context because companies usually manage large product quantities. Thus, by design, they have quite large stocking spaces and entities. The second assumption is also realistic since in many ports worldwide (e.g., Singapore port in Figure 6.4), vessels wait close to the port until authorized to enter for loading.

The first assumption allows the company to produce and stock, and load after a delayed vessel arrives. This is very relevant and practical since it allows the company to maintain the production schedule as it is. The second assumption enables the possibility to advance some vessels (queuing close to the port) ahead of their schedules if others are delayed.

Next, we introduce an observation, which extends the work of [237] to larger and more complex supply chain problems.

Observation 6. *In practice and for any given large-scale optimization problem, there is*

Figure 6.4 Singapore Port Queue

always a minimal set of decisions that allows reaching all feasible solutions, including the optimal one(s), from a non-longer feasible solution.

While Observation 6 does not hold theoretically, it holds in practice since organizations can always recover from non-longer feasible solutions due to perturbations. We introduce the two decisions taken to face port perturbations. First, a delay (del) decision d_1 is taken when the weather at the port is bad or when a vessel is delayed. Second, let us consider two vessels v_1 and v_2 in $\mathcal{V}$ with similar products. If v_1 is delayed and v_2 is queuing close to the port, an advance (adv) decision d_2 occurs to allow loading vessel v_2 ahead of schedule.

It is worth mentioning that if a vessel $v \in \mathcal{V}$ is not expected to arrive within the planning horizon, it is simply delayed using decision d_1 beyond the planning horizon. It is worth that swapping or permuting two vessels $v_1, v_2 \in \mathcal{V}$ with v_1 being ahead of v_2 is equivalent to applying decision d_1 to vessel v_1 and decision d_2 to vessel v_2. Furthermore, for decisions d_1 and d_2, the whole space of feasible solutions is generated as shown in Proposition 6 below (all mathematical proofs are in Appendix E).

Proposition 6. *Any feasible schedule $\bar{q}$ can be reached from the no longer feasible solution $\not{q}^*$ using decisions d_1 and d_2.*

Proposition 6 is relevant because it ensures that the whole feasible space is explored. Thus, no feasible solution is discarded, including the new feasible schedule q* we are looking for.

When a perturbation happens, we enumerate the set of delayed (resp. advanced) vessels $\mathcal{V}^{del}$ (resp. $\mathcal{V}^{adv}$). From a modeling perspective, decision d_1 corresponds to adding new variables

q_i^{del}, $i \in \mathcal{I}_v$ to any delayed vessel $v \in \mathcal{V}^{del}$ with $i_1 \geq \mathit{loading}$, *loading* being the potential (delayed) loading start period of vessel v. Decision d_2 corresponds to adding new variables q_i^{adv}, $i \in \mathcal{I}_v$ to any advanced vessel $v \in \mathcal{V}^{adv}$ with $i_1 \geq \mathit{loading}$, *loading* being the potential (advanced) loading start period of vessel v. For advanced vessels, we keep also their initial assignment variables. To differentiate, we refer to these variables as q_i^{ini}, $i \in \mathcal{I}_v$ with $v \in \mathcal{V}^{adv}$.

6.4.2 Resilience

This section defines resilience and models it mathematically for our context.

Resilience Definition

Among many definitions [214], resilience can be defined as the ability of a system (e.g., company, organization, SC) to return to its original state or move to a new, more desirable state after being disturbed [215].

In our context, SCR is the ability to recover or reach a better schedule while remaining as *close* as possible to the original schedule. By *close*, we imply fulfilling as many vessels as the previously optimal schedule while maintaining the least distance possible to the previously optimal schedule.

Resilience Modeling

As per the definition above, we highlight two resilience indicators. The first is maximizing the total fulfillment (TF) and minimizing the distance between schedules (ΔD). The first indicator TF is the first KPI in [2]. Denoting Q_v^p as the quantity of product $p \in \mathcal{P}$ ordered by vessel $v \in \mathcal{V}$, TF can be modeled as follows:

$$TF(q) = 100 \times \frac{\sum_{v \in \mathcal{V}, p \in \mathcal{P}_v} \sum_{i \in I_v} Q_v^p q_i}{\sum_{v \in \mathcal{V}, p \in \mathcal{P}_v} Q_v^p}$$

The second indicator ΔD can be modeled as follows:

$$\Delta D(q) = -100 \times \frac{\sum_{v \in \mathcal{V}^{del}, p \in \mathcal{P}_v} \sum_{i \in I_v} Q_v^p q_i^{del} + \sum_{v \in \mathcal{V}^{adv}, p \in \mathcal{P}_v} \sum_{i \in I_v} Q_v^p q_i^{adv}}{\sum_{v \in \mathcal{V}^{del} \bigcup \mathcal{V}^{adv}, p \in \mathcal{P}_v} Q_v^p}$$

For a given schedule q, $\Delta D(q)$ measures the percentage of vessels delayed and advanced. The negative sign is added since we want to maximize resilience and minimize the distance.

After modeling these two conflicting indicators, resilience can be modeled as a weighted objective:

$$R(q) = \alpha_1 \times TF(q) + \alpha_2 \times \Delta D(q),\ (\alpha_1, \alpha_2) \in \mathbb{R}^2 \text{ such that } \alpha_1 + \alpha_2 = 1$$

6.4.3 Re-optimization Problem

Following the design above, the re-optimization problem is written as:

$$\max \; R(q) \tag{Re-Opt}$$

$$s.t.: \sum_{i \in I_v} q_i^{del} \leq 1 \qquad \forall v \in \mathcal{V}^{del} \tag{6.1}$$

$$\sum_{i \in I_v} q_i^{ini} + q_i^{adv} \leq 1 \qquad \forall v \in \mathcal{V}^{adv} \tag{6.2}$$

$$\sum_{i \in I_v} q_i \leq 1 \qquad \forall v \in \mathcal{V} \setminus \mathcal{V}^{del} \cup \mathcal{V}^{adv} \tag{6.3}$$

$$\sum_{v \in \mathcal{V}^{del}} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t \leq i_2}} \mathbf{L}_v q_i^{del} + \sum_{v \in \mathcal{V}^{adv}} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t \leq i_2}} \mathbf{L}_v (q_i^{ini} + q_i^{adv}) + \sum_{v \in \mathcal{V} \setminus \mathcal{V}^{del} \bigcup \mathcal{V}^{adv}} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t \leq i_2}} \mathbf{L}_v q_i \leq \mathbf{L}_k \quad \forall k \in \mathcal{K}, t \in \mathcal{T} \tag{6.4}$$

$$\sum_{v \in \mathcal{V}, p \in \mathcal{P}_v} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t' \leq i_2}} \sum_{t'=t}^{t+6} \mathbf{Q}_v^p q_i \leq \mathbf{MAX} \qquad \forall k \in \mathcal{K}, t \in \mathcal{T}^- \tag{6.5}$$

$$q_i \in \mathbb{B} \qquad \forall i \in I_v, v \in V \tag{6.6}$$

The objective function maximizes the resilience. Constraints (6.1) ensure that, at most, a single assignment is selected for each delayed vessel. Constraints (6.2) ensure that, at most, a single assignment is selected for each advanced vessel. Constraints (6.3) ensure that normal vessels must be fulfilled. Constraints (4.14) ensure that for a given quay $k \in \mathcal{K}$ and a given period $t \in \mathcal{T}$, the total length of all vessels assigned to quay k is within the quay length restrictions. The length of a quay is noted $\mathbf{L}_k$ with $k \in \mathcal{K}$ while the length of a vessel is noted $\mathbf{L}_v$ with $v \in \mathcal{V}$. Constraints (4.15) are operational rules (specified by the OCP Group for the quay cranes), which ensure that the maximum quantity that can be loaded on vessels in a given quay $k \in \mathcal{K}$ over a week (seven periods) is below a capacity (**MAX**). We note $\mathcal{T}^- = \mathcal{T} \setminus \{\overline{\mathcal{T}} - 5, ..., \overline{\mathcal{T}}\}$. Constraints (4.18) ensure the binary restrictions on the q variables.

In the next section, we develop the solution methodology. The objective is to find a feasible schedule that maximizes resilience under port constraints.

6.5 CASE SOLUTION METHODOLOGY

In this section, we highlight different ways to leverage *Primal Information.* Then, we provide the *Re-optimization Approach.*

6.5.1 Primal Information

There are many ways to leverage primal information using the company's history, the previously optimal solution $ꝗ^*$, and the original problem formulation.

Fixing

Fixing is the first option to leverage primal information. For unaffected (by perturbation(s)) vessels $v \in \mathcal{V} \setminus \mathcal{V}^{del} \cup \mathcal{V}^{adv}$, it is possible to fix them as per the previously optimal solution $ꝗ^*$. In such a case, the corresponding term in the objective function becomes constant and we can remove constraints (6.3). Constraints (6.4) and (6.5) are written as follows:

$$\sum_{v \in \mathcal{V}^{del}} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t \leq i_2}} \mathbf{L}_v q_i^{del} + \sum_{v \in \mathcal{V}^{adv}} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t \leq i_2}} \mathbf{L}_v (q_i^{ini} + q_i^{adv}) \leq \mathbf{L}_k - \sum_{v \in \mathcal{V} \setminus \mathcal{V}^{del} \bigcup \mathcal{V}^{adv}} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t \leq i_2}} \mathbf{L}_v ꝗ_i^* \qquad \forall k \in \mathcal{K}, t \in \mathcal{T} \tag{6.7}$$

$$\sum_{v \in \mathcal{V}^{del} \bigcup \mathcal{V}^{adv}, p \in \mathcal{P}_v} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t' \leq i_2}} \sum_{t'=t}^{t+6} \mathbf{Q}_v^p q_i \leq \mathbf{MAX} - \sum_{v \in \mathcal{V} \setminus \mathcal{V}^{del} \bigcup \mathcal{V}^{adv}, p \in \mathcal{P}_v} \sum_{\substack{i \in \mathcal{I}_{vk} \\ i_1 \leq t' \leq i_2}} \sum_{t'=t}^{t+6} \mathbf{Q}_v^p ꝗ_i^* \; \forall k \in \mathcal{K}, t \in \mathcal{T}^- \tag{6.8}$$

Fixing allows alleviating the model by eliminating a significant portion of binary variables q_i as well as $|\mathcal{V} \setminus \mathcal{V}^{del} \cup \mathcal{V}^{adv}|$ constraints.

Warm-start

Warm-starting is the second option to leverage primal information. For advanced vessels, they can be warm-started using their optimal assignment in $ꝗ^*$.

Proposition 7. *Under the fixing option,* $q_i^0 = \begin{cases} 0 & i \in \mathcal{I}_v, v \in \mathcal{V}^{del} \\ 0 \textit{ or } ꝗ_i^* & i \in \mathcal{I}_v, v \in \mathcal{V}^{adv} \end{cases}$ *is a feasible solution to Problem (Re-Opt).*

If fixing is not applied, the unaffected vessels can also be warm-started.

Proposition 8. *Without the fixing option,* $q_i^0 = \begin{cases} 0 & i \in \mathcal{I}_v, v \in \mathcal{V}^{del} \\ 0 \text{ or } \cancel{q}_i^* & i \in \mathcal{I}_v, v \in \mathcal{V}^{adv} \\ \cancel{q}_i^* & i \in \mathcal{I}_v, v \in \mathcal{V} \setminus \mathcal{V}^{del} \cup \mathcal{V}^{adv} \end{cases}$ *is a feasible solution to Problem (Re-Opt).*

Warm-starting allows for accelerating the solving process.

Valid Inequalities

Valid inequalities are the third option to leverage primal information. They allow the strengthening of the model and obtaining tighter relaxations. For our (Re-Opt) model, we add the following valid inequalities and update set Ω accordingly:

$$\sum_{i \in I_{vk}} q_i^{del} \leq 1 \qquad \forall v \in \mathcal{V}^{del}, k \in \mathcal{K} \tag{6.9}$$

$$\sum_{i \in I_{vk}} q_i^{ini} + q_i^{adv} \leq 1 \qquad \forall v \in \mathcal{V}^{adv}, k \in \mathcal{K} \tag{6.10}$$

$$\sum_{i \in I_{vk}} q_i \leq 1 \qquad \forall v \in \mathcal{V} \setminus \mathcal{V}^{del} \cup \mathcal{V}^{adv}, k \in \mathcal{K} \tag{6.11}$$

$$\sum_{\substack{i \in I_v \\ i_1 \leq t \leq i_2}} q_i^{del} \leq 1 \qquad \forall v \in \mathcal{V}^{del}, t \in \mathcal{T} \tag{6.12}$$

$$\sum_{\substack{i \in I_v \\ i_1 \leq t \leq i_2}} q_i^{ini} + q_i^{adv} \leq 1 \qquad \forall v \in \mathcal{V}^{adv}, t \in \mathcal{T} \tag{6.13}$$

$$\sum_{\substack{i \in I_v \\ i_1 \leq t \leq i_2}} q_i \leq 1 \qquad \forall v \in \mathcal{V} \setminus \mathcal{V}^{del} \cup \mathcal{V}^{adv}, t \in \mathcal{T} \tag{6.14}$$

Constraints (6.9) are a decomposition of Constraints (6.1) to quays. Constraints (6.10) are a decomposition of Constraints (6.2) to quays. Constraints (6.11) are a decomposition of Constraints (6.3) to quays. Constraints (6.12) are a decomposition of Constraints (6.1) by periods and vessels. Constraints (6.13) are a decomposition of Constraints (6.2) by periods and vessels. Constraints (6.14) are a decomposition of Constraints (6.3) by periods and vessels. Figure 6.5 illustrates these valid inequalities. In the original formulation, the constraints cover all the periods and quays (area in orange). The valid inequalities ensure the decomposition by period (vertical in yellow) using constraints (6.9) to (6.11) and the decomposition by vessel (horizontal in yellow) using constraints (6.12) to (6.14).

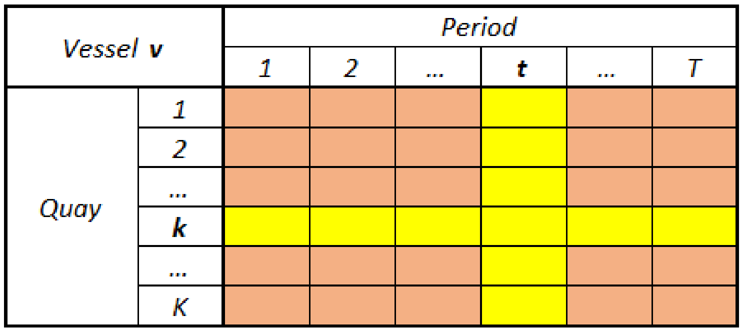


Figure 6.5 Valid Inequalities Illustration for Vessel v

Machine Learning

ML is the fourth option to leverage primal information and has been used intensively recently in OR [240]. It consists of learning from the company's history (e.g., the pool of optimal schedules, weather history in the port, and perturbations history). It also supports the capturing of hidden trends that can help in making the solving approach *quicker*. In this section, we present the target, the data, the features, and the network structure qualitatively. All experiments are kept for Section 6.7.

Target. The goal of our ML model can be stated as follows: Given a vessel $v \in \mathcal{V}^{del} \cup \mathcal{V}^{adv}$, estimate the probability y_{vk} that vessel v is assigned to a quay $k \in \mathcal{K}$. Given such probabilities, we can select the top κ-quays for each vessel $v \in \mathcal{V}^{del} \cup \mathcal{V}^{adv}$, κ being an integer parameter. The role of the ML model is alleviating model (Re-Opt) by selecting only the *promising* quays for each $v \in \mathcal{V}^{del} \cup \mathcal{V}^{adv}$ and thus reducing the number of binary q variables significantly. For a given vessel $v \in \mathcal{V}^{del} \cup \mathcal{V}^{adv}$, the target of our ML model is constructing a vector $Y_v \in \mathbb{R}^{|\mathcal{K}|}$ such that $Y_{v_k} = y_{vk}$. Then, ranking the elements of vector Y_v in decreasing order will allow us to extract the top κ-quays for vessel v.

Data & Features. Throughout the years, the OCP Group has accumulated several schedules (solutions) where various vessels are assigned to various quays. For many instances, optimal schedules are available, and for some difficult instances, near-optimal schedules are available. These solutions will be used to train, validate, and test the ML model.

Each vessel $v \in \mathcal{V}^{del} \cup \mathcal{V}^{adv}$ and quay $k \in \mathcal{K}$ have a feature vector X_{vk}, which contains the following features:

1. For each product $p \in \mathcal{P}$:

   (a) A binary feature indicating whether vessel v contains product p, denoted $f^{(a)}_{vp}$.

   (b) A numerical feature indicating the quantity of product p required by vessel v, denoted $f^{(b)}_{vp}$.

2. A numerical feature corresponding to the earliest arrival period of vessel v, denoted $f^{(2)}$.
3. A numerical feature corresponding to the latest arrival period of vessel v, denoted $f^{(3)}$.
4. A numerical feature corresponding to the average loading time of vessel v, denoted $f^{(4)}$.
5. A numerical feature corresponding to the ratio of vessel v length to quay k length, denoted $f^{(5)}$.
6. A categorical feature corresponding to the destination of vessel v, denoted $f^{(6)}$.

These features contain all the information related to a given vessel $v \in \mathcal{V}$ and quay $k \in \mathcal{K}$.

Network Structure. Each feature vector (entry) has a relatively large number of features. Furthermore, we did not find a strong correlation between any single feature and the target, suggesting that achieving a high prediction accuracy may require an ML model that can combine the features in a non-trivial way. Neural networks are known to perform well with entries containing many features. A neural network is composed of several neurons (also called perceptions) arranged in layers [241]. The first layer is called the input layer, and each neuron of this layer represents one feature. The last layer is called the output layer and holds the prediction $y_{vk}, k \in \mathcal{K}$ for $v \in \mathcal{V}$. A neural network may also contain one or several intermediate layers, called hidden layers, in which case it is called a deep neural network (DNN). A DNN has more layers that allow it to learn more complex relationships between the inputs and outputs. A DNN can also take as many features as possible [242]. Thus, we used a DNN to leverage the primal information. The neurons of one layer are generally connected to neurons of the next layer. When predicting an entry, the neurons of the input layer are initialized to the value of the entry's features. Those values are then propagated throughout the network to the output layer. Each neuron computes a weighted sum of its inputs and applies to the result an activation function, which introduces non-linearity in the model. This value is then transmitted to the neurons of the next layer. The weights and the parameters of the activation functions are adjusted in a training phase to achieve the best accuracy. We, therefore, train a DNN on the task of predicting the target of a new entry.

Following the four options that allow leveraging the primal information, we present next the re-optimization approach.

6.5.2 Re-optimization Approach

By leveraging the primal information to alleviate, strengthen, and warm-start model (Re-Opt), the goal of this section is to find the new optimal schedule(s), i.e.:

$$q^*_{new} = \arg\max_{q \in \Omega} R(q)$$

Before going to the solution, we introduce some relevant definitions. The first definition (Definition 4) highlights SCR mathematically. Since the resilience formula is a bi-objective function, the second definition (Definition 5) highlights the notion of a Pareto-optimal schedule.

Definition 4. *Let $TF(\cancel{q}^*)$ be the updated total fulfillment after the removal of all vessels $v \in \mathcal{V}$ delayed beyond the scheduling horizon. A supply chain is said resilient if there exists a schedule $\bar{q}$ such that $TF(\bar{q}) = TF(\cancel{q}^*)$ and $|\Delta D|$ is minimal (i.e., if $|\Delta D'| < |\Delta D|$ then $TF'(\bar{q}) < TF(\cancel{q}^*)$). We refer to schedule $\bar{q}$ as a resilient schedule.*

Definition 5. *If for any positive weights α_1 and α_2 such that $\alpha_1 + \alpha_2 = 1$, there exists a schedule $\bar{q} \in \Omega$ with the property:*

$$R(q) \leq R(\bar{q}) \;\; \forall q \in \Omega$$

Then schedule $\bar{q}$ is a Pareto-optimal solution for model (Re-Opt).

Next, we show that any schedule $\bar{q}$ achieving SCR is Pareto-optimal.

Proposition 9. *A resilient schedule $\bar{q}$ is a Pareto-optimal schedule.*

We refer to this schedule as the resilient schedule. The inverse is not necessarily correct since the choice of weights may generate non-resilient schedules. For instance, when choosing $\alpha_1 = 0$ and $\alpha_2 = 1$, the Pareto-optimal schedule will minimize the distance without fulfilling delayed vessels. To find a resilient schedule $\bar{q}$, the bi-objective function weights α_1 and α_2 must be tuned. Since model (Re-Opt) is convex (TF, ΔD, and Ω with relaxed Constraints 6.6 are convex), there exist appropriate positive weights, as suggested by the following theorem (Wierzbicki, 1986):

Theorem 6. *If $R(\Omega)$ is convex and $\bar{q}$ is Pareto-optimal for model (Re-Opt), then there exist positive weights α_1 and α_2 with the property:*

$$\alpha_1 \times TF(q) + \alpha_2 \times \Delta D(q) \leq \alpha_1 \times TF(\bar{q}) + \alpha_2 \times \Delta D(\bar{q}) \;\; \forall q \in \Omega$$

After finding appropriate weights and since we seek *quick* solving, we use the incremental large neighborhood search ($\mathcal{ILNS}$) metaheuristic of [2]. Briefly, $\mathcal{ILNS}$ takes a (Re-Opt) instance and iterates over four steps: the *Vessel Assignment* step, the *Problem Reduction* step, the *Solving* step, and the *Wrap-up* step. In the *Vessel Assignment* step, after partitioning the re-optimization time horizon into smaller time intervals (e.g., weeks), we assign each vessel to a time interval. Using these assignments, we reduce further the pool of binary variables related to vessel assignment in the *Problem Reduction* step. Then, we solve the reduced problem in the *Solving* step. We keep iterating over the time horizon, using previous solutions as a warm-start until completion. Before returning a solution, in case there are still unfulfilled vessels, we try to fulfill them in the *Wrap-up* step. Compared to the standard LNS, which does not work efficiently for very large-scale optimization problems, $\mathcal{ILNS}$ destroys only the part of the solution that can be improved. This is because we do not have time for backtracking in such a huge MILP problem. Thus, we break the problem down, solve it, fix part of the solution, and move forward to improve that solution further. Further details are available in [2].

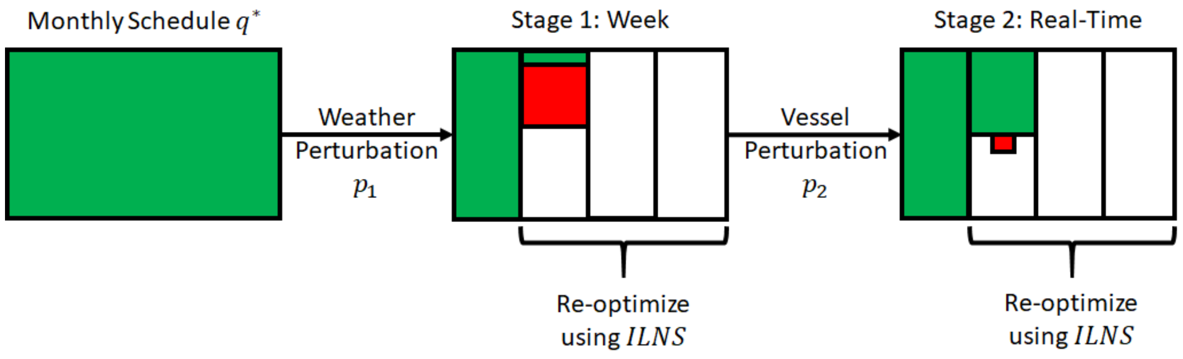


Figure 6.6 2-stage $\mathcal{ILNS}$

We illustrate the usage of $\mathcal{ILNS}$ in Figure 6.6. Starting from a monthly schedule q^* in green, we call $\mathcal{ILNS}$ on a two-stage approach. The first stage is the re-optimization after a weather perturbation p_1 (weekly basis). The second stage is the re-optimization after a vessel perturbation p_2 (real-time). For both stages, the schedule before the perturbation is maintained. The re-optimization using the two-stage $\mathcal{ILNS}$ is conducted on the periods that start from the perturbation period and continue until the end of the scheduling horizon. This is done by calling iteratively and when applicable, the $\mathcal{ILNS}$ just before the week begins for the weather perturbation p_1 and then whenever a vessel perturbation p_2 happens during the week. In what follows, we refer to the proposed approach as the 2-stage $\mathcal{ILNS}$.

6.6 CASE EXPERIMENTAL DESIGN

To study whether the proposed approach is computationally efficient, we complement the analysis presented in previous sections with an extensive computational study. In this section, we describe the general characteristics of the test instances, the machine learning model (quantitatively), the computational setting, and the implementation details.

6.6.1 Instances

We consider six real instances provided by the company. The features of these instances, including the time horizon (*Horizon*) in days, the number of vessels (*Vessels*), the total demand (*Demand*) in tonne, and the total fulfillment (~~TF~~*) corresponding to solution ~~q~~* of each instance are presented in Table 6.3. For these instances, we know the optimal values as well as the optimal solutions.

Table 6.3 Instances

Name	Horizon	Vessels	Demand	~~TF~~*
I_1	32	54	806360	92.52%
I_2	32	58	826460	92.65%
I_3	32	39	1044550	93.34%
I_4	24	40	1043330	95.68%
I_5	32	61	1066290	93.10%
I_6	31	91	1304370	99.08%
Avg	31	57	1015227	94.40%

These instances will be perturbed according to three scenarios: perturbations p_1 alone, perturbations p_2 alone, and perturbations p_1 and p_2. From each instance above, we generate three perturbed instances. The first one has only p_1-type perturbations. The second one has only p_2-type perturbations. The third one has both p_1-type and p_2-type perturbations. The way we generate these instances is as follows. For p_1-type perturbations, we perturb a given % of periods with the scheduling horizon. For p_2-type perturbations, we perturb a given % of the considered vessels. For p_1-type and p_2-type perturbations, we perturb both periods and vessels. We refer to the given % as the perturbation percentage. An example is provided below.

Example 6. *Let us consider an instance of a 2-week time horizon with ten vessels, two quays, and both p_1-type and p_2-type perturbations. With a perturbation percentage of 10% for both types, two periods out of 14 are randomly perturbed. Similarly, on the ten vessels, one vessel is* *randomly perturbed. This is visualized in red on the schedule in Figure 6.9. After perturbing periods 4 and 12, vessels with non-empty intersections with these two periods (i.e., v_2, v_4, v_6, v_8, and v_{10}) must be rescheduled. Similarly, for p_2-type perturbations, delayed vessel v_3 must also be rescheduled.*

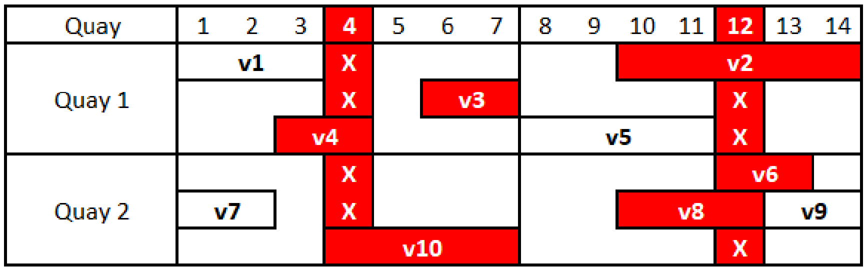


Figure 6.7 Example of a perturbed schedule

The re-optimizations are conducted in the order of appearance of perturbations following the 2-stage $\mathcal{ILNS}$ in Figure 6.6. Thus, starting with p_1-type perturbation in period 4, we reschedule vessels v_4 and v_{10}. This is the first re-optimization. Then, we re-optimize for a second time to reschedule v_3 (after p_2-type perturbation). Lastly, we re-optimize for a third time to reschedule vessels v_2, v_6, and v_8. The schedule obtained after completing all re-optimizations is referred to as the final schedule.

6.6.2 Machine Learning Model

As mentioned in the previous section, we use a ML model to construct Y_v for each vessel $v \in \mathcal{V}$. Our ML model is a DNN trained in a supervised training framework using data from several optimal solutions. To avoid confusion, we refer to these solutions as reference solutions. Each reference solution $\bar{q}$ from the history assigns several vessels to various quays. Thus, from a solution $\bar{q}$ with $|V_{\bar{q}}|$ vessels assigned, we can extract $|V_{\bar{q}}|$ pairs (X_{vk}, y_{vk}) where $v \in \mathcal{V}_{\bar{q}}$ and $k \in \mathcal{K}$. Furthermore, we can leverage the symmetry of the problem and collect all equivalent optimal reference solutions. Thus, we may have several possible assignments for each vessel $v \in V_{\bar{q}}$. Each X_{vk} is given a label y_{vk}. This label is computed as the frequency at which vessel $v \in V_{\bar{q}}$ is assigned to quay $k \in \mathcal{K}$ in all the considered reference solutions.

We train a DNN on the pool of reference solutions to predict the label of a new vector X_{vk}. We note y_{vk}^{prd} the predicted label of entry X_{vk}. The quay ranking is obtained by ordering the quays in decreasing order of y_{vk}^{prd}. Then, we can select the top κ-quays. The pairs (X_{vk}, y_{vk}) are split randomly into three disjoint subsets: a training set (60% of pairs), a validation set (20% of pairs), and a test set (20% of pairs).

The neural network is a feedforward fully connected DNN. Conforming with standard practices, the number of neurons in any hidden layer is less than or equal to that of the previous layer. All neurons except those of the input and output layers are rectilinear (ReLU) units. The output layer is composed of a single sigmoid unit to ensure that the output is in $[0, 1]$. The hyperparameters and their possible values are given in Table 6.4. We determine the appropriate hyperparameters of the neural network and the training algorithm by performing a random grid search over the hyperparameter space. We select the model that achieves the best validation performance based on a performance indicator presented next.

Table 6.4 ML Model Hyperparameters

Name	Range	Type
Training algorithm	{SGD, Adam}	Categorical
Number of hidden layers	{2, 3, 4}	Integer
Neurons per layer	{10, 50, 150,..., 500}	Integer
Learning Rate	{0.0001, 0.001, 0.01, 0.01}	Float
Dropout	{0.1, 0.2, 0.3, 0.4}	Float

A good performance indicator of the ML model can be obtained by taking the sum of the *true* labels in the top κ-quays in Y_v for all vessels. We use variable $j \in \{1, 2, ..., \kappa\}$ to refer to the ordered probabilities y_{vk_j} with y_{vk_1} being the top one and y_{vk_κ} the κ^{th} one. We measure the performance of our ML model by computing the label sum ratio in the top κ as follows:

$$TOP_\kappa = \frac{\sum_{v \in \mathcal{V}} \sum_{j=1}^{\kappa} y_{vk_j}}{\sum_{v \in \mathcal{V}} \sum_{k \in \mathcal{K}} y_{vk}}$$

Training is performed in a supervised fashion using either the stochastic gradient descent (SGD) [243] or the Adam algorithm [244]. Several strategies are used to prevent overfitting. The neurons have a dropout probability between 0.1 and 0.4. Also, the validation performance (TOP_κ) is computed every 10 epochs and the training algorithm is stopped when it degrades twice in a row, or after a fixed number of epochs.

Training a neural network took less than 2 hours. Generally, we note that a high training time is not an issue for real-world applications because each ML model would be trained once and then used for several months or years. Finally, retraining a neural network is significantly faster than training one from scratch (minutes instead of hours) because the neural network is already in a near-optimal state and only needs to be slightly adjusted. The ML model performance is presented in Table 6.5. We note that, in our case, the loading port contains four quays. Thus, κ takes values from 1 to 4.

Table 6.5 Average TOP_κ for the Test Pairs

κ	1	2	3	4
Average TOP_κ	55.28%	95.73%	97.62%	100%

The ML model above will be used to alleviate the (Re-Opt) model. It is implemented in Python using the PyTorch library. All experiments were performed on a 40-core machine with 384 GB of memory.

6.6.3 Computational Setting and Implementation Details

For the optimization part, the coding language is C++ and tests are conducted using version 12.10.0 of IBM ILOG CPLEX solver. All experiments were carried out on a 3.20GHz Intel(R) Core(TM) i7-8700 processor, with 64GiB System memory, running on Oracle Linux Server release 7.7. We use real-time to measure runtime.

We compare the following three methods:

- $\mathcal{ILNS}$: Solve the *Original Problem* in [2] from scratch using the $\mathcal{ILNS}$ metaheuristic as described in [2].
- MILP: Solve directly model (Re-Opt) with a MIP solver. In our case, we use the default CPLEX.
- 2-stage $\mathcal{ILNS}$: Solve using the proposed approach in Section 7.5.

The reason behind this choice of methods is the novelty of the problem, which was not tackled in the literature before, except in [2]. Thus, to measure the solving performance, we compare our approach with the default CPLEX. Furthermore, given that we also want to compare the difference between optimizing from scratch (solving *Original Problem*) and re-optimizing from a given solution, we compare with the $\mathcal{ILNS}$ in [2].

6.7 CASE COMPUTATIONAL RESULTS

In this section, we first find appropriate weights α_1 and α_2. Then, we compare the performance of the 2-stage $\mathcal{ILNS}$ against $\mathcal{ILNS}$ and MILP. After that, we conduct sensitivity analysis. We conclude with managerial insights.

6.7.1 Weights Tuning

To find appropriate weights α_1 and α_2, we conduct a sensitivity analysis on instance I_1 with 10% p_1-type perturbations and 10% p_2-type perturbations. Since $\alpha_1 + \alpha_2 = 1$, we consider α_1 for the dichotomic search.

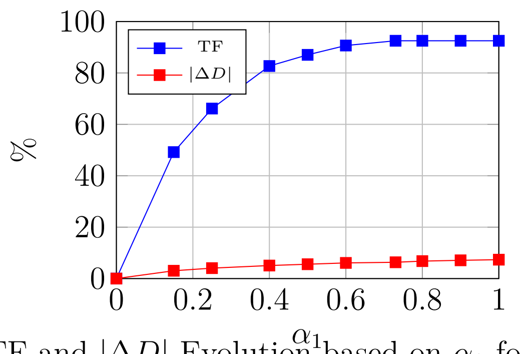


Figure 6.8 TF and $|\Delta D|$ Evolution based on α_1 for Instance I$_1$

Figure 6.8 shows the results for instance I$_1$, where $\alpha_1 = 0.73$ ensures the highest TF with the lowest $|\Delta D|$ possible. Given that the instances are representative of the same problem, the suitable weights for I_1 are likely to be suitable for all instances. To confirm, we conducted a sensitivity analysis for other instances as well. Table 6.6 shows the results obtained.

Table 6.6 Weight α_1 Sensitivity Analysis

Inst	α_1	0.00	0.15	0.25	0.40	0.50	0.60	0.73	0.80	0.90	1.00
I$_1$	TF	0.00%	49.20%	66.15%	82.69%	87.04%	90.67%	**92.52%**	92.52%	92.52%	92.52%
	$\lvert\Delta D\rvert$	0.00%	3.04%	4.06%	5.07%	5.58%	6.09%	6.34%	6.79%	7.10%	7.35%
I$_2$	TF	0.00%	49.69%	66.81%	83.52%	87.91%	91.31%	92.03%	**92.65%**	92.65%	92.65%
	$\lvert\Delta D\rvert$	0.00%	3.07%	4.10%	5.12%	5.63%	6.15%	6.44%	6.86%	7.17%	7.43%
I$_3$	TF	0.00%	50.18%	67.48%	84.34%	88.78%	92.48%	92.98%	**93.34%**	93.34%	93.34%
	$\lvert\Delta D\rvert$	0.00%	3.09%	4.11%	5.14%	5.66%	6.17%	6.51%	6.88%	7.20%	7.46%
I$_4$	TF	0.00%	51.17%	68.80%	86.00%	90.52%	94.30%	95.02%	**95.68%**	95.68%	95.68%
	$\lvert\Delta D\rvert$	0.00%	3.32%	4.42%	5.53%	6.08%	6.63%	6.91%	7.40%	7.74%	8.02%
I$_5$	TF	0.00%	50.09%	67.34%	84.18%	88.61%	92.30%	**93.10%**	93.10%	93.10%	93.10%
	$\lvert\Delta D\rvert$	0.00%	3.42%	4.56%	5.69%	6.26%	6.83%	6.98%	7.62%	7.97%	8.26%
I$_6$	TF	0.00%	61.50%	79.38%	89.31%	91.39%	92.48%	95.30%	**99.08%**	99.08%	99.08%
	$\lvert\Delta D\rvert$	0.00%	3.55%	4.74%	5.92%	6.51%	7.11%	7.15%	7.93%	8.29%	8.59%

As it can be observed, for all instances, the best α_1 value lies in the interval $[0.7, 0.8]$. Within this interval, all instances reach for the first time their optimal values. In what follows, we consider $\alpha_1 = 0.75$.

6.7.2 Performance Comparison

We compare $\mathcal{ILNS}$, MILP, and 2-stage $\mathcal{ILNS}$. We report in Table 6.7 the perturbation percentage for both types (p_1 and p_2), the previously optimal TF (~~TF~~*), the optimal TF obtained by each method (TF*), the optimal $|\Delta D|$ obtained by each method (ΔD^*), the optimal resilience (R^*), and the average time to re-optimize after each perturbation ($Time$) in seconds. For perturbations, we consider three scenarios with 10% as a percentage as explained in Section 6.6.

Table 6.7 Performance Comparison

p_1	p_2	$Inst$	~~TF~~*	$\mathcal{ILNS}$				MILP				2-stage $\mathcal{ILNS}$			
				TF^*	ΔD^*	R^*	$Time$	TF^*	ΔD^*	R^*	$Time$	TF^*	ΔD^*	R^*	$Time$
		I_1	92.52%	92.52%	53.60%	55.99%	518	92.52%	10.72%	66.71%	140	92.52%	10.72%	66.71%	10
		I_2	92.65%	92.65%	54.67%	55.82%	534	92.65%	10.93%	66.75%	168	92.65%	10.93%	66.75%	11
		I_3	93.34%	93.34%	55.77%	56.06%	550	93.34%	11.15%	67.22%	202	93.34%	11.15%	67.22%	13
10%	-	I_4	95.68%	95.68%	56.88%	57.54%	605	95.68%	11.38%	68.92%	242	95.68%	11.38%	68.92%	15
		I_5	93.10%	93.10%	62.28%	54.25%	635	93.10%	12.46%	66.71%	278	93.10%	12.46%	66.71%	16
		I_6	99.08%	99.08%	59.45%	59.45%	673	99.08%	11.89%	71.34%	305	99.08%	11.89%	71.34%	19
		Avg	**94.40%**	**94.40%**	**57.11%**	**59.45%**	**586**	**94.40%**	**11.42%**	**67.94%**	**222**	**94.40%**	**11.42%**	**67.94%**	**14**
		I_1	92.52%	92.52%	31.70%	61.47%	432	92.52%	6.34%	67.81%	98	92.52%	6.34%	67.81%	7
		I_2	92.65%	92.65%	32.20%	61.44%	454	92.65%	6.44%	67.88%	108	92.65%	6.44%	67.88%	8
		I_3	93.34%	93.34%	32.55%	61.87%	476	93.34%	6.51%	68.38%	119	93.34%	6.51%	68.38%	10
-	10%	I_4	95.68%	95.68%	33.85%	63.30%	500	95.68%	6.77%	70.07%	142	95.68%	6.77%	70.07%	11
		I_5	93.10%	93.10%	34.90%	61.10%	525	93.10%	6.98%	68.08%	185	93.10%	6.98%	68.08%	13
		I_6	99.08%	99.08%	35.75%	65.37%	561	99.08%	7.15%	72.52%	262	99.08%	7.15%	72.52%	16
		Avg	**94.40%**	**94.40%**	**33.49%**	**62.42%**	**491**	**94.40%**	**6.70%**	**69.12%**	**152**	**94.40%**	**6.70%**	**69.12%**	**11**
		I_1	92.52%	92.52%	52.54%	56.26%	475	92.52%	15.01%	65.64%	119	92.52%	15.01%	65.64%	8
		I_2	92.65%	92.65%	54.64%	55.83%	494	92.65%	15.61%	65.58%	138	92.65%	15.61%	65.58%	9
		I_3	93.34%	90.54%	57.37%	53.56%	513	90.54%	16.39%	63.81%	160	90.54%	16.39%	63.81%	11
10%	10%	I_4	95.68%	93.77%	59.63%	55.42%	552	93.77%	17.54%	65.94%	192	93.77%	17.54%	65.94%	13
		I_5	93.10%	90.31%	61.35%	52.39%	580	90.31%	18.59%	63.08%	232	90.31%	18.59%	63.08%	15
		I_6	99.08%	97.10%	64.14%	56.79%	617	97.10%	21.38%	67.48%	284	97.10%	21.38%	67.48%	17
		Avg	**94.40%**	**92.81%**	**58.28%**	**55.04%**	**538**	**92.81%**	**17.42%**	**65.26%**	**187**	**92.81%**	**17.42%**	**65.26%**	**12**

As observed in Table 6.7, the three methods reach the optimal TF value, which is equal to the previously optimal value, except for instances I_2, I_3, I_4, and I_5 for which the optimal TF value decreases slightly under the scenario (10%,10%), i.e., 10% p_1-type perturbation and 10% p_2-type perturbation. Under this scenario, it becomes difficult to fulfill all vessels within the given time horizon and thus TF decreases. For the $Time$, the 2-stage $\mathcal{ILNS}$ outperforms both $\mathcal{ILNS}$ and MILP by factors of 44 and 15 on average, respectively. The $Time$ is affected more by p_1-type perturbations than p_2-type perturbations because the former involves more vessels for rescheduling at each re-optimization. On the resilience aspect, both MILP and 2-stage $\mathcal{ILNS}$ (which obtain the same results) outperform $\mathcal{ILNS}$. The latter re-optimizes the original problem in [2] from scratch without considering the resilience aspect. Thus, while the same optimal TF value is reached, the number of changes to the previously optimal schedule is larger. This is due to the changes in the production schedule, which is impacted by the perturbation. This change impacts significantly the vessels.

To analyze further the performance, we compare the schedules obtained by $\mathcal{ILNS}$ and 2-stage $\mathcal{ILNS}$ with the previously optimal schedule $\not{q}^*$. To do so, we consider instance I_1 under scenario (10%,10%) in Table 6.7. Figure 6.9 shows the previously optimal schedule with all perturbations in red.

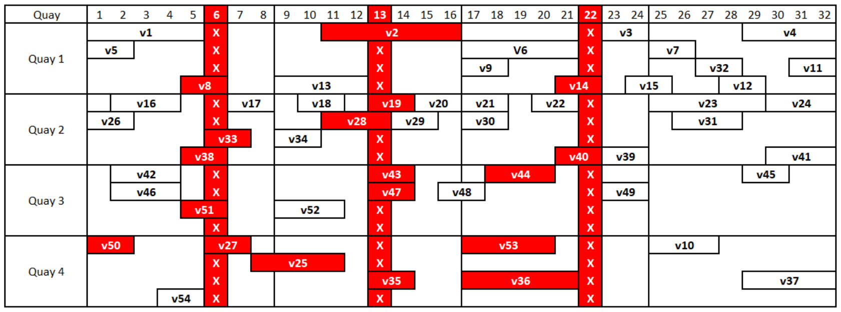


Figure 6.9 Perturbed Instance I_1 under scenario (10%,10%)

Figure 6.10 shows the final schedule obtained using the $\mathcal{ILNS}$ method. All the vessels in green were rescheduled to take into consideration all perturbations. As can be seen, when re-optimizing from scratch, several unperturbed vessels are re-scheduled as well. Of all 54 vessels, 22 vessels only remained as previously planned.

Figure 6.11 shows the final schedule obtained when re-optimizing while considering the resilience aspect. We observe that the 2-stage $\mathcal{ILNS}$ re-schedules only the perturbed vessels. Furthermore, it keeps several vessels close to their original schedule. For instance, vessel v_{27} in Quay 4 was moved to periods 7 and 8 from periods 6 and 7 on the same quay, i.e., the

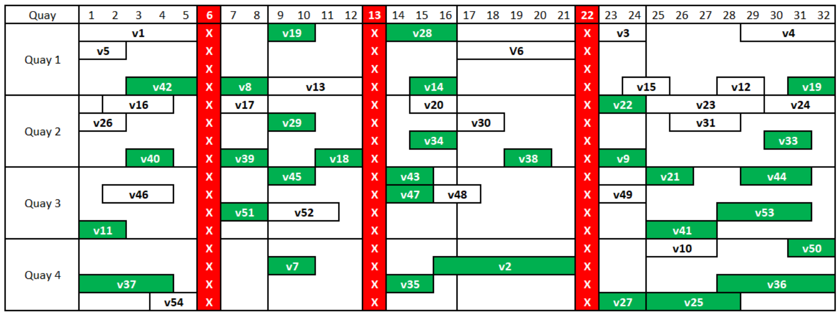


Figure 6.10 Final Schedule obtained by $\mathcal{ILNS}$ for Perturbed Instance I_1 under scenario (10%,10%)

period following the bad weather period. Out of all 54 vessels, only the 18 vessels perturbed were rescheduled.

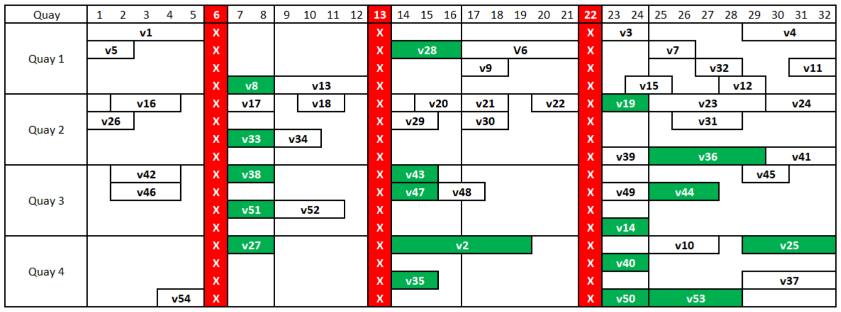


Figure 6.11 Final Schedule obtained by 2-stage $\mathcal{ILNS}$ for Instance I_1 under scenario (10%,10%)

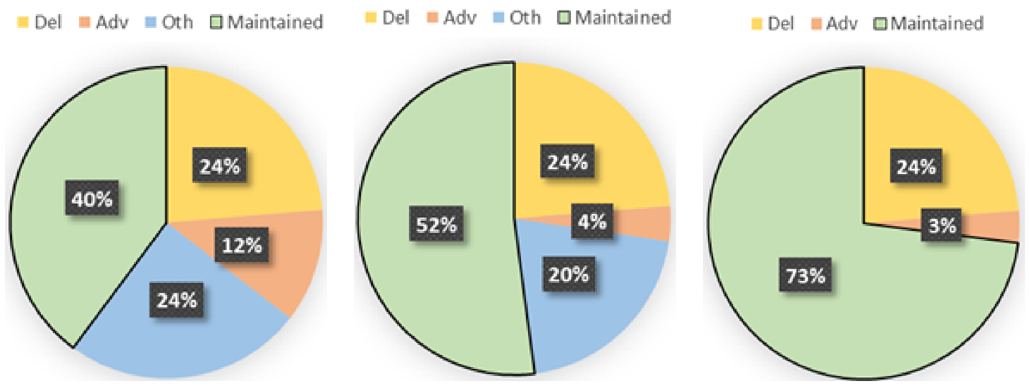
When observing the schedules in Figures 6.9, 6.10, and 6.11, one can infer that, when dealing with perturbations, optimizing locally can be more relevant than optimizing globally to achieve resilience, especially when the optimal value can be reached and the global problem is a large-scale optimization problem. Local optimization explores just the affected part of the mathematical model and tries to correct it while global optimization might correct the affected part of the mathematical model while making changes to unaffected parts. In our case, when re-optimizing from scratch, $\mathcal{ILNS}$ changes the production schedule, which in turn affects more vessels beyond the perturbed ones.

Table 6.8 shows the number of delayed (Del), advanced (Adv), and other (Oth) vessels rescheduled in the final schedules compared to the initial ones. As it can be observed, the 2-stage $\mathcal{ILNS}$ reschedules mainly the affected vessels. In tight scenarios, it reschedules advanced vessels to free room for delayed ones. The other vessels are kept as initially scheduled

Table 6.8 Number of Delayed, Advanced, and Other Vessels rescheduled in the Final Schedule

p_1	p_2	Inst	Horizon	Vessels	$\mathcal{ILNS}$			MILP			2-stage $\mathcal{ILNS}$		
					Del	Adv	Oth	Del	Adv	Oth	Del	Adv	Oth
		I_1	32	54	13	5	9	13	1	11	13	0	0
		I_2	32	58	14	4	9	14	2	12	14	0	0
		I_3	32	39	11	3	11	11	0	8	11	0	0
10%	-	I_4	24	40	12	6	12	12	3	8	12	0	0
		I_5	32	61	15	11	18	15	4	12	15	0	0
		I_6	31	91	18	15	27	18	5	18	18	0	0
		Avg	**31**	**57**	**14**	**7**	**14**	**14**	**1**	**11**	**14**	**0**	**0**
		I_1	32	54	5	4	8	5	0	9	5	0	0
		I_2	32	58	6	3	8	6	0	9	6	0	0
		I_3	32	39	4	2	10	4	0	6	4	0	0
-	10%	I_4	24	40	4	5	10	4	1	6	4	0	0
		I_5	32	61	6	9	15	6	2	10	6	0	0
		I_6	31	91	9	12	22	9	4	15	9	0	0
		Avg	**31**	**57**	**6**	**6**	**12**	**6**	**1**	**9**	**6**	**0**	**0**
		I_1	32	54	18	4	10	18	0	14	18	0	0
		I_2	32	58	22	8	10	22	5	15	22	7	0
		I_3	32	39	17	0	13	17	3	10	17	0	0
10%	10%	I_4	24	40	18	4	13	18	3	10	18	4	0
		I_5	32	61	23	11	19	23	5	16	23	9	0
		I_6	31	91	30	15	28	30	7	24	30	14	0
		Avg	**31**	**57**	**21**	**7**	**16**	**21**	**4**	**15**	**21**	**6**	**0**

because of the fixing strategy. On the other side, $\mathcal{ILNS}$ reschedules significantly more vessels (Oth) unaffected by perturbations and advanced vessels. The same observation holds for MILP because there is no fixing.



(a) $\mathcal{ILNS}$ (b) MILP (c) 2-stage $\mathcal{ILNS}$

Figure 6.12 Average Percentage of Vessels Delayed, Advanced, and Others rescheduled in the Final Schedule(s). Maintained Vessels are Vessels kept as per the Previously Optimal Solution(s).

We show in Figure 6.12 the average percentage of vessels rescheduled by each of the methods in the final schedule(s). It includes delayed, advanced, and other vessels. The vessels

Maintained are the ones that were maintained as per the initial schedule(s). Figure 6.12 highlights that the percentage of maintained vessels varies significantly among the three methods. When optimizing globally and considering the original problem, production is rescheduled leading to the rescheduling of around 60% of the vessels. The MILP reschedules the delayed vessels and some vessels that can be advanced. Still, it reschedules other vessels, which are not affected by perturbations. Compared to $\mathcal{ILNS}$, the MILP keeps 12% more vessels as previously planned. For the 2-stage $\mathcal{ILNS}$, only the vessels delayed are rescheduled with some advanced vessels (to free space). This shows, that when optimizing locally with resilience taken into consideration and primal information leveraged, the obtained schedules are closer to the initially planned ones.

6.7.3 Sensitivity Analysis

We conduct a sensitivity analysis on the perturbation percentage. We consider instance I_1 and vary the perturbation percentage from 5% to 25%. Table 6.9 shows the obtained results. We observe that the higher the perturbation percentage, the lower the resilience value. Indeed, with more perturbations, the TF value is more likely to decrease because there is less room to fulfill all vessels, and the ΔD^* value increases because there are more movements. In Table 6.9, the TF^* value decreases starting from a perturbation percentage of 20%.

Table 6.9 Perturbation Percentage impact on Instance I_1

p_1	p_2	~~TF~~*	$\mathcal{ILNS}$			MILP			2-stage $\mathcal{ILNS}$		
			TF^*	ΔD^*	R^*	TF^*	ΔD^*	R^*	TF^*	ΔD^*	R^*
5%	-	92.52%	92.52%	42.88%	58.67%	92.52%	8.58%	67.25%	92.52%	8.58%	67.25%
-	5%	92.52%	92.52%	25.36%	63.05%	92.52%	5.07%	68.12%	92.52%	5.07%	68.12%
5%	5%	92.52%	92.52%	42.03%	58.88%	92.52%	12.01%	66.39%	92.52%	12.01%	66.39%
10%	-	92.52%	92.52%	53.60%	55.99%	92.52%	10.72%	66.71%	92.52%	10.72%	66.71%
-	10%	92.52%	92.52%	31.70%	61.47%	92.52%	6.34%	67.81%	92.52%	6.34%	67.81%
10%	10%	92.52%	92.52%	52.54%	56.26%	92.52%	15.01%	65.64%	92.52%	15.01%	65.64%
15%	-	92.52%	92.52%	64.32%	53.31%	92.52%	12.86%	66.17%	92.52%	12.86%	66.17%
-	15%	92.52%	92.52%	38.04%	59.88%	92.52%	7.61%	67.49%	92.52%	7.61%	67.49%
15%	15%	92.52%	92.52%	63.04%	53.63%	92.52%	18.01%	64.89%	92.52%	18.01%	64.89%
20%	-	92.52%	89.76%	75.04%	48.56%	89.76%	15.01%	63.57%	89.76%	15.01%	63.57%
-	20%	92.52%	90.67%	44.38%	56.91%	90.67%	8.88%	65.78%	90.67%	8.88%	65.78%
20%	20%	92.52%	88.87%	73.55%	48.26%	88.87%	21.01%	61.40%	88.87%	21.01%	61.40%
25%	-	92.52%	87.07%	82.54%	44.67%	87.07%	16.51%	61.18%	87.07%	16.51%	61.18%
-	25%	92.52%	87.95%	48.82%	53.76%	87.95%	9.76%	63.52%	87.95%	9.76%	63.52%
25%	25%	92.52%	86.20%	80.90%	44.42%	86.20%	23.12%	58.87%	86.20%	23.12%	58.87%

We also check the impact of the primal information. To do so, we run additional tests considering no warm-start ($NoWS$), no fixing ($NoFix$), no valid inequalities ($NoVI$), and no machine learning ($NoML$). We report the results with all options (*All*), which are the same as instance I_1 results on Table 6.7, for comparison purposes. While warm-starting and valid inequalities do not impact the $Time$ a lot, the fixing and machine learning strategies impact significantly the $Time$. On average, $Time$ is reduced by a factor of 1.8 and 1.7 per perturbation when using fixing and machine learning options, respectively.

Table 6.10 Primal Information impact on $Time$ for Instance I$_1$

p_1	p_2	*Inst*	~~TF~~*	$NoWS$	$NoFix$	$NoVI$	$NoML$	*All*
		I$_1$	92.52%	11	17	11	16	10
		I$_2$	92.65%	12	19	12	18	11
		I$_3$	93.34%	14	23	14	21	13
10%	-	I$_4$	95.68%	17	27	17	25	15
		I$_5$	93.10%	18	29	18	27	16
		I$_6$	99.08%	20	33	20	31	19
		Avg	**94.40%**	**15**	**25**	**15**	**23**	**14**
		I$_1$	92.52%	7	12	7	11	7
		I$_2$	92.65%	9	14	9	13	8
		I$_3$	93.34%	11	18	11	17	10
-	10%	I$_4$	95.68%	13	21	13	19	11
		I$_5$	93.10%	15	24	15	22	13
		I$_6$	99.08%	18	29	18	27	16
		Avg	**94.40%**	**12**	**20**	**12**	**18**	**11**
		I$_1$	92.52%	9	15	9	14	8
		I$_2$	92.65%	10	17	10	16	9
		I$_3$	93.34%	13	21	13	19	11
10%	10%	I$_4$	95.68%	15	24	15	22	13
		I$_5$	93.10%	16	26	16	25	15
		I$_6$	99.08%	19	31	19	29	17
		Avg	**94.40%**	**14**	**22**	**14**	**21**	**12**

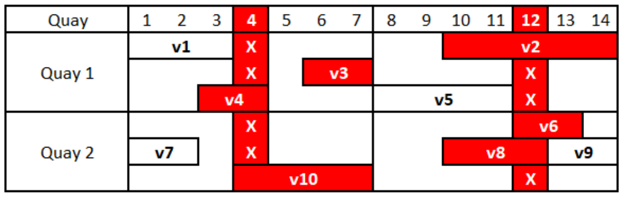
The ML model plays two roles. First, it computes assignment probabilities for each pair vessel and quay. Second, it identifies less promising quays for each vessel. Then, the corresponding q variables are removed from the mathematical model. It is worth mentioning that we could reach TF^* by selecting only the top 2-quays for each vessel. This is equivalent to reaching the optimal value when considering just 50% of the binary assignment variables for each vessel.

6.7.4 Managerial Insights

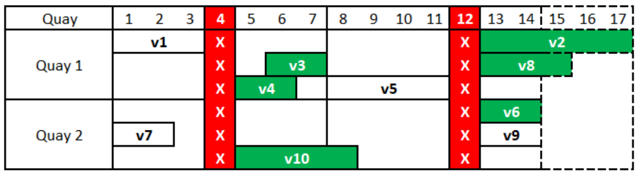
Through re-optimization, the OCP optimization system develops the ability to recover and tends towards more resilience and stability. In other words, this paper presents a way to seek resilience through re-optimization.

Furthermore, as shown above, the 2-stage $\mathcal{ILNS}$ reinforced with primal information ensures both quick re-optimization and resilient schedules. It also has several benefits. The first benefit is from a system standpoint, where the quick re-optimization capability allows the company's stakeholders to simulate various what-if scenarios during their meetings. Indeed, operators can run the optimization of the original problem during the night to have an optimal solution by the morning. Then, during the morning meeting, the stakeholders can perturb this solution and compare several what-if scenarios during the same meeting since the re-optimization is quick. The quick re-optimization provides a decision support system to decision makers (e.g., planners, managers, operators) and allows them to consider various what-if scenarios and cases. The resilient schedules permit to stay as close as possible to the previously planned schedule, thus involving fewer changes and sustaining operators' satisfaction.

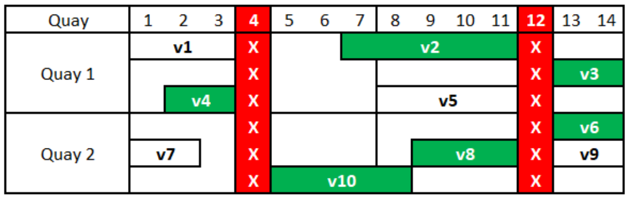
The second benefit is the advantage of local optimization. Indeed, when the perturbation is local, optimizing locally is better than optimizing globally, especially when the former reaches the optimal solution. First, local optimization involves a smaller problem. Second, it changes only the part of the problem affected by the perturbation. Third, it leverages the available primal information.



(a) Perturbed Schedule



(b) Preferred Schedule by Operators



(c) Preferred Schedule by Customers

Figure 6.13 Two Types of Schedules

The third benefit is the schedules' flexibility. We distinguish two types of schedules from which decision-makers can select. The first schedule delays all the perturbed vessels, as shown in the example in Figure 6.13b. Operators prefer this schedule because it involves changes that require less effort. Still, it may imply going beyond the planning time horizon. In Figure 6.13b, three periods are added to fulfill vessels v_2 and v_8. This may decrease customer satisfaction. The second schedule modifies the perturbed vessels and allows both delay and advance. This schedule is less preferred by operators compared to the first schedule because it involves working ahead of schedule as shown in Figure 6.13c. Still, it maintains customer satisfaction and may increase it if customers' vessels are fulfilled earlier than expected (e.g., vessel v_2). It also ensures that the planned vessels are fulfilled within the initial time horizon.

The fourth benefit is the ML support in identifying some hidden trends. For instance, it shows that some products are loaded on specific quays while others are loaded on other quays. An operational explanation is that liquid-type products are loaded on specific quays while solid-type products are loaded on different quays. The problem can then be decomposed by product type.

6.8 CONCLUSION

This research presents a generic and scalable resilience re-optimization framework. We highlight various ways of leveraging the primal information, including fixing, warm-start, valid inequalities, and machine learning. Using a real-world large-scale problem for illustration, we discuss uncertainties, confirm their impact on the model, and model recovery decisions, highlighting the need for resilience. Then, we model the re-optimization problem to maximize resilience. Finally, we conduct extensive computational experiments to demonstrate the power of our proposed framework and solution methodology. The proposed framework is scalable and generic to any large-scale company facing several perturbations and seeking quick re-optimization and resilient solutions. Future work includes extending the framework to tackle disruptions such as earthquakes, tsunamis, and wars, and considering cases where local optimization is no longer enough (suboptimal solutions, infeasible solutions, etc.). In such cases, global optimization becomes a must.

Acknowledgements

This work was supported by the *Fonds de Recherche du Québec - Nature et Technologies* (FRQNT) through a Ph.D. Excellence Scholarship, the *Institute for Data Valorisation* (IVADO) through another Ph.D. Excellence Scholarship, and the *Group for Research in Decision Analysis* (GERAD) through the provision of computing facilities for conducting experiments.

CHAPTER 7 ARTICLE 5: OCP OPTIMIZES ITS SUPPLY CHAIN FOR AFRICA

Authors: El Mehdi Er Raqabi, Ahmed Beljadid, Mohammed Ali Bennouna, Rania Bennouna, Latifa Boussaadi, Nizar El Hachemi, Issmaïl El Hallaoui, Michel Fender, Mohamed Anouar Jamali, Nabil Si Hammou, François Soumis

Submitted on May 9, 2024 to INFOMRS Journal on Applied Analytics

Abstract. Problem definition: Operations research specialists at the OCP Group, the Mohammed VI Polytechnic University, and the Polytechnique Montreal operationalized a system that optimizes the OCP downstream supply chain operations. The system simultaneously schedules production, inventory, and vessels while ensuring the highest demand fulfillment level. **Methodology:** To operationalize the system, the team equipped it with various heuristic and exact operations research tools. These tools provide the user with *satisfactory* schedules. Furthermore, inspired by the practice, the team implemented a novel hybrid variant of Benders decomposition, which consists of fixing some complicating variables related to confirmed orders and freeing others related to unconfirmed orders in the Benders subproblem. **Results and managerial implications:** The system has become central to the OCP planning process. Planners use the optimizer's solutions and insights to improve plans in different OCP sites. Initially, the system was a bottleneck, curbing the use of other supply chain management tools. OCP management now credits the system operationalization with providing operational benefits, contributing to over a $240 million increase in annual turnover.

Keywords. *Large-scale Optimization*, *MILP Solvers*, *Metaheuristics*, *Benders Decomposition*, *Supply Chain Management.*

History. This article is to appear in *INFORMS Journal on Applied Analytics.*

7.1 Introduction

7.1.1 Need for Fertilizers in Africa

By 2100, our population is expected to reach around 10.9 billion (Desa, 2019). With such growth, governments and international organizations must ensure a stable food supply for all of us. Unfortunately, the world is facing an unprecedented food deficit due to rising prices that occurred well before the onset of the Ukraine war and a fertilizer crisis that negatively impacted previous food production seasons. Indeed, half of the world's food production is made possible by the use of mineral fertilizers (Van Kauwenbergh 2010, Cooper et al. 2011). It is, therefore, essential to have sufficient quantities of fertilizers. The rising prices in developing countries have forced them to increase agricultural capacities more than before using fertilizers (Khan et al., 2007). It is particularly true in Africa, where the population will more than double in this century (Desa, 2019). Here, where the shortages are the most obvious and where the most devastating food-threatening events arise, fertilizers are the central part of the equation to increase the food supply (Cordell et al., 2009).

The African continent could potentially become a large market in the long term since, as of today, Africa has more than 65% of the arable land available on the planet with a diversity of agro-ecological zones and climates requiring different fertilizers [249]. This diversity creates vast potential in the combination of agricultural products that can be grown and marketed to the world. Still, Africa is far from the standard in terms of fertilizers (rate). For instance, only 30%-40% of Ethiopian smallholders use fertilizers, with only 37 to 40 kg per hectare, an amount substantially less than the recommended rate of 50 kg per hectare (Mekonnen and Kibret, 2021). This confirms the huge growth potential for fertilizers in Africa.

7.1.2 The Basic Challenge at OCP

Located in northwestern Africa, Morocco holds 70% of the world's phosphate rock reserves (Summaries, 2021), a crucial element for fertilizers' production, giving it a leading role in satisfying our planet's needs. The company in charge of the phosphate industry in Morocco is the OCP Group (formerly *Office Cherifien des Phosphates*), a Moroccan state-owned phosphate rock miner, phosphoric acid manufacturer, and fertilizer producer. Founded in 1920, the company is one of the largest phosphate, fertilizer, chemical, and mineral industrial companies worldwide. The OCP Group has five main sites (Jorf, Khouribga, Safi, Boukraa, and Benguerir).

Aware of its role in increasing the food supply through fertilizers, the OCP Group faces two challenges in its African supply chain: capacity and customization. The first challenge is capacity since OCP Group needs to produce fertilizers in sufficient quantities to remedy food shortages and withstand food shocks in Africa and worldwide. The second challenge is customization since each soil type requires a specific fertilizer type. Furthermore, given the importance of remaining environment-friendly, the OCP Group has been focusing more and more on promoting *precision farming*, i.e., determining the right fertilizer for the right soil (Auernhammer, 2001). While facing these two challenges, the OCP group believes Africa to be the center of solutions to global food security challenges. However, Africa is not a luxury market where farmers can pay for expensive fertilizers, so there is pressure to reduce the price, i.e., the OCP Group has to produce as much customized fertilizers as possible at low cost. While OCP makes donations and discounts to several African countries in need, it also seeks to meet margins by offering a correct price given the African particularity. To accomplish this, and since OCP Group covers, so far, 80% of fertilizer demand in Africa (*OCP Group*, 2021), the Moroccan company is committed to minimizing costs and optimizing supply chain operations [251].

7.1.3 The Specifics and the Need for a New Solution

To optimize operations, the OCP Group started with the downstream side of its supply chain. On the downstream side, the OCP faces a rich problem with two facets. The first facet is production scheduling with inventory management [252]. The second facet is the vessel assignment, which groups berth allocation [253, 254], and quay crane assignment and scheduling problems [255]. In the second facet, there are spatial and temporal dimensions. In the spatial dimension, the company deals with multi-quay and continuous vessel allocation. In the temporal dimension, the company deals with dynamic vessel arrivals. At OCP, we refer to this problem as the production scheduling, inventory management, and vessel assignment (PSIMVA) problem.

In the literature, supply chain optimization models have been around for a while [256, 257]. The trend is the same for optimization models involving mining, chemical plants, and bulk ports [258, 259, 260, 261, 262, 263, 264, 1, 3, 2, 4, 254]. Given that the PSIMVA problem integrates all the OCP Group downstream operations (with the spatial and temporal dimensions above), it is a highly customized large-scale mixed integer linear programming (MILP) model, with millions of constraints, and hundreds of thousands of variables, tens of thousands of which are integers, which cannot be solved using state-of-the-art solvers. Furthermore, the problem requires a different formulation from the existing ones in the literature.

To tackle the PSIMVA model, the OCP Group acquired the downstream logistic planner (DLP) from a third-party provider. The DLP is an off-the-shelf product bought to support the OCP Group in planning its downstream operations. The OCP Group uses the DLP tool to solve the PSIMVA problem, i.e., the DLP is the system in charge of solving the PSIMVA model. After formulating the PSIMVA problem, [1] and [2] designed two heuristic tools that efficiently tackle the PSIMVA instances. [3] designed an exact tool inspired by the Benders decomposition. These tools worked well as standalone tools. The planning department at OCP used these tools separately to tackle the PSIMVA problem after deploying them into the

DLP. To do so, the planning department took the order selection as input from the commercial department. The latter used an aggregate model to select orders. The aggregated model did not incorporate the planning constraints. It created a challenge for the planning department. The OCP management decided to do it all in one model, i.e., select and plan orders in one go. After this shift in planning, the designed tools [1, 2, 3] were no longer as efficient as before when used as a standalone. In fact, following the OCP management decision, the company distinguishes between confirmed and unconfirmed vessels. Confirmed shipments are those for which the OCP Group has already fixed the latest delivery date. Unconfirmed shipments are those that the commercial department still negotiates and wants to know if they fit in the current planning horizon and, as a result, their potential latest delivery dates. From a business perspective, the OCP Group wants to prioritize confirmed shipments over unconfirmed ones, i.e., fulfill confirmed ones over unconfirmed ones. Unfortunately, the designed tools did not guarantee such a prioritization.

7.1.4 Contribution

With the changing needs and the continuous improvement within the OCP Group, there was a need for new developments, in particular a novel solution methodology that improves all the algorithmic tools [1, 3, 2, 4] developed and supports the DLP operationalization while prioritizing confirmed shipments and dealing with the shift in planning. We outline in this paper how optimization and analytics played a key role in transforming the downstream supply chain at OCP. The paper's contribution is threefold: (1) we improve and deploy an efficient optimization system to deal with the shift in downstream operations planning and prioritize confirmed shipments at OCP, (2) we introduce a novel and generic implementation of Benders decomposition: the hybrid Benders decomposition ($\mathcal{HBD}$) method, and (3) we present several managerial insights that follow from the operationalization of the DLP system at OCP and highlight that the transformation has led to +\$240 million increase in turnover annually.

7.1.5 Organization

The rest of this paper is structured as follows. In the *Context* section, we describe the problem and present the investigation strategy findings. In the *Analytics Approach* section, we review the system's key components and highlight the improved DLP deployed at OCP. In the *Impact* section, we present the operationalization benefits and in the *Conclusion* section, we summarize our concluding remarks.

7.2 Context

7.2.1 Physical Flow & Storage

The OCP Group supply chain, as visualized in Figure 7.1, starts with the extraction of phosphate rocks from the mine. Trucks transport these rocks to the physical treatment facility, where they undergo the washing and floating processes. The obtained phosphate powder is transported for chemical treatment by a 187 Km slurry pipeline to the Jorf site. In the coastal processing plant of this site, OCP refines several derivative products through 32 various chemical processes. Conveyors supply these derivative products to 6 quays where clients' vessels are loaded. The supply chain is connected through 102 conveyors and pipelines through which products flow. The company has storage upstream, midstream, and downstream of the Jorf site. In particular, the final products are stored in 29 large tanks or vessels. The coastal processing factory and the loading port span 5 Km^2.

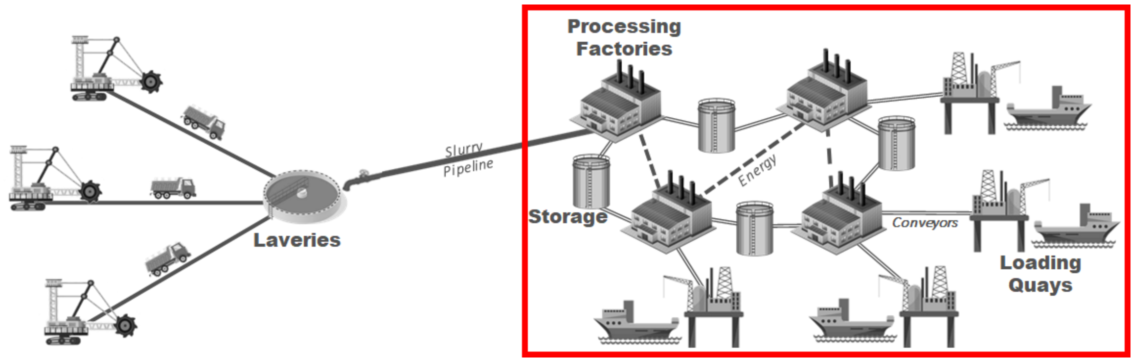


Figure 7.1 The OCP Supply Chain - From Extraction to the Jorf Site

7.2.2 Products

The promotion of *precision farming* has increased the number of finished products at the OCP Group from 3 to more than 30. As of today, the OCP Group has 45 raw, semi-finished, and finished products. The company expects that this number will exceed 75 shortly.

7.2.3 Orders

The OCP Group is a demand-driven company, i.e., it produces based on orders. OCP's commercial department receives orders from customers worldwide. We refer to each order of one or several products as a shipment. Each shipment needs a vessel and each vessel contains a unique shipment (we use both terms interchangeably). The commercial department provides shipments to the planning department for each planning horizon.

7.2.4 Decisions

The PSIMVA is a huge combinatorial MILP problem. Therefore, the OCP Group has invested intensively in operations research (OR) tools to integrate and centralize supply chain operations. Given the complexity of integrating the whole supply chain in one shot, OCP Group has been following a gradual integration approach starting from the downstream side of its supply chain. Thus, the focal location of this research is the Jorf site (El Jadida, Morocco) in the red frame in Figure 7.1. It is the largest among all OCP sites, where 90% of production happens.

The company wants to obtain satisfactory schedules for its downstream operations. The scheduling timescale is one month with the unit being one day (referred hereafter as a period). The key decisions that the PSIMVA model makes are as follows:

- Which shipments to fulfill and what is the loading schedule of each shipment selected for fulfillment, i.e., which quay crane, conveyor, product, and quantity at each period?
- Which product and quantity to produce in each production unit during each period (one day)?
- What is the changeover (shift from one product to another in a production unit) schedule in each production unit throughout the planning horizon?
- What is the level of each product at each storage point at the end of each period?

We present a detailed mathematical description of the PSIMVA model in Appendix F. More details about the OCP supply chain are in the paper of [2].

7.2.5 Solver

For the PSIMVA problem, no feasible solutions can be rapidly identified, even using off-the-shelf optimizers. OCP Group has been relying on manual planning for many years. With the increasing complexity of the supply chain, the company acquired the DLP in 2014. The initial version of DLP incorporates a database, a model generator, a mixed MILP solver, an output generator, and many other functionalities. To tackle the PSIMVA problem, the OCP planners provide the DLP with PSIMVA instances via a human-machine interface. The MILP solver within the DLP solves these instances and returns the obtained schedules to the planners. However, the DLP took more than 10 hours to generate feasible PSIMVA schedules, and the company maintained manual planning. Ideally, the OCP Group wants the DLP system to provide satisfactory PSIMVA solutions in less than 30 minutes. Towards the end of 2019, OCP opted to confirm the DLP's potential with optimization experts from the Polytechnique Montreal (Poly) and the Mohammed VI Polytechnic University (UM6P).

7.2.6 Exploration & Kernel

The OCP Group, UM6P, and Poly started exploring the PSIMVA instances (e.g., mathematical structure, constraints, decision variables) and found the following. The first version of the model had a weighted sum of various key performance indicators (KPIs) in the objective function. The weights are found by trial and error and are not mathematically founded. This implies a complex objective function, which is neither significant nor interpretable, leading the DLP to likely waste computational time exploring unpromising regions of the PSIMVA polyhedron, and consequently large integrality gaps. Furthermore, some constraints in the model assume the presence of some KPIs in the objective model and that we solve to optimality. Theoretically, the model was correct, however, since the OCP planner does not always solve optimally (a satisfactory solution is enough), the model was not practically correct. For the small-sized problems, this should not be problematic. However, in a large-scale case like the OCP Group, this created a problem. For instance, the initial PSIMVA model with the weighted objective function provided many changeovers (more than 100 in some cases) when it was not solved optimally. This was due to the presence of some constraints that fix changeovers to 1 (when a changeover is needed) and the absence of others that fix them to zero (when a changeover is not needed). The absence of constraints that fix changeovers to zero was due to the minimization of the changeovers in the objective function when we solve optimally. At OCP, a large number of changeovers is impractical as the number of changeovers required in practice over the one-month planning horizon is usually less than 15.

The research team gave PSIMVA instances to CPLEX (MILP solver within the DLP) to analyze further the MILP complexity. As shown in Figure 7.2, the default CPLEX fails to reach satisfactory schedules even after exploring more than 16,000 branch-and-bound nodes in more than 20,000 seconds, confirming that it consumes a significant amount of time exploring unpromising regions of the polyhedron (Er Raqabi et al., 2023).

To resolve these issues, the team ordered KPIs based on importance and defined a KPIs hierarchy. The first KPI is to *maximize the total fulfillment (TF) of shipments over the planning horizon.* The second KPI is to *minimize the total number of product changeovers (PC).* There are many other KPIs (e.g., safety stocks, demurrage costs). For conciseness purposes, we consider only these two KPIs in this paper (see Appendix F). Instead of optimizing a weighted objective function, the team shifted toward a lexicographic method, which consists of optimizing the first KPI and then optimizing the second KPI with a lower bound constraint

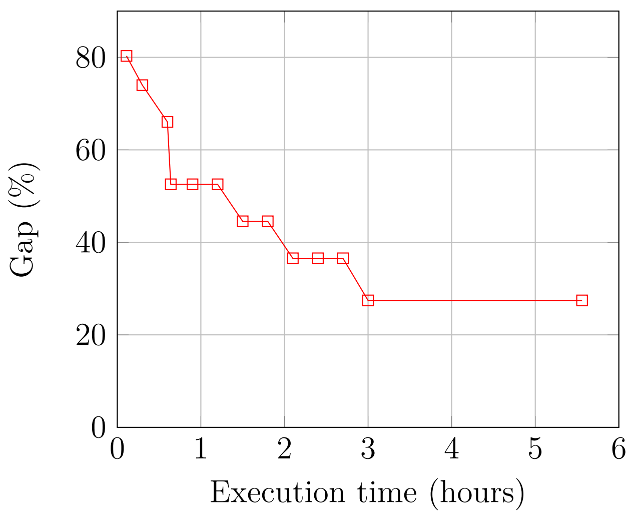


Figure 7.2 Integrality Gap Evolution with Execution Time

on the first KPI (Marler and Arora, 2004). After that, the team remodeled some PSIMVA constraints to ensure the new model is theoretically and practically correct. Furthermore, before developing sophisticated methods, the Poly experts manually tuned CPLEX parameters to improve the solver performance. By configuring CPLEX parameters for each new PSIMVA instance, the researchers could improve performance and reach feasible schedules with better quality and more rapidly (in less than 4 hours) compared to the 10 hours required by the DLP on the initial PSIMVA model.

Organizationally, the Poly experts elaborated a working strategy based on prototyping with a rapid cycle and quick feedback to show encouraging results and achieve early success. The early exploration results confirmed the DLP's potential, prompted the operationalization research effort, restored confidence, and established the OCP-UM6P-Poly alliance.

7.3 Analytics Approach

We present the optimization system's key components, which the OCP, UM6P, and Poly continuously improved. Then, we highlight the improved DLP.

7.3.1 Key Components

In this section, we highlight previous works developed to tackle the PSIMVA: the multi-phase iterated local search ($\mathcal{MPILS}$) tuner, the incremental large neighborhood search ($\mathcal{ILNS}$) metaheuristic, the primal Benders decomposition ($\mathcal{PBD}$) method, and the resilience re-optimization ($\mathcal{RRO}$) approach.

The $\mathcal{MPILS}$ Tuner. After noticing during the DLP exploration that manually configuring CPLEX parameters improves its performance on PSIMVA instances, [1] opted to configure the CPLEX solver automatically and designed the $\mathcal{MPILS}$ tuner, represented in the blue frame in Figure 7.3. The $\mathcal{MPILS}$ tuner searches for the best CPLEX configurations that improve the DLP performance on PSIMVA instances.

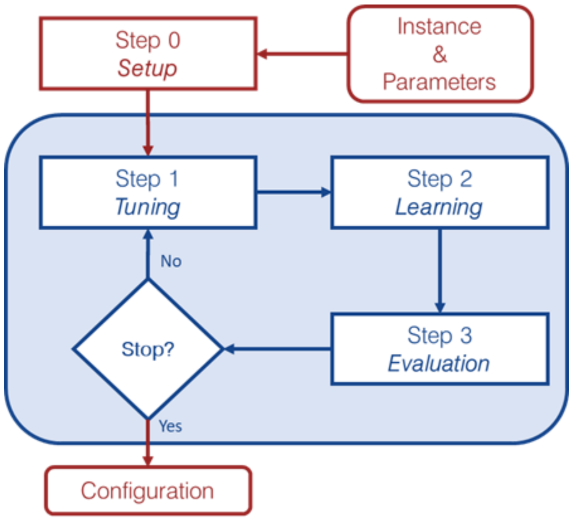


Figure 7.3 *$\mathcal{MPILS}$ Tuner in the Blue Frame with Three Steps:* Tuning *Step,* Learning *Step, and* Evaluation *Step*

The $\mathcal{MPILS}$ tuner starts with an initial pool of parameters identified a priori by the team in the *Setup* step as follows. The team runs initial tests on representative instances and analyzes the obtained log files. From these log files analysis, it is possible to infer an initial pool of parameters that can be used to construct an initial configuration θ^0. For instance, if the solver spends too much time adding cuts without any improvement, then the *CutsFactor* [128] parameter can be added to the initial pool of parameters. After identifying the initial pool of parameters, the *Tuning* step consists of searching for satisfactory configuration(s) in a reduced search space of configurations induced by a small subset of parameters (which is, in the beginning, the initial pool of parameters identified by the team). This is done by testing several configurations using an iterated local search metaheuristic. After that, the tuner uses statistical learning techniques to remove potential deteriorating configurations in the *Learning* step and evaluation methods for parameters based on two metrics (optimality gap and time to optimality) to insert promising ones in the *Evaluation* step. The cycle continues until no parameter to tune is identified in the *Evaluation* step.

Initial experiments on the considered PSIMVA problem have shown that there is a high chance that the best configuration obtained using the $\mathcal{MPILS}$ performs well on various PSIMVA instances. This can be explained by the fact that these large-scale industrial optimization problems have fixed installations, standard processes, and repetitive trends. For such MILP problems, instances show periodicity, similarity, and repetitiveness from one year to another. Compared to available state-of-the-art tuners such as paramILS [265] and irace [40], the $\mathcal{MPILS}$ does not consider all CPLEX parameters simultaneously, which makes tuning efficient in very large-scale contexts such as the OCP case. However, it has some limitations. First, the MPILS tuner can only tune one PSIMVA instance at a time. Second, the tuner design is specific to the MILP solvers' niche where the instances are clustered and only one instance from each cluster is selected for tuning. Third, it requires prior identification of an initial set of parameters based on problem knowledge. Further details about the $\mathcal{MPILS}$ tuner are available in Himmich et al. (2023).

The $\mathcal{ILNS}$ Metaheuristic. After analyzing the PSIMVA model, the team found that the complexity and high symmetry come from the loading variables (q_i binary variables in Appendix F), which represent around 80% of the PSIMVA binary variables. Each q_i variable corresponds to a possible assignment of a given vessel to a given quay during a given time interval (several periods). For each vessel, several q_i variables are generated by the OCP Group. [2] designed the $\mathcal{ILNS}$ metaheuristic (highlighted in the blue frame of Figure 7.4), which selects the q_i variables heuristically (vessel assignment side of the supply chain).

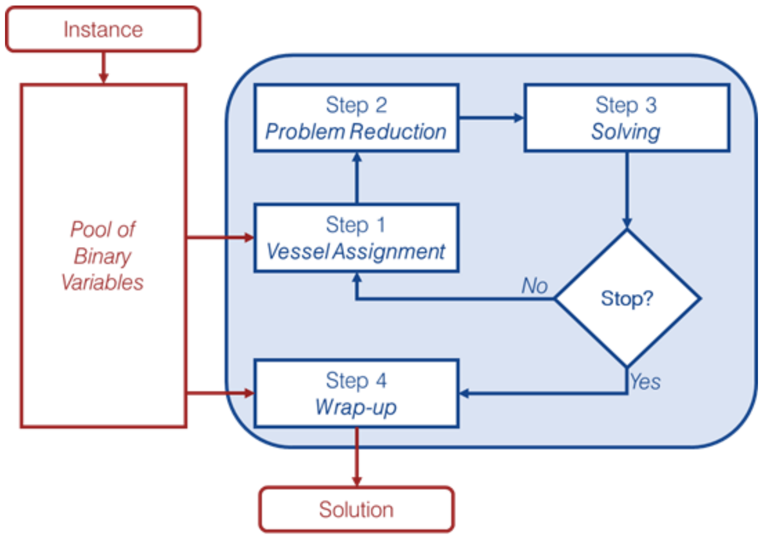


Figure 7.4 $\mathcal{ILNS}$ Metaheuristic in the Blue Frame with the Four Steps: *Vessel Assignment* Step, *Problem Reduction* Step, *Solving* Step, and *Wrap-up* Step

The $\mathcal{ILNS}$ metaheuristic takes a PSIMVA instance and iterates over four steps: the *Vessel Assignment* step, the *Problem Reduction* step, the *Solving* step, and the *Wrap-up* step. After partitioning the planning horizon into smaller time intervals (e.g., weeks), the *Vessel Assignment* step assigns each vessel to a time interval. After that, the metaheuristic first reduces the pool of binary variables related to vessel assignment in the *Problem Reduction* step. Then, it solves the reduced problem in the *Solving* step. The metaheuristic keeps iterating over the time horizon, using previous solutions as a warm-start until completion. Before returning a solution, in case there are still unfulfilled vessels, the metaheuristic attempts to fulfill them in the *Wrap-up* step.

Compared to the standard large neighborhood search (LNS) metaheuristic (Pisinger and Ropke, 2010), which does not work efficiently for very large-scale optimization problems, the $\mathcal{ILNS}$ destroys just the part of the solution to improve. This strategy is powerful since there is no time for backtracking in such a huge MILP problem. Thus, the metaheuristic breaks the problem down, solves it, fixes part of the solution, and then incrementally improves it. Further details about the $\mathcal{ILNS}$ metaheuristic are available in Er Raqabi et al. (2023).

The $\mathcal{PBD}$ Method. To select all complicating binary variables exactly (production scheduling and vessel assignment sides of the supply chain), [3] designed the $\mathcal{PBD}$, which is a primal variant of the Benders decomposition (BD) (Benders, 1962).

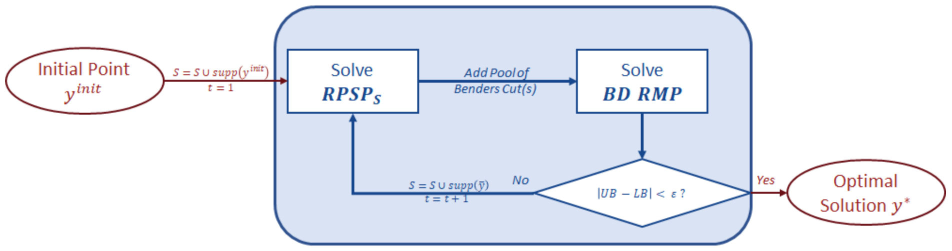

Initial Point y^{init}
$S = S \cup supp(y^{init})$ $t = 1$
Solve $RPSP_S$
Add Pool of Benders Cut(s)
Solve BD RMP
$|UB - LB| < \varepsilon$?
No
$S = S \cup supp(\bar{y})$ $t = t + 1$
Yes
Optimal Solution y^*

Figure 7.5 $\mathcal{PBD}$ Method in the Blue Frame with the Two Steps: *RPSP* Step and *PBD RMP* Step.

The $\mathcal{PBD}$, highlighted in Figure 7.5, inserts the complicating variables (i.e., when fixed the problem becomes significantly easier to solve) that are in the support (the set of indices where the solution vector has non-zero entries) of the initial solution (e.g., the solution(s) of the $\mathcal{ILNS}$ metaheuristic) into the PBD subproblem, referred to as the reduced subproblem (RPSP). Then, in the subsequent iterations, those in the support of the restricted master problem (PBD RMP) solution are inserted into the RPSP. The RPSP is thus a restriction of the original PSIMVA problem. We compute the Benders cuts to add to the PBD RMP, and loop on to insert gradually complicating variables in the RPSP, until convergence. To enhance the performance of the $\mathcal{PBD}$ method, [3] propose several acceleration strategies, including Pareto-optimal Benders cuts, warmstart, and a selection strategy for the PBD RMP variables.

Unlike $\mathcal{ILNS}$, which selects variables heuristically, the $\mathcal{PBD}$ selects subsets of variables exactly. Furthermore, being primal, the $\mathcal{PBD}$ method avoids the BD zigzagging behavior (of the primal bound) and slow convergence. It generates only optimality cuts, uses previous iteration solutions to warm-start the current iteration, and converges quickly towards satisfactory solutions. Further details are available in Er Raqabi et al. (2023).

The $\mathcal{RRO}$ Approach. The OCP supply chain faces several weather and vessel perturbations on the port (vessel assignment side of the supply chain). To remain resilient to perturbations and adapt to changing circumstances and challenges in real time, [4] developed an efficient re-optimization approach for the OCP Group: the $\mathcal{RRO}$ approach. It is highlighted in Figure 7.6.

To ensure the OCP Group remains resilient, the authors define and model resilience. OCP defines resilience as the ability to recover or reach a better schedule q^* while remaining as *close* as possible to the previously satisfactory schedule $\cancel{q}^*$. The term *close* implies fulfilling as many vessels as schedule $\cancel{q}^*$ while maintaining the least distance possible to this schedule. After that, the authors identify and model perturbation(s), which they classify into weather or vessel perturbations. A weather perturbation occurs when the weather in the port is bad enough to affect normal operations. A vessel perturbation occurs when there is a delay in a vessel's arrival at the port. Then, they identify the set of decisions to take. For instance, when the weather in the port is bad during a given period and some vessels are planned to be loaded during that same period, the OCP Group needs to adjust the loading schedule to ensure the loading of these vessels. Within the OCP Group, they distinguish two main types of decisions: the delay decision to be made when the weather at the port is bad or when there is a vessel delay, and the advance decision that occurs to allow loading a vessel ahead of schedule. Following these aspects, [4] formulate the re-optimization problem to maximize resilience.

After formulating the re-optimization problem, the authors design a re-optimization approach while leveraging the primal information using the previously satisfactory schedule, the original problem, and the company's history (e.g., previous vessels' assignments to quays and production schedules) to avoid re-optimizing from scratch. Since the perturbations happen on the port side, [4] fix the production and optimize locally by considering solely the port variables and constraints in the re-optimization problem formulation. The authors quantify resilience and model it as a weighted-objective function in the re-optimization problem.

With local optimization, the re-optimization approach provides schedules as close as possible to the previously optimal ones and with minimum changes in response to perturbations. This is computationally more efficient than optimizing from scratch. Further details about the

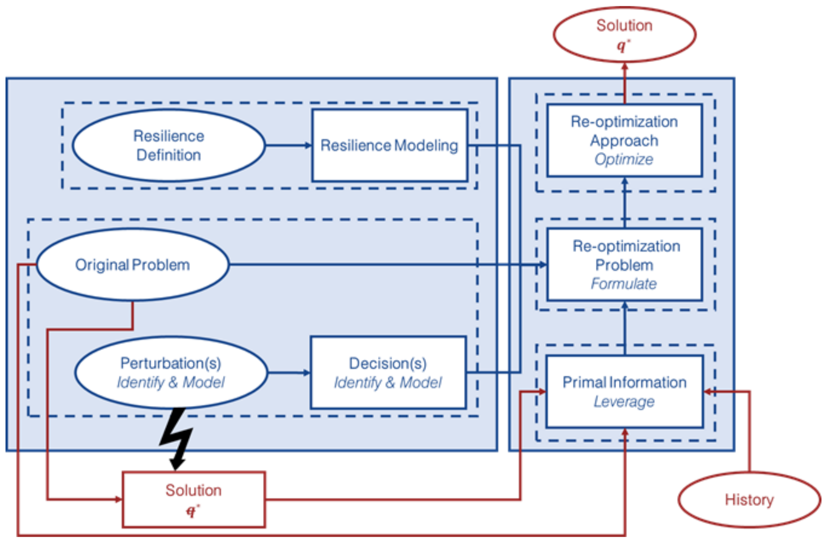


Figure 7.6 $\mathcal{RRO}$ Approach in the Blue Frames

$\mathcal{RRO}$ approach are available in [4].

7.3.2 The Improved DLP

While leveraging all the learned insights from the key components above, the team role was to improve the initial version of DLP (highlighted in red in Figure 7.7 and hereafter referred to as $\mathcal{DLP}$) on two metrics: runtime and quality, i.e., reaching near-optimal, if not optimal, schedules as quickly as possible. The team deployed the four methods (the $\mathcal{MPILS}$ tuner, the $\mathcal{ILNS}$ metaheuristic, the $\mathcal{HBD}$ method, and the $\mathcal{RRO}$ approach) in the $\mathcal{DLP}$ as visualized in Figure 7.7. We refer to the current version of the tool as the improved DLP ($\mathcal{IDLP}$).

In what follows, we highlight the $\mathcal{IDLP}$ tool. To do so, we introduce the $\mathcal{HBD}$ method. Then, we describe the improvement of the $\mathcal{MPILS}$ tuner and the adjustment of the $\mathcal{ILNS}$ metaheuristic. Lastly, we present the system deployment.

The $\mathcal{HBD}$ Method. Inspired by the practice at OCP, the $\mathcal{HBD}$ method, highlighted in Figure 7.8, is a hybridization of the BD [75] and the $\mathcal{PBD}$ [4] methods. The intuition behind the $\mathcal{HBD}$ is fixing some complicating variables (like BD) and keeping others free (like $\mathcal{PBD}$) in the $\mathcal{HBD}$ subproblem.

The $\mathcal{HBD}$ method starts with an initial point of complicating variables and gradually inserts promising complicating variables until convergence, i.e., $|UB-LB| < \epsilon$, where the $\mathcal{HBD}$ master problem provides UB (dual bound) and $\mathcal{HBD}$ subproblem provides LB (primal bound).

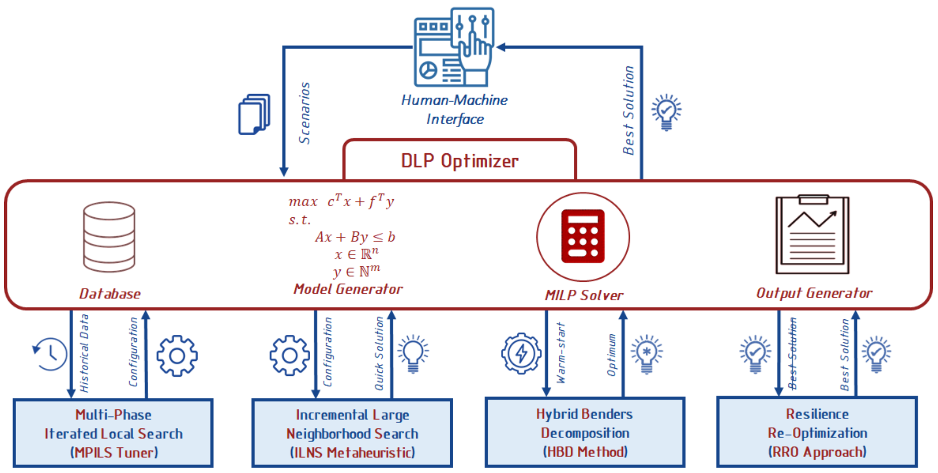

Human-Machine Interface
Scenarios
Best Solution
DLP Optimizer
Database
max $c^Tx + f^Ty$ s.t. $Ax + By \le b$ $x \in \mathbb{R}^n$ $y \in \mathbb{N}^m$
Model Generator
MILP Solver
Output Generator
Historical Data
Configuration
Configuration
Quick Solution
Warm-start
Optimum
Best Solution
Best Solution
Multi-Phase Iterated Local Search (MPILS Tuner)
Incremental Large Neighborhood Search (ILNS Metaheuristic)
Hybrid Benders Decomposition (HBD Method)
Resilience Re-Optimization (RRO Approach)

Figure 7.7 Overview of the $\mathcal{IDLP}$ System

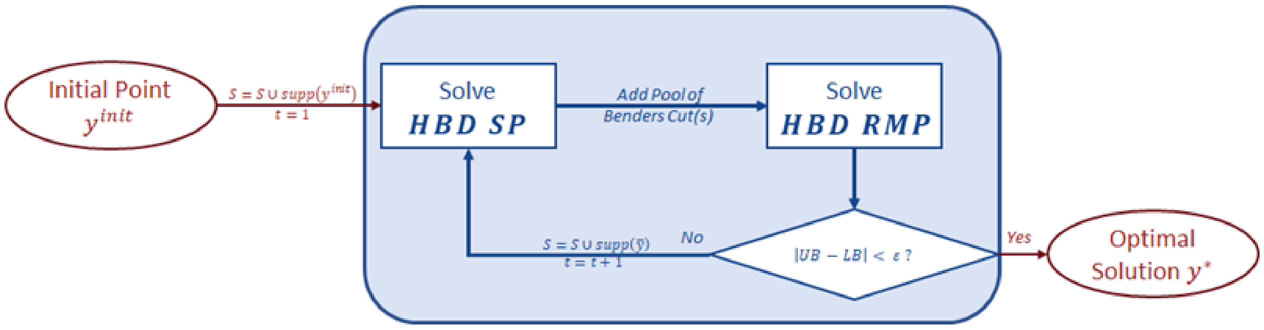

Initial Point y^{init}
$S = S \cup supp(y^{init})$
$t = 1$
Solve HBD SP
Add Pool of Benders Cut(s)
Solve HBD RMP
$|UB - LB| < \varepsilon$?
No
$S = S \cup supp(\bar{y})$
$t = t + 1$
Yes
Optimal Solution y^*

Figure 7.8 $\mathcal{HBD}$ Method in the Blue Frame

For the OCP case, the choice of the complicating variables to fix and the ones to keep free comes from the confirmed and unconfirmed shipments. Since the confirmed shipments have a higher priority, their corresponding complicating variables are fixed in the $\mathcal{HBD}$ subproblem to ensure their fulfillment, i.e., the corresponding q_i variables are fixed to 1 forcing their fulfillment. The corresponding complicating variables to unconfirmed shipments are kept free, and the $\mathcal{HBD}$ subproblem selects the ones to fulfill. It is worth highlighting that the complicating variables added to the subproblem are not all fixed, as in BD. Thus, the $\mathcal{HBD}$ subproblem is a restriction of the PSIMVA problem.

Unlike $\mathcal{ILNS}$, which selects subsets of variables heuristically, $\mathcal{HBD}$ selects subsets by mathematical optimization without incurring significant computational costs and can modify them to achieve optimality. Compared to $\mathcal{PBD}$, the $\mathcal{HBD}$ method ensures the prioritization of confirmed shipments over unconfirmed ones. Beyond the OCP case, $\mathcal{HBD}$ is generic enough to tackle several general large-scale optimization problems. We provide more details about the $\mathcal{HBD}$ method in Appendix G.

The $\mathcal{MPILS}$ Improvement and $\mathcal{ILNS}$ Adjustment. To deal with the shift in downstream operations planning (i.e., from order selection then order planning to order selection and planning in one model), we leveraged the $\mathcal{MPILS}$ tuner of [1] and the $\mathcal{ILNS}$ metaheuristic of [2]. While the $\mathcal{MPILS}$ worked well before the shift, it had two main limitations. First, it is a single-purpose tuner, i.e., it mainly focuses on improving a single metric: the optimality gap. Second, it provides a single configuration for the whole execution. We adjusted the $\mathcal{MPILS}$ tuner to become a multi-purpose tuner, which improves various metrics (e.g., primal bound, dual bound, feasibility, and optimality). For example, (1) using a first configuration to ensure feasibility and a second configuration to reach optimality, and (2) starting the solving with a configuration until stabilizing the primal bound and then shifting towards a configuration that improves the dual bound. To achieve this, we run executions to a certain level (e.g., achieving feasibility or improving the primal bound), and then we rerun from that level to improve further (e.g., reaching optimality or improving the dual bound).

To boost the $\mathcal{ILNS}$ metaheuristic performance, we rely on two techniques: parallelism and tuning. First, we run several variants of the $\mathcal{ILNS}$ metaheuristic in parallel. Second, for all the variants run in parallel, we use the best $\mathcal{MPILS}$ configurations to enhance the metaheuristic performance.

The System Deployment. The full-scale system required certain features to make it usable. We used CPLEX to solve the PSIMVA instances (and any inferred mathematical models like the $\mathcal{HBD}$ master problem and subproblem) since the DLP has the ILOG CPLEX Callable Library. For implementation, we used C++, Python, and R programming languages.

The system, designed for monthly planning with re-optimization as needed at each industrial site, works as follows: The planner updates the necessary planning data for the current planning horizon and provides a PSIMVA instance to the $\mathcal{IDLP}$, which returns the best schedule of production, stock, and loading found. We zoom inside the $\mathcal{IDLP}$ in Figure 7.9. The user takes the best configurations of the $\mathcal{MPILS}$ tuner from the *history* and provides them to the $\mathcal{ILNS}$, which uses them to solve the given PSIMVA instance. We execute several $\mathcal{ILNS}$ runs in parallel using these configurations and the $\mathcal{ILNS}$ parameters. For each run $j \in \{1, 2, ..., n\}$, we select a configuration, and an $\mathcal{ILNS}$ variant (when there is no decomposition, we obtain the MILP on the entire problem). We then collect the solution(s). If a satisfactory solution is reached within the specified runtime (e.g., confirmed shipments fulfilled over unconfirmed ones, near-optimal solutions, gap below a threshold), the $\mathcal{IDLP}$ returns it. Otherwise, the $\mathcal{IDLP}$ uses the obtained solution(s) to warm-start the $\mathcal{HBD}$ method with initial cut(s) for the $\mathcal{HBD}$ master problem and initial solution(s) for the $\mathcal{HBD}$ subproblem. The $\mathcal{IDLP}$ returns the $\mathcal{HBD}$ solution(s) to the user.

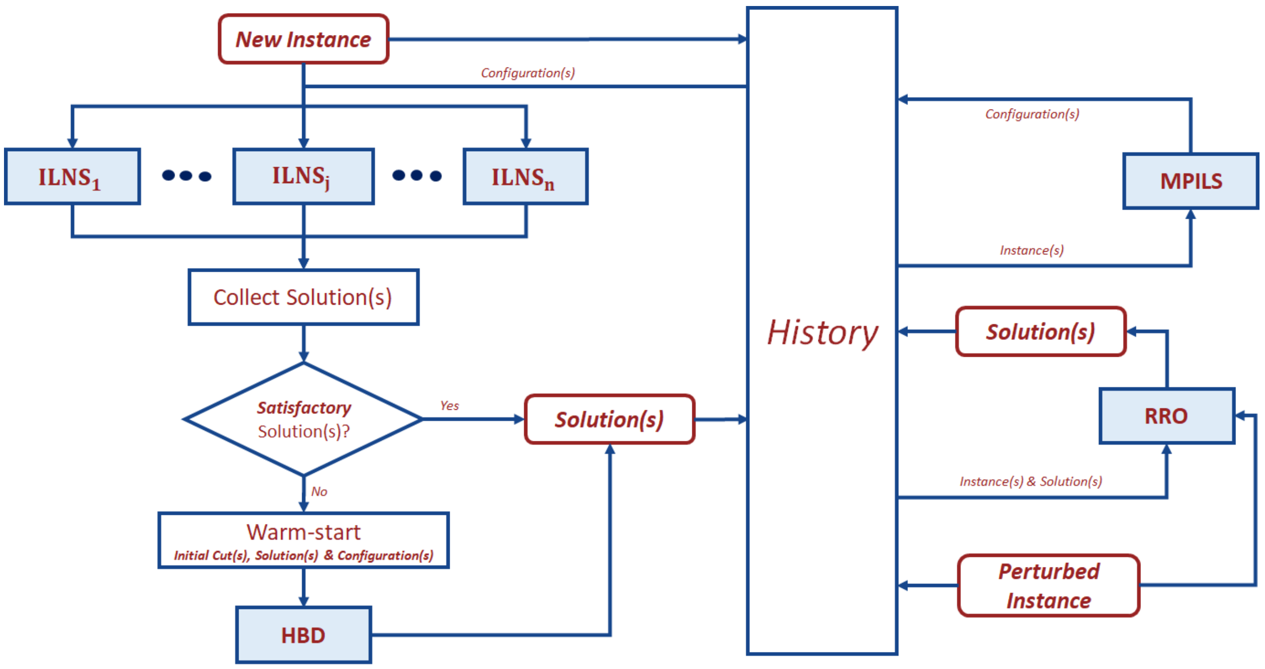


Figure 7.9 The $\mathcal{IDLP}$ System

The system stores all instances, configurations, and obtained solutions in history. These solution(s) and instances are used for training the $\mathcal{MPILS}$ tuner and for the $\mathcal{RRO}$ approach. In particular, the system allows for continuous training of the tuner for four reasons. First, except when running new instances through the system, there is plenty of runtime to train the tuner using PSIMVA instances stored in the history. The runtime availability is crucial since, for large-scale complex problems, testing each configuration requires several minutes. Second, benefiting from the runtime availability and computing power, we believe the tuner will converge towards the best configuration(s) for the OCP problem. Third, we expect these best configuration(s) will work well for several years since the downstream infrastructure at OCP does not change significantly, and the orders show seasonality and repetitiveness. Fourth, continuous training ensures we do not start from scratch each time new instances arrive, which sustains the capability to adapt to changes in the downstream supply chain and adjust quickly over time. If the tuner finds better configurations than existing ones, the system adds them to history. We allow the $\mathcal{MPILS}$ runs to communicate among them and improve the configurations dynamically.

Using the model generator, the planner can model perturbations in the downstream supply chain (e.g., bad weather in the port or vessel delay). It also allows for simulating several what-if scenarios. In both cases, the planner uses the model generator to add, remove, or fix variable(s) or constraint(s) and increase or decrease the values of model parameters (e.g., production capacity, loading capacity). When perturbing a given PSIMVA instance, the $\mathcal{RRO}$ tool takes the perturbed instance, its original instance, and its best solution(s) from history as input and returns the new best solution, which takes into account the perturbation(s).

Lastly, using the output generator of the DLP, the OCP planner can visualize any solution returned by the algorithmic tools as a monthly plan with production schedules, inventory management, and vessel assignments. The procedure in Figure 7.9 allows enough flexibility for the user to change parameters and run various trials, making it very practical. It is primal since we provide all the solutions obtained by a given algorithmic tool to the subsequent one.

7.4 Impact

In this section, we outline the impact of the DLP operationalization. We distinguish the following benefits: quantifiable benefits, planning process, involvement of local researchers, and theory inspired by practice.

7.4.1 Quantifiable Benefits

We categorize the quantifiable benefits into the following: runtime and quality metrics, optimal capacity, financial impact, and food security in Africa.

Runtime and Quality Metrics. The deployed system maximizes TF (the percentage of fulfilled shipments in terms of quantities) first before minimizing PC (the number of changeovers). Below, we highlight the computational results on real PSIMVA instances from the focal OCP site, the Jorf site (J instances). We provide a detailed description of the J instances in Appendix H.

For each instance and the first KPI (TF), we report in Table 7.1 the runtime (Runtime) in seconds and the optimality gap (Gap) in percentage using the $\mathcal{DLP}$, the MILP using the best configurations returned by the $\mathcal{MPILS}$ tuner of [1] as standalone, the $\mathcal{ILNS}$ metaheuristic of [2] as standalone as well, and the $\mathcal{IDLP}$ presented in this paper. For the $\mathcal{IDLP}$, we report the results obtained using the parallel $\mathcal{ILNS}$ enhanced with $\mathcal{MPILS}$ configurations, which we denote $\mathcal{ILNS}^{++}$. We allocate 10 hours for the $\mathcal{DLP}$ and 10 minutes for the MILP solving using the $\mathcal{MPILS}$ configurations, the $\mathcal{ILNS}$, and the $\mathcal{ILNS}^{++}$.

As seen in Table 7.1, we quickly obtain near-optimal (if not optimal) solutions using the $\mathcal{ILNS}^{++}$ on the first KPI, except for the instances J_{11} and J_{12} for which the gap is above 5%. On average, the runtime reduction factor is 67 for the Jorf site. For gap improvement, the $\mathcal{ILNS}^{++}$ outperforms the $\mathcal{DLP}$ by 23.9 %, the MILP using the $\mathcal{MPILS}$ configurations

Table 7.1 Results obtained by the $\mathcal{DLP}$, the $\mathcal{MPILS}$ tuner, the $\mathcal{ILNS}$ metaheuristic, and the $\mathcal{ILNS}^{++}$ method on J Instances for the TF KPI

Instance	$\mathcal{DLP}$		$\mathcal{MPILS}$		$\mathcal{ILNS}$		$\mathcal{ILNS}^{++}$	
	Runtime	**Gap**	**Runtime**	**Gap**	**Runtime**	**Gap**	**Runtime**	**Gap**
$\mathbf{J}_1$	36000	15.2	317	0.0	600	0.0	280	0.0
$\mathbf{J}_2$	36000	17.5	458	0.0	600	0.0	315	0.0
$\mathbf{J}_3$	36000	23.4	600	6.9	600	0.0	532	0.0
$\mathbf{J}_4$	36000	25.7	600	3.3	600	0.0	544	0.0
$\mathbf{J}_5$	36000	29.9	600	3.5	600	1.4	600	1.0
$\mathbf{J}_6$	36000	32.9	600	5.8	600	1.9	600	1.3
$\mathbf{J}_7$	36000	34.6	600	6.7	600	2.1	600	1.7
$\mathbf{J}_8$	36000	34.9	600	6.5	600	2.3	600	1.9
$\mathbf{J}_9$	36000	13.2	600	7.3	600	5.4	600	0.0
$\mathbf{J}_{10}$	36000	11.0	600	12.5	600	8.0	600	4.2
$\mathbf{J}_{11}$	36000	30.7	600	51.8	600	18.9	600	5.5
$\mathbf{J}_{12}$	36000	39.2	600	53.1	600	15.6	600	5.7
Avg	**36000**	**25.7**	**565**	**13.1**	**600**	**4.6**	**539**	**1.8**

as a standalone by 11.3%, and the $\mathcal{ILNS}$ metaheuristic as a standalone by 2.8%.

For the PSIMVA instances, we distinguish two cases: loosely coupled and narrowly coupled. In the first case, production scheduling and vessel assignment are loosely coupled, mainly because of stock abundance in between. In the other case, production scheduling and vessel assignment are narrowly coupled, mainly because of stock scarcity. For the loosely coupled instances (e.g., instances J_1 to J_4 and J_9), the $\mathcal{ILNS}^{++}$ can reach optimality. For the narrowly coupled instances (e.g., instances J_5 to J_8 and J_{10} to J_{12}), we use $\mathcal{HBD}$ (deployed in the $\mathcal{IDLP}$) to reach optimality, as shown in Table 7.2. For $\mathcal{HBD}$, we allocated 1 hour. To avoid the tailing effect of $\mathcal{HBD}$ (due to the tailing effect of BD), we stop within a gap of 0.1%. After maximizing TF, we minimize the second KPI (PC). We report the results on the second KPI in Appendix I.

Table 7.2 Results obtained by the $\mathcal{HBD}$ method on hard J Instances for the TF KPI

Instance	$\mathcal{ILNS}^{++}$		$\mathcal{HBD}$	
	Runtime	**Gap**	**Runtime**	**Gap**
$\mathbf{J}_5$	600	1.0	1800	0.1
$\mathbf{J}_6$	600	1.3	1800	0.1
$\mathbf{J}_7$	600	1.7	2700	0.1
$\mathbf{J}_8$	600	1.9	1800	0.1
$\mathbf{J}_{10}$	600	4.2	400	0.1
$\mathbf{J}_{11}$	600	5.5	3000	0.1
$\mathbf{J}_{12}$	600	5.7	3200	0.1
Avg	**600**	**3.0**	**2100**	**0.1**

The $\mathcal{IDLP}$ has allowed for the operationalization of the DLP, which was a bottleneck, curbing the usage of many other supply chain management tools (e.g., pricing, mine extraction, logistics, customer relationship management, etc.). It has also allowed the OCP planning team to reduce runtime from more than 10 hours to less than 2 hours. It has given planners more control over small iterations of 10 minutes. They no longer need to wait for hours to obtain the first solution. The obtained solutions show a gain of 5% in capacity, on average, compared to the manual solutions.

Optimal Capacity. Initially, when the commercial department at OCP selected orders, the intuition was to select the most profitable orders. This intuition leads to suboptimal decisions and does not provide any guarantee of their quality. After integrating order selection and planning in one model, the downstream side of the supply chain at the OCP Group started operating at near-optimal, if not optimal, capacity, as seen in Table 7.3, which reports the optimal TF percentage.

Table 7.3 Optimal TF for J Instances

Instance	$\mathbf{J_1}$	$\mathbf{J_2}$	$\mathbf{J_3}$	$\mathbf{J_4}$	$\mathbf{J_5}$	$\mathbf{J_6}$	$\mathbf{J_7}$	$\mathbf{J_8}$	$\mathbf{J_9}$	$\mathbf{J_{10}}$	$\mathbf{J_{11}}$	$\mathbf{J_{12}}$	**Avg**
$\mathbf{TF^*}$	100.0	93.4	99.1	93.1	95.7	99.2	97.3	100.0	100.0	100.0	100.0	95.7	**97.8**

The DLP becomes crucial to enabling maximum capacity utilization. As seen in Table 7.3, we distinguish two types of instances: fulfilled instances (for which $TF^*=100$) and unfulfilled instances (for which $TF^*<100$). For the fulfilled PSIMVA instances, the DLP informs the planning department about additional fulfillment opportunities. Since $TF^*=100$ for these instances, the planning department may suggest that the commercial department arranges additional shipments, which can be inserted into the fulfilled instances before re-optimizing using the $\mathcal{RRO}$ approach. For the unfulfilled PSIMVA instances, the DLP informs the planning department that the downstream side is operating at maximum capacity.

Financial Impact. An optimal PSIMVA production schedule generates higher volumes to sell. After DLP operationalization, many tools used by OCP's various departments became more effective as well, thus allowing OCP to increase its annual turnover by more than +5%, which translates into an additional 240 million dollars. This DLP operationalization occurred at the Jorf Lasfar site, which is OCP's largest, responsible for approximately 90% of total production. Although there are other OCP sites, the impact is significant due to the Jorf Lasfar's scale. The +5% in annual turnover generates space and allows OCP Group to produce more for Africa, especially during low demand season. As a result, OCP could sustain its capability to follow the increasing demand for customized fertilizers compared to generic fertilizers in Africa and worldwide.

Food Security in Africa. While the two challenges, i.e., capacity and customization, significantly augment the PSIMVA problem complexity, after tackling the MILP and operationalizing the DLP, OCP's production capacity and industrial flexibility increased. The DLP supports demand increase in Africa and precision farming while remaining environmentally friendly with over 30 fertilizer types. The gain in the annual turnover allows for producing enough fertilizers to feed a country like Nigeria.

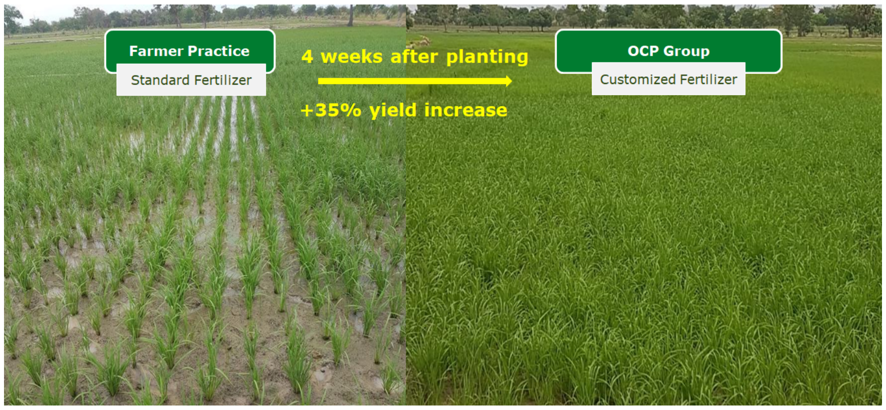


Figure 7.10 Rice Crops Ghana

As discussed in the beginning, fertilizers play a major role in modern agriculture. In particular, the DLP's capabilities in handling supply chain complexity and variability in an environment with a volatile sales pipeline enabled OCP to customize the portfolio of fertilizers. Given OCP leadership in Africa, the customized fertilizers culture evolved around the continent and improved yields for farmers at the lowest cost. For instance, in Ghana, rice yields increased by 35% (see Figure 7.10). While we are still far from achieving food security in Africa, the DLP operationalization is a step in the right direction.

7.4.2 Planning Process

Instead of relying on schedules designed by expert operators at the OCP Group, the DLP operationalization has enabled the shift to mathematically-based planning and scheduling. We categorize the impact on the planning process into OCP analytics culture disruption and extension to other sites.

OCP Analytics Culture Disruption. In addition to all the benefits of optimization and analytics, we highlight that the team did not deliver just the $\mathcal{IDLP}$, but also the best practices in the OR field. The planners use the system as a decision-making tool to check, simulate, and re-optimize schedules. They rely on it to enhance the supply chain resiliency to unexpected events and risks via what-if scenarios simulations. The DLP operationalization and weekly interactions with OCP people led to the iterative approach adoption as a way of thinking among OCP planners. This has led to changing the mindset at OCP Group and enhancing the analytics culture. Examples include optimizing KPI by KPI starting with the important ones, scheduling confirmed orders then unconfirmed ones, making decisions on the most certain schedules, and postponing the less certain ones.

Extension to Other Sites. The analytics and OR impact at the Jorf site has encouraged analytics adoption in other OCP sites. In particular, inspired by the PSIMVA instances, OCP planners at the Safi and Khouribga sites generated their instances. Planners started to adopt the OR mindset. In the Safi site, after noticing that solving instances using the default CPLEX takes more runtime than expected, a planner directly thought about changing the CPLEX configuration. In the Khouribga site, learning from the issues faced at the Jorf site, the runtime issue has been resolved in 48 hours compared to the six months spent fixing it at the Jorf site. While we have been implementing the developed solutions in the Jorf site, it is worth mentioning that we have started adapting them for other OCP sites and that the first results at these sites are promising.

7.4.3 Involvement of Local Researchers

One of the significant project benefits is the transfer of expertise through the involvement of local researchers. This involvement led us to acquire several insights. First, in applied research, local researchers support understanding and approximating factors and constraints, which are difficult to capture and model. Second, these researchers (who know the company's culture) ensure the social acceptability and adoption of the proposed system by final users through effective communication, OR concepts popularization, human factor management, and validation of solutions. Third, African researchers are the most suitable for leading projects related to the African continent. Still, the support of worldwide experts is necessary if there is no local expertise. Without the effective involvement of local researchers, we would have failed like so many others (Scott and Vessey, 2000; Xue et al., 2005; Danışman, 2010; Garg and Garg, 2013).

7.4.4 Theory Inspired by Practice

While we have been using OR theory to inform practice, this project generated new theoretical and methodological insights inspired by practice. These insights generalize well beyond the generated papers from this project [1, 2, 3, 4]. The developed system is generic and scalable to other contexts. Its intuition is quickly obtaining a near-optimal solution heuristically (using $\mathcal{ILNS}^{++}$ that combines $\mathcal{ILNS}$ with the best possible configurations fine-tuned over time) and proving optimality after that (using $\mathcal{HBD}$).

The $\mathcal{MPILS}$ tuner is generic and scalable beyond the CPLEX solver. Using observations from the practice, we conjecture that, for any configurable algorithm, there is a minimal subset of parameters, which allows converging towards the optimal solution(s) quickly (all other parameters being at their default values). With runtime availability, computing power, and parallelism, one can reach these solutions using techniques such as racing, statistical learning, and evaluation [1] without too much sophistication. The $\mathcal{HBD}$ method is an interesting methodological contribution generated from the practice of prioritizing confirmed orders (which corresponds mathematically to fixing their corresponding variables) and waiting for others to be confirmed (which corresponds mathematically to relaxing their corresponding variables). It can easily be adapted and applied to various contexts. For instance, in facility location type problems, we can use $\mathcal{HBD}$ to prioritize opening some facilities (e.g., based on cost or proximity reasons) over others. The $\mathcal{HBD}$ opens new research paths and horizons in the BD theory.

7.5 Conclusion

With applied analytics, the DLP tool has been revitalized as an efficient planning tool for the OCP Group. After overcoming several challenges and leveraging the $\mathcal{MPILS}$ tuner and the $\mathcal{ILNS}$ metaheuristic, we designed an efficient system for order selection and planning. In addition, after incorporating the $\mathcal{HBD}$ method, we reach near-optimal, if not optimal, solutions using the optimizer. These enhancements have contributed to increasing OCP's annual turnover by more than $240 million. The long-term impact could be even greater, as the optimizer's changes affect the whole supply chain and all OCP sites, leading to further profits and gains. Once again, operations research has demonstrated its ability to enhance processes that directly benefit the lives of humans. This project was a finalist at the IFORS Prize for OR in Development 2023 and won the EURO Excellence in Practice Award 2024, and we hope it will inspire the advancement of OR practices in developing countries.

Acknowledgments

The OCP Group, the Fonds de Recherche du Québec - Nature et Technologies (FRQNT), the Institute for Data Valorisation (IVADO), and the Group for Research in Decision Analysis (GERAD) have generously supported this project. We thank the stakeholders Boumahdi Salah-Eddine, Guellaf Hichame, Himmich Ilyas, Mahboubi El Miloudi, Metrane Abdelmoutalib, Rakhis Ilyas, Hafnaoui Anass, and El Mansouri Karim for their contributions

to the project.

CHAPTER 8 GENERAL DISCUSSION AND CONCLUSION

In this chapter, I summarize all the works presented in this thesis, mention the limitations, and highlight future research directions.

8.1 Summary of Works

This thesis highlights the potential of primal large-scale optimization. In particular, it opens the path for the *primalization* of dual methods. It also opens a new research path for MILP solvers. In this sense, the thesis contributes to the research community with new theoretical insights that can hopefully push state-of-the-art boundaries further. Beyond the theoretical component, this research goes with the spirit of applied mathematics in which research is conducted to deal with practical issues faced within the real world. While there is still a huge room for improvement, humans are already benefiting from large-scale optimization techniques in the real world. In my case, the methodological contributions in this thesis were quite crucial from our industrial partner's perspective since several results have been implemented within the OCP Group optimization software. Optimization comes then as an efficient way that will guarantee significant cost reductions. Furthermore, while the traditional approach is moving from fundamental research to practical applications, I believe that the reverse way is also possible, i.e., theory inspired from practice. This is in fact the beauty of OR, witnessed once again in this thesis.

8.2 Limitations

This thesis has some limitations that I could build on to improve OR techniques. For instance, the MPILS tuner (Chapter 3) is a deterministic tuner designed specifically for tuning MILP solvers, which are used to solve a variety of real-world optimization problems. Within this niche, although the MPILS shows significant performance compared to other general state-of-the-art tuners, it has some limitations. First, the MPILS tuner cannot tune more than one instance at a time. Second, the tuner design is specific to the MILP solvers' niche where I cluster the instances and select only one for tuning from each cluster. Third, it requires prior identification of an initial set of parameters based on problem knowledge. To tackle the above limitations, future research includes the implementation of the multi-instance (as a training set) option, automatic clustering of instances, and automatic troubleshooting. I will also explore the design and implementation of more advanced deep learning techniques to strengthen the identification and removal of non-promising combinations of parameter values. Furthermore, the PBD method (Chapter 5) performs well in real-life contexts, where organizations face sparse very large-scale problems for which they usually have good primal solutions close to the optimal solution (in terms of solution support). In such a case, I can use these solutions (primal information) to reach (near-)optimal solution(s) quickly. If the problems are not sparse, then the PBD subproblem quickly explodes and becomes as large as the original problem. In such a case the PBD method becomes obsolete.

8.3 Future Research

During these four years of my Ph.D., my colleagues and I also worked on other aspects of primal large-scale optimization, which led to the publications below. All these publications are methodological extensions of some of the thesis publications. Thus, they can be considered as ongoing/future research. They are highlighted below.

1. Er Raqabi El Mehdi*, Siwane Oussama, El Hallaoui Issmaïl and Beljadid Ahmed (2024). *A Parallel Multi-Purpose Tuner for MILP Solvers.* Soon to be submitted to **INFORMS Journal on Computing**.

2. Er Raqabi El Mehdi*, Bani Abderrahman, Morabit Mouad, Blondin Massé Alexandre, Besner Alexandre, Fournier Julien and El Hallaoui Issmaïl (2024). *An Efficient Decomposition Matheuristic for the Transient Stability Constrained Unit Commitment at Hydro-Quebec.* Submitted to **IEEE Transactions on Power Systems**.

3. Er Raqabi El Mehdi*, El Fassi Mohamed, Leus Roel and El Hallaoui Issmaïl (2024). *The Stochastic Improved Primal Simplex.* Soon to be submitted to **Mathematical Programming**.

The first paper extends the work in [1] (Chapter 3), where I design a parallel multi-purpose tuner that allows configurations to communicate among them in a multi-agent system. It dynamically combines the strengths of each configuration (e.g., one that improves the upper bound and another that improves the lower bound) to boost the solver performance and converge quickly. The second paper is an extension of the OCP Group success story in Morocco [2] (Chapter 4) to the Hydro-Quebec case in Canada. At Hydro-Quebec, we have achieved a significant reduction from a gap of 3% in 16 hours to a gap of 1.21% in 16 minutes on the provided instances. The third paper extends the work in [3] (Chapter 5) to the stochastic case. It is an attempt to primalize the L-shaped method for two-stage stochastic programming problems. For the fourth perspective, future research includes fusing AI and OR at large scales to tackle complex problems, revolutionize decision-making, and deliver scientific breakthroughs that the two fields cannot achieve independently [99]. This is referred to as AI for Optimization (AI4OPT). I expect to continue on these research paths: *Tuners*, *Primalization*, and *AI4OPT*.

The work presented in this thesis was a finalist at the IFORS Prize for OR in Development 2023 and a winner at the EURO Excellence in Practice Award 2024, and I hope it can contribute to boosting OR practices in Africa and all developing countries.

REFERENCES

[1] Ilyas Himmich et al. “MPILS: An automatic tuner for MILP solvers”. In: *Computers & Operations Research* 159 (2023), p. 106344.

[2] El Mehdi Er Raqabi et al. “Incremental LNS framework for integrated production, inventory, and vessel scheduling: Application to a global supply chain”. In: *Omega* 116 (2023), p. 102821.

[3] El Mehdi Er Raqabi, Issmail El Hallaoui, and François Soumis. “The Primal Benders Decomposition”. In: *Les Cahiers du GERAD* G-2023-27 (Aug. 2023). URL: https://www.gerad.ca/fr/papers/G-2023-27.

[4] El Mehdi Er Raqabi et al. “Towards resilience: Primal large-scale re-optimization”. In: *Transportation Research Part E: Logistics and Transportation Review* 192 (2024), p. 103819.

[5] El Mehdi Er Raqabi et al. “OCP optimizes its supply chain for Africa”. In: *Les Cahiers du GERAD ISSN* G-2023-29 (Aug. 2023), pp. 1–15. URL: https://www.gerad.ca/en/papers/G-2023-29.

[6] El Mehdi Er Raqabi et al. “A Parallel Multi-Purpose Tuner for MIP Solvers”. In: *Les Cahiers du GERAD* G-2024-xx (Nov. 2024), pp. 1–16. URL: https://www.gerad.ca/en/papers/G-2024-xx.

[7] El Mehdi Er Raqabi et al. “An efficient decomposition matheuristic for the transient stability constrained unit commitment at Hydro-Quebec”. In: *Les Cahiers du GERAD* G-2023-28 (July 2024), pp. 1–16. URL: https://www.gerad.ca/en/papers/G-2024-38.

[8] El Mehdi Er Raqabi et al. “The Stochastic Improved Primal Simplex”. In: *Les Cahiers du GERAD* G-2024-xx (Nov. 2024), pp. 1–16. URL: https://www.gerad.ca/en/papers/G-2024-xx.

[9] Abderrahman Bani et al. “Combining Benders decomposition and column generation for the petrol station replenishment problem with Inventory management”. In: *Les Cahiers du GERAD* G-2024-18 (Mar. 2024), pp. 1–31. URL: https://www.gerad.ca/fr/papers/G-2024-18.

[10] Abderrahman Bani et al. “The petrol station replenishment problem: A case study from West Africa”. In: *Les Cahiers du GERAD* G-2024-19 (Mar. 2024), pp. 1–16. URL: https://www.gerad.ca/en/papers/G-2024-19.

[11] El Mehdi Er Raqabi and Wenkai Li. “An electric vehicle transitioning framework for public fleet planning”. In: *Transportation Research Part D: Transport and Environment* 118 (2023), p. 103732.

[12] UN Desa. “World population prospects 2019: Highlights”. In: *New York (US): United Nations Department for Economic and Social Affairs* (2019).

[13] Francisco Martínez-Álvarez et al. “Coronavirus optimization algorithm: a bioinspired metaheuristic based on the COVID-19 propagation model”. In: *Big Data* 8.4 (2020), pp. 308–322.

[14] Dimitris Bertsimas et al. “From predictions to prescriptions: A data-driven response to COVID-19”. In: *Health care management science* (2021), pp. 1–20.

[15] George B Dantzig. “Linear programming”. In: *Operations research* 50.1 (2002), pp. 42–47.

[16] George L Nemhauser and Laurence A Wolsey. *Integer programming and combinatorial optimization*. Vol. 191. Springer, 1988.

[17] G Dantzig. *Linear programming and extensions*. 1998.

[18] Ailsa H Land and Alison G Doig. “An automatic method for solving discrete programming problems”. In: *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 105–132.

[19] John DC Little et al. “An algorithm for the traveling salesman problem”. In: *Operations research* 11.6 (1963), pp. 972–989.

[20] Jeff T Linderoth and Martin WP Savelsbergh. “A computational study of search strategies for mixed integer programming”. In: *INFORMS Journal on Computing* 11.2 (1999), pp. 173–187.

[21] Tobias Achterberg, Thorsten Koch, and Alexander Martin. “Branching rules revisited”. In: *Operations Research Letters* 33.1 (2005), pp. 42–54.

[22] Gilbert Laporte and Paolo Toth. “A gap in scientific reporting”. In: *4OR* 20.1 (2022), pp. 169–171.

[23] George L Nemhauser. “The age of optimization: Solving large-scale real-world problems”. In: *Operations Research* 42.1 (1994), pp. 5–13.

[24] *OCP Group. OCP Group Morocco.* `https://www.ocpgroup.ma/`. Accessed on 08.10.2021. Oct. 2021.

[25] Ambros Gleixner et al. "MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library". In: *Mathematical Programming Computation* (2021), pp. 1–48.

[26] John E Beasley. "An algorithm for solving large capacitated warehouse location problems". In: *European Journal of Operational Research* 33.3 (1988), pp. 314–325.

[27] Merve Bodur et al. "Strengthened Benders cuts for stochastic integer programs with continuous recourse". In: *INFORMS Journal on Computing* 29.1 (2017), pp. 77–91.

[28] Enda Ridge and Daniel Kudenko. "Tuning the Performance of the MMAS Heuristic". In: *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, International Workshop, SLS 2007, Brussels, Belgium, September 6-8, 2007, Proceedings.* Vol. 4638. Lecture Notes in Computer Science. Springer, 2007, pp. 46–60. DOI: `10.1007/978-3-540-74446-7\_4`. URL: `https://doi.org/10.1007/978-3-540-74446-7%5C_4`.

[29] L. F. Morán-Mirabal, José Luis González Velarde, and Mauricio G. C. Resende. "Automatic Tuning of GRASP with Evolutionary Path-Relinking". In: *Hybrid Metaheuristics - 8th International Workshop, HM 2013, Ischia, Italy, May 23-25, 2013. Proceedings.* Vol. 7919. Lecture Notes in Computer Science. Springer, 2013, pp. 62–77. DOI: `10.1007/978-3-642-38516-2\_6`. URL: `https://doi.org/10.1007/978-3-642-38516-2%5C_6`.

[30] Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss. "Sequential parameter optimization". In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK.* IEEE, 2005, pp. 773–780. DOI: `10.1109/CEC.2005.1554761`. URL: `https://doi.org/10.1109/CEC.2005.1554761`.

[31] Frank Hutter et al. "An experimental investigation of model-based parameter optimisation: SPO and beyond". In: *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009.* Ed. by Franz Rothlauf. ACM, 2009, pp. 271–278. DOI: `10.1145/1569901.1569940`. URL: `https://doi.org/10.1145/1569901.1569940`.

[32] Frank Hutter et al. "Time-Bounded Sequential Parameter Optimization". In: *Learning and Intelligent Optimization, 4th International Conference, LION 4, Venice, Italy, January 18-22, 2010. Selected Papers.* Vol. 6073. Lecture Notes in Computer Science. Springer, 2010, pp. 281–298. DOI: `10.1007/978-3-642-13800-3\_30`. URL: `https://doi.org/10.1007/978-3-642-13800-3%5C_30`.

[33] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. “Sequential model-based optimization for general algorithm configuration (extended version)”. In: *Technical Report TR-2010–10, University of British Columbia, Computer Science, Tech. Rep.* (2010).

[34] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. “Sequential Model-Based Optimization for General Algorithm Configuration”. In: *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*. Vol. 6683. Lecture Notes in Computer Science. Springer, 2011, pp. 507–523. DOI: `10.1007/978-3-642-25566-3\_40`. URL: `https://doi.org/10.1007/978-3-642-25566-3%5C_40`.

[35] Mustafa Baz et al. *Automated tuning of optimization software parameters*. Tech. rep. TR2007-7, University of Pittsburgh, Department of Industrial Engineering, 2007.

[36] Mustafa Baz, Brady Hunsaker, and Oleg A. Prokopyev. “How much do we "pay" for using default parameters?” In: *Comput. Optim. Appl.* 48.1 (2011), pp. 91–108. DOI: `10.1007/s10589-009-9238-5`. URL: `https://doi.org/10.1007/s10589-009-9238-5`.

[37] Belarmino Adenso-Díaz and Manuel Laguna. “Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search”. In: *Oper. Res.* 54.1 (2006), pp. 99–114. DOI: `10.1287/opre.1050.0243`. URL: `https://doi.org/10.1287/opre.1050.0243`.

[38] Mauro Birattari et al. “A Racing Algorithm for Configuring Metaheuristics”. In: *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002*. Morgan Kaufmann, 2002, pp. 11–18.

[39] Mauro Birattari et al. “F-Race and Iterated F-Race: An Overview”. In: *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010, pp. 311–336. DOI: `10.1007/978-3-642-02538-9\_13`. URL: `https://doi.org/10.1007/978-3-642-02538-9%5C_13`.

[40] Manuel López-Ibáñez et al. “The irace package: Iterated racing for automatic algorithm configuration”. In: *Operations Research Perspectives* 3 (2016), pp. 43–58.

[41] Manuel López-Ibáñez and Thomas Stützle. “Automatically improving the anytime behaviour of optimisation algorithms”. In: *Eur. J. Oper. Res.* 235.3 (2014), pp. 569–582. DOI: `10.1016/j.ejor.2013.10.043`. URL: `https://doi.org/10.1016/j.ejor.2013.10.043`.

[42] Frank Hutter et al. "AClib: A benchmark library for algorithm configuration". In: *International Conference on Learning and Intelligent Optimization*. Springer. 2014, pp. 36–40.

[43] Leslie Pérez Cáceres et al. "An Experimental Study of Adaptive Capping in irace". In: *Learning and Intelligent Optimization - 11th International Conference, LION 11, Nizhny Novgorod, Russia, June 19-21, 2017, Revised Selected Papers*. Vol. 10556. Lecture Notes in Computer Science. Springer, 2017, pp. 235–250. DOI: `10.1007/978-3-319-69404-7\_17`. URL: `https://doi.org/10.1007/978-3-319-69404-7%5C_17`.

[44] Marie Anastacio and Holger H. Hoos. "Model-Based Algorithm Configuration with Default-Guided Probabilistic Sampling". In: *Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I*. Vol. 12269. Lecture Notes in Computer Science. Springer, 2020, pp. 95–110. DOI: `10.1007/978-3-030-58112-1\_7`. URL: `https://doi.org/10.1007/978-3-030-58112-1%5C_7`.

[45] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "Automated Configuration of Mixed Integer Programming Solvers". In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010, Bologna, Italy, June 14-18, 2010. Proceedings*. Vol. 6140. Lecture Notes in Computer Science. Springer, 2010, pp. 186–202. DOI: `10.1007/978-3-642-13520-0\_23`. URL: `https://doi.org/10.1007/978-3-642-13520-0%5C_23`.

[46] Frank Hutter, Holger H. Hoos, and Thomas Stützle. "Automatic Algorithm Configuration Based on Local Search". In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*. AAAI Press, 2007, pp. 1152–1157. URL: `http://www.aaai.org/Library/AAAI/2007/aaai07-183.php`.

[47] Frank Hutter et al. "ParamILS: An Automatic Algorithm Configuration Framework". In: *J. Artif. Intell. Res.* 36 (2009), pp. 267–306. DOI: `10.1613/jair.2861`. URL: `https://doi.org/10.1613/jair.2861`.

[48] *ParamILS. ParamILS Website*. `http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/`. Accessed on 26.10.2021. Oct. 2021.

[49] Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. "A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms". In: *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*. Vol. 5732. Lecture Notes in

Computer Science. Springer, 2009, pp. 142–157. DOI: 10.1007/978-3-642-04244-7_14. URL: https://doi.org/10.1007/978-3-642-04244-7%5C_14.

[50] Thomas Bartz-Beielstein. “The New Experimentalism”. In: *Experimental Research in Evolutionary Computation: The New Experimentalism* (2006), pp. 13–39.

[51] Yasemin Eryoldaş and Alptekin Durmuşoglu. “A literature survey on offline automatic algorithm configuration”. In: *Applied Sciences* 12.13 (2022), p. 6316.

[52] Holger H. Hoos. “Automated Algorithm Configuration and Parameter Tuning”. In: *Autonomous Search*. Springer, 2012, pp. 37–71. DOI: 10.1007/978-3-642-21434-9_3. URL: https://doi.org/10.1007/978-3-642-21434-9%5C_3.

[53] Changwu Huang, Yuanxiang Li, and Xin Yao. “A Survey of Automatic Parameter Tuning Methods for Metaheuristics”. In: *IEEE Trans. Evol. Comput.* 24.2 (2020), pp. 201–216. DOI: 10.1109/TEVC.2019.2921598. URL: https://doi.org/10.1109/TEVC.2019.2921598.

[54] Elias Schede et al. “A Survey of Methods for Automated Algorithm Configuration”. In: *J. Artif. Intell. Res.* 75 (2022), pp. 425–487. DOI: 10.1613/jair.1.13676. URL: https://doi.org/10.1613/jair.1.13676.

[55] Celso C Ribeiro and Pierre Hansen. *Essays and surveys in metaheuristics*. Vol. 15. Springer Science & Business Media, 2012.

[56] Ilhem Boussaı************ERROR HERE!HERE!*******************d, Julien Lepagnot, and Patrick Siarry. “A survey on optimization metaheuristics”. In: *Information sciences* 237 (2013), pp. 82–117.

[57] Christian Blum et al. “Hybrid metaheuristics in combinatorial optimization: A survey”. In: *Applied soft computing* 11.6 (2011), pp. 4135–4151.

[58] Sedigheh Mahdavi, Mohammad Ebrahim Shiri, and Shahryar Rahnamayan. “Metaheuristics in large-scale global continues optimization: A survey”. In: *Information Sciences* 295 (2015), pp. 407–428.

[59] Paul Shaw. “A new local search algorithm providing high-quality solutions to vehicle routing problems”. In: *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK* 46 (1997).

[60] David Pisinger and Stefan Ropke. “Large neighborhood search”. In: *Handbook of metaheuristics*. Springer, 2019, pp. 99–127.

[61] Ravindra K Ahuja et al. “A survey of very large-scale neighborhood search techniques”. In: *Discrete Applied Mathematics* 123.1-3 (2002), pp. 75–102.

[62] John Baxter. “Local optima avoidance in depot location”. In: *Journal of the Operational Research Society* 32.9 (1981), pp. 815–819.

[63] Olivier C. Martin and Steve W. Otto. “Combining simulated annealing with local search heuristics”. In: *Ann. Oper. Res.* 63.1 (1996), pp. 57–75. DOI: `10.1007/BF02601639`. URL: `https://doi.org/10.1007/BF02601639`.

[64] Olivier Martin, Steve W Otto, and Edward W Felten. *Large-step Markov chains for the traveling salesman problem.* Citeseer, 1991.

[65] Eric B Baum. “Towards practical ‘neural’computation for combinatorial optimization problems”. In: *AIP Conference Proceedings*. Vol. 151. American Institute of Physics. 1986, pp. 53–58.

[66] David S. Johnson. “Local Optimization and the Traveling Salesman Problem”. In: *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*. Vol. 443. Lecture Notes in Computer Science. Springer, 1990, pp. 446–461. DOI: `10.1007/BFb0032050`. URL: `https://doi.org/10.1007/BFb0032050`.

[67] David S Johnson and Lyle A McGeoch. “The traveling salesman problem: A case study in local optimization”. In: *Local search in Combinatorial Optimization* 1.1 (1997), pp. 215–310.

[68] EB Baum. “Iterated descent: A better algorithm for local search in combinatorial optimization problems”. In: *Manuscript* (1986).

[69] Thomas Stützle. “Local search algorithms for combinatorial problems: analysis, improvements, and new applications”. In: (1999).

[70] Richard K. Congram, Chris N. Potts, and Steef L. van de Velde. “An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem”. In: *INFORMS J. Comput.* 14.1 (2002), pp. 52–67. DOI: `10.1287/ijoc.14.1.52.7712`. URL: `https://doi.org/10.1287/ijoc.14.1.52.7712`.

[71] Peter Brucker, Johann L. Hurink, and Frank Werner. “Improving Local Search Heuristics for Some Scheduling Problems-I”. In: *Discret. Appl. Math.* 65.1-3 (1996), pp. 97–122. DOI: `10.1016/0166-218X(95)00030-U`. URL: `https://doi.org/10.1016/0166-218X(95)00030-U`.

[72] Olivier C. Martin and Steve W. Otto. “Partitioning of unstructured meshes for load balancing”. In: *Concurr. Pract. Exp.* 7.4 (1995), pp. 303–314. DOI: `10.1002/cpe.4330070404`. URL: `https://doi.org/10.1002/cpe.4330070404`.

[73] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. "Iterated Local Search". In: *Handbook of Metaheuristics*. Vol. 57. International Series in Operations Research & Management Science. Kluwer / Springer, 2003, pp. 320–353. DOI: `10.1007/0-306-48056-5\_11`. URL: `https://doi.org/10.1007/0-306-48056-5%5C_11`.

[74] George B Dantzig and Philip Wolfe. "Decomposition principle for linear programs". In: *Operations research* 8.1 (1960), pp. 101–111.

[75] J. F. Benders. "Partitioning procedures for solving mixed-variables programming problems". In: *Numerische mathematik* 4.1 (1962), pp. 238–252.

[76] Monique Guignard and Siwhan Kim. "Lagrangean decomposition for integer programming: theory and applications". In: *RAIRO-Operations Research-Recherche Opérationnelle* 21.4 (1987), pp. 307–323.

[77] Antonio J Conejo et al. *Decomposition techniques in mathematical programming: engineering and science applications*. Springer Science & Business Media, 2006.

[78] Jacques Desrosiers, François Soumis, and Martin Desrochers. "Routing with time windows by column generation". In: *Networks* 14.4 (1984), pp. 545–565.

[79] Issmail Elhallaoui et al. "Dynamic aggregation of set-partitioning constraints in column generation". In: *Operations Research* 53.4 (2005), pp. 632–645.

[80] Marco E Lübbecke and Jacques Desrosiers. "Selected topics in column generation". In: *Operations research* 53.6 (2005), pp. 1007–1023.

[81] Yossiri Adulyasak, Jean-François Cordeau, and Raf Jans. "Benders decomposition for production routing under demand uncertainty". In: *Operations Research* 63.4 (2015), pp. 851–867.

[82] Yiling Zhang, Mengshi Lu, and Siqian Shen. "On the values of vehicle-to-grid electricity selling in electric vehicle sharing". In: *Manufacturing & Service Operations Management* 23.2 (2021), pp. 488–507.

[83] Vahid Zeighami and François Soumis. "Combining Benders' decomposition and column generation for integrated crew pairing and personalized crew assignment problems". In: *Transportation Science* 53.5 (2019), pp. 1479–1499.

[84] Jean-François Cordeau et al. "Benders decomposition for simultaneous aircraft routing and crew scheduling". In: *Transportation science* 35.4 (2001), pp. 375–388.

[85] Ximing Cai et al. "Solving large nonconvex water resources management models using generalized Benders decomposition". In: *Operations Research* 49.2 (2001), pp. 235–245.

[86] Sheng Liu and Zhixing Luo. “On-demand delivery from stores: Dynamic dispatching and routing with random demand”. In: *Manufacturing & Service Operations Management* 25.2 (2023), pp. 595–612.

[87] Ivan Contreras, Jean-François Cordeau, and Gilbert Laporte. “Benders decomposition for large-scale uncapacitated hub location”. In: *Operations research* 59.6 (2011), pp. 1477–1490.

[88] Jean-François Cordeau, François Soumis, and Jacques Desrosiers. “Simultaneous assignment of locomotives and cars to passenger trains”. In: *Operations research* 49.4 (2001), pp. 531–548.

[89] Gilbert Laporte, Francois V Louveaux, and Hélene Mercure. “A priori optimization of the probabilistic traveling salesman problem”. In: *Operations research* 42.3 (1994), pp. 543–549.

[90] Lingxiao Wu et al. “Vessel service planning in seaports”. In: *Operations Research* 70.4 (2022), pp. 2032–2053.

[91] Jeremy A Bloom. “Solving an electricity generating capacity expansion planning problem by generalized Benders’ decomposition”. In: *Operations Research* 31.1 (1983), pp. 84–100.

[92] Milad Keshvari Fard, Ivana Ljubić, and Felix Papier. “Budgeting in international humanitarian organizations”. In: *Manufacturing & Service Operations Management* 24.3 (2022), pp. 1562–1577.

[93] Thomas L Magnanti and Richard T Wong. “Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria”. In: *Operations research* 29.3 (1981), pp. 464–484.

[94] Gianni Codato and Matteo Fischetti. “Combinatorial Benders’ cuts for mixed-integer linear programming”. In: *Operations Research* 54.4 (2006), pp. 756–766.

[95] Matteo Fischetti, Domenico Salvagnin, and Arrigo Zanette. “A note on the selection of Benders’ cuts”. In: *Mathematical Programming* 124.1 (2010), pp. 175–182.

[96] Matteo Fischetti, Ivana Ljubić, and Markus Sinnl. “Redesigning Benders decomposition for large-scale facility location”. In: *Management Science* 63.7 (2017), pp. 2146–2162.

[97] Ragheb Rahmaniani et al. “The Benders dual decomposition method”. In: *Operations Research* 68.3 (2020), pp. 878–895.

[98] Ragheb Rahmaniani et al. “The Benders decomposition algorithm: A literature review”. In: *European Journal of Operational Research* 259.3 (2017), pp. 801–817.

[99] Pascal Van Hentenryck and Kevin Dalmeijer. “AI4OPT: AI Institute for Advances in Optimization”. In: *AI Magazine* (2024).

[100] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. “Machine learning for combinatorial optimization: a methodological tour d’horizon”. In: *European Journal of Operational Research* (2020).

[101] Elias B Khalil et al. “Learning to Run Heuristics in Tree Search”. In: *IJCAI*. 2017, pp. 659–666.

[102] Gregor Hendel. “Adaptive large neighborhood search for mixed integer programming”. In: *Mathematical Programming Computation* 14.2 (2022), pp. 185–221.

[103] Jian-Ya Ding et al. “Accelerating primal solution findings for mixed integer programs based on solution prediction”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 02. 2020, pp. 1452–1459.

[104] Álinson S Xavier, Feng Qiu, and Shabbir Ahmed. “Learning to solve large-scale security-constrained unit commitment problems”. In: *INFORMS Journal on Computing* (2020).

[105] Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. “A machine learning-based approximation of strong branching”. In: *INFORMS Journal on Computing* 29.1 (2017), pp. 185–195.

[106] Maxime Gasse et al. “Exact combinatorial optimization with graph convolutional neural networks”. In: *arXiv preprint arXiv:1906.01629* (2019).

[107] Giulia Zarpellon et al. “Parameterizing branch-and-bound search trees to learn branching policies”. In: *arXiv preprint arXiv:2002.05120* (2020).

[108] Maria-Florina Balcan et al. “Learning to branch”. In: *International conference on machine learning*. PMLR. 2018, pp. 344–353.

[109] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. “Reinforcement learning for integer programming: Learning to cut”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9367–9376.

[110] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. “Automated configuration of mixed integer programming solvers”. In: *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. Springer. 2010, pp. 186–202.

[111] Mouad Morabit, Guy Desaulniers, and Andrea Lodi. "Machine-learning-based column selection for column generation". In: *Transportation Science* 55.4 (2021), pp. 815–831.

[112] Adil Tahir et al. "An improved integral column generation algorithm using machine learning for aircrew pairing". In: *Transportation Science* 55.6 (2021), pp. 1411–1429.

[113] Frédéric Quesnel et al. "Deep-learning-based partial pricing in a branch-and-price algorithm for personalized crew rostering". In: *Computers & Operations Research* 138 (2022), p. 105554.

[114] Mouad Morabit, Guy Desaulniers, and Andrea Lodi. "Machine-learning-based arc selection for constrained shortest path problems in column generation". In: *INFORMS Journal on Optimization* 5.2 (2023), pp. 191–210.

[115] Marius Lindauer et al. "SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization". In: *J. Mach. Learn. Res.* 23 (2022), 54:1–54:9. URL: `http://jmlr.org/papers/v23/21-0888.html`.

[116] Thomas Stützle. "Local search algorithms for combinatorial problems - analysis, improvements, and new applications". PhD thesis. Darmstadt University of Technology, Germany, 1999. ISBN: 978-3-89601-220-3. URL: `https://d-nb.info/95811515X`.

[117] Frank Hutter, Holger H. Hoos, and Thomas Stützle. "Automatic Algorithm Configuration Based on Local Search". In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada.* AAAI Press, 2007, pp. 1152–1157. URL: `http://www.aaai.org/Library/AAAI/2007/aaai07-183.php`.

[118] Mauro Birattari. *Tuning Metaheuristics - A Machine Learning Perspective.* Vol. 197. Studies in Computational Intelligence. Springer, 2009. ISBN: 978-3-642-00482-7. DOI: `10.1007/978-3-642-00483-4`. URL: `https://doi.org/10.1007/978-3-642-00483-4`.

[119] Ed Klotz and Alexandra M Newman. "Practical guidelines for solving difficult mixed integer linear programs". In: *Surveys in Operations Research and Management Science* 18.1-2 (2013), pp. 18–32.

[120] Henry Scheffe. *The analysis of variance.* Vol. 72. John Wiley & Sons, 1999.

[121] Katherine McGuire and Kamala London. "Factorial ANOVA". In: *Research Methods in Psychology* (2020), p. 417.

[122] Kevin R Murphy, Brett Myors, and Allen Wolach. *Statistical power analysis: A simple and general model for traditional and modern hypothesis tests.* Routledge, 2014.

[123] Ronald L Wasserstein and Nicole A Lazar. “The ASA statement on p-values: Context, process, and purpose”. In: *The American Statistician* 70.2 (2016), pp. 129–133.

[124] Charu C Aggarwal. “Data classification”. In: *Data Mining*. Springer. 2015, pp. 285–344.

[125] Chanyeong Kwak and Alan Clayton-Matthews. “Multinomial logistic regression”. In: *Nursing Research* 51.6 (2002), pp. 404–410.

[126] Jason E King. “Binary logistic regression”. In: *Best Practices in Quantitative Methods* (2008), pp. 358–384.

[127] Manuel López-Ibáñez et al. *The irace package: User guide*. IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Université Libre de Bruxelles, 2016.

[128] IBM ILOG Cplex. “V12. 1: User’s manual for CPLEX”. In: *International Business Machines Corporation* 46.53 (2009), p. 157.

[129] Jade Man-yin Lee and Eugene Yin-cheung Wong. “Suez Canal blockage: An analysis of legal impact, risks and liabilities to the global supply chain”. In: *MATEC Web of Conferences*. Vol. 339. EDP Sciences. 2021.

[130] Ambros M. Gleixner et al. “MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library”. In: *Math. Program. Comput.* 13.3 (2021), pp. 443–490. DOI: 10.1007/s12532-020-00194-3. URL: https://doi.org/10.1007/s12532-020-00194-3.

[131] John T Mentzer et al. “Defining supply chain management”. In: *Journal of Business Logistics* 22.2 (2001), pp. 1–25.

[132] David H Taylor. *Global cases in logistics and supply chain management*. Cengage Learning EMEA, 1997.

[133] Philippe-Pierre Dornier et al. *Global operations and logistics: Text and cases*. John Wiley & Sons, 2008.

[134] Marc Humphries. *Rare earth elements: The global supply chain*. Diane Publishing, 2010.

[135] Morris A Cohen and Hau L Lee. “Designing the right global supply chain network”. In: *Manufacturing & Service Operations Management* 22.1 (2020), pp. 15–24.

[136] Scott W O’Leary-Kelly and Benito E Flores. “The integration of manufacturing and marketing/sales decisions: Impact on organizational performance”. In: *Journal of Operations Management* 20.3 (2002), pp. 221–240.

[137] Ram Narasimhan and Jayanth Jayaram. “Causal linkages in supply chain management: An exploratory study of North American manufacturing firms”. In: *Decision Sciences* 29.3 (1998), pp. 579–605.

[138] Markham T Frohlich and Roy Westbrook. “Arcs of integration: An international study of supply chain strategies”. In: *Journal of Operations Management* 19.2 (2001), pp. 185–200.

[139] Kathleen Iacocca, Stephen Mahar, and P Daniel Wright. “Strategic horizontal integration for drug cost reduction in the pharmaceutical supply chain”. In: *Omega* 108 (2022), p. 102589.

[140] Graham C Stevens. “Integrating the supply chain”. In: *International Journal of Physical Distribution & Materials Management* (1989).

[141] Helio Yochihiro Fuchigami and Socorro Rangel. “A survey of case studies in production scheduling: Analysis and perspectives”. In: *Journal of Computational Science* 25 (2018), pp. 425–436.

[142] Frederick A Rodammer and K Preston White. “A recent survey of production scheduling”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 18.6 (1988), pp. 841–851.

[143] Stephen C Graves. “A review of production scheduling”. In: *Operations Research* 29.4 (1981), pp. 646–675.

[144] Enrique Jélvez et al. “A new hybrid heuristic algorithm for the Precedence Constrained Production Scheduling Problem: A mining application”. In: *Omega* 94 (2020), p. 102046.

[145] Orlando Rivera Letelier et al. “Production scheduling for strategic open pit mine planning: a mixed-integer programming approach”. In: *Operations Research* 68.5 (2020), pp. 1425–1444.

[146] Edward Allen Silver, David F Pyke, and Rein Peterson. *Inventory Management and Production Planning and Scheduling, 3rd Edition.* Vol. 3. Wiley New York, 1998.

[147] Christian Bierwirth and Frank Meisel. “A follow-up survey of berth allocation and quay crane scheduling problems in container terminals”. In: *European Journal of Operational Research* 244.3 (2015), pp. 675–689.

[148] Christian Bierwirth and Frank Meisel. “A survey of berth allocation and quay crane scheduling problems in container terminals”. In: *European Journal of Operational Research* 202.3 (2010), pp. 615–627.

[149] Thys B Johnson. *Optimum open pit mine production scheduling.* Tech. rep. California Univ Berkeley Operations Research Center, 1968.

[150] Louis Caccetta and Stephen P Hill. "An application of branch and cut to open pit mine scheduling". In: *Journal of Global Optimization* 27.2-3 (2003), pp. 349–365.

[151] Renaud Chicoisne et al. "A new algorithm for the open-pit mine production scheduling problem". In: *Operations Research* 60.3 (2012), pp. 517–528.

[152] Akio Imai, Etsuko Nishimura, and Stratos Papadimitriou. "The dynamic berth allocation problem for a container port". In: *Transportation Research Part B: Methodological* 35.4 (2001), pp. 401–417.

[153] Akio Imai et al. "Berth allocation in a container port: Using a continuous location space approach". In: *Transportation Research Part B: Methodological* 39.3 (2005), pp. 199–221.

[154] Ali Diabat and Effrosyni Theodorou. "An integrated quay crane assignment and scheduling problem". In: *Computers & Industrial Engineering* 73 (2014), pp. 115–123.

[155] Agostinho Agra and Maryse Oliveira. "MIP approaches for the integrated berth allocation and quay crane assignment and scheduling problem". In: *European Journal of Operational Research* 264.1 (2018), pp. 138–148.

[156] Daina R Dennis and Jack R Meredith. "An analysis of process industry production and inventory management systems". In: *Journal of Operations Management* 18.6 (2000), pp. 683–699.

[157] M Abdur Rahim and Mohamed Ben-Daya. *Integrated models in production planning, inventory, quality, and maintenance.* Springer Science & Business Media, 2012.

[158] Henrik Andersson et al. "Creating annual delivery programs of liquefied natural gas". In: *Optimization and Engineering* 18.1 (2017), pp. 299–316.

[159] Gustavo Campos Menezes, Geraldo Robson Mateus, and Martín Gómez Ravetti. "A branch and price algorithm to solve the integrated production planning and scheduling in bulk ports". In: *European Journal of Operational Research* 258.3 (2017), pp. 926–937.

[160] X-T Sun, Sai Ho Chung, and Felix T-S Chan. "Integrated scheduling of a multi-product multi-factory manufacturing system with maritime transport limits". In: *Transportation Research Part E: Logistics and Transportation Review* 79 (2015), pp. 110–127.

[161] Arne Herzel, Stefan Ruzika, and Clemens Thielen. “Approximation methods for multiobjective optimization problems: A survey”. In: *INFORMS Journal on Computing* 33.4 (2021), pp. 1284–1299.

[162] R Timothy Marler and Jasbir S Arora. “Survey of multi-objective optimization methods for engineering”. In: *Structural and Multidisciplinary Optimization* 26.6 (2004), pp. 369–395.

[163] Songsong Liu and Lazaros G Papageorgiou. “Multiobjective optimisation of production, distribution and capacity planning of global supply chains in the process industry”. In: *Omega* 41.2 (2013), pp. 369–382.

[164] Joshua T Margolis et al. “A multi-objective optimization model for designing resilient supply chain networks”. In: *International Journal of Production Economics* 204 (2018), pp. 174–185.

[165] Ahmed M Ghaithan, Ahmed Attia, and Salih O Duffuaa. “Multi-objective optimization model for a downstream oil and gas supply chain”. In: *Applied Mathematical Modelling* 52 (2017), pp. 689–708.

[166] H Giray Resat and Berkcan Unsal. “A novel multi-objective optimization approach for sustainable supply chain: A case study in packaging industry”. In: *Sustainable Production and Consumption* 20 (2019), pp. 29–39.

[167] Fulya Altiparmak et al. “A genetic algorithm approach for multi-objective optimization of supply chain networks”. In: *Computers & industrial engineering* 51.1 (2006), pp. 196–215.

[168] Aliakbar Hasani, Hadi Mokhtari, and Mohammad Fattahi. “A multi-objective optimization approach for green and resilient supply chain network design: A real-life case study”. In: *Journal of Cleaner Production* 278 (2021), p. 123199.

[169] Ernesto Mastrocinque et al. “A multi-objective optimization for supply chain network using the bees algorithm”. In: *International Journal of Engineering Business Management* 5 (2013), p. 38.

[170] Aimin Zhou et al. “Multiobjective evolutionary algorithms: A survey of the state of the art”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 32–49.

[171] Ines Alaya, Christine Solnon, and Khaled Ghedira. “Ant colony optimization for multiobjective optimization problems”. In: *19th IEEE international conference on tools with artificial intelligence (ICTAI 2007).* Vol. 1. IEEE. 2007, pp. 450–457.

[172] Abbas Moradi, A Mirzakhani Nafchi, and A Ghanbarzadeh. "Multi-objective optimization of truss structures using Bees Algorithm". In: *Scientia Iranica* 22.5 (2015), pp. 1789–1800.

[173] Trisna Trisna et al. "Multi-objective optimization for supply chain management problem: A literature review". In: *Decision Science Letters* 5.2 (2016), pp. 283–316.

[174] Mostafa Naghavi, Ali Asghar Foroughi, and Masoud Zarepisheh. "Inverse optimization for multi-objective linear programming". In: *Optimization Letters* 13.2 (2019), pp. 281–294.

[175] Timothy CY Chan, Taewoo Lee, and Daria Terekhov. "Inverse optimization: Closed-form solutions, geometry, and goodness of fit". In: *Management Science* 65.3 (2019), pp. 1115–1135.

[176] Ravindra K Ahuja et al. "Very large-scale neighborhood search: Theory, algorithms, and applications". In: *Handbook of approximation algorithms and metaheuristics, Second Edition*. Chapman and Hall/CRC, 2018, pp. 311–326.

[177] Mineral Commodity Summaries. "Mineral commodity summaries". In: *US Geological Survey: Reston, VA, USA* 200 (2021).

[178] Hermann Auernhammer. "Precision farming: The environmental challenge". In: *Computers and Electronics in Agriculture* 30.1-3 (2001), pp. 31–43.

[179] Vilfredo Pareto. "Manuale di economia politica, Milano". In: *Società Editrice Libraria* (1906).

[180] K J Arrow, E W Barankin, and D_ Blackwell. "Admissible points of convex sets". In: *Contributions to the Theory of Games* 2 (1953), pp. 87–91.

[181] Kaisa Miettinen. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media, 2012.

[182] Andrzej P Wierzbicki. "On the completeness and constructiveness of parametric characterizations to vector optimization problems". In: *Operations-Research-Spektrum* 8.2 (1986), pp. 73–87.

[183] Merve Bodur et al. "Decomposition of loosely coupled integer programs: A multiobjective perspective". In: *Mathematical Programming* (2022), pp. 1–51.

[184] David Pisinger and Stefan Ropke. "Large neighborhood search". In: *Handbook of metaheuristics*. Springer, 2010, pp. 399–419.

[185] James Ostrowski. *Symmetry in integer programming*. Lehigh University, 2009.

[186] François Margot. “Symmetry in integer linear programming”. In: *50 Years of Integer Programming 1958-2008* (2010), pp. 647–686.

[187] Iain McDonald and Barbara Smith. “Partial symmetry breaking”. In: *International conference on principles and practice of constraint programming*. Springer. 2002, pp. 431–445.

[188] Jimmy H M Lee and Jingying Li. “Increasing symmetry breaking by preserving target symmetries”. In: *International conference on principles and practice of constraint programming*. Springer. 2012, pp. 422–438.

[189] Paul Shaw. “Using constraint programming and local search methods to solve vehicle routing problems”. In: *International conference on principles and practice of constraint programming*. Springer. 1998, pp. 417–431.

[190] Philippe Laborie and Daniel Godard. “Self-adapting large neighborhood search: Application to single-mode scheduling problems”. In: *Proceedings MISTA-07, Paris* 8 (2007).

[191] Golnar Behzadi et al. “Agribusiness supply chain risk management: A review of quantitative decision models”. In: *Omega* 79 (2018), pp. 21–42.

[192] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

[193] Arnold Neumaier and Oleg Shcherbina. “Safe bounds in linear and mixed-integer linear programming”. In: *Mathematical Programming* 99 (2004), pp. 283–296.

[194] Arthur M Geoffrion. “Elements of large-scale mathematical programming Part I: Concepts”. In: *Management Science* 16.11 (1970), pp. 652–675.

[195] François Vanderbeck. “On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm”. In: *Operations research* 48.1 (2000), pp. 111–128.

[196] David Murray Ryan and Michael Robert Osborne. “On the solution of highly degenerate linear programmes”. In: *Mathematical programming* 41 (1988), pp. 385–392.

[197] Jack Brimberg and Nenad Mladenovic. “Degeneracy in the multi-source Weber problem.” In: *Mathematical Programming* 85.1 (1999).

[198] Andrew R. Conn and Gerard Cornuejols. “A projection method for the uncapacitated facility location problem”. In: *Mathematical programming* 46 (1990), pp. 273–298.

[199] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. “An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts”. In: *Mathematical Programming* 115 (2008), pp. 351–385.

[200] José M Valério de Carvalho. “Using extra dual cuts to accelerate column generation”. In: *INFORMS Journal on Computing* 17.2 (2005), pp. 175–182.

[201] Hatem Ben Amor, Jacques Desrosiers, and José Manuel Valério de Carvalho. “Dual-optimal inequalities for stabilized column generation”. In: *Operations Research* 54.3 (2006), pp. 454–463.

[202] Paul Wentges. “Accelerating Benders’ decomposition for the capacitated facility location problem”. In: *Mathematical Methods of Operations Research* 44.2 (1996), pp. 267–290.

[203] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. “An analysis of approximations for maximizing submodular set functions—I”. In: *Mathematical programming* 14.1 (1978), pp. 265–294.

[204] Donald Erlenkotter. “A dual-based procedure for uncapacitated facility location”. In: *Operations Research* 26.6 (1978), pp. 992–1009.

[205] Wilfried Eckl-Dorna and Alex Longley. *Shipper Hapag-Lloyd Says Red Sea Route Still Too Dangerous*. 2023. URL: `https://www.bloomberg.com/news/articles/2023-12-27/hapag-lloyd-says-red-sea-route-is-still-too-dangerous` (visited on 12/27/2023).

[206] Ruth Endam Mbah and Divine Forcha Wasum. “Russian-Ukraine 2022 War: A review of the economic impact of Russian-Ukraine crisis on the USA, UK, Canada, and Europe”. In: *Advances in Social Sciences Research Journal* 9.3 (2022), pp. 144–153.

[207] Huayu Shen et al. “The impact of the COVID-19 pandemic on firm performance”. In: *Emerging Markets Finance and Trade* 56.10 (2020), pp. 2213–2230.

[208] Georgij Alekseev et al. “The effects of COVID-19 on US small businesses: evidence from owners, managers, and employees”. In: *Management Science* 69.1 (2023), pp. 7–24.

[209] Steve Gliessman. “Why is there a food crisis?” In: *Agroecology and Sustainable Food Systems* 46.9 (2022), pp. 1301–1303.

[210] Rosa Ferrer and Helena Perrone. “Consumers’ Costly Responses to Product-Harm Crises”. In: *Management Science* 69.5 (2023), pp. 2639–2671.

[211] Monika Winn et al. “Impacts from climate change on organizations: a conceptual foundation”. In: *Business Strategy and the Environment* 20.3 (2011), pp. 157–173.

[212] Nora Pankratz, Rob Bauer, and Jeroen Derwall. “Climate change, firm performance, and investor surprises”. In: *Management Science* (2023).

[213] Henk Akkermans and Luk N Van Wassenhove. “Supply chain tsunamis: research on low-probability, high-impact disruptions”. In: *Journal of Supply Chain Management* 54.1 (2018), pp. 64–76.

[214] AP Barroso et al. “Quantifying the supply chain resilience”. In: *Applications of Contemporary Management Approaches in Supply Chains* 13 (2015), p. 38.

[215] Martin Christopher and Helen Peck. “Building the resilient supply chain.” In: *International Journal of Logistics Management* 15.2 (2004), pp. 1–13.

[216] Masoud Kamalahmadi and Mahour Mellat Parast. “A review of the literature on the principles of enterprise and supply chain resilience: Major findings and directions for future research”. In: *International Journal of Production Economics* 171 (2016), pp. 116–133.

[217] Seyedmohsen Hosseini, Dmitry Ivanov, and Alexandre Dolgui. “Review of quantitative methods for supply chain resilience analysis”. In: *Transportation Research Part E: Logistics and Transportation Review* 125 (2019), pp. 285–307.

[218] Pravin Suryawanshi and Pankaj Dutta. “Optimization models for supply chains under risk, uncertainty, and resilience: A state-of-the-art review and future research directions”. In: *Transportation research part E: logistics and transportation review* 157 (2022), p. 102553.

[219] Zhibin Yang et al. “Supply disruptions, asymmetric information, and a backup production option”. In: *Management Science* 55.2 (2009), pp. 192–209.

[220] Tianjian Yang and Weiguo Fan. “Information management strategies and supply chain performance under demand disruptions”. In: *International Journal of Production Research* 54.1 (2016), pp. 8–27.

[221] Sunil Chopra, ManMohan Sodhi, and Florian Lücker. “Achieving supply chain efficiency and resilience by using multi-level commons”. In: *Decision Sciences* 52.4 (2021), pp. 817–832.

[222] Lichun Chen and Elise Miller-Hooks. “Resilience: an indicator of recovery capability in intermodal freight transport”. In: *Transportation Science* 46.1 (2012), pp. 109–123.

[223] Shi An et al. “Reliable emergency service facility location under facility disruption, en-route congestion and in-facility queuing”. In: *Transportation Research Part E: Logistics and Transportation Review* 82 (2015), pp. 199–216.

[224] Abdullah A Khaled et al. “Train design and routing optimization for evaluating criticality of freight railroad infrastructures”. In: *Transportation Research Part B: Methodological* 71 (2015), pp. 71–84.

[225] Seyed Mohammad Khalili, Fariborz Jolai, and Seyed Ali Torabi. “Integrated production-distribution planning in two-echelon systems: a resilience view”. In: *International Journal of Production Research* 55.4 (2017), pp. 1040–1064.

[226] Navid Sahebjamnia, S Ali Torabi, and S Afshin Mansouri. “Building organizational resilience in the face of multiple disruptions”. In: *International Journal of Production Economics* 197 (2018), pp. 63–83.

[227] Sahitya Elluru et al. “Proactive and reactive models for disaster resilient supply chain”. In: *Annals of Operations Research* 283 (2019), pp. 199–224.

[228] Seyedmohsen Hosseini et al. “Resilient supplier selection and optimal order allocation under disruption risks”. In: *International Journal of Production Economics* 213 (2019), pp. 124–137.

[229] Tadeusz Sawik. “Two-period vs. multi-period model for supply chain disruption management”. In: *International Journal of Production Research* 57.14 (2019), pp. 4502–4518.

[230] Giuseppe Bruno et al. “Reorganizing postal collection operations in urban areas as a result of declining mail volumes–A case study in Bologna”. In: *Journal of the Operational Research Society* 72.7 (2021), pp. 1591–1606.

[231] A D’Ariano et al. “Running time re-optimization during real-time timetable perturbations”. In: *Timetable Planning and Information Quality* 1 (2010), pp. 147–156.

[232] Claudia Archetti, Gianfranco Guastaroba, and Maria Grazia Speranza. “Reoptimizing the rural postman problem”. In: *Computers & Operations Research* 40.5 (2013), pp. 1306–1313.

[233] Yachao Dong, Christos T Maravelias, and Norman F Jerome. “Reoptimization framework and policy analysis for maritime inventory routing under uncertainty”. In: *Optimization and Engineering* 19 (2018), pp. 937–976.

[234] Baruch Schieber et al. “A theory and algorithms for combinatorial reoptimization”. In: *Algorithmica* 80 (2018), pp. 576–607.

[235] Benjamin Doerr, Carola Doerr, and Frank Neumann. “Fast re-optimization via structural diversity”. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* GECCO ’19. 2019, pp. 233–241.

[236] Rachid Hassani, Guy Desaulniers, and Issmail Elhallaoui. “Real-time personnel rescheduling after a minor disruption in the retail industry”. In: *Computers & Operations Research* 120 (2020), p. 104952.

[237] Rachid Hassani, Guy Desaulniers, and Issmail Elhallaoui. “Real-time bi-objective personnel re-scheduling in the retail industry”. In: *European Journal of Operational Research* 293.1 (2021), pp. 93–108.

[238] Gérard Cornuéjols. “Valid inequalities for mixed integer linear programs”. In: *Mathematical programming* 112.1 (2008), pp. 3–44.

[239] TK Ralphs and Menal Güzelsoy. “Duality and warm starting in integer programming”. In: *The proceedings of the 2006 NSF design, service, and manufacturing grantees and research conference*. 2006.

[240] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. “Machine learning for combinatorial optimization: a methodological tour d’horizon”. In: *European Journal of Operational Research* 290.2 (2021), pp. 405–421.

[241] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[242] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[243] Léon Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *19th International Conference on Computational Statistics, COMPSTAT 2010, Paris, France, August 22-27, 2010 - Keynote, Invited and Contributed Papers*. Physica-Verlag, 2010, pp. 177–186. DOI: `10.1007/978-3-7908-2604-3\_16`. URL: `https://doi.org/10.1007/978-3-7908-2604-3%5C_16`.

[244] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: `http://arxiv.org/abs/1412.6980`.

[245] Steven J Van Kauwenbergh. *World phosphate rock reserves and resources*. IFDC Muscle Shoals, 2010.

[246] James Cooper et al. “The future distribution and production of global phosphate rock reserves”. In: *Resources, Conservation and Recycling* 57 (2011), pp. 78–86.

[247] Mohammad Saghir Khan, Almas Zaidi, and Parvaze A Wani. “Role of phosphate-solubilizing microorganisms in sustainable agriculture: A review”. In: *Agronomy for Sustainable Development* 27.1 (2007), pp. 29–43.

[248] Dana Cordell, Jan-Olof Drangert, and Stuart White. “The story of phosphorus: Global food security and food for thought”. In: *Global Environmental Change* 19.2 (2009), pp. 292–305.

[249] Macky Sall, Akinwumi Adesina, and Patrick Verkooijen. *Time is running out to help Africa feed itself.* `https://www.afdb.org/en/news-and-events/time-running-out-help-africa-feed-itself-65923`. Accessed on 19.02.2024. Nov. 2023.

[250] Habtamu Mekonnen and Mulugeta Kibret. "The roles of plant growth promoting rhizobacteria in sustainable vegetable production in Ethiopia". In: *Chemical and Biological Technologies in Agriculture* 8.1 (2021), pp. 1–11.

[251] Adrian Becker et al. "Toward global food security: Transforming OCP through analytics". In: *Informs Journal on Applied Analytics* 52.1 (2022), pp. 90–107.

[252] Gaurav Singh et al. "Medium-term rail scheduling for an iron ore mining company". In: *Interfaces* 44.2 (2014), pp. 222–240.

[253] Yi Ding et al. "SGICT builds an optimization-based system for daily berth planning". In: *Interfaces* 46.4 (2016), pp. 281–296.

[254] Lu Zhen et al. "Mathematical Programming-Driven Daily Berth Planning in Xiamen Port". In: *INFORMS Journal on Applied Analytics* (2024).

[255] Michael F Gorman et al. "State of the Practice: A Review of the Application of OR/MS in Freight Transportation". In: *Interfaces* 44.6 (2014), pp. 535–554.

[256] Stephen C Graves, David B Kletter, and William B Hetzel. "A dynamic model for requirements planning with application to supply chain optimization". In: *Operations Research* 46.3-supplement-3 (1998), S35–S49.

[257] Stephen C Graves and Brian T Tomlin. "Process flexibility in supply chains". In: *Management Science* 49.7 (2003), pp. 907–919.

[258] Alexandra M Newman et al. "A review of operations research in mine planning". In: *Interfaces* 40.3 (2010), pp. 222–245.

[259] Bruno Santos Pimentel, Geraldo Robson Mateus, and Franklin Assunção Almeida. "Mathematical models for optimizing the global mining supply chain". In: *Intelligent Systems in Operations: Methods, Models and Applications in the Supply Chain.* IGI Global, 2010, pp. 133–163.

[260] P Bodon et al. "Modeling the mining supply chain from mine to port: A combined optimization and simulation approach". In: *Journal of Mining Science* 47.2 (2011), pp. 202–211.

[261] Alexandra Newman and Andres Weintraub. "Introduction to the Interfaces special issue on operations research in mining". In: *Interfaces* 44.2 (2014), pp. 125–126.

[262] Ioannis Fragkos and Bert De Reyck. “Improving the maritime transshipment operations of the Noble Group”. In: *Interfaces* 46.3 (2016), pp. 203–217.

[263] Sylvie C Bouffard et al. “Discrete-event simulation modeling unlocks value for the Jansen potash project”. In: *Interfaces* 48.1 (2018), pp. 45–56.

[264] Ted Gifford et al. “Dispatch optimization in bulk tanker transport operations”. In: *Interfaces* 48.5 (2018), pp. 403–421.

[265] Frank Hutter et al. “ParamILS: An Automatic Algorithm Configuration Framework”. In: *J. Artif. Intell. Res.* 36 (2009), pp. 267–306. DOI: `10.1613/jair.2861`. URL: `https://doi.org/10.1613/jair.2861`.

[266] Judy E Scott and Iris Vessey. “Implementing enterprise resource planning systems: The role of learning from failure”. In: *Information Systems Frontiers* 2.2 (2000), pp. 213–232.

[267] Yajiong Xue et al. “ERP implementation failures in China: Case studies with implications for ERP vendors”. In: *International Journal of Production Economics* 97.3 (2005), pp. 279–295.

[268] Ali Danışman. “Good intentions and failed implementations: Understanding culture-based resistance to organizational change”. In: *European Journal of Work and Organizational Psychology* 19.2 (2010), pp. 200–220.

[269] Poonam Garg and Atul Garg. “An empirical study on critical failure factors for enterprise resource planning implementation in Indian retail sector”. In: *Business Process Management Journal* (2013).

APPENDIX A TROUBLESHOOTING OF MILP SOLVING

Several remedies have been proposed in the CPLEX documentation as well as other literature [119] in order to address the different issues that may affect the performance of the MILP solver. These remedies are mainly based on the mastery of the solvers' parameters and the MILP problems at hand. In this section, we report the five common sources of performance loss for MILP algorithms with some potential remedies. A summary appears in Figure A.1.

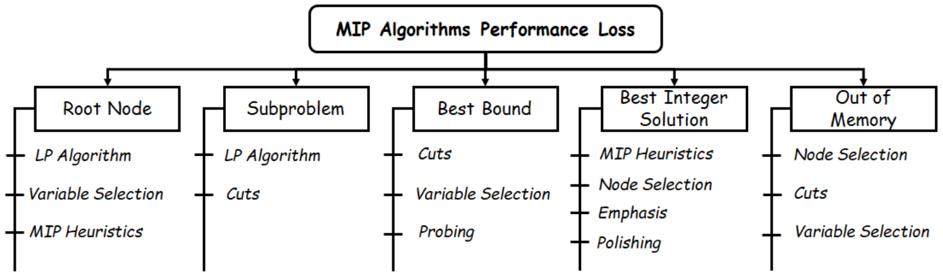


Figure A.1 Summary of MILP Performance Loss

Large solution time at the root node: for some models, the solver requires a significant amount of time before it starts branching. This time can be consumed solving the root relaxation, as it can be spent performing additional computations at the root node. In the first case, it can be helpful to change the linear program (LP) algorithm used to solve the root node to an interior point method. In the second case, two remedies can be tested. First, one could use a less expensive variable selection strategy which allows for a faster selection of the variable to branch. Second, one could turn off the MILP heuristics used to find integer feasible solutions around the current relaxed solutions.

Difficulties when solving the subproblems: the branch-and-bound type algorithms generate an enormous number of nodes. Each node has a proper MILP model whose linear relaxation is called a subproblem. In some cases, the solving of the subproblems may be more difficult than the solving of the initial problem at the root node. This weakness could be detected by comparing either the number of simplex iterations or the solution time needed to solve the initial LP problem and the subproblems. Similarly to the root node, one could alternate the algorithm used to solve the subproblems. Another possible source of this difficulty may be the use of a large number of unhelpful cuts. In this case, one could turn off a subset of cuts (or all of them) or reduce the frequency of their generation.

Lack of improvement in the best bound: for minimization problems, the best bound corresponds to the lower objective function value obtained after solving the LP relaxations in all active nodes. The ideal behavior of a MILP algorithm consists of tightening this limit more and more in order to reduce the gap it creates with the upper bound (the objective function value of the best integer solution available). For some MILP models, the best bound changes slightly or not at all. Some of the most recommended tuning changes are the following. First, one could add cuts aggressively to tighten the polyhedron of the linear relaxation. Second, consider a more-informed variable selection strategy such as strong branching. The strong branching selects a subset of the integer variables with fractional values in the node relaxation solution and explores both the up and down branches of each variable by running a modest number of simplex iterations before choosing the variable on which to branch. Third, by using a more aggressive probing, one can study the implication of fixing the binary variables to 0 and 1 before branching. The strong branching as well as probing are very expensive decisions, but they are worth trying for the difficult models.

Lack of progress in the best integer solution: for some models, the MILP algorithms struggle before finding the first integer solution, and even if they do, they fail to improve it. The first remedy consists of increasing the usage frequency of MILP heuristics. These heuristics are generally able to locate new feasible solutions sooner compared to branching. Another possible approach is to choose an alternative node selection strategy such as a depth-first search or best-estimate search. Furthermore, since the purpose here is to find more feasible solutions rather than optimal ones, it is recommended to focus on feasibility instead of optimality using the emphasis parameter. Finally, it can be useful to allow the algorithm to polish a solution to find additional improving solutions in the neighborhood of the existing one.

Out of memory: this problem occurs when the branching tree contains a large number of unexplored nodes. One of the remedies to alleviate the size of the branching tree is to switch the node selection strategy to a focused search such as a best-estimate search or depth search. The number of generated nodes can also be reduced using a more sophisticated variable selection strategy such as strong branching. Another memory-saving approach is to turn off the generated cuts. However, these remedies may drastically increase the execution time.

APPENDIX B ADDITIONAL EXPERIMENTS

Following the performance comparison using the best configurations obtained tuning on $N1$, we compare the MPILS and ParamILS$_1$ tuners by tuning on all N instances and highlighting the results obtained by the best configurations reached. We report in Table B.1 the total time (TT), the best gap after 10% of the total time ($Gap_{\frac{TT}{10}}$), the best gap at the end of the total time (Gap_{TT}), the best runtime to optimality at the end of the total time ($Time_{TT}$), and the time when the best configuration was found (TT^*). We recall that each tested configuration is run for 280s.

Table B.1 MPILS Tuner and ParamILS$_1$ Tuner Comparison

Inst	TT	*MPILS*				ParamILS$_1$			
		$Gap_{\frac{TT}{10}}$	Gap_{TT}	$Time_{TT}$	TT^*	$Gap_{\frac{TT}{10}}$	Gap_{TT}	$Time_{TT}$	TT^*
N1	38670	21.9	0.0	280	28471	65.6	7.1	-	23631
N2	44681	4.4	0.0	180	12757	70.2	18.5	-	20350
N3	41732	7.1	0.3	-	20538	31.6	31.6	-	12330
N4	52433	3.9	0.0	111	14684	22.1	4.9	-	15760
N5	52534	5.1	0.0	128	38003	55.1	5.1	-	13510
N6	46930	3.4	0.0	81	27642	42.6	42.6	-	14375
N7	41972	23.9	0.0	205	21207	86.2	86.2	-	10366
N8	49038	8.8	0.0	218	26548	100.0	100.0	-	-
N9	59186	4.7	0.0	70	26548	18.2	7.3	-	14800
N10	38485	7.3	0.0	82	15249	22.6	20.5	-	15759
N11	39505	10.5	0.0	156	26400	100.0	2.0	-	29877
N12	41972	19.9	0.0	204	21001	100.0	30.0	-	21150
N13	39490	4.2	0.0	200	16699	15.9	4.2	-	11477
N14	40272	7.3	0.0	73	13306	15.7	7.3	-	18660
N15	44267	3.8	0.0	182	25603	63.4	5.2	-	15489
Avg	**44745**	**9.1**	**$<$ 0.1**	**163**	**22229**	**53.9**	**24.8**	-	**16967**

The results show that the MPILS tuner outperforms the ParamILS$_1$ tuner on all N instances. For the MPILS tuner, the runtime to optimality is below 280s for all instances except $N3$. Furthermore, while the MPILS tuner requires on average 163s to reach optimality ($N3$ excluded), the ParamILS$_1$ tuner fails to reach optimality on all of them, with 2.0% being the best gap obtained. Although the best configuration is found, on average, within 50% of the total time, enough time is given to check the stability of the tuners, observe their ability to converge towards promising regions, and evaluate all residual parameters ($p \in \mathcal{R}$).

To provide further insights about the statistical learning, we report in Table B.2 the number of tested configurations (*# Tested Config.*), the % of deteriorating configurations for the MPILS and ParamILS$_1$ tuners (*% Deteriorating Config.*). These percentages are reported for five intervals: $[0, \frac{TT}{10}]$ (TT_{10}), $[\frac{TT}{10}, \frac{TT}{4}]$ (TT_{25}), $[\frac{TT}{4} \frac{TT}{2}]$ (TT_{50}), $[\frac{TT}{2}, \frac{3RT}{4}]$ (TT_{75}), and $[\frac{3RT}{4}, RT]$ (TT_{100}).

Table B.2 MPILS Tuner and ParamILS$_1$ Tuner Deteriorating Configurations % over Total Time for N Instances

Inst	# Tested Config.	% Deteriorating Config. *MPILS*					% Deteriorating Config. *ParamILS$_1$*				
		TT_{10}	TT_{25}	TT_{50}	TT_{75}	TT_{100}	TT_{10}	TT_{25}	TT_{50}	TT_{75}	TT_{100}
N1	143	90.2	79.3	13.9	86.7	3.4	100.0	100.0	100.0	95.2	88.4
N2	160	94.1	73.9	70.0	69.2	15.8	100.0	100.0	100.0	100.0	100.0
N3	149	100.0	67.8	34.7	4.3	3.7	100.0	100.0	100.0	100.0	100.0
N4	187	87.5	34.6	17.2	6.2	0.0	90.0	100.0	85.3	81.1	79.2
N5	188	82.6	26.0	18.6	16.3	5.4	100.0	100.0	93.4	90.8	88.1
N6	168	92.7	28.5	16.6	16.2	7.9	100.0	100.0	100.0	100.0	100.0
N7	150	100.0	94.5	88.0	42.8	12.1	100.0	100.0	100.0	100.0	100.0
N8	175	95.2	70.3	38.7	25.9	7.6	100.0	100.0	100.0	100.0	100.0
N9	211	93.4	49.1	43.6	33.0	6.9	100.0	100.0	87.1	83.7	78.3
N10	137	88.0	50.4	41.8	34.7	7.3	100.0	100.0	100.0	100.0	100.0
N11	141	93.3	57.9	46.6	32.5	5.2	100.0	100.0	91.0	87.9	85.4
N12	150	97.0	70.8	52.3	36.7	9.7	100.0	100.0	100.0	100.0	100.0
N13	141	85.0	30.3	17.9	11.2	2.7	100.0	100.0	89.1	85.8	82.5
N14	144	91.0	50.6	35.1	24.0	6.2	100.0	100.0	76.6	73.7	70.9
N15	158	88.4	43.7	24.8	16.1	4.3	100.0	100.0	82.8	79.8	76.7
Avg	**160**	**91.9**	**55.2**	**37.3**	**30.4**	**6.5**	**99.3**	**100.0**	**93.7**	**91.8**	**89.9**

As shown, it is clear that the percentage of deteriorating configurations is sharply decreasing over time for the MPILS tuner; this is contrary to the ParamILS$_1$ tuner, for which the percentage is either not changing or decreasing slightly. For the former, the percentage decreases from an average of 91.92% in the first interval to an average of 6.59% in the fifth interval. For the latter, the percentage remains high while decreasing by around 10% between the first and the fifth intervals. It is worth mentioning that, on average, 12 *1-value* and 15 *2-values* are identified in 5 phases by the *ExtractStats*() procedure. As seen in Section 3.3.4, removing a *1-value* or a *2-values* corresponds to removing hundreds of thousands of deteriorating configurations.

APPENDIX C THE STOCHASTIC CAPACITATED FACILITY LOCATION PROBLEM

Let us consider a set of potential facility locations J and a set of customers I. The objective is to open enough facilities to satisfy the customers with minimum cost. A customer must be served by at least one facility (or more). At each potential facility location $j \in J$, at most one facility with a service capacity of s_j can be opened. The corresponding fixed cost is f_j units. The variable service cost from facility $j \in J$ to customer $i \in I$ is c_{ij}. Each customer $i \in I$ has a stochastic demand d_i^k, where $k \in K$ is a scenario with probability p_k such that $\sum_{k \in K} p_k = 1$.

Let y_j be the binary variable equal to one if a facility is opened at location $j \in J$ and zero otherwise. Let $x_{ij}^k \geq 0$ be the quantity supplied from facility $j \in J$ to customer $i \in I$ given scenario $k \in K$. The stochastic capacitated facility location problem is then:

$$\textbf{min} \quad \sum_{j \in J} f_j y_j + \sum_{k \in K} \sum_{j \in J} \sum_{i \in I} p_k c_{ij} x_{ij}^k \tag{SFLP}$$

$$s.t.: \quad \sum_{j \in J} x_{ij}^k \geq d_i^k \qquad \forall i \in I, k \in K \tag{C.1}$$

$$\sum_{i \in I} x_{ij}^k \leq s_j y_j \qquad \forall j \in J, k \in K \tag{C.2}$$

$$\sum_{j \in J} s_j y_j \geq \max_{k \in K} \sum_{i \in I} d_i^k \tag{C.3}$$

$$x_{ij}^k \geq 0, \quad y_j \in \{0,1\} \qquad \forall i \in I, j \in J, k \in K \tag{C.4}$$

The objective minimizes the total costs (fixed and variable). The first constraint set controls the demand satisfaction for each customer under every scenario. The second constraint set controls the capacity level for each facility. The third constraint corresponds to the complete recourse property to the problem. The last constraints are positivity (x variables) and binary (y variables) conditions.

APPENDIX D THE PSIMVA FORMULATION AND PBD DECOMPOSITION

The optimization model presented in this section is the one in [2]. It is a complex multi-period MILP. We describe the notation, the decision variables, the objective function, the constraints, and the mathematical model. For the notation in Table D.1, indices are in lower-case, sets are in calligraphic style, and parameters are in bold style.

Decision Variables

The problem variables are defined below:

$a_{rt} \in \{0,1\}$	: binary variable equal to 1 if the routine $r \in \mathcal{R}$ is active during period $t \in \mathcal{T}$, 0 otherwise.
$d_{rt} \geq 0$	: real variable indicating the total quantity produced, transported, or loaded by routine $r \in \mathcal{R}$ over period $t \in \mathcal{T}$.
$q_i \in \{0,1\}$	: binary variable equal to 1 if the possible assignment $i \in \mathcal{I}$ is selected, 0 otherwise.
$u^p_{ho} \in \{0,1\}$	: binary variable equal to 1 if product $p \in \mathcal{P}_h$ of shipment $h \in \mathcal{H}$ is supplied from origin $o \in \mathcal{O}$, 0 otherwise.
$v^p_{ht} \geq 0$	: real variable indicating the total volume of product $p \in \mathcal{P}$ loaded onto the vessel associated with shipment $h \in \mathcal{H}$ at the end of period $t \in \mathcal{T}$.
$v^p_{st} \geq 0$	: real variable indicating the total volume of product $p \in \mathcal{P}$ stored at stocking point $s \in \mathcal{S}$ at the end of period $t \in \mathcal{T}$.
$g_{rt} \in \{0,1\}$	: binary variable equal to 1 if a changeover allowing to activate routine $r \in \mathcal{R}_j$ with $j \in \mathcal{J}^{tf}$ is executed during period $t \in \mathcal{T}$, 0 otherwise.
$z_h \in \{0,1\}$	: binary variable equal to 1 if the demand of shipment $h \in \mathcal{H}$ is fully satisfied, 0 otherwise.

Objective Function

The quality of a solution of the PSIMVA is evaluated using KPIs. These KPIs measure the overall commercial and industrial effectiveness of the manufacturing system. At the commercial level, the aim is to maximize shipments' total fulfillment (TF). A shipment is said to be fulfilled if the whole quantities requested are delivered by the due date. At the industrial level, we seek the lowest possible number of product changeovers (PC). The two

Table D.1 Notation

	Notation	Definition
Indices	h	shipment
	i	possible assignment
	j	unit
	k	quay
	o	origin
	p	product
	r	routine
	s	stocking point
	t	time period
Sets	$\mathcal{H}$	set of shipments
	$\mathcal{H}^m$	set of mono-source shipments
	$\mathcal{I}$	set of possible assignments
	$\mathcal{I}_h$	set of possible assignments for shipment h
	$\mathcal{I}_{hk}$	set of possible assignments for shipment h on quay k
	$\mathcal{J}$	set of units
	$\mathcal{J}^{ld}$	set of loading units
	$\mathcal{J}^{tf}$	set of transformation units
	$\mathcal{K}$	set of quays
	$\mathcal{O}$	set of origins
	$\mathcal{P}$	set of products
	$\mathcal{P}_h$	set of products required by shipment h
	$\mathcal{R}$	set of routines
	$\mathcal{R}_h$	set of shipment h loading routines
	$\mathcal{R}_j$	set of unit j routines
	$\mathcal{R}_k$	set of routines flowing into quay k
	$\mathcal{R}_s^{in}$	set of routines flowing into stocking point s
	$\mathcal{R}_s^{out}$	set of routines flowing from stocking point s
	$\mathcal{S}$	set of stocking points
	$\mathcal{T}$	set of time periods
Parameters	$\mathbf{A}_j$	daily availability of unit j (hours)
	$\mathbf{B}_{rt}$	production capacity of routine r during period t (tonne)
	$\mathbf{C}_j$	daily nominal capacity of unit j (tonne)
	$\mathbf{D}_r^p$	ratio of product p to produce, transport, or load one tonne using routine r
	$\mathbf{E}_r$	daily loading capacity of routine r (tonne)
	$\mathbf{F}_r^p$	ratio of product p in one tonne produced using routine r
	$\mathbf{G}_r$	required time to activate routine r (hours)
	$\mathbf{L}_k$	length of quay k (meter)
	$\mathbf{MAX}_j$	weekly loading capacity of unit j (tonne)
	$\mathbf{MIN}_{s0}^p$	initial safety stock of product p at stocking point s (tonne)
	$\mathbf{MIN}_{st}^p$	safety stock of product p at stocking point s during period t (tonne)
	$\mathbf{Q}_h^p$	quantity of product p ordered by shipment h (tonne)
	$\mathbf{T}_r$	time to transform, transport, or load one tonne using routine r (hours)
	$\mathbf{V}_h$	length of vessel h (meter)
	$\boldsymbol{\tau}_j$	binary parameter equal to 1 if $j \in \mathcal{J}^{tf}$, 0 otherwise.

KPIs are defined as follows:

- **TotalFulfillment** (TF). This variable computes the percentage of fulfilled shipments (in terms of quantities): $TF(z) = 100 \times \frac{\sum_{p \in \mathcal{P}_h, h \in \mathcal{H}} \mathbf{Q}_h^p z_h}{\sum_{p \in \mathcal{P}_h, h \in \mathcal{H}} \mathbf{Q}_h^p}$
- **ProductChangeovers** (PC). This variable counts the number of changeovers to be minimized: $PC(g) = \sum_{r \in \mathcal{R}_{tf}, t \in \mathcal{T}} g_{rt}$

Model

Since the $\mathcal{PBD}$ subproblem is a restriction of the original problem, we describe just the $\mathcal{PBD}$ subproblem to alleviate the text. In a given feasible node and using Proposition 2, the ($\mathcal{PBD}$ Subproblem) is:

$$\textbf{Max} \quad w_{TF}TF(z) - w_{PC}PC(g) \qquad (\mathcal{PBD}\text{ Subproblem})$$

s.t.:

$$\textit{Prod. cstr.:} \quad d_{rt} \leq \mathbf{B}_{rt}\bar{a}_{rt} \qquad \forall r \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \tag{D.1}$$

$$\textit{Capa. cstr.:} \quad \sum_{r\in\mathcal{R}_j} d_{rt} \leq \mathbf{C}_j \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{D.2}$$

$$\sum_{r\in\mathcal{R}_j} \mathbf{T}_r d_{rt} \leq \mathbf{A}_j - \boldsymbol{\tau}_j \sum_{r\in\mathcal{R}_j} \mathbf{G}_r \bar{g}_{rt} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{D.3}$$

$$\textit{Inv. cstr.:} \quad v_{s0}^p = \mathbf{MIN}_{s0}^p \qquad \forall p \in \mathcal{P}, s \in \mathcal{S} \tag{D.4}$$

$$\sum_{r\in\mathcal{R}_s^{in}} \mathbf{F}_r^p d_{rt} + v_{s,t-1}^p = \sum_{r\in\mathcal{R}_s^{out}} \mathbf{D}_r^p d_{rt} + v_{st}^p \qquad \forall p \in \mathcal{P}, s \in \mathcal{S}, t \in \mathcal{T} \tag{D.5}$$

$$v_{st}^p \geq \mathbf{MIN}_{st}^p \qquad \forall p \in \mathcal{P}, s \in \mathcal{S}, t \in \mathcal{T} \tag{D.6}$$

$$v_{h0}^p = 0 \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H} \tag{D.7}$$

$$\sum_{r\in\mathcal{R}_h} \mathbf{F}_r^p d_{rt} + v_{h,t-1}^p = \begin{cases} v_{ht}^p & t \neq \overline{T} \in \mathcal{T} \\ \mathbf{Q}_h^p z_h & t = \overline{T} \end{cases} \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H} \tag{D.8}$$

$$\textit{Chg. cstr.:} \quad \sum_{r\in\mathcal{R}_j} \bar{a}_{rt} \leq 1 \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{D.9}$$

$$\sum_{\substack{r\in\mathcal{R}_j \\ r\neq r'}} \bar{a}_{r,t-1} + \bar{a}_{r't} \leq 1 + \bar{g}_{r't} \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \setminus \{1\} \tag{D.10}$$

$$\bar{g}_{r't} \leq \frac{1}{2}\Big(\sum_{\substack{r\in\mathcal{R}_j \\ r\neq r'}} \bar{a}_{r,t-1} + \bar{a}_{r't}\Big) \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \setminus \{1\} \tag{D.11}$$

$$\textit{Ld. cstr.:} \quad d_{rt} \leq \sum_{\substack{i\in\mathcal{I}_{hk} \\ i_1\leq t\leq i_2}} \mathbf{E}_r \bar{q}_i \qquad \forall r \in \mathcal{R}_k, k \in \mathcal{K}, t \in \mathcal{T}, h \in \mathcal{H} \tag{D.12}$$

$$z_h \geq \sum_{i\in\mathcal{I}_h} \bar{q}_i \qquad \forall h \in \mathcal{H} \tag{D.13}$$

$$\sum_{h\in\mathcal{H}} \sum_{\substack{i\in\mathcal{I}_{hk} \\ i_1\leq t\leq i_2}} \mathbf{V}_h \bar{q}_i \leq \mathbf{L}_k \qquad \forall k \in \mathcal{K}, t \in \mathcal{T} \tag{D.14}$$

$$\sum_{r\in\mathcal{R}_j} \sum_{t'=t}^{t+6} d_{rt'} \leq \mathbf{MAX}_j \qquad \forall j \in \mathcal{J}^{ld}, t \in \mathcal{T} \setminus \{\overline{T}-5, ..., \overline{T}\} \tag{D.15}$$

$$\textit{Mono. cstr.:} \quad z_h \geq \sum_{o\in\mathcal{O}} \bar{u}_{ho}^p \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H}^m \tag{D.16}$$

$$\sum_{t\in\mathcal{T}} d_{rt} \geq \mathbf{Q}_h^p \bar{u}_{ho}^p \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H}^m, r \in \mathcal{R}_o, o \in \mathcal{O} \tag{D.17}$$

Non-negativity in D (D.18)

where $\bar{g}_{rt} = g_{rt}$ if $g^*_{rt} = 1$, and 0 otherwise, $\bar{q}_i = q_i$ if $q^*_i = 1$, and 0 otherwise, $\bar{u}^p_{ho} = u^p_{ho}$ if $u^{p^*}_{ho} = 1$, and 0 otherwise, $\bar{a}_{rt} = a_{rt}$ if $a^*_{rt} = 1$, and 0 otherwise.

The objective function is a weighted function (with positive weights w_{TF} and w_{PC}) that maximizes the TF and minimizes the PC. We distinguish six groups of constraints, which are production (Prod. cstr.), capacity (Capa. cstr.), inventory (Inv. cstr.), changeovers (Chg. cstr.), loading (Ld. cstr.), and mono-source (Mono. cstr.) constraints. Constraints (D.1) ensure that the quantity produced by an active routine during a period is bounded; otherwise, it is equal to zero. Constraints (D.2) impose the respect of production capacities in terms of quantities, and constraints (D.3) impose the respect of production capacities in terms of time. For transformation units, we add the time consumed by changeovers. Constraints (D.4) correspond to the initial stocks at stocking points. Constraints (D.5) ensure inflow and outflow conservation for each stocking point. Constraints (D.6) control safety stock levels. Constraints (D.7) assume initial stocks in vessels are equal to zero, while Constraints (D.8) track the flow conservation on the vessels. For $t \neq \overline{T} \in \mathcal{T}$, these constraints update the stock level within the vessel while for $t = \overline{T}$, they ensure that the stock level of product $p \in \mathcal{P}_h$ corresponding to shipment $h \in \mathcal{H}$ is equal to the shipment request for the same product. Constraints (D.9) guarantee that at most one routine can be active on each unit during a specific period. Constraints (D.10) and (D.11) count the shift from one routine to another on the same unit. In such a case, a new changeover is active. Constraints (D.12) guarantee the assignment of vessels to quays for loading. Constraints (D.13) ensure that each shipment has only one possible assignment of its corresponding vessel. These constraints do not allow partial loading on different quays and time intervals. Constraints (D.14) check the respect of quays' length while assigning vessels. Constraints (D.15) make sure that the sum of the loaded quantities by the loading routines in the interval $[t, t+6]$ (over a week) should not exceed a maximal capacity. This capacity can change based on periods because of meteorological conditions. Constraints (D.16) ensure the satisfaction of shipments $h \in \mathcal{H}^m$ from at most one origin. If an origin is selected, constraints (D.17) ensure the activation of its corresponding routines. It is worth mentioning that the integration of production scheduling, inventory management, and vessel assignment is ensured through the variables d_{rt}, $r \in \mathcal{R}$, $t \in \mathcal{T}$, which control the quantities produced in the units, transported through conveyors, or loaded in stocking points or vessels.

Decomposition

The PSIMVA has several complicating variables, which make production scheduling, vessel assignment, and the whole problem challenging. In what remains, to alleviate the notations, we remove the $\mathcal{S}$ notation (i.e., we write y instead of y_S, x instead of x_S, etc.).

The PSIMVA complicating variables are the set of binary variables q_i assigning vessels to quays, the set of production variables a_{rt} ruling the production inside the transformation units, the set of mono-source variables u^p_{ho}, and the set of changeover variables g_{rt}. Table D.2 shows the statistics related to each class of these binary variables in the optimal solution of a representative PSIMVA instance. We consider two metrics: the percentage of each class among all the binary variables (% Class) and the percentage of non-zero variables per class in the solution of the representative instance with mono-source (% Non-zeros).

Table D.2 Complicating Binary Classes in the PSIMVA Representative Instance

Class	Index	#Variables	#Non-zeros	%Non-zeros	%Class
q_i	1	13402	64	0.48	50.38
a_{rt}	2	5832	1870	32.06	21.92
u^p_{ho}	3	1434	83	5.79	5.39
g_{rt}	4	5868	714	12.17	22.06

Table D.2 confirms that the PSIMVA is a sparse very large-scale problem. Among all the 26536 complicating variables, only 2731 variables take a non-zero value in the optimal solution of the representative PSIMVA instance, i.e., the ratio $\frac{supp(y^*)}{n} = 0.1$.

Proposition 10. *For $h \in \mathcal{H}$, solving the ($\mathcal{PBD}$ Subproblem) problem with $z_h \in \{0, 1\}$ is equivalent to solving the ($\mathcal{PBD}$ Subproblem) problem with z_h relaxed, i.e., $0 \leq z_h \leq 1$.*

Proof. Let $h \in \mathcal{H}$. Given that the q_i for $i \in \mathcal{I}_h$ variables are binary, we distinguish two cases:

- If $\exists i \in \mathcal{I}_h$ such that $q_i = 1$, then constraints (G.13) with $z_h \leq 1$ ensure $z_h = 1$.
- If $\forall i \in \mathcal{I}_h$, $q_i = 0$, constraints (G.12) ensure that we cannot load on any vessel. Then, constraints (G.7) and (G.8) ensure z_h=0.

Thus, we can relax the binary constraints on the $z_h \in \{0, 1\}$, $h \in \mathcal{H}$. $\square$

Proposition 10 ensures that the $z_h \in \{0, 1\}$, $h \in \mathcal{H}$ variables can be relaxed and kept in the $\mathcal{PBD}$ subproblem. We denote $y = (q, a, u, g)$ as the complicating binary variables and x all the remaining real variables. The PSIMVA can be written in the form (OP), and since it is sparse, we can apply the $\mathcal{PBD}$ method to solve it. The $\mathcal{PBD}$ master problem minimizes the changeovers and provides the changeovers' and vessel assignments' information to the

$\mathcal{PBD}$ subproblem, i.e., we insert in the $\mathcal{PBD}$ subproblem all variables in the support of the $y^* = (a^*, q^*, u^*, g^*)$ vector, where y^* is the $\mathcal{PBD}$ master problem solution.

Proposition 11. *The $\mathcal{PBD}$ subproblem of PSIMVA is always feasible and bounded.*

Proof. It is feasible because the null solution is feasible since the model allows for not fulfilling any shipments. This can be easily seen by fixing $q_i = 0$, $\forall i \in \mathcal{I}_h$. It is bounded because the system has limited capacities, the requested quantities are limited, and $z_h \leq 1$, $\forall h \in \mathcal{H}$. $\square$

Proposition 11 ensures that only optimality cuts are required by the $\mathcal{PBD}$ method.

Let $\alpha = (\alpha^{(G.1)}, \alpha^{(G.2)}, ..., \alpha^{(G.18)})$ be the dual variables corresponding to constraints (G.1) to (G.18). Let $\bar{\alpha} = (\bar{\alpha}^{(G.1)}, \bar{\alpha}^{(G.2)}, ..., \bar{\alpha}^{(G.18)})$ be the optimal dual solution corresponding to ($\mathcal{PBD}$ Subproblem). The corresponding optimal dual objective value is then:

$$\sum_{\substack{r\in\mathcal{R}_j\\ j\in\mathcal{J}^{tf}\\ t\in\mathcal{T}}} \mathbf{B}_{rt}\bar{a}_{rt}\bar{\alpha}^{(G.1)}_{rjt} + \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} \mathbf{C}_j\bar{\alpha}^{(G.2)}_{jt} + \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}}(\mathbf{A}_j - \boldsymbol{\tau}_j \sum_{r\in\mathcal{R}_j} \mathbf{G}_r\bar{g}_{rt})\bar{\alpha}^{(G.3)}_{jt} + \sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}}} \mathbf{MIN}^p_{s0}\bar{\alpha}^{(G.4)}_{ps} -$$
$$\sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}\\ t\in\mathcal{T}}} \mathbf{MIN}^p_{st}\bar{\alpha}^{(G.6)}_{pst} + \sum_{\substack{p\in\mathcal{P}\\ h\in\mathcal{H}}} \mathbf{Q}^p_h z_h\bar{\alpha}^{(G.8)}_{ph} + \sum_{\substack{r\in\mathcal{R}\\ k\in\mathcal{K}\\ t\in\mathcal{T}\\ h\in\mathcal{H}}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\leq t\leq i_2}} \mathbf{E}_r\bar{q}_i\bar{\alpha}^{(G.12)}_{rkth} - \sum_{h\in\mathcal{H}}\sum_{i\in\mathcal{I}_h} \bar{q}_i\bar{\alpha}^{(G.13)}_h +$$
$$\sum_{\substack{j\in\mathcal{J}^{ld}\\ t\in\mathcal{T}\setminus\{\overline{T}-5,\dots,\overline{T}\}}} \mathbf{MAX}_j\bar{\alpha}^{(G.15)}_{jt} - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m}}\sum_{o\in\mathcal{O}} \bar{u}^p_{ho}\bar{\alpha}^{(G.16)}_{ph} - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m\\ r\in\mathcal{R}_o\\ o\in\mathcal{O}}} \mathbf{Q}^p_h\bar{u}^p_{ho}\bar{\alpha}^{(G.17)}_{phro} \qquad \text{(D.19)}$$

Let $\mathcal{F}$ be the feasible region of the dual of ($\mathcal{PBD}$ Subproblem) and $\Gamma_{\mathcal{F}}$ be the set of extreme points of $\mathcal{F}$. Introducing the additional free variable μ, the ($\mathcal{PBD}$ Master Problem) can be formulated as follows:

$$\mathbf{Max} \quad w_{TF}\mu - w_{PC}PC(g) \qquad \text{(}\mathcal{PBD}\text{ Master Problem)}$$

s.t.:

Chg. cstr.:
$$\sum_{r\in\mathcal{R}_j} a_{rt} \leq 1 \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \qquad \text{(D.20)}$$

$$\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}} a_{r,t-1} + a_{r't} \leq 1 + g_{r't} \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T}\setminus\{1\} \qquad \text{(D.21)}$$

$$g_{r't} \leq \frac{1}{2}(\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}} a_{r,t-1} + a_{r't}) \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T}\setminus\{1\} \qquad \text{(D.22)}$$

$$\textit{Ld. cstr.:} \sum_{h\in\mathcal{H}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\le t\le i_2}} \mathbf{V}_h q_i \le \mathbf{L}_k \qquad \forall k\in\mathcal{K}, t\in\mathcal{T} \tag{D.23}$$

$$\begin{aligned}\textit{PBD Cuts: } \mu \le & \sum_{\substack{r\in\mathcal{R}_j\\ j\in\mathcal{J}^{tf}\\ t\in\mathcal{T}}} \mathbf{B}_{rt} a_{rt} \bar{\alpha}^{(G.1)}_{rjt} + \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} \mathbf{C}_j \bar{\alpha}^{(G.2)}_{jt} + \\ & \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} (\mathbf{A}_j - \boldsymbol{\tau}_j \sum_{r\in\mathcal{R}_j} \mathbf{G}_r g_{rt}) \bar{\alpha}^{(G.3)}_{jt} + \sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}}} \mathbf{MIN}^p_{s0} \bar{\alpha}^{(G.4)}_{ps} - \\ & \sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}\\ t\in\mathcal{T}}} \mathbf{MIN}^p_{st} \bar{\alpha}^{(G.6)}_{pst} + \sum_{\substack{p\in\mathcal{P}\\ h\in\mathcal{H}}} \mathbf{Q}^p_h z_h \bar{\alpha}^{(G.8)}_{ph} + \\ & \sum_{\substack{r\in\mathcal{R}\\ k\in\mathcal{K}\\ t\in\mathcal{T}\\ h\in\mathcal{H}}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\le t\le i_2}} \mathbf{E}_r q_i \bar{\alpha}^{(G.12)}_{rkth} - \sum_{h\in\mathcal{H}} \sum_{i\in\mathcal{I}_h} q_i \bar{\alpha}^{(G.13)}_h \\ & + \sum_{\substack{j\in\mathcal{J}^{ld}\\ t\in\mathcal{T}\setminus\{\overline{T}-5,\dots,\overline{T}\}}} \mathbf{MAX}_j \bar{\alpha}^{(G.15)}_{jt} - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m}} \sum_{o\in\mathcal{O}} u^p_{ho} \bar{\alpha}^{(G.16)}_{ph} \\ & - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m\\ r\in\mathcal{R}_o\\ o\in\mathcal{O}}} \mathbf{Q}^p_h u^p_{ho} \bar{\alpha}^{(G.17)}_{phro} \qquad \bar{\alpha}\in\Gamma_{\mathcal{F}}\end{aligned} \tag{D.24}$$

$$\textit{Non-negativity and binary conditions in 4.4.1} \tag{D.25}$$

To maintain the shipment information in the ($\mathcal{PBD}$ Master Problem), we make the following remark.

Remark 2. *We strengthen the ($\mathcal{PBD}$ Master Problem) using the following valid inequalities:*

$$\sum_{i\in I_{hk}} q_i \le 1 \qquad \forall h\in\mathcal{H}, k\in\mathcal{K} \tag{D.26}$$

$$\sum_{\substack{i\in I_h\\ i_1\le t\le i_2}} q_i \le 1 \qquad \forall h\in\mathcal{H}, t\in\mathcal{T} \tag{D.27}$$

$$\sum_{\substack{i\in I_{hk}\\ i_1\le t\le i_2}} q_i \le 1 \qquad \forall h\in\mathcal{H}, k\in\mathcal{K}, t\in\mathcal{T} \tag{D.28}$$

Constraints (D.26) *are a decomposition of Constraints* (G.13) *by quays. Constraints* (D.27) *are a decomposition of Constraints* (D.13) *by periods. Constraints* (D.28) *are a decomposition of Constraints* (D.13) *by periods and quays.*

APPENDIX E MATHEMATICAL PROOFS

Proof of Proposition 6

Consider a schedule $\bar{q}$. Delay beyond scheduling horizon using decision d_1 all vessels belonging to schedule ꝗ^* and not to $\bar{q}$. If vessel $v \in \mathcal{V}$ is loaded earlier in schedule ꝗ^* than $\bar{q}$, use decision d_1 to delay it. Similarly, if vessel $v \in \mathcal{V}$ is loaded later in schedule ꝗ^* compared to schedule $\bar{q}$, use decision d_2 to advance it.

Proof of Proposition 7

The delayed vessels cannot be warm-started because their previous assignments are removed. Their new assignment q_i^{del}, $i \in \mathcal{I}_v$ $v \in \mathcal{V}^{del}$ can be initiated with a 0. For the advanced vessels, the new assignment q_i^{adv}, $i \in \mathcal{I}_v$ $v \in \mathcal{V}^{adv}$ can be initiated with a 0. Still, the advanced vessels have their q_i^{ini}, $i \in \mathcal{I}_v$ $v \in \mathcal{V}^{adv}$ variables (from the previous schedule) in Re-Opt model. These variables can be warm-started using their values in ꝗ^*.

Proof of Proposition 8

Adding to Proposition 7, without fixing, q_i, $i \in \mathcal{I}_h$ $v \in \mathcal{V} \setminus \mathcal{V}^{del} \cup \mathcal{V}^{adv}$ can be warm-started using their assignment in ꝗ^*.

Proof of Proposition 9

As per Definition 4, schedule $\bar{q}$ ensures that $\forall q \in \Omega$:

$$TF(q) \leq TF(\text{ꝗ}^*) = TF(\bar{q}) \text{ and } \Delta D(q) \leq \Delta D(\bar{q})$$

Given two positive weights α_1 and α_2 such that $\alpha_1 + \alpha_2 = 1$, we have:

$$R(q) = \alpha_1 \times TF(q) + \alpha_2 \Delta D(q) \leq \alpha_1 \times TF(\bar{q}) + \alpha_2 \times \Delta D(\bar{q}) = R(\bar{q}) \;\; \forall q \in \Omega$$

APPENDIX F THE OPTIMIZATION MODEL

The optimization model presented in this section is a complex multiperiod MILP. We describe the notation, the decision variables, the objective function, and the mathematical model. For the notation in Table F.1, indices are in lower-case, sets are in calligraphic style, and parameters are in bold style. More details are in [2].

Table F.1 PSIMVA Notation

Notation	**Definition**
Sets	
$h \in \mathcal{H}$	set of shipments
$h \in \mathcal{H}^m$	set of mono-source shipments
$i \in \mathcal{I}$	set of possible assignments
$i \in \mathcal{I}_h$	set of possible assignments for shipment h
$i \in \mathcal{I}_{hk}$	set of possible assignments for shipment h on quay k
$j \in \mathcal{J}$	set of units
$j \in \mathcal{J}^{ld}$	set of loading units
$j \in \mathcal{J}^{tf}$	set of transformation units
$k \in \mathcal{K}$	set of quays
$o \in \mathcal{O}$	set of origins
$p \in \mathcal{P}$	set of products
$p \in \mathcal{P}_h$	set of products required by shipment h
$r \in \mathcal{R}$	set of routines
$r \in \mathcal{R}_h$	set of shipment h loading routines
$r \in \mathcal{R}_j$	set of unit j routines
$r \in \mathcal{R}_o$	set of origin o routines
$r \in \mathcal{R}_k$	set of routines flowing into quay k
$r \in \mathcal{R}_s^{in}$	set of routines flowing into stocking point s
$r \in \mathcal{R}_s^{out}$	set of routines flowing from stocking point s
$s \in \mathcal{S}$	set of stocking points
$t \in \mathcal{T}$	set of time periods
Parameters	
$\mathbf{A}_j$	daily availability of unit j (hours)
$\mathbf{B}_{rt}$	production capacity of routine r during period t (tonne)
$\mathbf{C}_j$	daily nominal capacity of unit j (tonne)
$\mathbf{D}_r^p$	ratio of product p to produce, transport, or load one tonne using routine r
$\mathbf{E}_r$	daily loading capacity of routine r (tonne)
$\mathbf{F}_r^p$	ratio of product p in one tonne produced using routine r
$\mathbf{G}_r$	required time to activate routine r (hours)
$\mathbf{L}_k$	length of quay k (meter)
$\overline{\mathbf{Q}}_j$	weekly loading capacity of unit j (tonne)
$\underline{\mathbf{Q}}_{s0}^p$	initial safety stock of product p at stocking point s (tonne)
$\underline{\mathbf{Q}}_{st}^p$	safety stock of product p at stocking point s during period t (tonne)
$\mathbf{Q}_h^p$	quantity of product p ordered by shipment h (tonne)
$\mathbf{T}_r$	time to transform, transport, or load one tonne using routine r (hours)
$\mathbf{V}_h$	length of vessel h (meter)

Decision Variables

The problem variables are defined below:

Table F.2 Decision Variables

Variable	Definition
$a_{rt} \in \{0,1\}$	binary variable equal to 1 if the routine $r \in \mathcal{R}$ is active during period $t \in \mathcal{T}$, 0 otherwise.
$g_{rt} \in \{0,1\}$	binary variable equal to 1 if a changeover allowing to activate routine $r \in \mathcal{R}_j$ with $j \in \mathcal{J}^{tf}$ is executed during period $t \in \mathcal{T}$, 0 otherwise.
$q_i \in \{0,1\}$	binary variable equal to 1 if the possible assignment $i \in \mathcal{I}$ is selected, 0 otherwise. To each assignment $i \in \mathcal{I}$ corresponds a quadruple $(h, k, \underline{t}, \overline{t})$ where h is the shipment, k is the quay, $\underline{t}$ is the starting period, and $\overline{t}$ is the ending period.
$u^p_{ho} \in \{0,1\}$	binary variable equal to 1 if product $p \in \mathcal{P}_h$ of shipment $h \in \mathcal{H}$ is supplied from origin $o \in \mathcal{O}$, 0 otherwise.
$v^p_{ht} \geq 0$	real variable indicating the total volume of product $p \in \mathcal{P}$ loaded onto the vessel associated with shipment $h \in \mathcal{H}$ at the end of period $t \in \mathcal{T}$.
$v^p_{st} \geq 0$	real variable indicating the total volume of product $p \in \mathcal{P}$ stored at stocking point $s \in \mathcal{S}$ at the end of period $t \in \mathcal{T}$.
$x_{rt} \geq 0$	real variable indicating the total quantity produced, transported, or loaded by routine $r \in \mathcal{R}$ over period $t \in \mathcal{T}$.
$z_h \in \{0,1\}$	binary variable equal to 1 if the demand of shipment $h \in \mathcal{H}$ is fully satisfied, 0 otherwise.

Objective Function

We use OCP KPIs to evaluate the quality of a PSIMVA solution. These KPIs measure the overall commercial and industrial effectiveness of the manufacturing system. At the industrial level, the goal is to achieve the lowest possible number of product changeovers (PC). A changeover is a switch from one product to another on a production line. At the commercial level, the goal is to maximize vessels' total fulfillment (TF). We say that a shipment is *fulfilled* if the delivery of all the quantities requested by this shipment is by the due date. The two KPIs are defined as follows:

- **TotalFulfillment** (TF). This variable represents the percentage of fulfilled shipments (in terms of quantities) to be maximized: $TF(z) = 100 \times \frac{\sum_{p \in \mathcal{P}_h, h \in \mathcal{H}} \mathbf{Q}^p_h z_h}{\sum_{p \in \mathcal{P}_h, h \in \mathcal{H}} \mathbf{Q}^p_h}$
- **ProductChangeovers** (PC). This variable counts the number of changeovers to be minimized: $PC(g) = \sum_{r \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T}} g_{rt}$

Model

The sets, parameters, and decision variables being introduced, the MILP model is as follows:

$$\textbf{Max} \quad w_{TF} TF(z) - w_{PC} PC(g) \qquad \text{(Model)}$$

s.t.:

$$\textit{Prod. cstr.:} \quad d_{rt} \leq \mathbf{B}_{rt} a_{rt} \qquad \forall r \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \qquad \text{(F.1)}$$

$$\textit{Capa. cstr.:} \quad \sum_{r \in \mathcal{R}_j} d_{rt} \leq \mathbf{C}_j \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \qquad \text{(F.2)}$$

$$\sum_{r\in\mathcal{R}_j}\mathbf{T}_r d_{rt}+\boldsymbol{\tau}_j\sum_{r\in\mathcal{R}_j}\mathbf{G}_r g_{rt}\le\mathbf{A}_j \qquad \forall j\in\mathcal{J},t\in\mathcal{T} \tag{F.3}$$

Inv. cstr.:
$$v^p_{s0}=\mathbf{MIN}^p_{s0} \qquad \forall p\in\mathcal{P},s\in\mathcal{S} \tag{F.4}$$

$$\sum_{r\in\mathcal{R}^{in}_s}\mathbf{F}^p_r d_{rt}+v^p_{s,t-1}=\sum_{r\in\mathcal{R}^{out}_s}\mathbf{D}^p_r d_{rt}+v^p_{st} \qquad \forall p\in\mathcal{P},s\in\mathcal{S},t\in\mathcal{T} \tag{F.5}$$

$$v^p_{st}\ge\mathbf{MIN}^p_{st} \qquad \forall p\in\mathcal{P},s\in\mathcal{S},t\in\mathcal{T} \tag{F.6}$$

$$v^p_{h0}=0 \qquad \forall p\in\mathcal{P}_h,h\in\mathcal{H} \tag{F.7}$$

$$\sum_{r\in\mathcal{R}_h}\mathbf{F}^p_r d_{rt}+v^p_{h,t-1}=\begin{cases}v^p_{ht} & t\neq\overline{T}\in\mathcal{T}\\ \mathbf{Q}^p_h z_h & t=\overline{T}\end{cases} \qquad \forall p\in\mathcal{P}_h,h\in\mathcal{H} \tag{F.8}$$

Chg. cstr.:
$$\sum_{r\in\mathcal{R}_j}a_{rt}\le 1 \qquad \forall j\in\mathcal{J},t\in\mathcal{T} \tag{F.9}$$

$$\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}}a_{r,t-1}+a_{r't}\le 1+g_{r't} \qquad \forall r'\in\mathcal{R}_j,j\in\mathcal{J}^{tf},t\in\mathcal{T}\setminus\{1\} \tag{F.10}$$

$$g_{r't}\le\frac{1}{2}(\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}}a_{r,t-1}+a_{r't}) \qquad \forall r'\in\mathcal{R}_j,j\in\mathcal{J}^{tf},t\in\mathcal{T}\setminus\{1\} \tag{F.11}$$

Ld. cstr.:
$$\sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\le t\le i_2}}\mathbf{E}_r q_i\ge d_{rt} \qquad \forall r\in\mathcal{R}_k,k\in\mathcal{K},t\in\mathcal{T},h\in\mathcal{H} \tag{F.12}$$

$$\sum_{i\in\mathcal{I}_h}q_i\le z_h \qquad \forall h\in\mathcal{H} \tag{F.13}$$

$$\sum_{h\in\mathcal{H}}\sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\le t\le i_2}}\mathbf{V}_h q_i\le\mathbf{L}_k \qquad \forall k\in\mathcal{K},t\in\mathcal{T} \tag{F.14}$$

$$\sum_{r\in\mathcal{R}_j}\sum_{t'=t}^{t+6}d_{rt'}\le\mathbf{MAX}_j \qquad \forall j\in\mathcal{J}^{ld},t\in\mathcal{T}\setminus\{\overline{T}-5,...,\overline{T}\} \tag{F.15}$$

Mono. cstr.:
$$\sum_{o\in\mathcal{O}}u^p_{ho}\le z_h \qquad \forall p\in\mathcal{P}_h,h\in\mathcal{H}^m \tag{F.16}$$

$$\sum_{t\in\mathcal{T}}d_{rt}\ge\mathbf{Q}^p_h u^p_{ho} \qquad \forall p\in\mathcal{P}_h,h\in\mathcal{H}^m,r\in\mathcal{R}_o,o\in\mathcal{O} \tag{F.17}$$

Non-negativity and binary conditions in F (F.18)

The objective function is a weighted function (with positive weights w_{TF} and w_{PC}) that maximizes the TF and minimizes the PC. We distinguish six groups of constraints, which are production (Prod. cstr.), capacity (Capa. cstr.), inventory (Inv. cstr.), changeovers (Chg. cstr.), loading (Ld. cstr.), and mono-source (Mono. cstr.) constraints. Constraints (F.1) ensure that the quantity produced by an active routine during a period is bounded; otherwise, it is equal to zero. Constraints (F.2) impose the respect of production capacities in terms of quantities, and constraints (F.3) impose the respect of production capacities in terms of time. For transformation units, we add the time consumed by changeovers. Constraints (F.4) correspond to the initial stocks at stocking points. Constraints (F.5) ensure inflow and outflow conservation for each stocking point. Constraints (F.6) control safety stock levels. Constraints (F.7) assume initial stocks in vessels are equal to zero, while Constraints (F.8) track the flow conservation on the vessels. For $t \neq \overline{T} \in \mathcal{T}$, these constraints update the stock level within the vessel while for $t = \overline{T}$, they ensure that the stock level of product $p \in \mathcal{P}_h$ corresponding to shipment $h \in \mathcal{H}$ is equal to the shipment request for the same product. Constraints (F.9) guarantee that at most one routine can be active on each unit during a specific period. Constraints (F.10) and (F.11) count the shift from one routine to another on the same unit. In such a case, a new changeover is active. Constraints (F.12) guarantee the assignment of vessels to quays for loading. Constraints (F.13) ensure that each shipment has only one possible assignment of its corresponding vessel. These constraints do not allow partial loading on different quays and time intervals. Constraints (F.14) check the respect of quays' length while assigning vessels. Constraints (F.15) make sure that the sum of the loaded quantities by the loading routines in the interval $[t, t+6]$ (over a week) should not exceed a maximal capacity. This capacity can change based on periods because of meteorological conditions. Constraints (F.16) ensure the satisfaction of shipments $h \in \mathcal{H}^m$ from at most one origin. If an origin is selected, constraints (F.17) ensure the activation of its corresponding routines. It is worth mentioning that the integration of production scheduling, inventory management, and vessel assignment is ensured through the variables d_{rt}, $r \in \mathcal{R}$, $t \in \mathcal{T}$, which control the quantities produced in the units, transported through conveyors, or loaded in stocking points or vessels.

APPENDIX G THE HBD METHOD

To apply the $\mathcal{HBD}$ method in the OCP case, we keep all binary variables in the master problem, except variables $z_h, h \in \mathcal{H}$, which are relaxed (integrality) and kept (with other real variables) in the subproblem. The intuition behind relaxing these variables is the following: For the unconfirmed vessels, the company does not necessarily have to fulfill them. Hence, we can relax these variables in the mathematical model. For the confirmed vessels, the company must fulfill them. Thus, the corresponding $q_i, i \in \mathcal{I}$ are fixed in the subproblem. The resulting Constraints (F.13) with $z_h \leq 1, h \in \mathcal{H}$ ensure that the corresponding confirmed vessels variables $z_h, h \in \mathcal{H}$ take 1 in the $\mathcal{HBD}$ subproblem.

We can then apply the $\mathcal{HBD}$ method. We denote $y = (a, q, u, g)$ as the vector of complicating variables. The $\mathcal{HBD}$ master problem minimizes the changeovers and provides the changeovers' and vessel assignments' information to the $\mathcal{HBD}$ subproblem, i.e., we insert in the $\mathcal{HBD}$ subproblem all variables in the support of vector $y^* = (a^*, q^*, u^*, g^*)$, where y^* is the $\mathcal{HBD}$ master problem solution. The $\mathcal{HBD}$ subproblem maximizes the total fulfillment. The vessel assignment information corresponding to the confirmed vessels is fixed in the $\mathcal{HBD}$ subproblem, while the same information corresponding to the unconfirmed vessels is kept free. The $\mathcal{HBD}$ Subproblem is:

$$\textbf{Max} \quad w_{TF}TF(z) - w_{PC}PC(g) \qquad (\mathcal{HBD}\ \text{Subproblem})$$

s.t.:

$$\textit{Prod. cstr.:}\quad d_{rt} \leq \mathbf{B}_{rt}\bar{a}_{rt} \qquad \forall r \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T} \tag{G.1}$$

$$\textit{Capa. cstr.:}\quad \sum_{r \in \mathcal{R}_j} d_{rt} \leq \mathbf{C}_j \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{G.2}$$

$$\sum_{r \in \mathcal{R}_j} \mathbf{T}_r d_{rt} \leq \mathbf{A}_j - \boldsymbol{\tau}_j \sum_{r \in \mathcal{R}_j} \mathbf{G}_r \bar{g}_{rt} \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{G.3}$$

$$\textit{Inv. cstr.:}\quad v^p_{s0} = \mathbf{MIN}^p_{s0} \qquad \forall p \in \mathcal{P}, s \in \mathcal{S} \tag{G.4}$$

$$\sum_{r \in \mathcal{R}^{in}_s} \mathbf{F}^p_r d_{rt} + v^p_{s,t-1} = \sum_{r \in \mathcal{R}^{out}_s} \mathbf{D}^p_r d_{rt} + v^p_{st} \qquad \forall p \in \mathcal{P}, s \in \mathcal{S}, t \in \mathcal{T} \tag{G.5}$$

$$v^p_{st} \geq \mathbf{MIN}^p_{st} \qquad \forall p \in \mathcal{P}, s \in \mathcal{S}, t \in \mathcal{T} \tag{G.6}$$

$$v^p_{h0} = 0 \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H} \tag{G.7}$$

$$\sum_{r \in \mathcal{R}_h} \mathbf{F}^p_r d_{rt} + v^p_{h,t-1} = \begin{cases} v^p_{ht} & t \neq \overline{T} \in \mathcal{T} \\ \mathbf{Q}^p_h z_h & t = \overline{T} \end{cases} \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H} \tag{G.8}$$

$$\textit{Chg. cstr.:}\quad \sum_{r \in \mathcal{R}_j} \bar{a}_{rt} \leq 1 \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{G.9}$$

$$\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}} \bar{a}_{r,t-1} + \bar{a}_{r't} \leq 1 + \bar{g}_{r't} \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T}\setminus\{1\} \tag{G.10}$$

$$\bar{g}_{r't} \leq \frac{1}{2}(\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}} \bar{a}_{r,t-1} + \bar{a}_{r't}) \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T}\setminus\{1\} \tag{G.11}$$

$$\textit{Ld. cstr.:}\ d_{rt} \leq \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\leq t\leq i_2}} \mathbf{E}_r \bar{q}_i \qquad \forall r \in \mathcal{R}_k, k \in \mathcal{K}, t \in \mathcal{T}, h \in \mathcal{H} \tag{G.12}$$

$$z_h \geq \sum_{i\in\mathcal{I}_h} \bar{q}_i \qquad \forall h \in \mathcal{H} \tag{G.13}$$

$$\sum_{h\in\mathcal{H}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\leq t\leq i_2}} \mathbf{V}_h \bar{q}_i \leq \mathbf{L}_k \qquad \forall k \in \mathcal{K}, t \in \mathcal{T} \tag{G.14}$$

$$\sum_{r\in\mathcal{R}_j} \sum_{t'=t}^{t+6} d_{rt'} \leq \mathbf{MAX}_j \qquad \forall j \in \mathcal{J}^{ld}, t \in \mathcal{T}\setminus\{\overline{T}-5,...,\overline{T}\} \tag{G.15}$$

$$\textit{Mono. cstr.:}\ z_h \geq \sum_{o\in\mathcal{O}} \bar{u}^p_{ho} \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H}^m \tag{G.16}$$

$$\sum_{t\in\mathcal{T}} d_{rt} \geq \mathbf{Q}^p_h \bar{u}^p_{ho} \qquad \forall p \in \mathcal{P}_h, h \in \mathcal{H}^m, r \in \mathcal{R}_o, o \in \mathcal{O} \tag{G.17}$$

$$\textit{Non-negativity in } F \tag{G.18}$$

where $\bar{g}_{rt} = g_{rt}$ if $g^*_{rt} = 1$, and 0 otherwise, $\bar{q}_i = q^*_i$ if $q^*_i = 1$ and the corresponding shipment is confirmed, $\bar{q}_i = q_i$ if $q^*_i = 1$ and the corresponding shipment is unconfirmed, and 0 otherwise, $\bar{u}^p_{ho} = u^{p^*}_{ho}$ if $u^{p^*}_{ho} = 1$ and the corresponding shipment is confirmed, $\bar{u}^p_{ho} = u^p_{ho}$ if $u^{p^*}_{ho} = 1$ and the corresponding shipment is unconfirmed, and 0 otherwise, $\bar{a}_{rt} = a_{rt}$ if $a^*_{rt} = 1$, and 0 otherwise.

We solve the $\mathcal{HBD}$ subproblem and obtain the Benders cuts as in [3]. Let $\alpha = (\alpha^{(G.1)}, \alpha^{(G.2)}, ..., \alpha^{(G.18)})$ be the dual variables corresponding to constraints G.1 to G.18. Let $\bar{\alpha} = (\bar{\alpha}^{(G.1)}, \bar{\alpha}^{(G.2)}, ..., \bar{\alpha}^{(G.18)})$ be the optimal dual solution corresponding to $\mathcal{HBD}$ Subproblem when all complicating variables are fixed. The corresponding optimal dual objective value is then:

$$\sum_{\substack{r\in\mathcal{R}_j\\ j\in\mathcal{J}^{tf}\\ t\in\mathcal{T}}} \mathbf{B}_{rt}\bar{a}_{rt}\bar{\alpha}^{(G.1)}_{rjt} + \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} \mathbf{C}_j \bar{\alpha}^{(G.2)}_{jt} + \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} (\mathbf{A}_j - \boldsymbol{\tau}_j \sum_{r\in\mathcal{R}_j} \mathbf{G}_r \bar{g}_{rt})\bar{\alpha}^{(G.3)}_{jt} + \sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}}} \mathbf{MIN}^p_{s0}\bar{\alpha}^{(G.4)}_{ps} -$$
$$\sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}\\ t\in\mathcal{T}}} \mathbf{MIN}^p_{st}\bar{\alpha}^{(G.6)}_{pst} + \sum_{\substack{p\in\mathcal{P}\\ h\in\mathcal{H}}} \mathbf{Q}^p_h z_h \bar{\alpha}^{(G.8)}_{ph} + \sum_{\substack{r\in\mathcal{R}\\ k\in\mathcal{K}\\ t\in\mathcal{T}\\ h\in\mathcal{H}}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\leq t\leq i_2}} \mathbf{E}_r \bar{q}_i \bar{\alpha}^{(G.12)}_{rkth} - \sum_{h\in\mathcal{H}} \sum_{i\in\mathcal{I}_h} \bar{q}_i \bar{\alpha}^{(G.13)}_h +$$
$$\sum_{\substack{j\in\mathcal{J}^{ld}\\ t\in\mathcal{T}\setminus\{\overline{T}-5,...,\overline{T}\}}} \mathbf{MAX}_j \bar{\alpha}^{(G.15)}_{jt} - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m}} \sum_{o\in\mathcal{O}} \bar{u}^p_{ho}\bar{\alpha}^{(G.16)}_{ph} - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m\\ r\in\mathcal{R}_o\\ o\in\mathcal{O}}} \mathbf{Q}^p_h \bar{u}^p_{ho}\bar{\alpha}^{(G.17)}_{phro} \tag{G.19}$$

Let $\mathcal{F}$ be the feasible region of the dual of $\mathcal{HBD}$ Subproblem (with fixed complicating variables) and $\Gamma_{\mathcal{F}}$ be the set of extreme points of $\mathcal{F}$ and $\Upsilon_{\mathcal{F}}$ be the set of extreme rays of $\mathcal{F}$. Introducing the additional free variable μ, the $\mathcal{HBD}$ Master Problem can be formulated as follows:

$$\textbf{Max} \quad w_{TF}\mu - w_{PC}PC(g) \qquad (\mathcal{HBD}\text{ Master Problem})$$

s.t.:

Chg. cstr.:
$$\sum_{r\in\mathcal{R}_j} a_{rt} \le 1 \qquad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{G.20}$$

$$\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}} a_{r,t-1} + a_{r't} \le 1 + g_{r't} \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T}\setminus\{1\} \tag{G.21}$$

$$g_{r't} \le \frac{1}{2}\Big(\sum_{\substack{r\in\mathcal{R}_j\\ r\neq r'}} a_{r,t-1} + a_{r't}\Big) \qquad \forall r' \in \mathcal{R}_j, j \in \mathcal{J}^{tf}, t \in \mathcal{T}\setminus\{1\} \tag{G.22}$$

Ld. cstr.:
$$\sum_{h\in\mathcal{H}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\le t\le i_2}} \mathbf{V}_h q_i \le \mathbf{L}_k \qquad \forall k \in \mathcal{K}, t \in \mathcal{T} \tag{G.23}$$

HBD Cuts:
$$\mu \le \sum_{\substack{r\in\mathcal{R}_j\\ j\in\mathcal{J}^{tf}\\ t\in\mathcal{T}}} \mathbf{B}_{rt} a_{rt} \bar{\alpha}^{(G.1)}_{rjt} + \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} \mathbf{C}_j \bar{\alpha}^{(G.2)}_{jt} +$$
$$\sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} (\mathbf{A}_j - \boldsymbol{\tau}_j \sum_{r\in\mathcal{R}_j} \mathbf{G}_r g_{rt}) \bar{\alpha}^{(G.3)}_{jt} + \sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}}} \mathbf{MIN}^p_{s0} \bar{\alpha}^{(G.4)}_{ps} -$$
$$\sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}\\ t\in\mathcal{T}}} \mathbf{MIN}^p_{st} \bar{\alpha}^{(G.6)}_{pst} + \sum_{\substack{p\in\mathcal{P}\\ h\in\mathcal{H}}} \mathbf{Q}^p_h z_h \bar{\alpha}^{(G.8)}_{ph} +$$
$$\sum_{\substack{r\in\mathcal{R}\\ k\in\mathcal{K}\\ t\in\mathcal{T}\\ h\in\mathcal{H}}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\le t\le i_2}} \mathbf{E}_r q_i \bar{\alpha}^{(G.12)}_{rkth} - \sum_{h\in\mathcal{H}} \sum_{i\in\mathcal{I}_h} q_i \bar{\alpha}^{(G.13)}_h$$
$$+ \sum_{\substack{j\in\mathcal{J}^{ld}\\ t\in\mathcal{T}\setminus\{\overline{T}-5,\ldots,\overline{T}\}}} \mathbf{MAX}_j \bar{\alpha}^{(G.15)}_{jt} - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m}} \sum_{o\in\mathcal{O}} u^p_{ho} \bar{\alpha}^{(G.16)}_{ph}$$
$$- \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m\\ r\in\mathcal{R}_o\\ o\in\mathcal{O}}} \mathbf{Q}^p_h u^p_{ho} \bar{\alpha}^{(G.17)}_{phro} \qquad \bar{\alpha} \in \Gamma_{\mathcal{F}} \tag{G.24}$$

$$0 \le \sum_{\substack{r\in\mathcal{R}_j\\ j\in\mathcal{J}^{tf}\\ t\in\mathcal{T}}} \mathbf{B}_{rt} a_{rt} \bar{\alpha}^{(G.1)}_{rjt} + \sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} \mathbf{C}_j \bar{\alpha}^{(G.2)}_{jt} +$$
$$\sum_{\substack{j\in\mathcal{J}\\ t\in\mathcal{T}}} (\mathbf{A}_j - \boldsymbol{\tau}_j \sum_{r\in\mathcal{R}_j} \mathbf{G}_r g_{rt}) \bar{\alpha}^{(G.3)}_{jt} + \sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}}} \mathbf{MIN}^p_{s0} \bar{\alpha}^{(G.4)}_{ps} -$$

$$
\begin{aligned}
&\sum_{\substack{p\in\mathcal{P}\\ s\in\mathcal{S}\\ t\in\mathcal{T}}} \mathbf{MIN}^p_{st}\bar{\alpha}^{(G.6)}_{pst} + \sum_{\substack{p\in\mathcal{P}\\ h\in\mathcal{H}}} \mathbf{Q}^p_h z_h \bar{\alpha}^{(G.8)}_{ph} + \\
&\sum_{\substack{r\in\mathcal{R}\\ k\in\mathcal{K}\\ t\in\mathcal{T}\\ h\in\mathcal{H}}} \sum_{\substack{i\in\mathcal{I}_{hk}\\ i_1\le t\le i_2}} \mathbf{E}_r q_i \bar{\alpha}^{(G.12)}_{rkth} - \sum_{h\in\mathcal{H}} \sum_{i\in\mathcal{I}_h} q_i \bar{\alpha}^{(G.13)}_h \\
&+ \sum_{\substack{j\in\mathcal{J}^{ld}\\ t\in\mathcal{T}\setminus\{\overline{T}-5,\dots,\overline{T}\}}} \mathbf{MAX}_j \bar{\alpha}^{(G.15)}_{jt} - \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m}} \sum_{o\in\mathcal{O}} u^p_{ho}\bar{\alpha}^{(G.16)}_{ph} \\
&- \sum_{\substack{p\in\mathcal{P}_h\\ h\in\mathcal{H}^m\\ r\in\mathcal{R}_o\\ o\in\mathcal{O}}} \mathbf{Q}^p_h u^p_{ho} \bar{\alpha}^{(G.17)}_{phro} \qquad \bar{\alpha}\in\Upsilon_{\mathcal{F}} \qquad \text{(G.25)}
\end{aligned}
$$

Non-negativity and binary conditions in F (G.26)

To keep the vessel assignment information in the $\mathcal{HBD}$ Master Problem, we strengthen it using the following valid inequalities:

$$\sum_{i\in I_{hk}} q_i \le 1 \qquad \forall h\in\mathcal{H}, k\in\mathcal{K} \qquad \text{(G.27)}$$

$$\sum_{\substack{i\in I_h\\ i_1\le t\le i_2}} q_i \le 1 \qquad \forall h\in\mathcal{H}, t\in\mathcal{T} \qquad \text{(G.28)}$$

The $\mathcal{HBD}$ Subproblem provides a lower bound (LB), while the $\mathcal{HBD}$ Master Problem provides an upper bound (UB). The method stops when the $|UB - LB| < \epsilon$, where ϵ is a given threshold.

APPENDIX H PSIMVA INSTANCES DESCRIPTION

We report in Table H.1 the scheduling horizon (in days), the number of vessels, the demand (in tonne), the number of variables, integers, binaries, and constraints.

Table H.1 PSIMVA Instances Description

Instance	**Horizon**	**Vessels**	**Demand**	**Variables**	**Integers**	**Binaries**	**Constraints**
$\mathbf{J}_1$	32	48	1320580	470310	170772	12314	1560843
$\mathbf{J}_2$	32	62	2031400	947598	366330	17966	3506473
$\mathbf{J}_3$	32	61	1066290	936657	359570	11155	3610913
$\mathbf{J}_4$	24	40	1043330	298693	142362	33284	1489253
$\mathbf{J}_5$	30	58	1797910	450772	192268	15144	1964908
$\mathbf{J}_6$	30	58	1797910	450772	192276	15237	1964908
$\mathbf{J}_7$	30	58	1740100	450789	192292	15237	1957865
$\mathbf{J}_8$	32	61	957338	948009	360419	18264	3679588
$\mathbf{J}_9$	30	38	856686	126314	42105	16695	110922
$\mathbf{J}_{10}$	30	38	846702	125880	44058	16292	109977
$\mathbf{J}_{11}$	34	34	926476	106020	40408	17289	197896
$\mathbf{J}_{12}$	30	46	2340500	513300	141894	14894	1843150
Avg	**31**	**50**	**1393769**	**485426**	**187063**	**16981**	**1833058**

APPENDIX I RESULTS ON THE SECOND KPI

We report in Table I.1 the runtime (Runtime) in seconds and the number of product changeovers (Change) using the $\mathcal{DLP}$ and the $\mathcal{IDLP}$. For the $\mathcal{DLP}$, we recall that it relies on a weighted sum objective function. Thus, we collect the PC obtained after 10 hours. We allocate 10 minutes for the $\mathcal{IDLP}$. As shown in Table I.1, we obtain optimal solutions quickly. After maximizing TF in the first stage, PC minimization becomes easy. On the other hand, the $\mathcal{DLP}$ does not reach optimality in all instances.

Table I.1 Results obtained on J Instances using the $\mathcal{DLP}$ and the $\mathcal{IDLP}$ for the PC KPI

Instance	$\mathcal{DLP}$		**This Paper**	
	Runtime	**Change**	**Runtime**	**Change**
$\mathbf{J}_1$	36000	0	1	0
$\mathbf{J}_2$	36000	0	2	0
$\mathbf{J}_3$	36000	4	15	0
$\mathbf{J}_4$	36000	4	13	2
$\mathbf{J}_5$	36000	4	15	2
$\mathbf{J}_6$	36000	4	17	2
$\mathbf{J}_7$	36000	8	20	2
$\mathbf{J}_8$	36000	12	28	2
$\mathbf{J}_9$	36000	8	31	2
$\mathbf{J}_{10}$	36000	12	20	2
$\mathbf{J}_{11}$	36000	15	50	4
$\mathbf{J}_{12}$	36000	15	60	4
Avg	**36000**	**7**	**24**	**2**