Universidade Federal de Minas Gerais
Escola de Engenharia
Departamento de Engenharia de Produção

Formulations and algorithms for a rich production-routing problem

Allexandre Fortes da Silva Reis

Belo Horizonte MG
2020

Allexandre Fortes da Silva Reis

Formulations and algorithms for a rich production-routing problem

Tese de Doutorado apresentada ao Curso de Engenharia de Produção da Universidade Federal de Minas Gerais como parte dos requisitos necessários para a obtenção do grau em Doutor em Engenharia de Produção.

Supervisor: Prof. Dr. Ricardo Saraiva de Camargo.

Belo Horizonte - MG
13 de novembro de 2020

R375f

Reis, Allexandre Fortes da Silva.
Formulations and algorithms for a rich production-routing problem [recurso eletrônico] / Allexandre Fortes da Silva Reis. - 2020.
1 recurso online (xi, 114 f. : il., color.) : pdf.

Orientador: Ricardo Saraiva de Camargo.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 86-114.

Bibliografia: f. 77-85.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia de produção - Teses. 2. Logística - Modelos matemáticos - Teses. 3. Transportes - Modelos matemáticos - Teses. 4. Veículos - Teses. 5. Otimização matemática - Teses. I. Camargo, Ricardo Saraiva de. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 658.5(043)

Ficha catalográfica: Biblioteca Profº Mário Werneck, Escola de Engenharia da UFMG

FOLHA DE APROVAÇÃO

Formulations and algorithms for a rich production-routing problem

ALLEXANDRE FORTES DA SILVA REIS

Tese submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA DE PRODUÇÃO, como requisito para obtenção do grau de Doutor em ENGENHARIA DE PRODUÇÃO, área de concentração PESQUISA OPERACIONAL E INTERVENÇÃO EM SISTEMAS SOCIOTÉCNICOS, linha de pesquisa Otimização e Simulação de Sistemas Logíst. e de Grande Porte.

Aprovada em 13 de novembro de 2020, pela banca constituída pelos membros:

Prof(a). Ricardo Saraiva de Camargo - Orientador
UFMG

DocuSigned by:
Gilberto Miranda

Prof(a). Gilberto de Miranda Junior
UFOP

Prof(a). Ricardo Poley Martins Ferreira
UFMG

Prof(a). Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto

Prof(a). Alexandre Xavier Martins
UFOP

Prof(a). George Henrique Godim da Fonseca
UFOP

Belo Horizonte, 13 de novembro de 2020.

Dedico esse trabalho a minha esposa Fernanda, minha mãe Cláudia e irmãs Anna Carolina e Anna Cláudia, grandes incentivadoras desse caminho. Em especial, ao meu filho Bernardo que chegou ao final desta caminhada. Aos meus mestres que contribuiram para minha formação e a todos aqueles que lutam pela educação pública, inclusiva e de qualidade.

Acknowledgements

Agradeço ao meu orientador Prof. Ricardo Camargo, por toda ajuda, dedicação, paciência e alternativas propostas para o desenvolvimento deste estudo, além de todos os conselhos e apoio a situações fora deste trabalho.

Aos membros da banca por disponibilizarem seu tempo e por suas valiosas contribuições, sugestões e suporte ao desenvolvimento de conhecimento.

À minha família, meu maior alicerce, que foram as minhas grandes incentivadoras, me acompanhando e apoiando em todas as conquistas e momentos, sempre com paciência quando me ausentei para me dedicar aos estudos. Agradeço à minha amada esposa Fernanda, que ao longo desses 15 anos, sempre me ofereceu tudo que podia e me deu meu maior presente, nosso filho Bernardo. À minha fortaleza e mãe Cláudia, que se sacrificou tanto por minhas irmãs e por mim, não nos deixando faltar nada, sempre presente em nossas vidas e nos lembrando que o conhecimento é a única coisa que não pode ser tomado de nós. Às minhas queridas irmãs Anna Carolina e Anna Cláudia por todas as conversas, planos, carinho e amizade. Amo vocês!

Às famílias Fortes, Cota e Matos por todo incentivo e preocupação. Especialmente tia Anjinha, tio Beto, Vinícius e Vanessa, por sempre me apoiarem tanto, e tia Élida por acompanhar e incentivar esta caminhada.

Queridos amigos, Leila, Tom, Amanda, Karina, Vinicíus Lage, Gabi, Arthur, Edson, Júlio, Felipe, Guga e Téis (*in memoriam*), e especialmente ao Leandro Reis, pelas conversas, conselhos, estudos, apadrinhamento, sala e caronas divididas.

Aos meus caros colegas da Engenharia de Produção da Universidade Federal de São João del-Rei, Robson Dutra, Wilson Trigueiro, Guilherme Germano, Lincoln Brandão, Kívia Nascimento, Roberta Alves e Flávio Napolitano, e demais professores do Departamento de Engenharia Mecânica e Produção, pelo suporte durante este trabalho.

À Universidade Federal de Ouro Preto, por me formar engenheiro e professor. À Universidade Federal de Minas Gerais por me conceder ambos graus de pós-graduação com excelência. À Universidade Federal de São João del-Rei, onde me realizo profissionalmente e recebi apoio através do PQualis e demais políticas de capacitação.

Ao Magno Silvério, pela imensa ajuda na realização dos testes estatísticos e ao Paganini Barcelos pelas conversas e ajuda nas análises.

Aos grandes mestres que tive, especialmente, ao professor Gustavo Peixoto Silva pela iniciação à pesquisa científica.

A todos vocês, o meu muito obrigado!

Fortis Fortuna adiuvat

Publius Terentius Afer (Pliny Epistles 6 16)

Abstract

This work studies a rich production-routing problem, which consists of determining, at minimal cost, a production and distribution plan for a mix of products to be delivered via routes of heterogeneous fleet to supply different demand patterns of scattered clients over time while controlling the inventory levels at the plant and the customers. The problem still allows back-order deliveries and limit the riding time of the vehicles. Two formulations were proposed for the problem. Given that the problem scales quickly with the number of the customers, periods, products and vehicles, different approaches were devised. First, three hybrid two-level decomposition using a top-down strategy were developed. The top tier decides the production and inventory levels, and the distribution of goods via CPLEX; whereas the bottom one routes the heterogeneous fleet heuristically in each period. The methods rely on an iterated local search framework combined with new perturbations schemes that operate in both tactical and operational levels. At the tactical level, feasible moves modify the production plans, shifting quantities manufactured between different periods, while at the operational the perturbations change the routing design. The top-down algorithms adopt a new implicit cost for the delivered loads, that reflects their influence of over the production, holding and transportation decisions, providing an important aid in yielding better solutions. An adaptive matheuristic working with a bottom-up strategy is proposed. It selects operators to modify the distribution and routing plans, which are reoptimized in the sequence. The best plans are then fixed for the tactical problem. Finally, a column generation approach is provided to achieve lower bounds better than the ones attained by the formulations. An *ad hoc* labeling algorithm is proposed, and the columns are heuristically priced. The algorithms were tested over a proposed set of instances and the achieved results found more, better and faster solutions than CPLEX, where the methods that focus on the operational level reach the best results. Having the bottom-up algorithm performed better than the others.

Keywords: Iterated local search, Hybrid methods, Matheuristics, Column generation, Production-routing problem.

Resumo

Este trabalho estuda um problema enriquecido e integrado de produção e roteamento, que consiste em determinar, a um custo mínimo, os planos de produção e distribuição para um mix de produtos a serem entregues através de rotas percorridas por uma frota heterogênea para suprir diferentes padrões de demanda de clientes espalhados ao longo do tempo enquanto controla os níveis de estoques na planta e nos clientes. O problema ainda permite a postergação das entregas e limita o tempo de viagem de cada veículo. Duas formulações foram propostas para o problema. Dado que os problema cresce rapidamente com o número de clientes, períodos, produtos e veículos, diferentes métodos foram desenvolvidos. Primeiro, são devenvolvidos três métodos híbridos que decompõem o problema em dois níveis usando uma estratégia top-down. O nível superior decides os níveis de produção e inventário e a distribuição dos produtos via CPLEX; enquanto o nível inferior roteia a frota heterogêna heuristicamente para cada período. Os métodos são imbutidos em um escopo da busca local iterativa combinado com novos esquemas de perturbação que operam em ambos níveis tático e operacional. No nível tático, movimentos viáveis modificam os planos de produção, transferindo quantidades produzidas entre diferentes períodos, enquanto no nível operacional as perturbações alteram o arranjo das rotas. Estes algoritmos top-down adotam um novo e implícito custo para as cargas entregues, que reflete sua influência sobre as decisões relativas a produção, inventário e transporte, provendo um importante auxílio no alcance de soluções melhores. Uma mateurística trabalhando com uma estratégia bottom-up é proposta. Ela seleciona operadores para modificar os planos de distribuição e roteamento, que são então reotimizados em sequência. Os melhores planos são então fixados no nível tático do problema. Finalmente, uma aproximação por geração de colunas e provida para alcançar limites inferiores melhores do que os obtidos pelas formulações. Um algoritmo de rotulamento ad hoc é proposto e as colunas são precificadas heuristicamente. Os algorimos foram testas em um conjunto proposto de instâncias e alcançaram resultados melhores e mais rapidamente do que o CPLEX, onde os métodos que focam no nível operacional obtiveram os melhores resultados. Sendo que o algoritmo bottom-up teve desempenho melhor do que os demais.

Palavras-chave: Busca local iterativa, Métodos híbridos, Mateurísticas, Geração de colunas, Problema de produção e roteamento.

List of Figures

List of Tables

List of Algorithms

Contents

1 Introduction

"True optimization is the revolutionary contribution of modern research to decision processes".

George Dantzig

1.1 Background

Most industries have complex production and logistics systems. These systems usually process several types of raw materials generating other ones that need to be transferred for the next echelons of the supply chain. Then, there are periodic decisions to be taken about what to produce, to store, and to distribute. The challenge to simultaneously make these decisions can be intimidating. Furthermore, to perform better, companies have also to consider their interactions with different levels of the logistics network, or to do what is known as supply chain management. Traditionally in these environments, the decisions about production and transportation have been made sequentially and independently (DÍAZ-MADROÑERO et al., 2015).

A supply chain integrates operations of manufacturing and logistics to fulfill orders of clients more effectively and efficiently. During this process, feed-stock coming from suppliers are converted to intermediate or final products at one or more industrial plants. After this, they are transported to their final destination being or not stored at intermediary distribution centers (ALMEIDA, 2015). Due to these facts, an integrated planning system is a powerful tool used to jointly optimize several decisions thereby, capturing all the benefits of coordination of the chain (ADULYASAK et al., 2015b), which usually happens in a natural way when the companies belong to the same group (COELHO et al., 2013).

Even in this context, the Production-Routing Problem (PRP) arises to assist this integrated decision making. The PRP is relatively recent, and its seminal papers are considered the studies of Chandra (1993) and Chandra and Fisher (1994), and it can be stated as the combination of two well-known combinatorial optimization problems, the lot-sizing (LSP) and the vehicle-routing (VRP). Over a planning horizon, the LSP considers decisions about production, holding, and distribution of one or more items to a set of geographically dispersed customers. Generally, it minimizes the costs related to setup, production, storage, and back-ordering (BRAHIMI; AOUAM, 2016). Periodic deliveries capable of satisfying the customer's demands are realized, and the VRP is responsible for provides these decisions besides how much to load on each vehicle and where to send it, developing routes that minimize the related operational cost (SIMCHI-LEVI et al., 2014).

Due to its relevance, the studies over the PRP have been intensified in the last years. But, because of the complexity of the problem, the majority of these works are focused on heuristic procedures (ADULYASAK et al., 2015b), and models are mostly limited to less complex cases that consider a single product and homogeneous fleet (MOSTAFA; ELTAWIL, 2015). However, in real life, many more complex cases exist, but that does not prevent successful examples like the following already reported in the literature. Brown et al. (2001) presented Kellogg's case, where an integrated production and distribution planning system was implemented, leading to potential savings between $35-40 million. Another example is the decrease of 10% in the logistics costs achieved by the Frito-Lay company after the implementation of an optimization system integrating production, inventory, distribution, and routing decisions (ÇETINKAYA et al., 2009).

1.2 Motivation and relevance of this research

The benefits of the integrated decisions are inviting, but they considerably elevate the complexity of the problem, making the PRP extremely challenging. Aiming to contribute to fill some literature lacks, this thesis aims to propose a rich PRP, in the sense that the proposed problem variant considers features that are not usually adopted simultaneously in the literature, delivering a more complex problem.

To treat this rich PRP, high-performance algorithms are needed. First, considering that the decisions realized by the PRP are carried out at different organizational levels, the tactical (production and inventory) and the operational (distribution and routing), hybrid algorithms, of the solver and heuristics, are proposed. They help to fill this gap in the literature with fast and precise results. Even linearly relaxed, the proposed problem states as a huge challenge, then this work also provides a column generation approach with columns heuristically priced. Thus, to make clear the purposes of this thesis and its contributions, the objectives that guide the work are presented in the next section, as well as the organization of the thesis in Section 1.4.

1.3 Objectives

The main objective of this work is to propose an innovative and challenging variant of the production-routing problem. This variant considers simultaneously representative features, besides efficient approaches to treat it, more specifically:

- To introduce a PRP with back-ordering, multi-products, mixed loads, a heterogeneous fleet, and a maximum riding time;
- To propose a vehicle-indexed model;
- To propose a two-commodity flow model;
- To solve the proposed problem with a top-down approach;
- To solve the proposed problem with a bottom-up approach;

- To attain lower bounds via a column generation approach;

In other words, to propose a model closer to reality but providing means to solve it and assessing the quality of the attained solutions.

1.4 Thesis organization

Chapter 1 introduces the theme of study. Chapter 2 reviews the main works of PRP and its importance. Chapter 3 presents our proposed models, comparisons between their performances, and describes the set of used instances, and the impacts of the adopted features in the solutions. Chapter 4 presents a set of efficient matheuristics, based on an iterated local search framework, their results, and analysis. These hybrid methods are based on top-down and bottom-up decision making. Chapter 5 describes the devised columns generation approach to attain lower bounds for the problem via heuristically priced columns, besides the proposition of an *ad hoc* labeling algorithm. Final remarks and an outline for future researches are presented in Chapter 6.

2 Literature review

"What we know is a drop, what we do not know is an ocean".

Sir Isaac Newton

2.1 Production-routing problem

Integrated production and distribution planning problems focus on the tactical and operational decision levels. According to Bard and Nananukul (2010), there are four critical decisions to be made: *(a)* how many items to manufacture each day; *(b)* when to visit each customer; *(c)* how much to deliver to a customer during a visit; and *(d)* which delivery routes to use. We can also include as a fifth decision, *(e)* the amount of each product to store at the manufacturer plants and clients.

The tactical level decides on the production, storage, and back-ordering, whereas the operational level, determines the product distribution and routing. The production-routing problem (PRP) optimizes the decision periodically and simultaneously being seen as a combination of the lot-sizing problem with the direct shipment (LSPDS) and the inventory-routing problem (IRP), as stated by Adulyasak et al. (2015b). Figure 2.1 (ADULYASAK et al., 2014b) illustrates how the decisions relate to the involved problems .

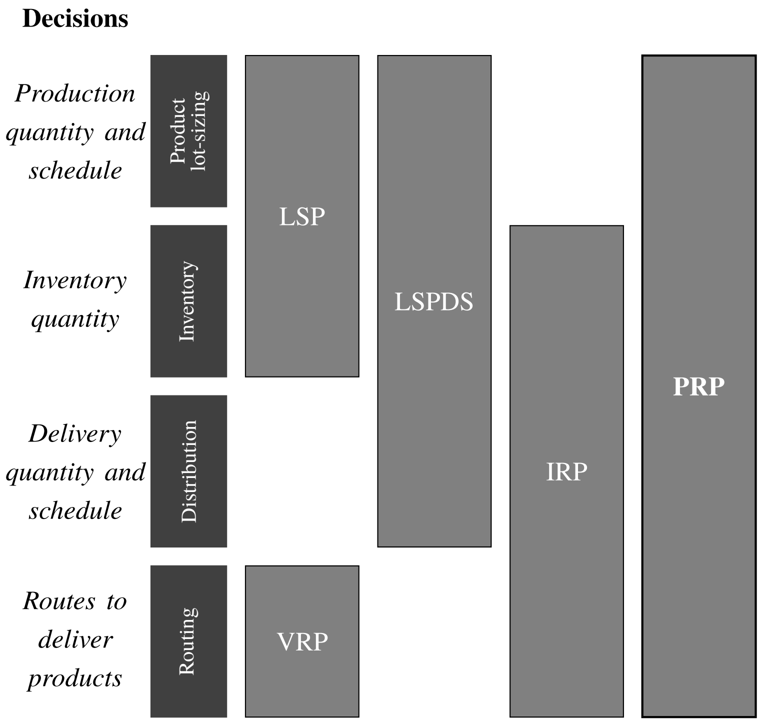
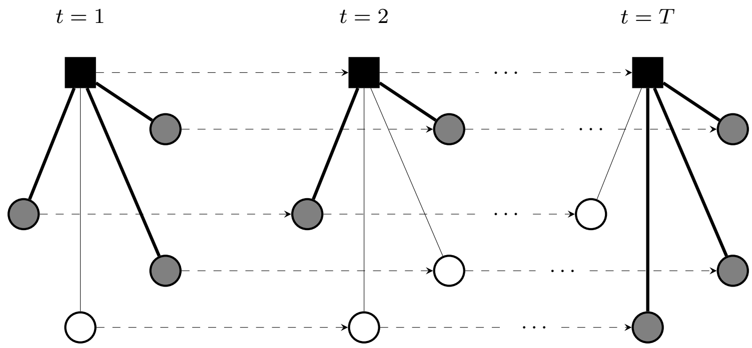
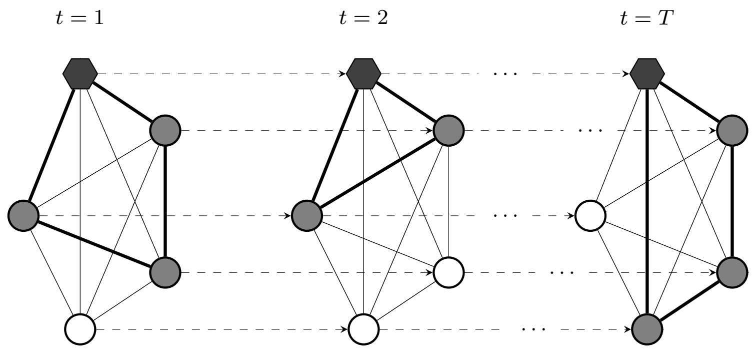


Figure 2.1 – Supply chain planning models.

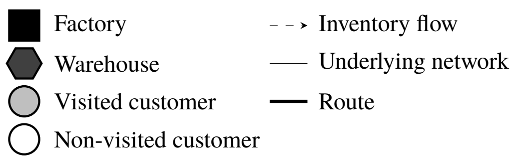
The LSPDS encompass decisions on how much a plant produces and dispatches, while minimizing the respective setup, production, holding, and fixed direct deliveries costs, for all periods. Network representation is illustrated in Figure 2.2a, where, represented by a square, a plant is directly connected to its clients. On the other hand, the IRP decides on the routing and inventory control while neglects the production aspects. Usually, the IRP consists of determining at a minimal total cost, which products to ship by which routes while controlling their corresponding inventory levels at the clients (DÍAZ-MADROÑERO et al., 2015). Network representation is shown in Figure 2.2b (ADULYASAK et al., 2014b), where a warehouse, represented by a bold hexagon, serves as the start and final points for routes that can visit more than one customer per period.



(a) LSPDS



(b) IRP



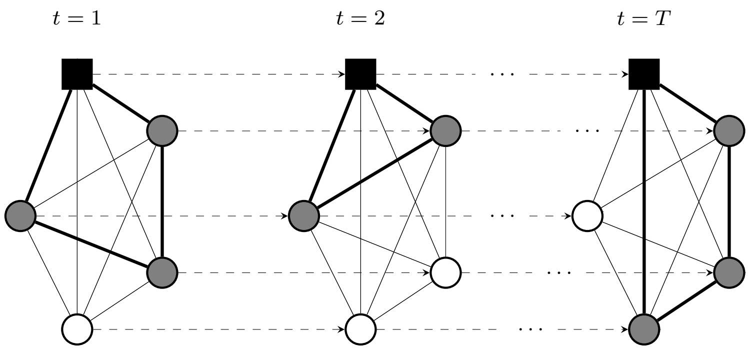
(c) Notation

Figure 2.2 – Network representations of the integrated problems.

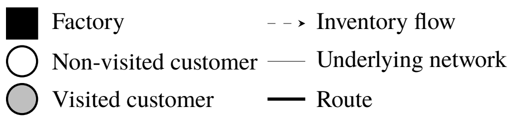
For further references on LSP, VRP, and IRP, please refer to the extensive reviews of Glock et al. (2014), Toth and Vigo (2014), and Coelho et al. (2013), respectively.

The production-routing problem is an integrated operational planning application that jointly optimizes production, inventory, distribution, and routing decisions to produce an optimal solution when considering the total system cost (ADULYASAK et al., 2015b). Its practical relevance relies upon the savings it grants to a supply chain by efficiently use the available resources. It is an NP-hard problem as it contains a variant of the VRP (BOUDIA et al., 2007; ARCHETTI et al., 2011).

A network structure of PRP is illustrated in Figure 2.3a (ADULYASAK et al., 2015b). A central plant, represented by a bold square, produces a set of items, which are delivered to satisfy the demands of each client periodically. Both plant and clients have their stocks of finished products. These inventories can be carried or not through the time horizon while being resupplied. The distribution is done by a set of vehicles. All these activities have an associated cost.



(a) PRP



(b) Notation

Figure 2.3 – Network representation of PRP.

Chandra (1993) is considered the seminal work on the PRP since previous works optimized production, inventory management, and routing decisions simultaneously but only for one period at a time, whereas Chandra (1993) did for all the periods at once.

2.2 Basic formulations for the PRP

Accordingly to Adulyasak et al. (2015b), a PRP network is defined on a complete directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ represents the set of the plant and the clients indexed by $i \in \{0, \ldots, n\}$ and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}, i \neq j\}$ is the set of arcs. The plant is represented by node 0 and we further define the set of customers $\overline{\mathcal{N}} = \mathcal{N} \setminus 0$. Over a finite set of periods $\mathcal{T} = \{1, \ldots, T\}$, a single product can be manufactured at the plant and delivered by a set of identical vehicles $\mathcal{V} = \{1, \ldots, V\}$ to the customers to satisfy the demands in each period.

The parameters are defined as follows: l is the fixed setup cost, u is the unit production cost, h_i is the unit holding cost at node i, c_{ij} is the transportation cost from node i to node j, d_i^t is the demand at customer i in period t, C is the production capacity, Q is the vehicle capacity, U_i is the maximum inventory level at node i, I_i^0 is the initial inventory at node i. At a given period t, we further let $\overline{M}^t = \min\{C, \sum_{\tau=t}^{T} \sum_{i\in\overline{\mathcal{N}}} d_i^\tau\}$, and $\widetilde{M}_i^t = \min\{U_i, Q, \sum_{i\in\overline{\mathcal{N}}} d_i^t\}$ represent the maximum quantities to manufacture at the plant node 0 and to deliver at client i, respectively.

The decision variables are described as follows: y^t is equal to 1 if there is production at the plant in the period t, 0 otherwise; p^t is the production quantity in period t; I_i^t is the inventory at node i at the end of period t; x_{ij}^t is equal to 1 if a vehicle travels directly from node i to node j in period t, 0 otherwise; q_i^t is the quantity delivered to customer i in period t; z_i^t is equal to 1 if customer i is visited in period t, 0 otherwise; z_0^t is the number of vehicles leaving the plant in period t, o_i^t is the load of a vehicle before making a delivery to customer i in period t.

2.2.1 A lot-sizing and vehicle-routing based formulation

This model is based on the basic LSP and VRP formulations. It is also the most compact one as it contains a polynomial number of constraints. The PRP is formulated with variables that control the amounts delivered by a homogeneous fleet of vehicles. This model was first introduced by Bard and Nananukul (2009a).

$$\min \sum_{t\in\mathcal{T}} \left(ly^t + up^t + \sum_{i\in\mathcal{N}} h_i I_i^t + \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}^t \right) \tag{2.1}$$

s.t.:

$$p^t \leq \overline{M}^t y^t \qquad \forall t \in \mathcal{T} \tag{2.2}$$

$$I_0^t = I_0^{t-1} + p^t - \sum_{i\in\overline{\mathcal{N}}} q_i^t \qquad \forall t \in \mathcal{T} \tag{2.3}$$

$$I_i^t = I_i^{t-1} + q_i^t - d_i^t \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{2.4}$$

$$I_0^t \leq U_0 \qquad \forall t \in \mathcal{T} \tag{2.5}$$

$$I_i^t + q_i^t \leq U_i \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{2.6}$$

$$q_i^t \leq \widetilde{M}_i^t z_i^t \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{2.7}$$

$$\sum_{j\in\mathcal{N}} x_{ij}^t = z_i^t \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{2.8}$$

$$\sum_{j\in\mathcal{N}} x_{ji}^t + \sum_{j\in\mathcal{N}} x_{ij}^t = 2z_i^t \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{2.9}$$

$$z_0^t \leq V \qquad \forall t \in \mathcal{T} \tag{2.10}$$

$$o_i^t - o_j^t \geq q_i^t - \widetilde{M}_i^t (1 - x_{ij}^t) \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{2.11}$$

$$0 \leq o_i^t \leq Q z_i^t \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{2.12}$$

$$p^t, I_i^t, q_i^t \geq 0 \qquad \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{2.13}$$

$$y^t, x_{ij}^t \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{2.14}$$

$$z_i^t \in \{0,1\} \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{2.15}$$

$$z_0^t \in \mathbb{Z}^+ \qquad \forall t \in \mathcal{T} \tag{2.16}$$

The objective function (2.1) minimizes the total production, setup, inventory, and routing costs. Constraints (2.2)-(2.6) represent the lot-sizing part of the problem. Constraints (2.2) ensure activation of the production setup capacity constraints while limiting the production lot size to the minimum value between the production capacity and the total remaining demand. Constraints (2.3) e (2.4) are the inventory flow balance at the plant and customers, respectively. Constraints (2.5) and (2.6) limit the maximum inventory at the plant and customers, respectively. The inventory part of this model is controlled by the so-called maximum level (ML) policy, where the delivered quantity cannot exceed the maximum inventory level. Constraints (2.7)-(2.12) represent the vehicle loading and routing constraints. Constraints (2.7) allow a positive delivered quantity only if customer i is visited in period t. Constraints (2.8) define that each customer can be visited by at most by one vehicle. Constraints (2.9) are degree constraints. Constraints (2.10) limit the number of vehicles that can be used. Constraints (2.11) prevent vehicle overloading and the formation of subtours.

2.2.2 A vehicle-indexed formulation

A formulation with a vehicle index can impose routing constraints on each vehicle separately. In this formulation, the variables q_i^{vt}, z_i^{vt} and x_{ij}^{vt} have the same interpretation as q_i^t, z_i^t and x_{ij}^t but they are associated with vehicle v only. The main advantage of this formulation is the possibility to solve the problem with a heterogeneous fleet of vehicles, i.e., a fleet with different capacities and activation costs. The formulation with a vehicle index is based on the ones presented by Boudia et al. (2007), Boudia et al. (2008), and it is described as follows.

$$\min \sum_{t\in\mathcal{T}} \left(ly^t + up^t + \sum_{i\in\mathcal{N}} h_i I_i^t + \sum_{(i,j)\in\mathcal{A}} c_{ij} \sum_{v\in\mathcal{V}} x_{ij}^{vt}\right) \tag{2.17}$$

s.t.:

$$p^t \le \overline{M}^t y^t \qquad \forall t\in\mathcal{T} \tag{2.18}$$

$$I_0^t = I_0^{t-1} + p^t - \sum_{\in\mathcal{V}} \sum_{i\in\overline{\mathcal{N}}} q_i^{vt} \qquad \forall t\in\mathcal{T} \tag{2.19}$$

$$I_i^t = I_i^{t-1} + \sum_{\in\mathcal{V}} q_i^{vt} - d_i^t \qquad \forall i\in\overline{\mathcal{N}}, t\in\mathcal{T} \tag{2.20}$$

$$I_0^t \le U_0 \qquad \forall t\in\mathcal{T} \tag{2.21}$$

$$I_i^t + \sum_{\in\mathcal{V}} q_i^{vt} \le U_i \qquad \forall i\in\overline{\mathcal{N}}, t\in\mathcal{T} \tag{2.22}$$

$$q_i^{vt} \le \widetilde{M}_i^{vt} z_i^{vt} \qquad \forall i\in\overline{\mathcal{N}}, v\in\mathcal{V}, t\in\mathcal{T} \tag{2.23}$$

$$\sum_{v\in\mathcal{V}} z_i^{vt} \le 1 \qquad \forall i\in\overline{\mathcal{N}}, t\in\mathcal{T} \tag{2.24}$$

$$\sum_{j\in\mathcal{N}} x_{ji}^{vt} + \sum_{j\in\mathcal{N}} x_{ij}^{vt} = 2z_i^{vt} \qquad \forall i\in\overline{\mathcal{N}}, v\in\mathcal{V}, t\in\mathcal{T} \tag{2.25}$$

$$\sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}} x_{ij}^{vt} \le \sum_{k\in\mathcal{S}} z_k^{vt} - 1 \qquad \forall \mathcal{S}\subseteq\overline{\mathcal{N}} : |\mathcal{S}|\ge 2, v\in\mathcal{V}, t\in\mathcal{T} \tag{2.26}$$

$$\sum_{i\in\overline{\mathcal{N}}} q_i^{vt} \le Q^v z_0^{vt} \qquad \forall v\in\mathcal{V}, t\in\mathcal{T} \tag{2.27}$$

$$p^t, I_i^t, q_i^{vt} \ge 0 \qquad \forall i\in\mathcal{N}, v\in\mathcal{V}, t\in\mathcal{T} \tag{2.28}$$

$$y^t, x_{ij}^{vt}, z_i^{vt} \in \{0,1\} \qquad \forall i,j\in\mathcal{N}, v\in\mathcal{V}, t\in\mathcal{T} \tag{2.29}$$

The objective function (2.17) and constraints (2.18)-(2.25) have the same meaning as (2.1), (2.2)-(2.9), respectively. Constraints (2.26) are subtour elimination, while constraints (2.27) limit the maximum carried load by each vehicle.

2.3 Solutions approaches

The PRP has delivered a very high computational challenge since its first publication by Chandra (1993). Through years, many methods were used to solve it, trying to reach optimally, feasible solutions, or even to compute lower bounds for further researches.

2.3.1 Heuristics and metaheuristics

Decomposition approaches are among the most intuitive and adopted methods by PRP literature (ADULYASAK et al., 2015b). Chandra (1993) followed this strategy and presented a PRP with several products and an unlimited homogeneous fleet to serve client demand partially or not on the rolling horizon time, but not back-logged. Extending this study, Chandra and Fisher (1994) developed two approaches, the first problem still is solved separately, whereas, in the second, it was solved in an integrated manner. For the uncoupled approach the authors used a derivative method from Barany et al. (1984), and Leung et al. (1989) for production planning. To solve the routing subproblems, they employed different heuristics such as the sweep procedure (GILLETT; MILLER, 1974), nearest neighbor (ROSENCRANTZ; STEARNS; LEWIS, 1974) or feasible insertion (CHANDRA, 1991) to select the overall best. Next, they modified the local search procedure with a new feasibility test to allow the shifting of the production between periods while considering the integration of the subproblems. The analysis showed that depending on the relative scale of the cost parameters, the attained solutions by the methods differed substantially, from 3% to 20% of the difference in favor of the integrated approach.

Bertazzi et al. (2005) developed a nonlinear programming approach to solve the PRP after its decomposition. Their method was based on two hierarchical algorithms. The first method was referred to as VMI-PDP, similar to the one proposed by Chandra and Fisher (1994). It solved the production subproblem assuming that all the retailers were served daily. For a given production plan, the distribution subproblem was solved. The production subproblem was solved again with the updated with the quantities shipped to each retailer. The second was known as VMI-DP. It firstly solved the distribution plan, fixing sufficient quantities to serve retailers as the initial production quantity. Next, it solved the production plan. An acyclic network reformulation was built and determined the shortest path it solved the production problem to optimality, as described in Lee and Nahmias (1993).

Aiming to maximize the profits, Park (2005) solved a decoupled PRP under the direct shipment policy with a capacitated fleet. The proposed heuristic was partially based on the local search done by Chandra and Fisher (1994), and worked transferring loads to earlier periods.

There were two phases, first, a production-distribution plan was established and in the second, improvements of these plans were realized. The improvements were done by consolidating partial loads and reducing stock-outs at retailers. After established, the production plan was enabled to change for integrated planning but, it was fixed for decoupled planning.

Boudia et al. (2005) presented a heuristic to solve the PRP separately. The production plan was solved by the Wagner and Whitin (1958) procedure, and the distribution was defined in the sequence and optimized with the 2-opt (LIN; KERNIGHAN, 1973). An update was realized in a manner that the production was determined after the routing, allowing a small integration of the solution. A benchmark set of instances was proposed considering 20 periods, 50, 100, or 200 clients, homogeneous fleet, one item, and a single-capacitated plant. This set was adopted by several works, as Boudia et al. (2007), Boudia and Prins (2009), Bard and Nananukul (2009b), and Adulyasak et al. (2014b).

Lei et al. (2006) solved the maritime PRP with a two-phase heuristic. The problem was characterized by a single commodity, multiple plants, and a heterogeneous fleet. The first phase was solved with a commercial solver considering an LSPDS. As the solution of this phase was always feasible, it served as an upper bound for the solution. Next, in the second phase, the tactical decisions from the previous phase were fixed, and two heuristics determined the routes for each vehicle at each plant in each period. The heuristics tested neighbor solutions and built new ones using the closest neighbor. Good results were achieved in considerable computational times when compared with the CPLEX, that not even find feasible solutions for some problems.

The PRP allows us to integrate decisions for the production of perishable goods, as Chen et al. (2009) and Piewthongngam et al. (2013). Chen et al. (2009) assumed that the goods could not be stored, which required them to formulate the problem with short-term planning decisions with routing operations. The authors modeled the problem as a non-linear program with a profit maximization objective function, while considering traveling times between each pair of nodes, and the service times related to the loading and unloading of goods. They also assumed soft time windows with penalty costs for the vehicles that arrived late at a node or made them wait if they arrived. To solve, they decomposed the model into two, one regarding the production and the other the VRPTW. The production problem held the non-linearity and it was handled with the Nelder-Mead method with boundary constraints. The resulting solution attained a set of fixed variables for the VRPTW, an initial solution was found based on the cheapest insertion algorithm and improved by a 2-opt (LIN; KERNIGHAN, 1973) and Or-opt (OR, 1977) local search methods.

Piewthongngam et al. (2013) adapted the PRP to a real case of the swine supply chain. Multiple feel mills supplied swine food for a set of swine farms. The ration could be delivered in bags, boxes, or both for the clients. These pig farms had to control their inventories to accommodate the specific food received. To deliver, each one of the trucks available could operate in a limited set of clients and must be always fully loaded. As the problem size increased quickly, solvers were not able to perform. Then the authors proposed a heuristic that has two parts, the first to determine the order quantity (based on analysis of the data), the number of trucks required in each farm group, and the production batch size. The second part was used to arrange the trucks to service the swine farms. The results compared to the solver were only 5% worst, but much faster.

Hein and Almeder (2016) studied and solved a PRP approach named capacitated lot sizing and supply-side vehicle routing problem. In this problem, there were many suppliers, each providing just one product, to only one customer. The suppliers had limited production capacity and the fleet was composed of a set of limited homogeneous vehicles. They incorporated the just-in-time (JIT) philosophy, with an inventory equal zero at the end of each period. As the idea was to consume the input material per period, the replacement cannot be classified as ML or OU. The first step solved the CLSP in two different scenarios, considering and not the JIT inventory policy. While the second was an IRP or VRP, accordingly with the presence of JIT, respectively. The problems were solved using the commercial solver IBM CPLEX. The study showed that companies that follows JIT may expect higher gains from coordinated plans.

Absi et al. (2014) designed a two-phase iterative mixed-integer program (MIP) based heuristic to solve the uncapacitated version of the problem. The MIP was reformulated by replacing the variables and costs of the routing component in the original model with approximated fixed costs for the visitation of the retailers. After solved, it obtained the production, inventory, and customer visit decisions. Next, a routing heuristic was called. When compared their results with Archetti et al. (2011) and Adulyasak et al. (2014b), their algorithm outperformed both. But, this work was outperformed in both solutions and gaps by the studies of Solyalı and Süral (2017) and Russell (2017).

Solyalı and Süral (2017) worked with a five-phase heuristic. It first solved a huge TSP considering the plant and retailers. With this information, the second step defined the production quantities, the retailers that must be visited, and the delivered quantities, using a MIP. After, the routes were found with the resolution of a Capacitated Vehicle Routing Problem (CVRP). To improve the routing, another MIP was solved allowing the insertion or removal of retailers from the routes and alterations over the production planning. The last phase solved a TSP for each vehicle to improve their routes.

Russell (2017) adopted a matheuristic that considered predetermined or seeded routes. The approach with predetermined routes relied on a set partitioning formulation, where the routes were previously generated with artificial demands. The solution to this problem served as an input for a reactive tabu search, which improved the solution by using exchange moves. The last phase employed a multi-iteration improvement search that estimated the insertion of retailers for each period. The seeded routes method estimated the cost of inserting some retailers on a vehicle. This insertion calculated the saving to cluster clients, very similar to the Clarke and Wright (1964).

Miranda et al. (2018) and Qiu et al. (2018b) divided their problem into an LSP and VRP, in which the former was solved by a solver whereas the latter heuristically. Miranda et al. (2018) presented a formulation with products sharing the production line during some periods. Their objective function minimized the costs related to the setups of interchanging products on the production line, to holding inventory at the plant, and to the routing of vehicles. Their clients were served only one time over the planning horizon. The first phase solved a lot-scheduling with direct shipments problem (LSDSP) which also allocated customers to the routes. Next, routing decisions were made by solving a multi-trip VRP with time-windows. Customers could be reallocated between routes if the movement kept the feasibility of the solution while improving it. The method achieved near-optimal solutions when compared with CPLEX.

Qiu et al. (2018b) applied a skewed general variable neighborhood search and guided variable neighborhood descent (GVND) (HANSEN; MLADENOVIĆ, 2001) to the delivery and routing variables, respectively. During the construction phase, a production-distribution problem was solved near optimality. Then, Clarke and Wright (1964) algorithm was applied to obtain a set of routes that were improved by GVND. The method outperformed existing methods on solving benchmark instances of Archetti et al. (2007), Boudia et al. (2007).

Due to the combinatorial nature of the PRP, the Tabu Search (TS), proposed by Glover (1986), plays an important role in some approaches. Van Buer et al. (1999) studied a newspaper PRP with a short-term planning horizon in which an extremely perishable good could not be stored. To solve it, they proposed a non-linear programming approach with an objective function that minimized the total travel time instead of the holding and routing costs. Multiple trips per period by the same vehicle were allowed, and they also considered the use of the empty trucks at the end of the routes to perform recycling pickups to obtain further cost savings. Two local search algorithms, TS and Simulated Annealing were applied to solve it. The initial solution was obtained by a heuristic combined with a neighborhood search with full insertion moves for lots and trucks. Through computational experiments with real data of a newspaper producer, the authors conclude that the use of recycling pickups impacted more on the objective function that the incorporation of better local searches.

Bard and Nananukul (2009b) adopted a three-phase approach based on a Reactive Tabu Search (RTS) to solve the PRP. An allocation model with the VRP constraints dropped was used to find good starting feasible points for the RTS, followed by a CVRP subroutine. Swap moves examined two customers in two consecutive periods to exchange the maximum possible number of goods between these two customers. Transfer moves found quantities to be delivered to a customer that was combined with deliveries from previous periods to reduce the transportation costs. The moves that led to an improved solution were stored in a tabu list. Infeasible solutions were not allowed during the search. After a solution was found, the path relinking search was applied to improve the results. Lower bounds for the optimum were obtained by solving the linear programming relaxation of the allocation model. Bard and Nananukul (2009a) extended their work to investigate an IRP with production decisions. Their approach minimized the costs for transportation, production setup, holding costs at the factory, and at the customer sites through the determination of production quantity, inventory at plant and customers, delivery quantity, and routing. A branch-and-price algorithm was used to solve, with the columns being generated to the master problem periodically. They devised a three two-step heuristics for solving the IRP component. The first step determined the quantity to be delivered to each customer in each period. The second step planned the routes by an adjustable VRP tabu search. They solved instances with up to 50 clients within one hour.

Shiguemoto and Armentano (2010) proposed a TS algorithm with a relaxation mechanism that allowed the evaluation of infeasible solutions to guide a solution search by jointly contemplating the production and setup costs. Only the vehicle capacity was allowed to be violated during the searching process, but not for the final solution. The initial solution was constructed using an initial inventory and production plan to meet the demand exactly, to be then routed. Their TS algorithm had a short and long-term memory which forbade certain moves and solutions previously visited.

Armentano et al. (2011) proposed a TS with a path relinking search for the PRP. The initial solution was created by setting delivered quantities equal to demands and applying two algorithms based on savings: the Wagner and Whitin (1958) for the production, and Clarke and Wright (1964) for routing. The neighborhood search moves were similar to the ones made by Bard and Nananukul (2009b), but allowing a combination of quantities to be delivered for future periods only. During every iteration, linear programming optimized the production and inventory quantities at the plant. The local search used the maximum number of iterations. Two algorithms were devised, the first used only short-term memory, and the second used a path relinking to diversify the search within long-term memory.

Boudia et al. (2007) used a greedy randomized adaptive search procedure (FEO; RESENDE, 1989) on a problem with a single plant and product, delivered by a homogeneous limited fleet. A two-phase algorithm created production and delivery plans period-by-period. The algorithm shifted some deliveries on the time horizon to achieve a compromise between setup and storage costs at the plant. They used an adaptation of the Wagner and Whitin (1958) algorithm. For that, for each period, the Clarke and Wright (1964) saving algorithm was used to create the routes. The local search step applied 3-opt, inserting, and swapping neighborhoods improved the initial solution. To further improve the routes, a path relinking procedure was also developed (BOUDIA et al., 2008).

Approaches using the adaptive large neighborhood search (ALNS), see Ropke and Pisinger (2006), also achieved good results. Adulyasak et al. (2014b) were the first to use it for the PRP. An initial phase treated the production and routing variables separately and generated a pool of solutions. Next, an ALNS improved these solutions using the selection and transformation operators. These operators handled the setup and routing decisions with an enumeration scheme and upper-level search. After fixing these variables, the continuous ones were adjusted by solving a minimum cost flow problem. New solutions were accepted using a simulated annealing (SA) criterion. Extensive computational experiments were performed on benchmark instances from Boudia et al. (2005) and Archetti et al. (2011). The proposed algorithm generally outperformed the existing heuristics at that time and produced high-quality solutions within short computational times.

Belo-Filho et al. (2015) addressed a case with perishable goods also adopting the ALNS and SA acceptance criterion. They assumed that the shelf life of the perishable goods was shorter than the planning horizon enforcing then a periodic serving time-window or and interval in which a client was restricted to be served. An initial solution was constructed via the concept of first to come, first to serve, prioritizing earlier time windows. The results were compared with CPLEX to be near-optimal solutions.

Some authors also experimented with nature-based heuristics, for example, Boudia and Prins (2009). They solved a problem with a single plant and product. The plant had limited capacity and a homogeneous fleet to serve clients without delay. The authors used a memetic algorithm with population management, and it simultaneously dealt with production and distribution decisions. An initial population was created with a heuristic procedure that first set a production plan for each period equal to the total demand. Then, a savings heuristic was used to generate the delivery and production plans, adjusted by a Wagner and Whitin (1958) algorithm. The next step generated new offsprings through a crossover. The local search of Boudia et al. (2007) was chosen to improve the offsprings, while population management was used to accept new solutions only if they improved the current one.

Calvete et al. (2011) considered a bi-level approach wherein the first level problem was solved as a multi-depot vehicle routing problem (MDVRP), while the second having the routing variables fixed decided on how much to produce in each plant. They solved the problem using an ant colony algorithm which first found a feasible solution for MDVRP using the nearest neighbor heuristic. Then the pheromone trail was updated for the arcs and, while the lower level decided the amounts to be produced by each plant and delivered to each retailer.

Kumar et al. (2016) incorporated time windows and the reduction of carbon footprint to the PRP. The authors modeled the problem as a bi-objective problem. Also, they incorporated a fleet of identical and capacitated vehicles. The population was represented as chromosomes that were under mutation and crossover operations. The algorithm ran over a set of real instances of the United Kingdom that were analyzed. To solve, they used both a self-learning swarm particle optimization (SLSPO) and NSGA-II, wherein most of the cases the SLSPO outperformed the NSGA-II.

Hybrid algorithms also have their space with the PRP, mixing the quality of both methods, the speed of heuristics with the determinism of the exact approaches. Bard and Nananukul (2010) introduced a heuristic based on a branch-and-price framework using the restricted master problem and subproblems. They added fixed costs per delivery made to a customer and developed a two-phase approach to design a reactive tabu search algorithm. In the first part of Phase I, an initial solution was found by solving an allocation model that determined customer delivery quantities. In the second part, these values become the demand for T independent routing problems, where T was the number of periods in the planning horizon. An efficient CVRP subroutine was called to find the solutions. In Phase II, a neighborhood search was performed to improve the allocations and routing assignments found in Phase I. The results obtained by performing computational experiments using the benchmark instances by Boudia et al. (2005) showed improvements in all cases which range from 10% to 20% if compared to those obtained by the previous GRASP procedure of Boudia et al. (2007). However, it was emphasized the increase up to 5 times in the running times.

Archetti et al. (2011) discussed the PRP under the ML and OU policies and developed a hybrid heuristic to solve the problem. Focused on the PRP with uncapacitated production and a single capacitated vehicle, the algorithm was composed of three successive steps. A distribution problem was solved assuming infinite production capacity at the plant. This plan provided which quantities must be delivered at each period to the customers. Next, the production plan sought to minimize holding and production costs. The improvement and last phase iteratively removed and reinserted two retailers as long a new and best solution was found. They also studied single retailers and single-vehicle variants, besides developing a branch-and-cut approach similar to that of Archetti et al. (2007) to solve them.

Considering a two-level supply chain with multiple items, production sites, and client areas and a discrete-time horizon, Melo and Wolsey (2012) proposed two formulations, whereas the second one provided better bounds and was used by a heuristic. The MIP heuristic proposed first solved the linear relaxation of the strong formulation and used its solution to generate a neighborhood. This neighborhood fixed the values of some integer variables, and the MIP was solved again considering only the set of neighbors generated. The method was tested considering sales and its absence, as defying capacities for the production. The major conclusion was that for a multi-commodity problem was that tighter was the capacity harder was to reach an optimal solution or to get close.

Introduced by Pochet and Wolsey (2006), the heuristics relax-and-fix (RF) and fix-and-optimize (FO) gained prominence to solve production-routing problems. These methods usually decompose the PRP period by period, sequentially by production and routing problems or combining periods and problems. Brahimi and Aouam (2016) used the RF to build a solution for the lot-sizing problem, and a record-to-record local search to optimize routes and make adjustments to the lot sizes, if necessary. Their heuristic outperformed the solver, solving the two formulations proposed while providing good solutions in a few minutes. The formulations differed where the first was a classical aggregate LSP and the second was based on a facility location problem. The second formulation showed itself stronger than the first but resulted in a non-linear model which were linearized. These authors were the first to address the PRP considering the possibility of the backorder.

Watanabe et al. (2017) used the RF to find an initial solution and the FO to improve it. Their problem was described following the vehicle-indexed formulation from Armentano et al. (2011). The RF partitioned took the binary variables which indicate production and distribution to apply the decomposition. The FO procedure adopted these decomposition schemes, and this procedure was repeated up to the last subset of variables was solved. Promising results were found for instances with up to 30 customers, but beyond that, neither the heuristics nor CPLEX was capable to solve.

Neves-Moreira et al. (2019) solved a large multi-perishable-item problem considering time-windows for a real meat producer with a three-phase FO methodology, The first phase simplified the problem dimensions trying to decrease the number of products, locations, and routes. It was done aggregating products with similar characteristics and low demands into sets of minor priority and clustering retail nodes with close geographical coordinates. The second built an initial solution to the problem, decomposing into one lot-sizing and many inventory routing problems. The last phase solved the IRP cluster by cluster and the disaggregating the products for the LSP with three strategies of decomposition for the FO: *(i)* periodically each VRP cost, *(ii)* each product LSP and *(iii)* local PRP, for each cluster.

2.3.2 Exact methods

Metters (1996) addressed the inter-dependency between transportation and production in the United States Postal Service. They only considered inventories at the production plants (regional and central post office and distribution facilities) as the routing component was related to outsourced partners, and the clients were not contemplated. So, the decision variables were representing inventory levels, their corresponding warehousing costs were not included in the objective function, which included fixed costs per vehicle used in each route. The approach also allowed to remove low potential or infeasible routes based on the areas that they were. It was done analogously to the "cluster-first, route-second" proposed by Fisher and Jaikumar (1981).

Jolayemi and Olorunniwo (2004) developed a deterministic model for planning production and transportation quantities in multi-plant and multi-warehouse environments with extensible capacities. The extensions of capacity could occur at warehouses on any period, if necessary. The distribution was accounted for direct shipments from depots to customers. Their model determined the product mix that maximized the total profit, and it was solved to optimality using the solver LINDO in a set of real data instances.

Bard and Nananukul (2010) proposed an RMP and subproblem formulations for the PRP and developed a branch-and-price procedure. Their subproblem was the delivery schedule generator, which decomposed into a VRP for each period. At each branching node, starting from the initial solution, variables in the RMP were fixed and column generation was performed to add variables to the RMP and solve it again until an optimal solution was found. The branching process went until a new optimal solution was attained to the original problem. A new branching rule for dealing with an unstudied form of master problem degeneracy, reducing the effects of symmetry, obtained feasible solutions by combining a rounding heuristic and TS.

Ruokokoski et al. (2010) explored different lot-sizing reformulations for the PRP with uncapacitated production and a single non-capacitated vehicle to determine their efficiency. The stronger LSP reformulations relied on facility location and shortest path reformulations, similar to Boudia et al. (2007), where the vehicle index was dropped and the subtour elimination constraints were replaced. The authors also strengthened these formulations with two families of valid inequalities, proposing a new heuristic separation algorithm for the generalized comb inequalities and adapted a heuristic algorithm from the literature to find high-quality integer feasible solutions.

Adulyasak et al. (2014a) introduced multi-vehicle PRP formulations with and without vehicle-index to solve the problem under both the maximum level (ML) and order-up-to-level (OU) (COELHO et al., 2013) inventory replenishment policies. An initial solution and upper bound was provided by an ALNS procedure (ADULYASAK et al., 2014b). The formulation was strengthened the inventory constraints, breaking symmetry, generalized fractional subtour elimination, and others, while the model was solved under a branch-and-cut (B&C) algorithm. The vehicle-indexed formulation outperformed the non-vehicle-index one, offering better lower bounds at root node as overall gaps. They also performed tests using multi-core processors, which reduced the elapsed time but increased the number of B&B nodes.

Adulyasak et al. (2015a) were the first to solve the PRP under demand uncertainty. They chose the Benders (1962) decomposition in a B&C scheme. The problem received two formulations, called two-stage and multi-stage decision processes. The difference between them was that for the two-stage problem, the demands for the entire planning horizon become known once the first-stage decisions were done, while for the multi-stage problem, the demands for a given stage become known only after the decisions for all previous stages have been made. After each formulation was decomposed using the Benders' method, each subproblem was solved at the B&B nodes generating cuts for the master problem. The computational results show that, for both the two-stage and the multi-stage problems, the B&C was efficient in handling small instances with a small number of scenarios while the Benders-based B&C with the enhancements outperformed the B&C on larger instances and particularly on instances with a large number of scenarios.

Qiu et al. (2018a) proposed a formulation for the PRP considering time-windows. They strengthened the formulation replacing the MTZ constraints (MILLER; TUCKER; ZEMLIN, 1960) and introduced four more families of cuts. This formulation was solved using a B&C algorithm which solved the model at each node disregarding the generalized capacity and path constraints. A heuristic combined the ideas of large neighborhood search, local search, tabu search, and simulated annealing, and accelerated the solution process. After the tests, results showed an increase of lower bounds by the root node as the reduction of CPU times.

Darvish et al. (2016) worked together a company that produced and sold furniture. They decided to approach only online customers, as other deliveries were made for stores. In this way, only the factories had an inventory which also allowed the exchange of products among themselves. The company had an outsourced partner that make the deliveries, then the distribution was treated with a proportional cost to the delivered quantity per period. With real data, they were able to create different situations to analyses the size of the time windows, costs, and capacities of production and storage. Although they did not find optimal solutions using a B&B algorithm, the gap was less than 3% and much better than the methods used by the company. The total savings reach between 4% and 14% of total costs.

Senoussi et al. (2016) approached the problem using the concept of supply chain management, whereas there was only one supplier, with limited production capacity and infinity inventory limit. The customers must have their demands satisfied periodically. They unconsidered distance between retailers and used a major fixed cost each vehicle to travel to some customers' clusters. Two formulations were proposed, one based on basic lot-sizing, and the other relied on the idea of echelon stocks. Both were strengthened with six valid inequalities and solved using a commercial solver. Despite the good results, especially after the addition of cuts, the approaches would not be suitable for very large instances of the problem.

There is also a concern about the impact of supply chain activities over the environment, like carbon footprints and green/reverse logistics. A PRP approach with carbon and cap trade was proposed by Qiu et al. (2017) which also accepted partial delivery with lost-sales. They assumed emissions generated by both echelons, production-inventory, and routing, rewriting the objective function with a linear approximation that considered the proportional carbon price. They solved the problem using a branch-and-price (B&P) algorithm based on Dantzig and Wolfe (1960) decomposition, while the branching was realized over the variables of visit, setup, and connection. The pricing was solved through subproblem using an elementary shortest path labeling algorithm. Their approach proved capable to reduce simultaneously emissions levels of carbon dioxide and operational costs.

Fang et al. (2017) investigated a PRP with a reverse logistic approach under the reduction of carbon emissions also considering simultaneous pickup and delivery. They proposed an arc-flow-based model that was solved with a guided branch-and-cut (B&C) method. The authors adopted the same approximation for a carbon price as Qiu et al. (2017). During the execution of the guided B&C, three families of valid inequalities were added to the root node of the search tree. The first strengthens the size of delivery and pickup lots, the second the routing constraints, and the last the inventory quantities. The approach reached most of their bounds under 1%.

Qiu et al. (2018) studied a PRP under reverse logistics and remanufacturing conditions. The solution method was closely related to Fang et al. (2017), with a guided B&C. The generated valid inequalities were based on the residual delivery requests by customers and as a consequence the number of visits, at each period. Their main results show that high pickup requests turn the algorithm more effective and the optimal solutions were insensitive to the remanufacturing depot location.

Darvish, Archetti and Coelho (2018) worried about the carbon and other gas emissions, evaluated the IRP and PRP under the minimization of three different objective functions: total costs, routing costs, or emissions. A B&C algorithm was used to solve the total cost and distance minimization objectives. For the minimization of emissions, and enhanced exact algorithm called Variable MIP Neighborhood Descent (VMND) (LARRAIN; COELHO; CATALDO, 2017) was used, with the B&C working within. To guide these evaluations some business key performance indicators were adopted. One of the main results was that to perform deliveries with a vehicle lighter was an important factor in reducing emissions.

2.3.3 Lower bounds approaches

The study from Fumero and Vercellis (1999) proposed a Lagrangian relaxation for a variant of the PRP, where unit transportation costs were assumed. The idea was to decompose the LSP and a multi-commodity flow VRP into four subproblems (production, inventory, distribution, and routing). The authors dualized the inventory and vehicle capacity constraints, then they could solve the product and inventory subproblems by inspection, the distribution using a solver, and the routing with a heuristic.

Solyalı and Süral (2009) developed a Lagrangian relaxation approach to obtain lower bounds based on the multi-commodity flow to solve the PRP with the order-up-to level policy. To compute the lower bounds, the authors used a set of subproblems. The first was called ORDER and involved only lot-sizing and setup variables. To make it tighter two sets of inequalities were proposed, one for ensuring that the amount of product ordered do not exceed the maximum requirement of customers less the initial inventory, and the other stipulates that the total amount of product to be ordered to supplier up to period t should be at least the total of minimal requirements of retailers less the initial amount at the supplier. The second subproblem was the SINV and it consisted of only inventory variables of the supplier with non-negativity constraints. The third was DIST and involved routing variables and periods. Five sets of inequalities were added to the subproblem involving strengthened the routing decisions. The fourth and last subproblem was RET that contained variables concerning delivered amounts to retailers as the inventory levels of retailers. It decomposed into N separate single retailer problems which could be solved by the shortest path problem with resource constraints (FEILLET et al., 2004). However, the lower bounds obtained by this approach were weak compared to the case where the unit transportation costs are used as in Fumero and Vercellis (1999). On the instances with 8 customers and 5 periods, the lower bounds produced by the Lagrangian relaxation had an average deviation of 33.16% from the optimal value.

2.4 Summary of the papers

Tables 2.1 and 2.2 summarize the information about the reviewed papers. As can be seen, the problem has received more attention in the past twelve years (2009-2020) that can be perceived by the presence of 32 of the 47 works belonging to this interval. We adopted classifications about production, inventory, distribution, and routing characteristics. Production may have single or multiple plants and products, as some capacity to the size of the lot. Inventory may also be limited and works with maximum level or order-up-to policies (COELHO et al., 2013). Distribution considers if the delivered loads can be realized by different vehicles in a given time-period t and if the demand can be back-ordered to a future period. Routing shows if the fleet is homogeneous or heterogeneous concerning the capacity and if its number of vehicles is unlimited, limited, single, or multiple (DÍAZ-MADROÑERO et al., 2015). Solution distinguishes if the adopted approach was exact, heuristic or metaheuristic, or hybrid that blends exact and heuristics in someway. The heuristics and metaheuristics approaches have a major role to solve the PRP as 35 of 48 works reviewed use them or combine with some exact method. When Table 2.1 and 2.2 are observed, it is clear that there is a lack of works considering the idea of back-ordering deliveries and routing characteristics as a heterogeneous fleet. Also, hybrids methods that take advantage of the combinations of mathematical procedures and heuristics have not been explored. Our work follows this direction proposing a formulation and algorithms to contribute to filling these gaps. As we far know, four reviews about the PRP were done by Reimann et al. (2014),Adulyasak et al. (2015b),Díaz-Madroñero et al. (2015) and Mostafa and Eltawil (2015), where more details can be found about the papers published.

AUTHORS	CHARACTERISTICS											
	PRODUCTION			*INVENTORY*		*DISTRIBUTION*		*ROUTING*			*SOLUTION*	
	Plants	Product	Capacity	Policy	Capacity	Partial	Back-ordering	Vehicles	Fleet	Capacity	Type	Approach
Chandra (1993)	S	M	x	ML		yes	no	U	HO	x	HR	Decomposition
Chandra and Fisher (1994)	S	M	x	ML		yes	no	U	HO	x	HR	Decomposition
Metters (1996)	S	S	x	-		no	no	U	HE	x	EX	Mixed integer programming (MIP)
Fumero and Vercellis (1999)	S	M	x	ML		yes	no	L	HO	x	HR/LB	Lagrangian Relaxation
Van Buer et al. (1999)	S	M		-		yes	no	M	HO	x	HR	Tabu Search/Simulated Annealing
Jolayemi and Olorunniwo (2004)	M	M	x	-	x	no	no	-	O	-	EX	LINDO®
Park (2005)	M	M	x	ML	x	yes	no	U	HO	x	HR	Decomposition
Boudia et al. (2005)	S	S	x	ML	x	no	no	L	HO	x	HR	Decomposition
Bertazzi et al. (2005)	S	S		ML	x	no	no	M	HO	x	HR	Decomposition
Lei et al. (2006)	M	S	x	ML	x	yes	no	L	HE	x	HR	Decomposition
Boudia et al. (2007)	S	S	x	ML	x	no	no	L	HO	x	HR	GRASP
Boudia et al. (2008)	S	S	x	ML	x	no	no	L	HO	x	HR	Decomposition
Boudia and Prins (2009)	S	S	x	ML	x	no	no	L	HO	x	HR	Memetic heuristic
Bard and Nananukul (2009a)	S	S	x	ML	x	no	no	L	HO	x	HR/LB	Branch-and-price
Bard and Nananukul (2009b)	S	S	x	ML	x	no	no	L	HO	x	HR	Tabu Search
Chen et al. (2009)	S	M				no	no	M	HO	x	HR	Decomposition
Solyalı and Süral (2009)	S	S		OU	x	no	no	L	HO	x	HR/LB	Lagrangian Relaxation
Bard and Nananukul (2010)	S	S	x	ML	x	no	no	L	HO	x	HR/LB	Branch-and-price/Heuristic
Ruokokoski et al. (2010)	S	S		ML		no	no	S	HO		EX	Branch-and-cut
Shiguemoto and Armentano (2010)	S	M	x	ML		yes	no	M	HO	x	HR	Tabu Search
Archetti et al. (2011)	S	S		ML/OU	x	no	no	S	HO	x	HY	Branch-and-cut/Heuristic
Armentano et al. (2011)	S	M	x	ML	x	no	no	L	HO	x	HR	Tabu Search
Calvete et al. (2011)	M	S	x	-	-	no	no	M	HO	x	HR	Ant Colony
Melo and Wolsey (2012)	S	S		ML	x	no	no	M	HO		HR	MIP heuristic
Piewthongngam et al. (2013)	M	M	x	ML	x	no	yes	M	HE	x	HR	Decomposition
Absi et al. (2014)	S	S		ML	x	no	no	M	HO	x	HR	Iterative MIP Heuristic

(S) single, (M) multiple, (ML) maximum level, (OU) order-up-to, (HO) homogeneous, (HE) heterogeneous, (U/L) (un)limited, (O) outsourced, (EX) exact, (HR) heuristic or metaheuristic, (LB) lower bounds, (HY) hybrid

Table 2.1 – Summary of the reviewed papers - Part 1.

AUTHORS	CHARACTERISTICS											
	PRODUCTION			*INVENTORY*		*DISTRIBUTION*		*ROUTING*			*SOLUTION*	
	Plants	Product	Capacity	Policy	Capacity	Partial	Back-ordering	Vehicles	Fleet	Capacity	Type	Approach
Adulyasak et al. (2014a)	S	S	x	ML/OU	x	no	no	M	HO	x	HY	Branch-and-cut/Adaptive large neighborhood search
Adulyasak et al. (2014b)	S	S	x	ML	x	no	no	M	HO	x	HR	Adaptive large neighborhood search
Adulyasak et al. (2015a)	S	S	x	ML/OU	x	no	no	M	HO	x	HY	Benders decomposition/Branch-and-cut
Belo-Filho et al. (2015)	M	M	x	-	x	no	no	M	HO	x	HR	Adaptive large neighborhood search
Hein and Almeder (2016)	M	M		-		no	no	M	HO	x	EX	Decomposition
Kumar et al. (2016)	S	S	x	ML	x	no	no	M	HO	x	HR	SLPSO/NSGA-II
Senoussi et al. (2016)	S	S	x	OU	x	no	no	L	HO	x	EX	CPLEX®
Brahimi and Aouam (2016)	S	M	x	ML	x	no	yes	L	HO	x	HR	Relax-and-Fix
Darvish et al. (2016)	M	M		ML	x	no	no	-	O	-	EX	Branch-and-bound
Darvish et al. (2017)	S	S	x	ML	x	no	no	S	HO	x	EX	Branch-and-cut
Watanabe et al. (2017)	S	M	x	ML	x	no	no	L	HO	x	HY	Relax-and-Fix/Fix-and-Optmize
Fang et al. (2017)	S	S	x	ML	x	no	no	S	HO	x	EX	Branch-and-cut
Qiu et al. (2017)	S	S	x	ML	x	yes	no	M	HO	x	EX	Branch-and-price
Russell (2017)	S	S	x	ML	x	no	no	M	HO	x	HY	MIP-based metaheuristic
Solyalı and Süral (2017)	S	S	x	ML	x	no	no	M	HO	x	HR	Five-phase heuristic
Watanabe et al. (2017)	S	M	x	ML	x	no	no	L	HO	x	HY	Relax-and-Fix/Fix-and-Optmize
Miranda et al. (2018)	S	M	x	ML	x	no	no	M	HE	x	HY	Decomposition
Qiu et al. (2018)	M	S	x	ML	x	no	no	M	HO	x	EX	Branch-and-cut
Qiu et al. (2018a)	S	S	x	ML	x	no	no	M	HO	x	EX	Branch-and-cut
Qiu et al. (2018b)	S	S	x	ML	x	no	no	M	HO	x	HY	Skewed general variable neighborhood search
Neves-Moreira et al. (2019)	S	M	x	ML	x	no	no	M	HE	x	HR	Fix-and-Optmize
This study *- Chapter 4*	*S*	*M*	*x*	*ML*	*x*	*no*	*yes*	*M*	*HE*	*x*	*HY*	*Iterated local search matheuristics*
This study *- Chapter 5*	*S*	*M*	*x*	*ML*	*x*	*no*	*yes*	*M*	*HE*	*x*	*HY/LB*	*Column generation*

(S) single, (M) multiple, (ML) maximum level, (OU) order-up-to, (HO) homogeneous, (HE) heterogeneous, (U/L) (un)limited, (O) outsourced, (EX) exact, (HR) heuristic or metaheuristic, (LB) lower bounds, (HY) hybrid

Table 2.2 – Summary of the reviewed papers - Part 2.

3 Formulations for a rich production-routing problem

"Fortuna favors the bold".

Publius Terentius Afer

Some production-routing variants have characteristics that are usually not considered together when modeled, e.g., a heterogeneous fleet, multiple products, or back-order. When considered separately, they result in problems that fall short from reality. Nonetheless, combining them all leads to a challenging and complex problem seldom approached in the literature, which is here addressed.

The problem consists of determining at the minimal total cost, a production and distribution plant that fulfills periodic demands of multiple products by different clients while controlling the inventory levels at the production plant and clients, observing the production capacity, allowing back-orders for the unfilled demands in a timely fashion, and assuming a heterogeneous capacitated fleet with a maximum riding time to deliver the products in each period of the planning horizon. The total cost is made up of production and distribution related costs. The former includes the production, setup, holding, and inventory costs, while the latter accounts for the vehicle fixed and routing costs. From this point forward, the proposed problem will be identified as a rich production-routing problem (RPRP).

The problem relies upon the following notations and definitions. Let $\mathcal{T} = \{1, \ldots, T\}$ be a set of finite and discrete planning horizon. Let $\overline{\mathcal{N}} = \{1, \ldots, n\}$ be the set of clients that are supplied by a plant with products of the set $\mathcal{P} = \{1, \ldots, P\}$. Each client $i \in \overline{\mathcal{N}}$ demands $d_{ik}^t \geq 0$ units of product $k \in \mathcal{P}$ in period $t \in \mathcal{T}$. Let also $\mathcal{N} = \{0\} \cup \overline{\mathcal{N}}$ be the set of nodes with the plant labeled 0, and its copy $n + 1$.

Every time period $t \in \mathcal{T}$ the plant manufactures product $k \in \mathcal{P}$ while respecting production capacity C^k, it incurs in setup l^k and unitary u^k costs. A product $k \in \mathcal{P}$ can be stored at the plant or clients up to the limit of $U_i^k, \forall i \in \mathcal{N}$ units but incurring in an unitary holding cost h_{ik}^t for each period $t \in \mathcal{T}$. Every time the demand of client $i \in \overline{\mathcal{N}}$ for product $k \in \mathcal{P}$ can not be filled in a timely fashion in a time period $t \in \mathcal{T}$, the unfilled quantity can be back-ordered but at the unitary cost B_i^k.

A limited heterogeneous fleet, given by setting $\mathcal{V} = \{1, \ldots, V\}$ of vehicles, delivers the products by means of routes that start at the plant (0) and end in its copy $(n + 1)$ at the end of each time period $t \in \mathcal{T}$. Every vehicle $v \in \mathcal{V}$ that starts a route in a time period $t \in \mathcal{T}$ incurs in an activation cost e^v, and a routing cost associated with the used arcs of the directed graph

$\mathcal{G} = (\mathcal{N}_0, \mathcal{A})$ in which $\mathcal{N}_0 = \mathcal{N} \cup \{n+1\}$ and $\mathcal{A} = \{(i,j) : i,j \in \mathcal{N}_0, i \neq j\}$. Every arc $(i,j) \in \mathcal{A}$ has a routing cost $c_{ij} > 0$ and a traversing time $a_{ij} > 0$. To unload the products from the vehicles at the clients it takes a total of s_i units of service time.

To easy the reading, Table 3.1 lists the parameters of the problem with parameters $\overline{M}_k^t$ and $\widetilde{M}_{ik}^{vt}$ representing the maximum quantities to manufacture and deliver of the product $k \in \mathcal{P}$ to client $i \in \overline{\mathcal{N}}$ using vehicle $v \in \mathcal{V}$ in period $t \in \mathcal{T}$, respectively. Here, we propose two different formulations for the problem.

Table 3.1 – Problem parameters.

Notation	Meaning
C^k, l^k, u^k	The production capacity, setup and unitary costs of product $k \in \mathcal{P}$
U_i^k, h_i^k, B_i^k	The inventory capacity, holding and back-order costs of product $k \in \mathcal{P}$ for client $i \in \overline{\mathcal{N}}$
e^v, Q^v	The cost and capacity of each vehicle $v \in \mathcal{V}$
c_{ij}, a_{ij}	The connection costs and traveling times of each arc $(i,j) \in \mathcal{A}$
H, s_i	The maximum riding time and service times of each customer $i \in \overline{\mathcal{N}}$
d_{ik}^t	The periodic demand of each client $i \in \overline{\mathcal{N}}$ of product $k \in \mathcal{P}$ in period $t \in \mathcal{T}$
	$\overline{M}_k^t = \min\{C_k, \sum_{i \in \overline{\mathcal{N}}} \sum_{e=t}^{T} d_{ik}^e\}, \forall t \in \mathcal{T}, k \in \mathcal{P}$
	$\widetilde{M}_{ik}^{vt} = \min\{U_i^k, Q^v, \sum_{e=t}^{T} d_{ik}^e\}, \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T}$

3.1 A vehicle-indexed formulation

A vehicle-indexed formulation is here proposed. This model extends the one of Section 2.2.2 to our RPRP. It uses the following decision variables. The production variables are $y_k^t \in \{0,1\}$ is equal to one if a setup is realized to manufacture $p_k^t \geq 0$ units of product $k \in \mathcal{P}$ in period $t \in \mathcal{T}$, zero, otherwise. Variables $I_{ik}^t \geq 0$ indicates the inventory level for product $k \in \mathcal{P}$ at node $i \in \mathcal{N}$ in period $t \in \mathcal{T}$. Variables $b_{ik}^t \geq 0$ represents the quantity of the unfilled demand for product $k \in \mathcal{P}$ at node $i \in \overline{\mathcal{N}}$ in period $t \in \mathcal{T}$. The aforementioned are production related variables. The routing counterpart variables are: variable $g^{vt} \in \{0,1\}$ is equal to one if vehicle $v \in \mathcal{V}$ is set to serve the clients' demands of period $t \in \mathcal{T}$, zero, otherwise. Variable $z_i^{vt} \in \{0,1\}$ is equal to one if client $i \in \mathcal{N}$ is visited by vehicle $v \in \mathcal{V}$ to deliver $q_{ik}^{vt} \geq 0$ units of product $k \in \mathcal{P}$ in period $t \in \mathcal{T}$, zero, otherwise. Finally, variable $x_{ij}^{vt} \in \{0,1\}$ is equal to one if arc $(i,j) \in \mathcal{A}$ is used by vehicle $v \in \mathcal{V}$ in period $t \in \mathcal{T}$, zero, otherwise, while variable $w_{ij}^{vt} \geq 0$ represents the arrival time at node $j \in \mathcal{N}$, of vehicle $v \in \mathcal{V}$ after traversing arc $(i,j) \in \mathcal{A}$ in period $t \in \mathcal{T}$. To facilitate the representation, Table 3.2 lists the decision variables.

Table 3.2 – Decision variables of the vehicle-indexed formulation.

Notation	**Meaning**
$b^t_{ik} \geq 0$	The quantity back-ordered at customer $i \in \overline{\mathcal{N}}$ of product $k \in \mathcal{P}$ at period $t \in \mathcal{T}$
$I^t_{ik} \geq 0$	The inventory level at node $i \in \mathcal{N}$ of product $k \in \mathcal{P}$ at final of period $t \in \mathcal{T}$
$p^t_k \geq 0$	The quantity of product $k \in \mathcal{P}$ manufactured at period $t \in \mathcal{T}$
$q^{vt}_{ik} \geq 0$	The quantity of product $k \in \mathcal{P}$ delivered to customer $i \in \overline{\mathcal{N}}$ at period $t \in \mathcal{T}$ using vehicle $v \in \mathcal{V}$
$w^{vt}_{ij} \geq 0$	The arrival time at node $j \in \mathcal{N}_0$ after vehicle $v \in \mathcal{V}$ traverses arc $(i,j) \in \mathcal{A}$ at period $t \in \mathcal{T}$
$g^{vt} \in \{0,1\}$	It is equal to one if vehicle $v \in \mathcal{V}$ perform a route at period $t \in \mathcal{T}$, zero, otherwise
$x^{vt}_{ij} \in \{0,1\}$	It is equal to one if arc $(i,j) \in \mathcal{A}$ is used by vehicle $v \in \mathcal{V}$ at period $t \in \mathcal{T}$, zero, otherwise
$y^t_k \in \{0,1\}$	It is equal to one if the product $k \in \mathcal{P}$ is manufactured at period $t \in \mathcal{T}$, zero, otherwise
$z^{vt}_i \in \{0,1\}$	It is equal to one if node $i \in \mathcal{N}$ is visited by vehicle vehicle $v \in \mathcal{V}$ at period $t \in \mathcal{T}$, 0 otherwise

$$\min \sum_{t\in\mathcal{T}} \left\{ \sum_{k\in\mathcal{P}} \left[l^k y^t_k + u^k p^t_k + \sum_{i\in\mathcal{N}} h^k_i I^t_{ik} + \sum_{i\in\overline{\mathcal{N}}} B^k_i b^t_{ik} \right] + \sum_{v\in\mathcal{V}} \left[e^v g^{vt} + \sum_{(i,j)\in\mathcal{A}} c_{ij} x^{vt}_{ij} \right] \right\} \tag{3.1}$$

s.t.:

$$p^t_k \leq \overline{M}^t_k y^t_k, \qquad \forall k \in \mathcal{P}, t \in \mathcal{T} \tag{3.2}$$

$$I^t_{0k} = I^{t-1}_{0k} + p^t_k - \sum_{i\in\overline{\mathcal{N}}} \sum_{v\in\mathcal{V}} q^{vt}_{ik}, \qquad \forall k \in \mathcal{P}, t \in \mathcal{T} \tag{3.3}$$

$$I^t_{ik} = I^{t-1}_{ik} + \sum_{v\in\mathcal{V}} q^{vt}_{ik} - d^t_{ik} + b^t_{ik} - b^{t-1}_{ik}, \qquad \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, t \in \mathcal{T} \tag{3.4}$$

$$I^t_{ik} \leq U^k_i, \qquad \forall i \in \mathcal{N}, k \in \mathcal{P}, t \in \mathcal{T} \tag{3.5}$$

$$q^{vt}_{ik} \leq \widetilde{M}^{vt}_{ik} z^{vt}_i, \qquad \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.6}$$

$$\sum_{k\in\mathcal{P}} \sum_{i\in\overline{\mathcal{N}}} q^{vt}_{ik} \leq Q^v g^{vt}, \qquad \forall v \in \mathcal{V}, t \in \mathcal{T} \tag{3.7}$$

$$\sum_{v\in\mathcal{V}} z^{vt}_i \leq 1, \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.8}$$

$$z^{vt}_0 + z^{vt}_{n+1} = 2g^{vt}, \qquad \forall v \in \mathcal{V}, t \in \mathcal{T} \tag{3.9}$$

$$\sum_{i\in\overline{\mathcal{N}}} x^{vt}_{0i} = z^{vt}_0, \qquad \forall v \in \mathcal{V}, t \in \mathcal{T} \tag{3.10}$$

$$\sum_{i\in\overline{\mathcal{N}}} x^{vt}_{i,n+1} = z^{vt}_{n+1}, \qquad \forall v \in \mathcal{V}, t \in \mathcal{T} \tag{3.11}$$

$$\sum_{(i,j)\in\mathcal{A}} x^{vt}_{ij} = z^{vt}_i, \qquad \forall i \in \overline{\mathcal{N}}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.12}$$

$$\sum_{(j,i)\in\mathcal{A}} x^{vt}_{ji} = z^{vt}_i, \qquad \forall i \in \overline{\mathcal{N}}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.13}$$

$$w^{vt}_{0i} = a_{0i} x^{vt}_{0i}, \qquad \forall i \in \overline{\mathcal{N}}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.14}$$

$$\sum_{(i,j)\in\mathcal{A}} w^{vt}_{ij} - \sum_{(j,i)\in\mathcal{A}} w^{vt}_{ji} = \sum_{(i,j)\in\mathcal{A}} (s_i + a_{ij}) x^{vt}_{ij}, \qquad \forall i \in \overline{\mathcal{N}}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.15}$$

$$0 \leq w^{vt}_{ij} \leq H x^{vt}_{ij}, \qquad \forall (i,j) \in \mathcal{A}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.16}$$

$$p^t_k, I^t_{ik}, b^t_{ik}, q^{vt}_{ik} \geq 0, y^t_k \in \{0,1\}, \qquad \forall i \in \mathcal{N}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.17}$$

$$g^{vt}, z^{vt}_i \in \{0,1\} \qquad \forall i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.18}$$

$$x^{vt}_{ij} \in \{0,1\}, w^{vt}_{ij} \geq 0, \qquad \forall (i,j) \in \mathcal{A}, v \in \mathcal{V}, t \in \mathcal{T} \tag{3.19}$$

The objective function (3.1) minimizes the production, setup, holding, back-ordering, fleet, and routing costs, respectively. Constraints (3.2) limit the quantity to be produced for each item either the plant's production capacity for that item or its total remaining demand for the horizon planning. Constraints (3.3)-(3.5) control the inventory level of each product in each period. For each product $k \in \mathcal{P}$, both constraints (3.3) and (3.4) guarantee the balance of the inventory level at the end of period $t \in \mathcal{T}$ at the plant and customers, respectively. The maximum inventory levels at the factory and clients are stated by (3.5). The distribution and routing constraints are given by (3.6)-(3.16). The delivery lot size of each item $k \in \mathcal{P}$ for each client $i \in \overline{\mathcal{N}}$ in each period $t \in \mathcal{T}$ is limited by constraints (3.6). While constraints (3.7) assure that the vehicles' capacity is not exceeded. Constraints (3.8) limit the maximum number of times that a client $i \in \overline{\mathcal{N}}$ is visited per period $t \in \mathcal{T}$, while constraints (3.9) stipulates that if a vehicle $v \in \mathcal{V}$ leaves the plant node 0 it must return to its node copy $n + 1$. Constraints (3.10)-(3.13) are the degree related constraints for the plant, its node copy, and the clients visited by the same vehicle $v \in \mathcal{V}$ in a period $t \in \mathcal{T}$. Proposed by Bianchessi et al. (2018), constraints (3.14)-(3.16) bound the riding time of the vehicle while eliminating subtours. The vehicle's arrival time at the clients after leaving the plant is defined by (3.14), while constraints (3.15) balances the vehicles' time continuity. Constraints (3.16) bound the vehicles' maximum riding time. The remaining constraints (3.17)-(3.19) are variable domain related. It is here that the back-orders at the last period are equal to zero just as done by Brahimi and Aouam (2016). For simplicity, we call this formulation VINDX.

3.2 A two-commodity flow formulation

A two-commodity flow formulation based on Baldacci et al. (2009) is proposed. This formulation allows the omission of the vehicle index by rewriting the routing and distribution constraints for the heterogeneous fleet as a two-commodity flow pattern. As far as we know, it is the first time that such a modeling strategy is extended to the PRP.

To formulate using such flow pattern, we associate each vehicle $v \in \mathcal{V}$ with its own plant node copy resulting in set $\mathcal{V}^+ = \{n+1, n+2, \ldots, n+V\}$. We also introduce a new set of nodes $\mathcal{N}^+ = \{0\} \cup \overline{\mathcal{N}} \cup \mathcal{V}^+$, i.e., $\mathcal{N}^+ = \{0, 1, 2, \ldots, n, n+1, n+2, \ldots, n+V\}$ and arcs $\mathcal{A}^+ = \{(i, j) \in \mathcal{N}^+ \times \mathcal{N}^+ : i \neq j \wedge (i, j) \notin \{0\} \times \mathcal{V}^+ \cup \mathcal{V}^+ \times \{0\}\}$. Note that there no arcs connecting the plant and its copies, and vice-versa. We also define parameter $\overline{Q} = \max_{v \in \mathcal{V}}\{Q^v\}$ to represent the largest vehicle capacity, and redefine $\widetilde{M}_{ik}^t = \min\{\overline{Q}, U_i^k, \sum_{\tau=t}^{T} d_{ik}^\tau\}, \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, t \in \mathcal{T}$.

To design the routes without the vehicle indexes, we introduce a new set of flow variables $f_{ij}^t \geq 0$ to represent either the total amount of flow traversing arc $(i, j) \in \mathcal{A}^+$ in period $t \in \mathcal{T}$ or the residual capacity of the vehicle when using arc $(j, i) \in \mathcal{A}^+$. Here variables $q_{ik}^t, z_i^t, w_{ij}^t$ and x_{ij}^t still have the same meaning as before, but with the vehicles' indices suppressed. To avoid using the vehicle activation variables g^{vt}, we redefine the costs c_{ij} with the help of the function

$\pi(i) = i - n$ to return the vehicle associated to the plant node copy $i \in \mathcal{V}^+$. We assume that the vehicle leaves its node copy and moves towards the plant node 0. Hence, the costs for the new arcs $(i,j) \in \mathcal{A}^+$ with $i \in \mathcal{V}^+$ are set to $c_{ij} = e^{\pi(i)} + c_{0j}$ to properly compute the vehicles' activation cost. The remaining arcs retain their previous cost. Given the aforementioned, the two-commodity flow formulation, for short, can be written as follows.

$$\min \sum_{t \in \mathcal{T}} \left\{ \sum_{k \in \mathcal{P}} \left[l^k y_k^t + u^k p_k^t + \sum_{i \in \mathcal{N}} h_i^k I_{ik}^t + \sum_{i \in \overline{\mathcal{N}}} B_i^k b_{ik}^t \right] + \sum_{(i,j) \in \mathcal{A}^+} c_{ij} x_{ij}^t \right\} \tag{3.20}$$

s.t.:

$$p_k^t \le \overline{M}_k^t y_k^t, \qquad \forall k \in \mathcal{P}, t \in \mathcal{T} \tag{3.21}$$

$$I_{0k}^t = I_{0k}^{t-1} + p_k^t - \sum_{i \in \overline{\mathcal{N}}} q_{ik}^t, \qquad \forall k \in \mathcal{P}, t \in \mathcal{T} \tag{3.22}$$

$$I_{ik}^t = I_{ik}^{t-1} + q_{ik}^t - d_{ik}^t + b_{ik}^t - b_{ik}^{t-1}, \qquad \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, t \in \mathcal{T} \tag{3.23}$$

$$I_{ik}^t \le U_i^k, \qquad \forall i \in \mathcal{N}, k \in \mathcal{P}, t \in \mathcal{T} \tag{3.24}$$

$$q_{ik}^t \le \widetilde{M}_{ik}^t z_i^t, \qquad \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, t \in \mathcal{T} \tag{3.25}$$

$$\sum_{(v,j) \in \mathcal{A}^+} f_{vj}^t = \sum_{k \in \mathcal{P}} \sum_{i \in \overline{\mathcal{N}}} q_{ik}^t, \qquad \forall t \in \mathcal{T} \tag{3.26}$$

$$\sum_{i \in \overline{\mathcal{N}}} f_{i0}^t = 0, \qquad \forall t \in \mathcal{T} \tag{3.27}$$

$$\sum_{(i,j) \in \mathcal{A}^+} (f_{ji}^t - f_{ij}^t) = 2 \sum_{k \in \mathcal{P}} q_{ik}^t, \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.28}$$

$$\sum_{(i,j) \in \mathcal{A}^+} (f_{ji}^t + f_{ij}^t) = 2\overline{Q}, \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.29}$$

$$f_{ij}^t + f_{ji}^t = \overline{Q}(x_{ij}^t + x_{ji}^t), \qquad \forall (i,j) \in \mathcal{A}^+, t \in \mathcal{T} \tag{3.30}$$

$$f_{vj}^t \le Q^{\pi(v)} x_{vj}^t, \qquad \forall v \in \mathcal{V}^+, j \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.31}$$

$$\sum_{j \in \overline{\mathcal{N}}} x_{vj}^t \le 1, \qquad \forall v \in \mathcal{V}^+, t \in \mathcal{T} \tag{3.32}$$

$$\sum_{v \in \mathcal{V}^+} \sum_{j \in \overline{\mathcal{N}}} x_{vj}^t - \sum_{i \in \overline{\mathcal{N}}} x_{i0}^t = 0, \qquad \forall t \in \mathcal{T} \tag{3.33}$$

$$\sum_{(i,j) \in \mathcal{A}^+} x_{ij}^t = z_i^t, \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.34}$$

$$\sum_{(j,i) \in \mathcal{A}^+} x_{ji}^t = z_i^t, \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.35}$$

$$w_{vj}^t = a_{0j} x_{vj}^t, \qquad \forall v \in \mathcal{V}^+, j \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.36}$$

$$\sum_{(i,j) \in \mathcal{A}^+} w_{ij}^t - \sum_{(j,i) \in \mathcal{A}^+} w_{ji}^t = \sum_{(i,j) \in \mathcal{A}^+} (s_i + a_{ij}) x_{ij}^t, \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} \tag{3.37}$$

$$0 \le w_{ij}^t \le H x_{ij}^t, \qquad \forall (i,j) \in \mathcal{A}^+, t \in \mathcal{T} \tag{3.38}$$

$$p_k^t, I_{ik}^t, b_{ik}^t, q_{ik}^t \ge 0, z_i^t, y_k^t \in \{0,1\}, \qquad \forall i \in \mathcal{N}, k \in \mathcal{P}, t \in \mathcal{T} \tag{3.39}$$

$$x_{ij}^t \in \{0,1\}, w_{ij}^t \ge 0, \qquad \forall (i,j) \in \mathcal{A}^+, t \in \mathcal{T} \tag{3.40}$$

The objective function (3.20) is similar to (3.1) but now with the vehicle indices suppressed and the vehicles' activation cost embedded within the costs $c_{ij}, \forall (i,j) \in \mathcal{A}^+$. Constraints

(3.21)-(3.25) have the same meaning as constraints (3.2)-(3.6).

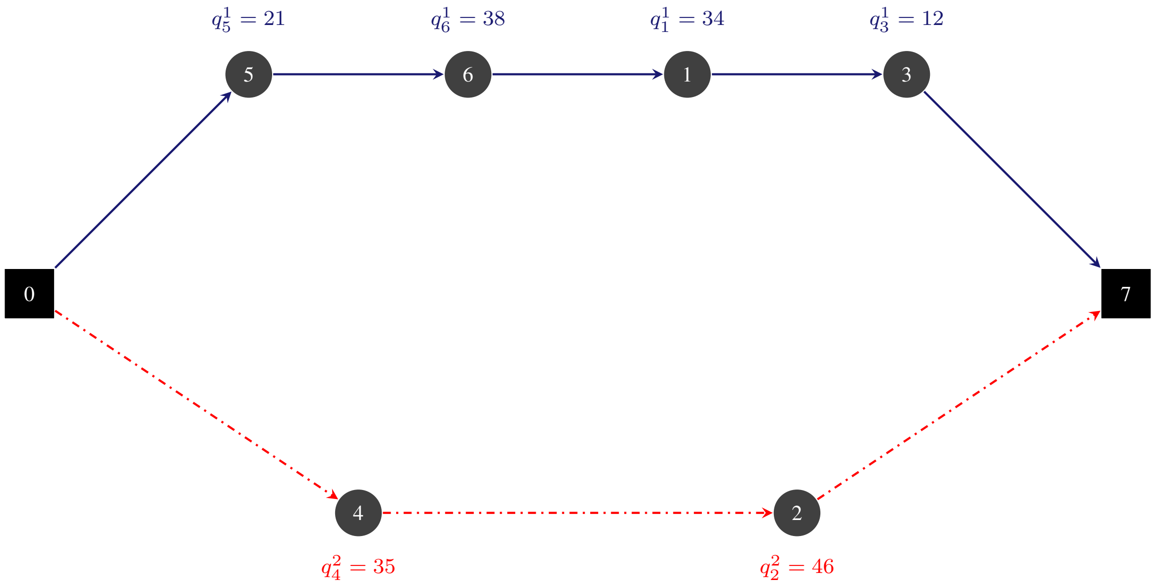
Constraints (3.26)-(3.31) define a feasible flow pattern. Constraints (3.26) ensure that the total amount to be delivered leaves the plant through the vehicles' associate plant node copy. Constraints (3.27) guarantee that the vehicles return to the plant node 0 empty. Constraints (3.28) and (3.29) state that the difference of inflows and outflows and their sum at a client node $i \in \overline{\mathcal{N}}$ in a time period $t \in \mathcal{T}$ is equal to twice the amount delivered and the largest vehicle capacity $\overline{Q}$, respectively. Constraints (3.30) define the relationships between flow and routing variables. They guarantee that the sum of the flow traversing an arc $(i, j) \in \mathcal{A}^+$ and the vehicles' residual capacity represented by the flow going in the opposite direction is equal to the vehicle's capacity if any of the forward or reverse arcs are connecting nodes i and $j, \forall i, j \in \mathcal{N}^+$, is used. The inequalities (3.31) allow flows traversing an arc $(i, j) \in \mathcal{A}^+$ only if that arc is activated in a route, and limits the maximum flow leaving the artificial node $i \in \mathcal{V}^+$ to the corresponding vehicle's capacity $Q^v, v = \pi(i) = i - n$. Constraints (3.32) limit the maximum number of vehicles leaving a plant node copy $v \in \mathcal{V}^+$. The remaining constraints (3.33)-(3.40) have the same meaning of the VINDX formulation but with the vehicles' indices suppressed. Formulation (3.20)-(3.40) is named henceforth as 2COMM.

To illustrate and clarify the routing modeling part of the aforementioned formulations, Figure 3.1 exemplifies two routes with vehicles' capacities $Q^1 = 105, Q^2 = 100$ to serve six clients with their respective delivery lots (q_i^v or q_i) for a period.

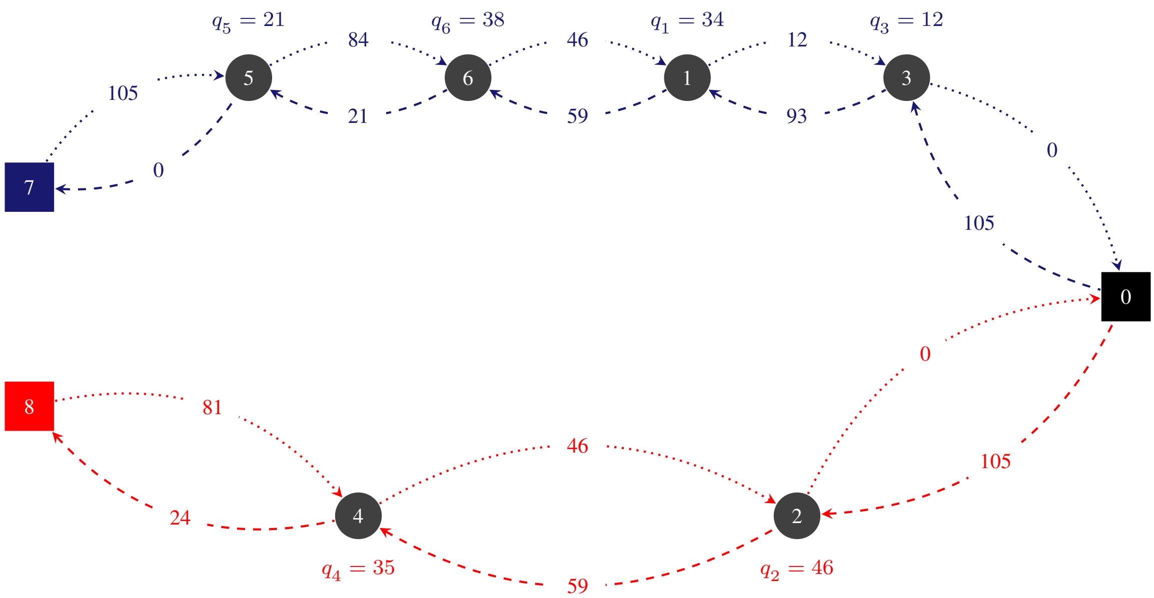
Figure 3.1a illustrates a vehicle-indexed solution in which labels 0 and 7 represent the plant node and its copy. For sake of representation, we show variables q_{ik}^{vt} as q_i^v to indicate the size of the delivery lots. In solid blue arrows, we can see the route of the first vehicle leaving the plant 0 fully loaded and visiting customers $5, 6, 1$ and 3, before returning to the plant copy 7. In dash-and-dotted red arrows, the second vehicle, with 81 units of load, visiting customers 4 and 2, before its return to 7.

Figure 3.1b shows a two-commodity flow solution. As previously defined, we have $\overline{Q} = \max\{Q^1, Q^2\} = Q^1 = 105$. Here, the two vehicles leave their respective plant copies, nodes 7 and 8, to visit the client nodes before returning to the plant node 0. The dotted blue and red arrows show the routes for both vehicles and their loads when traversing an arc. Reversed dashed arrows display the residual "artificial" capacity of the vehicles. Note that the sum of a forward arc and its reverse counterpart is exactly $\overline{Q}$. Furthermore, observe also that the difference between the inflow and outflows, and their sum are equal to twice the amount delivered, and twice the value of $\overline{Q}$, respectively, for each client.

Unfortunately, both formulations have poor linear programming relaxation bounds due to the big-M constraints such as (3.2), (3.6), (3.21) and (3.25), that, associated with the fact that both formulations scale very quickly with the number of clients, vehicles, products and periods, and that the problem is NP-Hard, limit their applicability to small instances only. Hence our motivation to devise hybrid methods capable of reaching good upper bounds in reasonable computational times, while a column generation approach computes lower bounds. Chapter 4 presents three top-down and one bottom-up ILS-based matheuristics. Chapter 5 introduces a column generation approach, that combines a compact formulation as a master problem with columns heuristically priced, besides the introduction of a pricing algorithm.



(a) Vehicle-indexed solution.



(b) Two-commodity flow solution.

Figure 3.1 – Example of routing solutions.

3.3 Generation of instances

As far as we know, no set of benchmark instances gathers simultaneously all the necessary information for our formulation and algorithms, then they were generated using as a basis the works of Coelho and Laporte (2013) and Brahimi and Aouam (2016). We developed 108 instances, varying four main aspects: (1) numbers of retailers ($n = 10c, c \in \{2, \ldots, 5\}$), (2) number of periods ($T \in \{5, 10, 15\}$), (3) number of products ($P \in \{3, 5, 7\}$) and (4) number of available vehicles ($V \in \{7, 9, 11\}$).

Parameters and costs were drawn from discrete uniform distributions within intervals shown in Table 3.3. Two different intervals were used to characterize low and high levels of demands. We assumed as in Brahimi and Aouam (2016) that the initial back-order and inventory at the clients were equal zero, and the clients' first-period demand, respectively. The codes to generate and read the instances are all available at *https://tinyurl.com/rprp-instances*, and in Appendix A. To ease the identification, the instances are labeled from 1 to 108, accordingly to Table B.1.

Table 3.3 – Parameters and costs values.

Values	**Meaning**
$x_i, y_i \in [0, 1000], \forall i \in \mathcal{N}$	The node coordinates
$d_{ik}^t \in [0, 25]$ or $d_{ik}^t \in [30, 55], \forall (i, k, t) \in \{\overline{\mathcal{N}}, \mathcal{P}, \mathcal{T}\}$	The clients' periodic demand
$I_{0k}^0 \in [100, 150], \forall k \in \mathcal{P}$	The plant initial inventory
$C_k = nA_k, A_k \in [50, 140], \forall k \in \mathcal{P}$	The production capacity
$u^k \in [2, 8], l^k = 10^4 u^k, \forall k \in \mathcal{P}$	The production setup and unitary costs
$U_i^k = \kappa F_k : F_k \in [140, 190], \kappa \in \{2, 3, 4\}, \forall (i, k) \in \{\mathcal{N}, \mathcal{P}\}$	The holding capacity
$h_0^k \in [1, 5], h_i^k \in [6, 10], \forall (i, k) \in \{\overline{\mathcal{N}}, \mathcal{P}\}$	The holding costs
$B_i^k = \gamma h_i^k, \gamma \in \{8, \ldots, 12\}, \forall (i, k) \in \{\overline{\mathcal{N}}, \mathcal{P}\}$	The back-order costs
$\overline{D} = \left\lceil \frac{\sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{P}} \sum_{i \in \overline{\mathcal{N}}} d_{ik}^t}{\lvert \mathcal{T} \rvert} \right\rceil$	The average demand per period
$e^v \in [500, 1000], Q^v \in \frac{2\overline{D}}{TV}[0.8, 1], \forall v \in \mathcal{V}$	The vehicles' cost and capacity
$c_{ij} = round(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}), \forall (i, j) \in \mathcal{A}$	The traveling costs
$a_{ij} = c_{ij}, \forall (i, j) \in \mathcal{A}$	The traveling times
$H = 6000$	The maximum riding time
$s_i = 50, \forall i \in \overline{\mathcal{N}}$	The client's service time

3.4 Bounds comparisons

Before we proceed to the devised methods, we analyze the bounds obtained for the generated instances when the VINDX and 2COMM formulations are solved by CPLEX 12.9. The experiments were performed on an Intel® Xeon™ CPU E5-2687W v3 @ 3.10GHz computer with 160 GB of RAM and running Ubuntu Linux 18.04. The formulations were coded in C++ using the CPLEX 12.9 Concert technology.

We first examined how the CPLEX performed on solving the 108 instances with a time-stopping criterion of 21600 seconds or six hours. Tables B.2-B.7 report the attained results. They show the obtained upper (UB) and lower (LB) bounds, the percentage gaps between the UB and the linear programming relaxation (LP GAP%), the optimality gaps between the UB and LB, the total number of nodes explored by the branch-and-bound tree, and the total number of nodes left until the reaching of the stopping criterion.

To compare the formulations, a benchmark profile (Figure 3.2) is performed with respect to the upper bounds (UB) achieved by CPLEX on solving the 108 instances but set to stop after six hours of execution. Performance profiles are graphic tools that aid to evaluate and compare the performance of a set of algorithms on solving a set of problems over a given performance measure, more details can be found in Dolan and Moré (2002). The 2COMM (solid and blue lines) formulation outperformed the VINDX (dashed and magenta lines) obtaining the best upper bounds for 98% of them after six hours of which, 12 were optimal. The VINDX formulation found valid upper bounds for only 87 of the instances. These solution values are up to 25% worse than those found by 2COMM. The smallest, average, and largest attained optimality gaps for the 2COMM (VINDX) formulation were 0.62% (2.49%), 15.11% (36.68%), and 41.25% (76.27%), respectively.

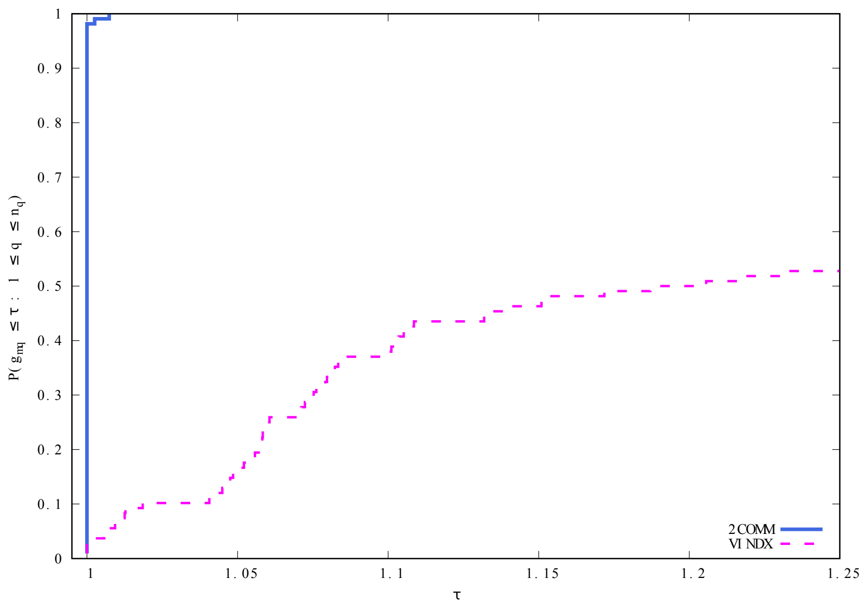


Figure 3.2 – Benchmark profile for the attained upper bounds.

The 2COMM formulation outperformed the VINDX one on reaching the best lower bounds (LB) for all instances. To allow a comparison of these lower limits, we analyze the linear programming gaps (LP GAP). Figures 3.3a-3.3d illustrates the attained LP GAP for all the 108 instances, considering from the smaller to the larger. The LP GAP is calculated for each instance considering the percentage of the difference between the upper bound (integer solution) and the lower bounds (relaxed solution) divided by the upper bound, as shown in Equation 3.41.

$$LPGAP(\%) = \frac{UB - LB}{UB}\% \tag{3.41}$$

Observing Figure 3.3, it is clear that 2COMM (blue boxes) provides tighter bounds than VINDX (magenta boxes), and comparing their average linear programming gaps, the 2COMM achieved

30,55% against 53,23% of the VINDX. Due to the superior performance of the 2COMM formulation, the outlined methods presented in Chapters 4-5 are compared to it.

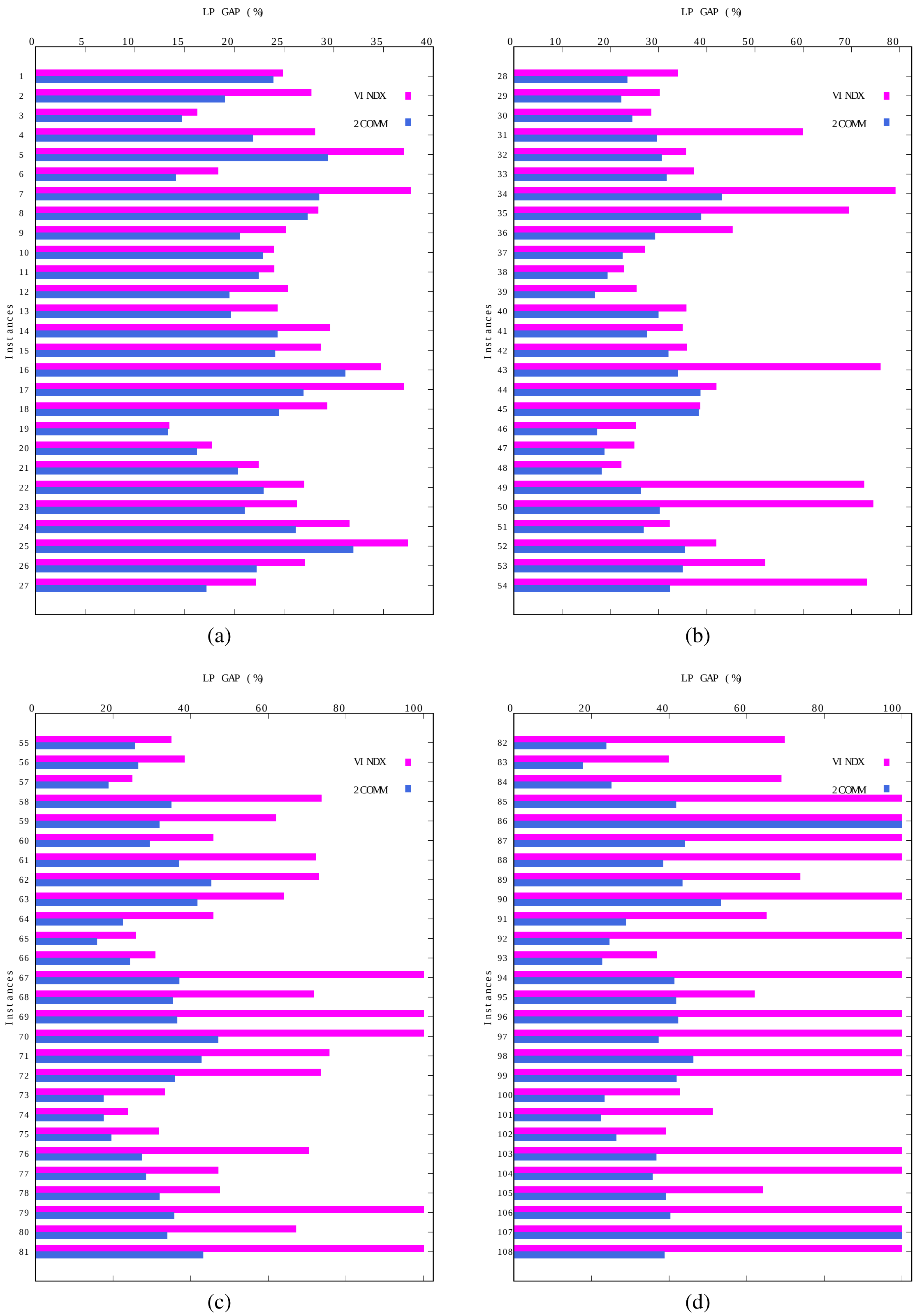


Figure 3.3 – Comparison of the linear programming gaps for the proposed formulations.

3.5 Impacts of back-ordering and a heterogeneous fleet

Back-order and heterogeneous fleet are not features commonly found in PRP studies. Given that a regular PRP already packs a large number of complicating assumptions, the presence of further certainty hinders not only the solution of a test instance but the development of algorithms as well. Nevertheless, their presence bridges the applicability gap of such solution frameworks.

In industrial applications, back-orders represent a risk of losing sales or even some patrons, but when judiciously allowed, they pose as to reduce the delivery or inventory costs. However, determining the quantities that can be back-ordered without negatively impacting costs or substantially affecting the current production plan is a non-trivial task. Moreover, delivery fleets can be owned or outsourced, and may not always be homogeneous.

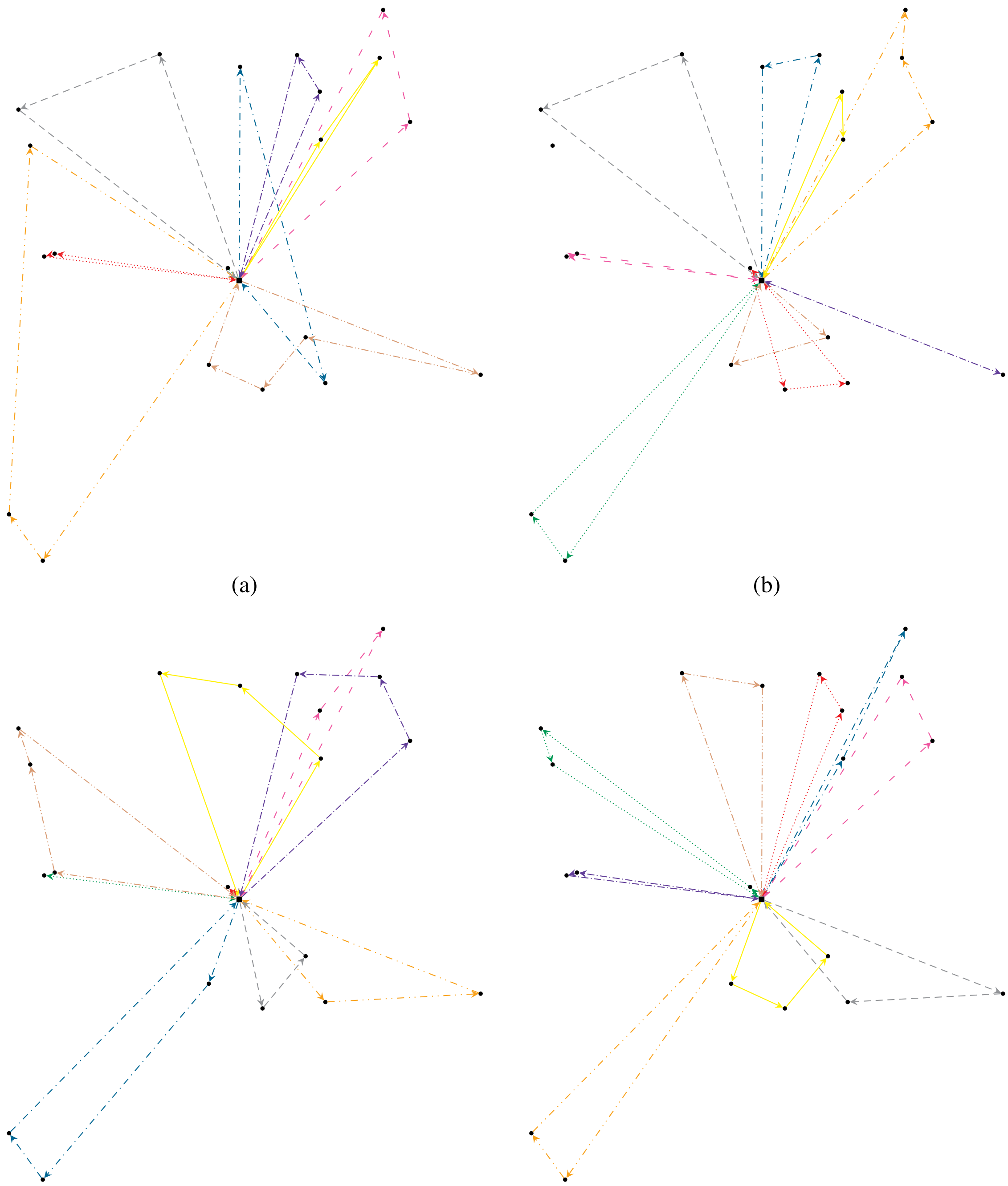
To illustrate how these aspects influence the obtained solutions and their costs, we solved instance P20n3p10t9v with 2COMM formulation considering four cases: *(a)* no back-ordering and homogeneous fleet, *(b)* no back-ordering and heterogeneous fleet, *(c)* back-ordering and homogeneous fleet, and the proposed RPRP considering *(d)* back-ordering and heterogeneous fleet. The data about the solution values can be found in the Section B.3, and some metrics are presented in the Table 3.4, further discussed.

Table 3.4 – Solutions' comparison among the cases *(a), (b)* and *(c)* with *(d)*.

	Cases		
Metrics	***(a)***	***(b)***	***(c)***
Holding costs	-25,57%	-22,75%	-2,43%
Back-ordering costs	100,00%	100,00%	100,00%
Traveling costs	-5,03%	-1,73%	-10,13%
Vehicle activation costs	18,01%	-1,95%	14,80%
Total cost	-2,14%	-2,75%	1,16%
Activated vehicles	-4,08%	0,00%	-8,16%

Table 3.4 presents the percentage difference among the aforementioned cases *(a),(b), and (c)* when compared with the proposed RPRP, represented by case *(d)*. There, negative values imply that a worse metric value was obtained when compared with case *(d)*, used as a reference. For the tested cases, all of them obtained the same setup and production costs, which does not interfere in the following analysis. Observing the back-ordering cost metric, it shows that the three alternatives do not incur in such kind of cost, even in the case *(c)*, when it is allowed. But, it can lead to the false idea of savings. The absence of back-ordering incurs higher traveling and holding costs, as well as the total cost, for all three alternative cases. When properly accounted for, as here modeled, it helps in achieving lower total costs.

Furthermore, the use of a homogeneous fleet can lead to longer traveled distances, and a larger number of vehicles required over the periods for cases *(a)* and *(c)*. These settings most definitely raise environmental concerns, as highlighted by Fang et al. (2017), Qiu et al. (2017), and Qiu et al. (2018). Another important aspect to observe is how the routing design can be influenced without the back-order or with a homogeneous fleet. In Figure 3.4, routing solutions for the same period are illustrated. These solutions were obtained with a stopping criterion of six hours. For each case, the square node represents the plant, while the circle ones the clients. Except for Figure 3.4d, Figures 3.4a-3.4c reach solutions with an elevated presence of longer and overlapping edges, suggesting that the absence of the proposed back-ordering and heterogeneous fleet features can produce solutions with higher quality.



(a) (b) (c) (d)

Figure 3.4 – Back-order and heterogeneous fleet impacts over the routing solutions.

4 Efficient matheuristics to solve a rich production-routing problem

"Divide and rule".

Julius Caesar

4.1 Introduction

Due to the difficulty to solve the RPRP with the proposed formulations, this chapter outlines four hybrid algorithms. Hybrid methods seek to blend the speed of heuristics with the determinism of the exact methods. Such a solution framework is normally called a matheuristic. Matheuristic approaches explore mathematical programming techniques in (meta)heuristic frameworks or on granting to mathematical programming approaches the cross-problem robustness and constrained-CPU-time effectiveness which characterize metaheuristics (CASERTA; VOSS, 2009).

They devised algorithms work in a two-level decision making within an iterated local search (ILS) framework (LOURENÇO; MARTIN; STÜTZLE, 2003). These levels are tactical, which is responsible for the production, inventory plans, and operational plans of the distribution and routing. Three of the proposed hybrid methods follow a top-down hierarchical approach, presented in Section 4.6.1. The fourth approach adopts a bottom-up decision approach, outlined in Section 4.6.2. They share a set of characteristics that are presented in Sections 4.2, 4.3, 4.4, and 4.5.

4.2 Building a solution

Algorithm 4.1 provides a new, or initial, solution based on the problem information and on the parameter $\boldsymbol{\Delta}$ to the top-down methods. It constructs a feasible plan for the tactical problem and then proposes delivery routes for each period. Throughout the execution of the algorithm, the parameter $\boldsymbol{\Delta} = [\Delta_{ik}^{vt}], \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T}$ is a vector of estimated costs, and plays an important role. The meaning of Δ_{ik}^{vt} is that each unit of the delivery lots q_{ik}^{vt} has an indirect cost over the production, inventory, and transportation decisions since it affects or is affected by them.

To find an initial solution for the problem, Algorithm 4.1 computes Δ_{ik}^{vt} as shown in Equation (4.1). The visitation cost Δ_i is calculated as in Qiu et al. (2018b). The other components consider how much it costs to transport a unit by each vehicle $v \in \mathcal{V}$ thorough the fraction ${}^{e^v}/_{Q^v}$, and if produced, whether this unit is stored either at the plant or the customer. The parameter

$\omega \in [1, \mathrm{maxWeight}]$, belonging to a discrete and uniform distribution, adds a random component to the problem to help to escape non-promising search areas.

$$\Delta_{ik}^{vt} = \omega\left(\Delta_i + \frac{e^v}{Q^v} + \min\{u^k + h_0^k, u^k + h_i^k\}\right), \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T} \tag{4.1}$$

Algorithm 4.1: Build and optimize

Data: Problem information, costs $\boldsymbol{\Delta}$

1 $\overline{q} \leftarrow$ BuildAndSolve($\mathrm{PI_v}(\boldsymbol{\Delta})$);
2 $\mathrm{VR_t} \leftarrow \emptyset$;
3 **for** $t \in \mathcal{T}$ **do**
4   vr($\overline{q}^t$) $\leftarrow$ BuildVRP($\overline{q}^t$);
5     $\mathrm{VR_t} \leftarrow \mathrm{VR_t} \cup \{$NRS(vr($\overline{q}^t$))$\}$;
6 **end**
7 $s \leftarrow \mathrm{PI_v} \cup \mathrm{VR_t}$;
8 **return** s

Algorithm 4.1 solves in line 1 the top tier problem $\mathrm{PI_v}$ with the indirect costs Δ_{ik}^{vt}, as written in (4.2)-(4.4) via CPLEX. The variables and constraints of formulation (4.2)-(4.4) have the same meaning as before, please see Table 3.2 for the meaning of the variables. Note that with the adoption of $\boldsymbol{\Delta}$ it is possible to solve $\mathrm{PI_v}$ even without the visitation (z) and vehicle activation (g) binary variables, since $\boldsymbol{\Delta}$ carries routing information. It eases the resolution by the solver.

$$\min \sum_{t\in\mathcal{T}} \left\{ \sum_{k\in\mathcal{P}} \left[l^k y_k^t + u^k p_k^t + \sum_{i\in\mathcal{N}} h_i^k I_{ik}^t + \sum_{i\in\overline{\mathcal{N}}} \left(B_i^k b_{ik}^t + \sum_{v\in\mathcal{V}} \Delta_{ik}^{vt} q_{ik}^{vt} \right) \right] \right\} \tag{4.2}$$

s.t.: (3.2) – (3.5)

$$\sum_{k\in\mathcal{P}} \sum_{i\in\overline{\mathcal{N}}} q_{ik}^{vt} \le Q^v, \forall v \in \mathcal{V}, \forall t \in \mathcal{T} \tag{4.3}$$

$$0 \le q_{ik}^{vt} \le \widetilde{M}_{ik}^{vt}, \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T} \tag{4.4}$$

Given the to-be-delivered loads $\overline{q}^t$ per product in each period t to the clients i, that are provided by $\mathrm{PI_v}$, they are concatenated into $\overline{q}_i^t = \sum_{k\in\mathcal{P}} \sum_{v\in\mathcal{V}} q_{ik}^{vt}$. Next, the $\mathrm{VR_t}$ problem is initialized (Alg. 4.1, line 2), it corresponds to the set of the independent routing solutions for the periods $t \in \mathcal{T}$.

For each period, the BuildVRP function generates feasible routes vr($\overline{q}^t$) (Alg. 4.1, line 4). It is done via randomly selected constructive methods chosen within the following: the Clarke-Wright (CLARKE; WRIGHT, 1964) savings heuristic (parallel or sequential, see Lysgaard (1997)); the sequential-lexicographic and load-ordered insertions. For all these methods, the vehicles with the largest capacities are filled first. The reason for using these four constructive methods is to allow diversification in the search space In line 5, the routing solution vr($\overline{q}^t$) is improved by the procedure neighborhood routing search (NRS), described in Section 4.3, and added the resulting routing solution added to $\mathrm{VR_t}$.

Thought the first constructive routing methods are well known in the literature, the last two are proposed by this study. The Sequential-lexicographic insertion starts from a node defined as 1, and tries to add it to a vehicle v, considering the feasibility of vehicle capacity and riding time. The procedure investigates all customers until no further client can be added to v. When a vehicle is full or reaches the maximum riding time of H, a new vehicle is activated. The procedure stops when all retailers are allocated. The load-ordered insertion algorithm takes the individual lots q_i of each node i and orders them from largest to smallest. The largest loads are then added iteratively to the largest vehicle available until no further additions can be done.

For instance, suppose that there are five retailers $\overline{\mathcal{N}} = \{\mathrm{i_1}, \dots, \mathrm{i_5}\}$ with delivery lots equal to $\mathrm{q_1} = 3, \mathrm{q_2} = 2, \mathrm{q_3} = 4, \mathrm{q_4} = 2, \mathrm{q_5} = 1$, respectively, and that there are two vehicles $\mathcal{V} = \{\mathrm{v_1}, \mathrm{v_2}\}$ with capacities equal to $\mathrm{Q^1} = 7, \mathrm{Q^2} = 5$ are available. The sequential-lexicographic solution would be vehicle $\mathrm{v_1}$ visiting retailers $\mathrm{i_1}, \mathrm{i_2}$ and $\mathrm{i_4}$, while vehicle $\mathrm{v_2}$ visits $\mathrm{i_3}$ and $\mathrm{i_5}$. Now, for the load-ordered insertion procedure, the attained solution would be vehicle $\mathrm{v_1}$ visiting retailers $\mathrm{i_3}$ and $\mathrm{i_1}$, while vehicle $\mathrm{v_2}$ visits $\mathrm{i_2}, \mathrm{i_4}$ and $\mathrm{i_5}$.

Algorithm 4.2 explains how the initial solution for the bottom-up method is obtained. It starts defining the objective function of the best solution as a large enough value (line 1). In order to achieve better results, the initial solution function generates different initial solutions (lines 2 to 9). A solution s is generated with the Algorithm 4.1, and if it is better than the current s^*, it is retained (line 4). Now, to allow different arrangements, the $\boldsymbol{\Delta}$ set of coefficients at the production-inventory problem $\mathrm{PI_v}$ are updated with the information of s^* using the Equation (4.8), at line 8. This process is repeated until the stop criterion is met, which for this case is running for up to twenty minutes.

Algorithm 4.2: Bottom-up initial solution

Data: Problem information, costs $\boldsymbol{\Delta}$
1 $f^* \leftarrow +\infty$;
2 **while** *stop criterion is not met* **do**
3     $s \leftarrow$ Build and optimize($\boldsymbol{\Delta}$); //Alg. 4.1
4   **if** $f^* > f(s)$ **then**
5         $s^* \leftarrow s$;
6         $f^* \leftarrow f(s)$;
7   **end**
8     $\boldsymbol{\Delta} \leftarrow \mathrm{Update}\boldsymbol{\Delta}(\mathrm{s}^*)$; //Sec. 4.5
9 **end**
10 **return** s^*;

4.3 Neighborhood routing searches

The procedure NRS showed in Algorithm 4.3 applies a set of routing local searches on a given solution. It is based on the random variable neighborhood descent (RVND) procedure (SOUZA et al., 2010; PENNA; SUBRAMANIAN; OCHI, 2013), which prevents method favoritism, and applies each routing local search as neighborhood structure. The combination of the aforementioned constructive routing heuristics with the following neighborhood routing local searches achieved good upper bounds with reduced computational effort and running times.

First, set RS of inter-route searches is created (line 1). While this set RS is not empty, i.e., while solution improvements are obtained (lines 2-10), a inter-route local search rs is randomly chosen from the set RS and applied on vr($\overline{q}^t$) (line 4). In the case of improvement, an intra-route search procedure is performed (line 6). Otherwise, local search rs is removed from the set rs (line 8).

Algorithm 4.3: Neighborhood routing search (NRS)

Data: vehicle routing solution vr($\overline{q}$)
1 $\mathrm{RS} \leftarrow$ inter-route local search list;
2 **while** $\mathrm{RS} \neq \emptyset$ **do**
3 $\quad \mathrm{rs} \leftarrow random(\mathrm{RS})$;
4 $\quad \mathrm{vr}'(\overline{\mathrm{q}}) \leftarrow \mathrm{InterRoute}(\mathrm{rs}, \mathrm{vr}(\overline{\mathrm{q}}))$;
5 $\quad$ **if** $f(\mathrm{vr}'(\overline{\mathrm{q}})) < f(\mathrm{vr}(\overline{\mathrm{q}}))$ **then**
6 $\quad\quad \mathrm{vr}(\overline{\mathrm{q}}) \leftarrow \mathrm{IntraRoute}(\mathrm{vr}'(\overline{\mathrm{q}}))$;
7 $\quad$ **else**
8 $\quad\quad \mathrm{RS} \leftarrow \mathrm{RS} \setminus \{\mathrm{rs}\}$
9 $\quad$ **end**
10 **end**
11 **return** $\mathrm{vr}(\overline{\mathrm{q}})$;

The adopted routing local searches may be separated into two groups, intra-route, and inter-routes (KYTÖJOKI et al., 2007). Every time an improvement is found by some inter-route method, a set of intra-route heuristics is applied (Alg. 4.3, line 6). They are organized in an RVND strategy but only employed on modified routes by the inter-route procedures. Intra-route local searches work reordering the visited nodes by each route, and we used five: the *1, 2* or *3-point* move, *2* or *Or-opt*, with *Or=3,4,5*. Groër et al. (2010) describe them as follows, the 1-point move relocates an existing node into a new position (Fig. 4.1a), the 2-point move swaps the position of two nodes (Fig. 4.1b), the 3-point move swaps the position of a pair of adjacent nodes with the position of a third node (Fig. 4.1c), the 2-opt move removes two edges from the solution and replaces them with two new edges (Fig. 4.1d), the Or-opt move removes a string of two, three, or four nodes and inserts the string into a new position (Fig. 4.1e).

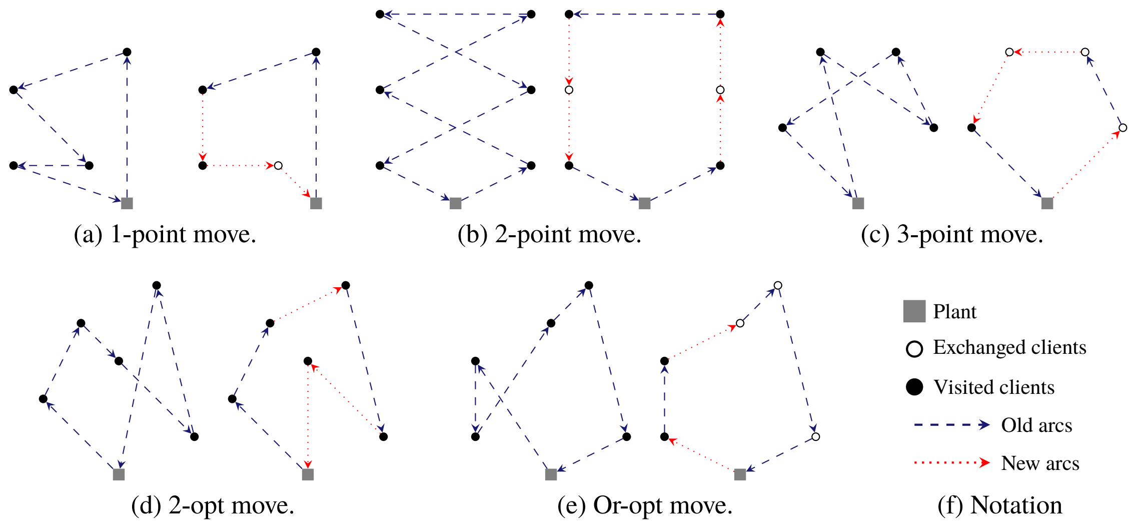
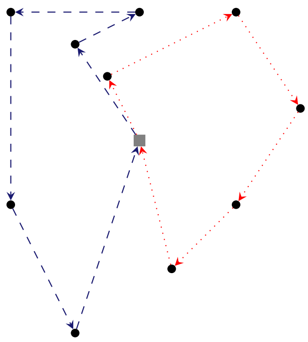
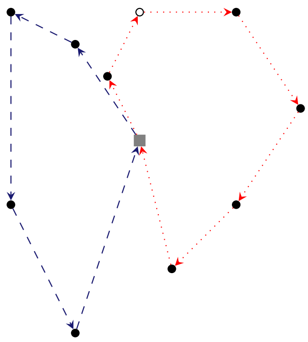
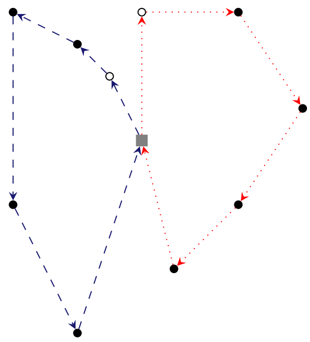


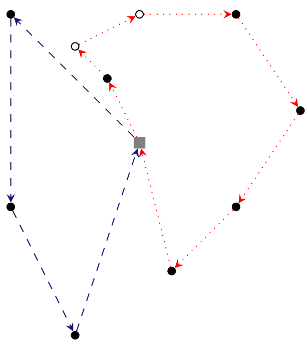
Figure 4.1 – Examples of intra-route local searches.

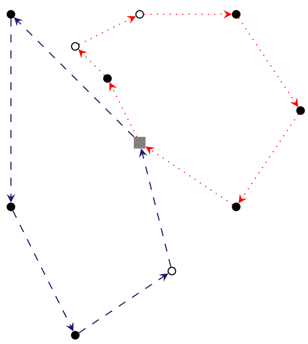
For the inter-route neighborhood structure, we adopted seven well-described moves by Penna, Subramanian and Ochi (2013). They are the Shift(1,0), Swap(1,1), Shift(2,0), Swap(2,1), Swap(2,2), Cross, and K-Shift (K=3,4,5) and they work selecting two different routes and exchanging or allocating customers between them. These moves are realized only if the new route arrangement is feasible and improves the routing cost. The moves are illustrated in Figure 4.2. Figure 4.2a shows an initial arrangement for the example, with route $r1$ in dashed and blue arrow and route $r2$ in dotted and red arrows, while the notation is presented in Figure 4.2i. In the Shift(1,0) move, a customer i is transferred from route $r1$ to route $r2$ (Figure 4.2b). Move Swap(1,1) picks a client i from route $r1$ and a client j from route $r2$ and exchange them (Figure 4.2c). In Shift(2,0) move, two adjacent customers $i - j$ from route $r1$ are transferred to route $r2$ (Figure 4.2d). The move Swap(2,1) exchanges two adjacent customers $i - j$ from route $r1$ by a customer k from route $r2$ (Figure 4.2e). Swap(2,2) swaps a pair $i - j$ of adjacent customers belonging to route $r1$ by a second pair of adjacent customers $k - l$ from route $r2$ (Figure 4.2f). The Cross move removes arcs connecting two adjacent $i - j$ clients belonging to route $r1$ and the one between clients $k - l$ that belongs to route $r2$. Next, arcs connecting i and l and k and j are inserted (Figure 4.2g). The move K-Shift removes a subset of consecutive customers K, in this example $K = 3$, which is transferred from route $r1$ to the end of route $r2$. It should be pointed out that the move is also applied if the second route is empty (Figure 4.2h).

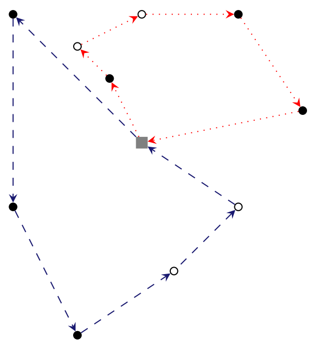

(a) Initial example.

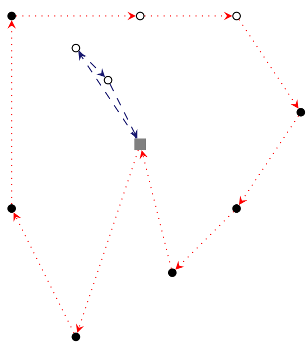

(b) Shift(1,0).

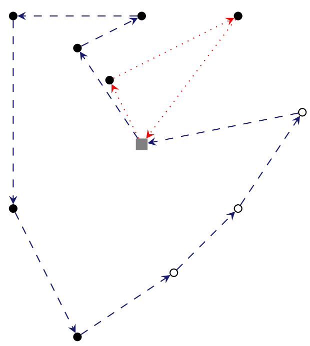

(c) Swap(1,1).

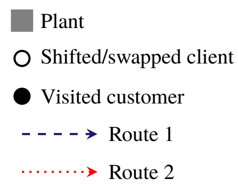

(d) Shift(2,0).


(e) Swap(2,1).


(f) Swap(2,2).


(g) Cross exchange.


(h) K-shift.



(i) Notation

Figure 4.2 – Examples of inter-route local searches.

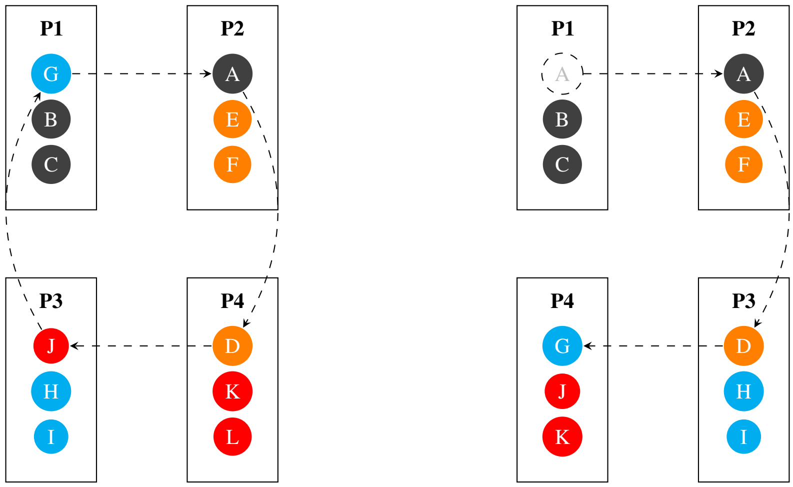
Aiming for better results, we also applied the very large-scale neighborhood search (VLNS)[1]proposed by Ahuja et al. (2000). It replaces both moves Shift(1,0) and Swap(1,1), with substantially more neighbors when compared to them. This occurs because VLNS contemplates simultaneously swap and shift moves not only between a pair of routes, as done by Shift(1,0) and Swap(1,1) but among all the sets of routes if the move is feasible. Its design is extremely efficient for the family of set partitioning problems, once the VLNS neighborhood structure is classified as a network flow-based improvement algorithm, once it builds an improvement graph that contains all feasible moves of retailers between routes.

The improvement graph for a neighborhood with multiple exchanges is defined over a

[1] The *VLNS* algorithm must not be confused with the Large Neighborhood Search (*LNS*) proposed by Shaw (1998). The *LNS* belongs to the *VLNS* class of heuristics (PISINGER; ROPKE, 2010).

feasible routing solution s for the problem, being represented by $\mathbf{G}(s)$. Let $r[i_j]$ be a route that contains the client i_j. The graph $\mathbf{G}(s)$ is a directed graph with $n+V+1$ nodes, where n is the number of clients belonging to the V routes and one more artificial node. Then, each node $i_1,\ldots,i_n$ corresponds to the customers, nodes $i_{n+1},\ldots,i_{n+m}$ to respective vehicles/routes and artificial node i_{n+m+1}. A directed arc $(l,k)\in\mathbf{G}(s)$ means that client i_l leaves its current route and is transferred to the route that contains item i_k, i.e., the route $r[i_k]$. Simultaneously, the client i_k leaves $r[i_k]$. To build $\mathbf{G}(s)$, all the pairs of elements $i_l, i_k \in s$ are considered. Then, an arc (l,k) is added to $\mathbf{G}(s)$ if and only if: *(i)* the clients i_l and i_k belong to different routes; *(ii)* the route $r[i_k]\setminus\{i_k\}\cup\{i_l\}$ is feasible. The cost $\bar{c}_{lk}$ of the arc (l,k) is defined by the difference of the cost of the modified route and its previous version, i.e., $\bar{c}(r[i_k]\setminus\{i_k\}\cup\{i_l\})-\bar{c}(r[i_k])$.

Let D be a directed cycle on the improvement graph $\mathbf{G}(s)$ if the elements which compose D belong to different partitions. A valid cycle can be defined as a directed cycle with a negative cost of $\mathbf{G}(s)$. Thus, a valid cycle corresponds to a cyclic or path exchange often leading to an improvement in the objective function of the problem with respect to all constraints. If no improvement is found, the search stops. To reach these improvements it is necessary to efficiently identify these valid cycles in $\mathbf{G}(s)$. It is done with a label-correcting algorithm, in this work the Bellman-Ford[2]algorithm is adopted, that finds the minimum path from a given (and artificial) source node to all nodes of the network.



(a) Cyclic exchange. (b) Path exchange.

Figure 4.3 – Very large-scale neighborhood exchanges.

These cyclic and path exchanges (Figure 4.3) are a generalization of the two-exchange neighborhood (THOMPSON; ORLIN, 1989). Given a subset of routes, the cyclic exchange removes one node from each route that composes this subset and exchange them. Figure 4.3 illustrates the cyclic and path exchanges, respectively. Shifting the nodes i_A, i_D, i_G and i_J, among their respective partitions P_1, P_2, P_4 and P_3, in the sequence $i_A\to i_D\to i_J\to i_G\to i_A$,

[2] The Bellman-Ford label-correcting algorithm correspond to the ones proposed in Bellman (1958) and Ford Jr. (1956).

characterizes a cycle exchange (Figure 4.3a). While moving the nodes i_A, i_D and i_G in the sequence $i_A \rightarrow i_D \rightarrow i_G$, among partitions P_1, P_2 and P_3, characterizes a path exchange (Figure 4.3b). The cardinality of the routes involved in the exchange is kept for the cyclic, but for the path, one route has its cardinality increased by one while others decreased. Further details about the VLNS method can be found in Ahuja et al. (1998), Ahuja et al. (2000), Ahuja (2017), and a real crew scheduling problem application in Silva and Reis (2014).

4.4 Perturbation operators

The local searches can get trapped into basins of attractions, leading to non-promising neighborhoods. To avoid that, a solution perturbation is applied, perturbation operators must be strong enough to allow the exploration of new regions or have their effort undone by a local search (LOURENÇO; MARTIN; STÜTZLE, 2019). This study considers these recommendations and outlines four algorithms, having different strategies of perturbation mechanisms. The main idea of these mechanisms is to modify either or both decision levels while verifying how these changes lead to better solutions. The proposed perturbation operators modify the tactical level through modifying the production plans, or the operational level changing the distribution and routing plans.

4.4.1 Production plan operator

For the tactical level, we adopted moves to change the production plans of the $\mathrm{PI_v}$, called production plan (PP). It shifts part of the lot produced at period t' to a period t'' in which the production also takes place. We consider the periods where the production occurs as this shift takes place if the following conditions are met: *(i)* period t'' corresponds to the beginning of the adjacent production stages, *(ii)* there is idle production capacity at t'', and *(iii)* there is the possibility to store the excess production at the plant.

To understand the concept of production stages, let's see the following example illustrated in Figure 4.4. Suppose that there are 14 periods the production occurs at periods $t = 1, 5, 10$, and 13 (blue and solid lines). It implies that the production lot manufactured in $t = 1$ fulfills the demands of subsequent periods $1, 2, 3$ and 4. The production of period $t = 5$ fulfills the demands of periods $5, 6, 7, 8$ and 9. The same reasoning is applied to periods of 10 and 13. The inventory level of each stage is represented by red and dashed lines. Once the proposed RPRP allows back-order, and the tactical problem $\mathrm{PI_v}$ does not contemplate the routing constraints, the production and inventory plans are found after solving $\mathrm{PI_v}$ may not be properly fitting the demands.

Lets us consider that part of the demand $d^{t''}$ of the period $t'' = 4$ is back-ordered in $b^{t''}$ units to period $t' = 5$. Then, to fulfill $d^{t''}$ fully inside its current stage, part of the production occurred in $t' = 5$, which corresponds to the quantity $b^{t''}$ back-ordered, can be transferred to

$t'' = 4$. Lets us also considers that the manufactured lot in period $t' = 10$ contemplates part of the demand of period $t'' = 13$, yielding to not necessary inventory levels, i.e., $I^{12} > d^{12}$. Thus, part of the production lot of period $t' = 10$ can be transferred to $t'' = 13$ to meet demand in due time, i.e., $I^{12} = 0, p^{13} = d^{13} + d^{14}$.

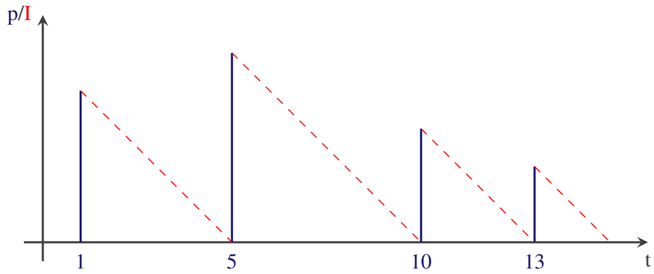


Figure 4.4 – Production stages and inventory levels vs. planning horizon.

With this example, it is possible to see that two possible moves can be realized on the production lots, shifting the lot partially or completely to other adjacent periods where production takes place. If $t'' < t'$, then the move is anticipating the production, otherwise ($t'' > t'$), the production is delayed. The possible shifted amount is calculated with Equation (4.5), which defines the move feasibility. For each product $k \in \mathcal{P}$, it is possible to shift from period t' to an adjacent moment t'' the minimum value among its entire production lot $p_k^{t'}$, the residual production capacity $\overline{M}_k^{t''} - p_k^{t''}$ of t'' or the residual holding capacity $U_0^k - I_{0k}^{t''}$ at the plant during t''.

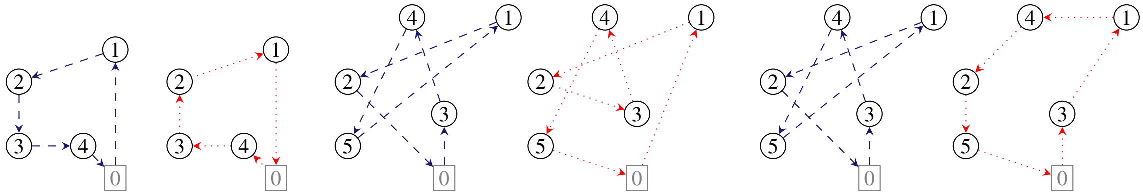
$$m_k^{t't''} = \min\{p_k^{t'}, \overline{M}_k^{t''} - p_k^{t''}, U_0^k - I_{0k}^{t''}\}, \forall k \in \mathcal{P}, t', t'' \in \mathcal{T}, t'' \neq t' \tag{4.5}$$

Every time the operator PP is selected, it lists all feasible moves of the production lots. Then, it randomly draws up to $maxPertPP$ moves for different products and periods. To keep the solution feasibility, it allows the modification of the production plan of a product $k \in \mathcal{P}$ up to one time per perturbation call. The selected moves are applied to $\mathrm{PI_v}$ by changing the variables' bounds while taken into the consideration the value of Δ_{ik}^{vt}.

4.4.2 Routing operators

Exchanges on the routing design characterize perturbation on the operational level. To modify the routes, five operators are outlined. For each of them, between 25% and 75% of the time periods are randomly selected, and perturbed, always keeping the solution feasible. They are separated into intra-route and inter-route operators. After the execution of routing perturbations following presented, the NRS (Section 4.3) procedure is applied to the attained solution.

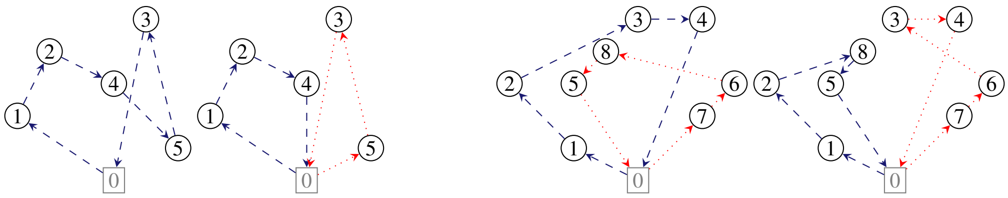
The intra-routes perturbations modify one route individually, they are the reverse-route (RR), lexicographic (LX), and randomize route (RD). Figure 4.5 illustrates them. The route connection is represented in dashed and blue arrows, and after the perturbation applied over a single route, its arcs are in dotted and red. Operator RR inverts the sense of selected routes, giving more opportunity to the last visited nodes to be first explored by the routing local searches (Fig. 4.5a). It reverts the sense of up to $maxV$ routes of a period time $t \in \mathcal{T}$. This parameter is randomly select from interval $[0, V]$ every time the operator is called. As an example for LX e RD, suppose that a route visits a subset of customers in the following order $3, 4, 5, 1, 2$. The operator LX turns the sequence into $1, 2, 3, 4, 5$ (Fig. 4.5b), while RD could turn into $3, 1, 4, 2, 5$ (Fig. 4.5c). Both operators LX and RD were calibrated, and they are executed in each selected time period $t \in \mathcal{T}$ up to $maxPertLX$ and $maxPertLX$ times, respectively.



(a) Reverse-route (RR). (b) Lexicographic (LX). (c) Randomize (RD).

Figure 4.5 – Examples of intra-route perturbation operators.

The inter-route perturbation operators modify simultaneously two routes. Figure 4.6 illustrates the adopted inter-route operators. One of the routes is in blue and dashed arrows and the other is in red and dotted arrows. The split-route (SR) operator looks for some empty vehicle and allocates to it some or all retailers visited by another route. For example, a route that visits the following customers $1, 2, 4, 5, 3$ could be divided into $1, 2, 4$ and $5, 3$ (Fig. 4.6a). This operator search for some empty vehicle $v_1 \in \mathcal{V}$ and randomly selects another one $v_2 \in \mathcal{V}, v_1 \neq v_2$ that is activated and transfer as much as possible load from v_t to v_1. The KK-swap (KK) operator randomly selects two routes, and also strings of sequential nodes with sizes K_1 and K_2 belonging to them. These strings are exchanged between the routes, being allocated from a route to the end of the other, for example, suppose that a route visiting clients $1, 2, 3, 4$ and a second route visiting $7, 6, 8, 5$. Applying operator KK, they could be turn into $1, 2, 8, 5$ and $7, 6, 3, 4$ (Fig. 4.6b). This perturbation operator is applied up to $maxPertKK$ over each selected time period $t \in \mathcal{T}$.



(a) Split-route (SR). (b) KK-swap (KK).

Figure 4.6 – Examples of inter-route perturbation operators.

4.4.3 Distribution plan operator

The operational level may be perturbed by modifying the distribution plan. Qiu et al. (2018b) outlined that the delivery loads could be moved totally or partially between periods. The authors formulated that these movements can anticipate or delay deliveries, aiming to form as many full-load vehicles as possible.

Following this concept, we also apply these moves, but it is extended to incorporate important features of our work which are the existence of multiple products and a heterogeneous fleet. For this, a move $m_{ik}^{tt'}$ is based on the transfer of some quantity of the item k delivered for the client i at period t to another period t', and must not violate the bounds on the inventory levels I_{ik}^{t} and $I_{ik}^{t'}$, as shown in Armentano et al. (2011). If $t' < t$, then the method is anticipating the delivery, otherwise $t' > t$, the delivery is delayed, and they also occur inside the respective production stage.

Based in the example illustrated in Figure 4.4, with 14 periods and production occurring in periods $t = 1, 5, 10$, and 13. As aforementioned, the manufactured lot of period $t = 1$ fulfill the demands of subsequent periods $1, 2, 3$ and 4. The same reasoning is applied to periods $5, 10$ and 13. As an example of moving a delivered lot, from $t = 7$, then a move $m_{ik}^{tt'}$ could happen to a period such as $t' = \{1, 2, 3, 4\} \cup \{5, 6, 8, 9\} \cup \{10, 11, 12\}$, i.e., transferring to the predecessor, current or successor production stage.

Equation 4.6 computes the anticipating moves. This kind of movement causes a decrease in the inventory levels of the plant I_{0k}^{t}, and an increase in the inventory level of the clients $I_{ik}^{t}, \forall i \in \overline{\mathcal{N}}$. For each product $k \in \mathcal{P}$ and customer $i \in \overline{\mathcal{N}}$ is possible to move from period t to t' the minimum value among the following: the current delivery load q_{ik}^{t}, or the minimum residual inventory level value $\min_{\tau}\left\{U_i^k - I_{ik}^{\tau}\right\}$, and the maximum residual vehicle capacity $\max_{v\in\mathcal{V}}\{Q^v - \sum_{j\in\overline{\mathcal{N}}}\sum_{k\in\mathcal{P}} q_{jk}^{vt'}\}$.

$$m_{ik}^{tt'} = \min\left\{q_{ik}^{t}, \min_{\tau}\left\{U_i^k - I_{ik}^{\tau}\right\}, \max_{v\in\mathcal{V}}\left\{Q^v - \sum_{j\in\overline{\mathcal{N}}}\sum_{k\in\mathcal{P}} q_{jk}^{vt'}\right\}\right\}, \tau = t', \ldots, t-1, \forall t' < t \tag{4.6}$$

Equation 4.7 describes the delaying moves. This kind of move provokes an increase in the inventory levels of the plant I_{0k}^{t}, and a decrease in the inventory level of the clients $I_{ik}^{t}, \forall i \in \overline{\mathcal{N}}$. For each product $k \in \mathcal{P}$ and customer $i \in \overline{\mathcal{N}}$ is possible to move from period t to t' the minimum value among the following: the current delivery load q_{ik}^{t}, or the minimum inventory level value $\min_{\tau}\left\{I_{ik}^{\tau}\right\}$, and the maximum residual vehicle capacity $\max_{v\in\mathcal{V}}\{Q^v - \sum_{j\in\overline{\mathcal{N}}}\sum_{k\in\mathcal{P}} q_{jk}^{vt'}\}$.

$$m_{ik}^{tt'} = \min\left\{q_{ik}^{t}, \min_{\tau}\left\{I_{ik}^{\tau}\right\}, \max_{v\in\mathcal{V}}\left\{Q^v - \sum_{j\in\overline{\mathcal{N}}}\sum_{k\in\mathcal{P}} q_{jk}^{vt'}\right\}\right\}, \tau = t, \ldots, t'-1, \forall t' > t \tag{4.7}$$

The aforementioned moves try to delay deliveries of those clients with inventory costs bigger than the plant ones and to advance deliveries for clients with inventory costs smaller than the plant ones. It is important to note that all generated moves $m_{ik}^{tt'}$ are feasible, and they only occur if there is delivery in the period t from which the cargo originates, and in the period t'' where the quantity moved will be allocated, there is availability to be absorbed by the residual capacity and cargo of the respective stocks and vehicles available.

With a set of moves generated, a maximum of $maxPertDL$ moves $m_{ik}^{tt'}$ are performed. Then, a new solution for the $\mathrm{VR_t}$ problem is built as done in Algorithm 4.1 (line 4) and re-optimized with the adaptive version of the inter-route algorithm 4.3. For simplicity, this operator is called DL.

4.5 Updating Δ

Building an initial solution for the bottom-up method or every time a perturbation operator is applied on the solution from the three top-down algorithms, the indirect costs $\mathbf{\Delta}$ are updated according to Equation 4.8. They offer much more accurate information about the problem and assess the influence that the transported lots have on the production, storage, and routing components.

$$\Delta_{ik}^{vt} = \omega\left(\frac{\Delta_i}{\Delta_r} + \frac{e^v}{Q^v} + \frac{h_i^k}{(\overline{I}_{ik}^t - \overline{I}_{ik}^{t-1})} + \frac{u^k}{(\overline{p}_k^t - \overline{p}_k^{t-1})}\right), \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T} \tag{4.8}$$

Equation 4.8 reflects the sum of the following components. Parameter $\Delta_i = c_{hi} + c_{ij}, \forall i, j, k \in r \subset \mathcal{N}$, i.e., it is equal to the sum of the costs of the corresponding arcs that connects the node i inside the route r, whose total length is equal to Δ_r. The second component is the fraction ${}^{e^v}/_{Q^v}$, it shows how much it costs to transport a unit by each vehicle $v \in \mathcal{V}$. The component that divides the holding cost h_i^k per the difference between the inventory levels at period t and $t-1$ estimates how much each load unity impacts the inventory from a period to another. The reasoning is analogous for the last component considering the production costs and manufactured lots.

By adding more information to the $\mathbf{\Delta}$, the decision making at the tactical level is enriched and able to achieve better solutions. This is presented in Section 4.7.2, where the proposed new Δ_{ik}^{vt} is compared with the one proposed by Qiu et al. (2018b).

4.6 Iterated local search matheuristics

The outlined methods split the problem into two, a top tier deciding about the production and inventory plans, as the distribution of goods, and the bottom tier solving the routing component. They are embedded within an ILS metaheuristic, whose core-concept involves improving a current solution by generating new initial solutions through perturbations combined with local searches (LOURENÇO; MARTIN; STÜTZLE, 2019).

4.6.1 Top-down framework

Three of the devised hybrid approaches follow a top-down decision-making process, usually adopted by production planning systems (BITRAN; TIRUPATI, 1993). The Algorithm 4.4 shows the main steps of the devised top-down ILS (TDILS) framework. An initial solution s^* is provided by Algorithm 4.1 and declared as the current best solution (line 1). Each iteration of the outer loop (lines 2-20) tries to improve the current best solution s^*. In line 4, the setup variables y have their values fixed for each period to the corresponding products being manufactured in solution s^*. It aims to make the resulting problem easier and quicker to be solved. Line 5 initializes the set P of perturbation operators.

In the inner loop (lines 6-19), a perturbation operator is randomly selected to avoid favoritism and applied to the current solution. The chosen perturbation operator modifies either the tactical or the operational part of the solution, adopting the operators presented in Section 4.4.1 and 4.4.2.

Parameter $\boldsymbol{\Delta}$ and its updating function are the core of the TDILS algorithm. As aforementioned, parameter $\boldsymbol{\Delta}$ is a vector of indirect costs, used for the to-be-delivered loads, and to guide the solution of the tactical problem $\mathrm{PI_v}$. As the RPRP is here solved in two stages, first the production, inventory, and distribution plans, followed by the routing one, the parameter $\boldsymbol{\Delta}$ estimate the impact that the delivery lots have on the production plan and routing design. It works as a feedback parameter to guide the solution of the top tier problem. Function UpdateΔ renews parameter $\boldsymbol{\Delta}$ at the top tier problem $\mathrm{PI_v}$ considering the modifications done by the operator ϵ^κ, please see Section 4.5.

In line 10, the problem $\mathrm{PI_v}$ is optimized considering the renewed $\boldsymbol{\Delta}$, and examined if it reached a better objective function value. If a better solution is found, the search continues from it, otherwise, the current solution has its objective function value tested, and if it is less than $(1 + \alpha)$ times the objective function value, this solution is explored in the next iteration. This step allows diversification, exploring non-local optimal solutions, and are adapted from the Skewed variable neighborhood search (HANSEN; MLADENOVIĆ, 2001).

A perturbation operator is discarded from the list if a solution improvement is not attained in the current iteration. This strategy of perturbation operator can be understood as a combination of the cyclic and pipe neighborhood exchange steps (HANSEN et al., 2017). Because if there is an improvement, the algorithm does not immediately return to the first perturbation operator, i.e., it can explore a different perturbation operator from the current one unless it is drawn again. But if there is not an improvement, the remaining operators have the same opportunity of being selected. The results show that no operator was at a disadvantage to the others.

If the setup variables y at the tactical problem are fixed, then they are unfixed, otherwise, they are fixed. It works closely related to a relax-and-fix procedure (POCHET; WOLSEY, 2006). Recall that, with y fixed, it is expected that the problem will be easily and quickly solved.

The proposed top-down algorithms were named after the adopted perturbation strategy, TILS, OILS, and IILS referring to tactical, operational, and integrated, respectively. TILS adopted the production plan operator PP from Section 4.4.1. OILS picked the five routing operators RR, LX, RD, SR, and KK presented in Section 4.4.2. IILS works with perturbations the same adopted perturbation operators by TILS and OILS.

Algorithm 4.4: Top-down ILS (TDILS)

Data: Problem information, costs $\boldsymbol{\Delta}$

$s^* \leftarrow$ BuildAndOptimize ($\boldsymbol{\Delta}$); //Alg. 4.1
for $i \leftarrow maxIter$ **do**
    $s \leftarrow s^*$;
    fix the setup variables y;
    $\mathrm{P} \leftarrow \{(\epsilon^\kappa)\}$;
    while $\mathrm{P} \neq \emptyset$ **do**
        $\epsilon^\kappa \leftarrow$ Random(P);
        $s' \leftarrow$ Perturb(s, ϵ^κ);
        $s' \leftarrow$ Update$\Delta(s')$;
        $s'' \leftarrow$ Opt(s');
        if $f(s'') < f(s)$ **then**
            $s \leftarrow s''$;
            if $f(s) < f(s^*)$ **then** $s^* \leftarrow s$;
        else
            $\mathrm{P} \leftarrow \mathrm{P} \setminus \{\epsilon^\kappa\}$;
            if $f(s'') < (1+\alpha)f(s)$ **then** $s \leftarrow s''$;
            unfix/fix y;
        end
    end
end
return s^*;

4.6.2 Adaptive bottom-up framework

Accordingly to Darvish and Coelho (2018), in a bottom-up approach is supposed that the distribution managers have the most power in the decisions and can, therefore, determine how the rest of the system works. It is done optimizing the distribution decisions and right after, fixing them in the production-inventory problem. Our approach jointly optimized routing and distribution decisions.

The following bottom-up approach aims to provide an alternative to the usual top-down decision making, making the distribution and routing decisions take place before the production-inventory ones. In the proposed algorithm, the focus is on modifying the delivery and routing plans, and shortly thereafter to fix the renewed delivery decisions in the top tier problem $\mathrm{PI_v}$.

Algorithm 4.5 summarizes how the proposed adaptive bottom-up iterated local search (ABUILS) works. The proposed procedure starts with an initial solution provided by the Algorithm 4.2 and it is declared as the current best solution s^* (line 1). In line 2, the linear coefficients $\boldsymbol{\Delta}$ of the load variables q in the problem $\mathrm{PI_v}$ are set to 0. The setup variables y are fixed in their respective values (line 3).

Each iteration of the outer loop (lines 4-23) explores the current best solution s^* for up to $maxIter$ iterations. First, s^* is assigned to an incumbent solution s (line 5). The operator mechanism set is initialized with possible moves (line 6). In line 7, different from the TDILS approaches (Alg. 4.4), the perturbation operators are adaptively selected using a roulette wheel (RW) procedure, discussed in Appendix C. Here, vector $\mathbb{S}$ registers the number of times that an operator κ lead to some solution improvement. Parameter $\mathbf{S}$ accounts the total of improvements achieved.

Algorithm 4.5: Adaptive bottom-up ILS (ABUILS)

Data: Problem information, costs $\boldsymbol{\Delta}$

1 $s^* \leftarrow$ InitialSolution($\boldsymbol{\Delta}$); //Alg. 4.2
2 $s^* \leftarrow$ Update$\Delta(s^*, 0)$;
3 fix the setup variables y;
4 **for** $i \leftarrow maxIter$ **do**
5     $s \leftarrow s^*$;
6     $\mathrm{R} \leftarrow \{(\epsilon^\kappa)\}$;
7   RW $\leftarrow$ RouletteWheel ($\mathbb{S}$,**S**,|R|); //Alg. C.1
8   **while** $\mathrm{R} \neq \emptyset$ **do**
9         $\epsilon^\kappa \leftarrow$ AdaptiveSelection(RW, $|\mathrm{R}|$); //Alg. C.2
10         $s' \leftarrow$ PerturbAndOptVRP(s, ϵ^κ);
11         $s'' \leftarrow$ OptPI$(s', \overline{q})$;
12   **if** $f(s'') < f(s)$ **then**
13             $s \leftarrow s''$;
14   **if** $f(s) < f(s^*)$ **then** $s^* \leftarrow s$;
15             $\mathbb{S}[\kappa] \leftarrow \mathbb{S}[\kappa] + 1$;
16             $\mathbf{S} \leftarrow \mathbf{S} + 1$;
17   **else**
18             $\mathrm{R} \leftarrow \mathrm{R} \setminus \{\epsilon^\kappa\}$;
19   **if** $f(s'') < (1+\alpha)f(s)$ **then** $s \leftarrow s''$;
20   unfix/fix y;
21   **end**
22   **end**
23 **end**
24 **return** s^*;

While some move can improve the solution, the following steps are done (lines 8-22). With the accumulative success RW of each of the perturbations, one is selected from set R, all of them keeping the solution feasible (line 9). Five of them consist of the operators Reverse-route (RR), Split-route (SR), KK-swap (KK), Lexicographic (LX), and Random (RD), introduced at Section 4.4.2, and each of them modifying between 25% and 75% of the periods, generating s'. The sixth is the operator DL that modify the distribution plans, described in Section 4.4.3.

In line 10, if the distribution plan is modified, the routing solutions of the changed periods are rebuilt with the BuildVRP method, and optimized with an adaptive version of the NRS procedure (Alg. 4.3). If the routes are changed by routing operators, the same adaptive NRS method optimizes them. The scheme adopted to select the operators adaptively is applied on the inter-route selection and can be found in Appendix C.

The attained routing solution deliveries which it considers the best assignment of vehicle-customer. At the top tier problem $\mathrm{PI_v}$), the delivery load variables $q_{ik}^{vt}, \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T}$ that do not match this vehicle-customer assignment are fixed to 0 (line 11). This corresponds to defining which retailers will be visited in each period and by which vehicles (routes).

An example to the vehicle-customer assignment and fixation is presented as follows, Suppose that for period $t = 2$, vehicle $v = 3$ visits the retailer $i = 1$. Thereby, all variables $q_{ik}^{vt}, \forall k \in \mathcal{P}$ with $t = 2$ and $i = 1$, but not with vehicle $v = 3$ are reset to 0. This guarantee simultaneously the feasibility of $\mathrm{PI_v}$ and $\mathrm{VR_t}$.

The new obtained solution s'' is tested, and if better than the current, in the next iteration the search continues from it (line 12). This test also updates two parameters used by the roulette wheel procedure. Parameter $\mathbb{S}$ is a vector that stores the number of times that each perturbation ϵ^κ leads to some improvement, while **S** stores the total of improvements. If s'' is not better than s, but not worst than α times $f(s)$ the search is allowed to modify s again in the next iteration. But, the operator is eliminated from the list R. As done for the Algorithm 4.4, if the solution found is worse, setup **y** variables are unfixed, if they are fixed, otherwise, they are fixed.

4.7 Computational results and analysis

This section presents the parameters adjustment, the results, and the analysis of the computational experiments performed by the TILS, OILS, IILS, and ABUILS algorithms. All the examined data are available in Appendix D.

4.7.1 Parameters adjustment and implementation details

All experiments were performed on an Intel® Xeon™ CPU E5-2687W v3 @ 3.10GHz computer with 160 GB of RAM and running Ubuntu Linux 18.04. The matheuristics were coded in C++ and used the Concert Technology of CPLEX to solve the production planning problem. We solved each instance by matheuristic 10 times as planned by a power of sample test with a power of 95% to detect the differences on the averages. We set the stopping criterion of 7200 seconds for all algorithms.

The TILS variant of the Algorithm 4.4 only works with PP operator. Then, the inner loop (lines 6-19) was modified to repeat at most $maxIterTILS$ number of times. Every time an improvement was found at line 13, the counter $nIter \in [0, maxIterTILS]$ was reset to 0, otherwise it was incremented by one.

Parameters α, $\mathrm{maxWeight}$, $\mathrm{maxIter}$, $\mathrm{maxPertPP}$, $\mathrm{maxPertRD}$, $\mathrm{maxPertLX}$, $\mathrm{maxPertKK}$, $\mathrm{maxPertDL}$ were all calibrated using irace package (LÓPEZ-IBÁÑEZ et al., 2016), an automatic configuration tool. Parameter α, in Algorithms 4.4 and 4.5, allows the method to explores more configurations of the basin of attraction without straying too far from the best current solution. Parameter $\mathrm{maxWeight}$ adds an aleatory component to the problem, modifying the $\boldsymbol{\Delta}$ costs. Parameter $\mathrm{maxIter}$ defines the maximum number of iterations of loop between lines 2 and 20 in Algorithm 4.4. Parameter $\mathrm{maxPertPP}$ bounds the maximum number of moves that perturbation operator PP realizes (Section 4.4.1). Parameters $\mathrm{maxPertRD}, \mathrm{maxPertLX}, \mathrm{maxPertKK}$ limit the number of times that the routing perturbation operators RD, LX, and KK are applied on the routing solution (Section 4.4.2). Parameter $\mathrm{maxPertPP}$ borders the maximum number of moves that perturbation operator DL realizes (Section 4.4.3). Each algorithm was executed a thousand times and the irace picked the best values for each parameter (Table 4.1). The irace reports, the possible values for each parameter (see Table D.2), the instances used in the tests, as the matheuristics results are available are in Appendix D.

Table 4.1 – Adopted parameter values by the devised methods.

Parameter	TILS	OILS	IILS	ABUILS
α	10%	20%	15%	20%
$maxWeight$	5	5	5	5
$maxIter$	300	300	300	500
$maxPertPP$	15	-	5	-
$maxPertRD$	-	7	7	3
$maxPertLX$	-	3	3	7
$maxPertKK$	-	15	10	15
$maxPertDL$	-	-	-	3

4.7.2 The importance of $\boldsymbol{\Delta}$

Recall that our parameter Δ_{ik}^{vt} estimates the impact of the routing and distribution decisions when making a new production plan. It helps to guide CPLEX to find more promising production and inventory plans. Here, we compare our parameter with the one introduced in Qiu et al. (2018b). As far as we know, it is the first time that indirect costs related to delivery variables q are used.

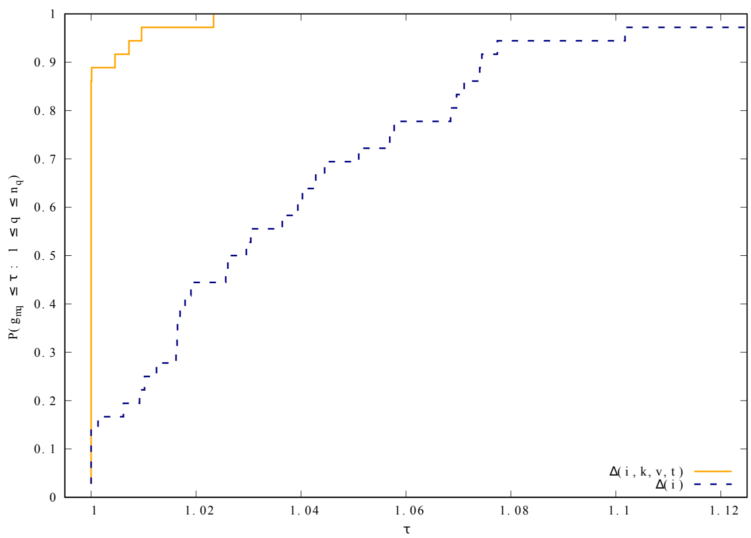


Figure 4.7 – Comparison of the new Δ_{ik}^{vt} with the Δ_i of Qiu et al. (2018b).

Using both theirs and our parameter $\boldsymbol{\Delta}$, the TDILS algorithms were tested on twelve instances PX0n5p15tYv (where $\mathrm{X} = 2, 3, 4, 5,$ and $\mathrm{Y} \in \mathcal{V}$, see Table D.1). Figure 4.7 brings the attained benchmark profile (DOLAN; MORÉ, 2002), where the plots only use the overall best solutions of the algorithms with respect to the adopted parameter $\boldsymbol{\Delta}$. Our adaptive indirect cost Δ_{ik}^{vt} (solid and orange lines) reaches 90% of the best solution values, which demonstrates its superiority over Δ_i (dashed and blue lines).

To draw further insights, about the contribution of the $\boldsymbol{\Delta}$, we performed statistical tests (Table 4.2). Here, due to the smaller sample size and the absence of normality, please see the results of the Kolmogorov-Smirnov tests of Table 4.2a which shows that the p-value is less than 0.05, the non-parametric tests of Wilcoxon (WX) was applied. The alternative hypothesis of the WX signed-rank test verifies if the results are statistically different,i.e., it verifies if the Δ_{ik}^{vt} results are better than the Δ_i ones. Table 4.2b shows the p-values of the Wilcoxon tests, and that the results are indeed significantly different besides showing a clear superior performance of Δ_{ik}^{vt} over Δ_i, because both p-values tend to zero.

Table 4.2 – Statistical tests for the comparison of the new Δ_{ik}^{vt} with Δ_i of Qiu et al. (2018b).

(a) Kolmogorov-Smirnov tests.

$\boldsymbol{\Delta}$	**Statistic**	**p-value**
Δ_{ik}^{vt}	1.0	0.0
Δ_i	1.0	0.0

(b) Wilcoxon tests.

Alternative	**Statistic**	**p-value**
default	36.0	3.1×10^{-6}
greater	630.0	1.5×10^{-6}

4.7.3 Comparing the algorithms

We assessed the performance of the algorithms with respect to the solutions found, and time spent to reach them. First, the best solution values attained by the algorithms are compared to the 2COMM formulation upper bounds, due to 2COMM clearly outperforming the VINDX formulation, please see Section 3.4. Hereafter, the four devised algorithms, i.e., TILS, OILS, IILS, and ABUILS are compared among themselves.

As shown in Section 3.4, the 2COMM found solutions for 106 of 108 instances but running for up to six hours (21600 seconds). Observing the benchmark profile plot (DOLAN; MORÉ, 2002) illustrated in Figure 4.8a, it is possible to see that ABUILS founds solutions that are comparable or better for all proposed instances.

Observing the time, the ABUILS (TILS, OILS, IILS) average time until the best solution (Opt) was 1229.60 (729.60, 621.70, 729.60) seconds (see Table 4.7), which means approximately 5.7% (3.38%, 2.88%, 3.38%) of the time spent by the solver (21,600 seconds). ABUILS was responsible to reach around 54.65% of the best solutions. Disregarding the 2 instances that 2COMM found no upper bounds, on average, the bottom-up method was 2.65% better than the solver. With the best distance found of 18.77% and worse of -7.7% (negative sign means that 2COMM performed better). Analyzing the performance of the TDILS methods, we can see that

40% of their best-known solution (BKS) are better than 2COMM, but 60% are far worst.

To conclude this comparative analysis about the objective function values, Figure 4.8b illustrates the comparison of the averages obtained by the ILS in comparison with the best values found by the solver. For the ABUILS, it can be seen that around 35% of the instances had averages better than the best solutions found with the 2COMM. In contrast, the OILS and IILS methods had only 20% of their objective function averages better than 2COMM. The TILS had the poorest average performance. Tables D.3-D.5 present the percentage distance between the BKS of the proposed matheuristics and the 2COMM formulation.

The matheuristics behavior is studied by analyzing the objective values and running times comparing the top-down and bottom-up approaches. The benchmark profile plot in Figure 4.9 shows that the ABUILS reaches 93.5% of the best solution values when compared to the other matheuristics. Followed by OILS (3.7%), IILS (1.8%), and TILS (1.0%). ABUILS was on average 1.76% better than these methods, with the best distance equals 6.05% and the worse equals to -4.61%.

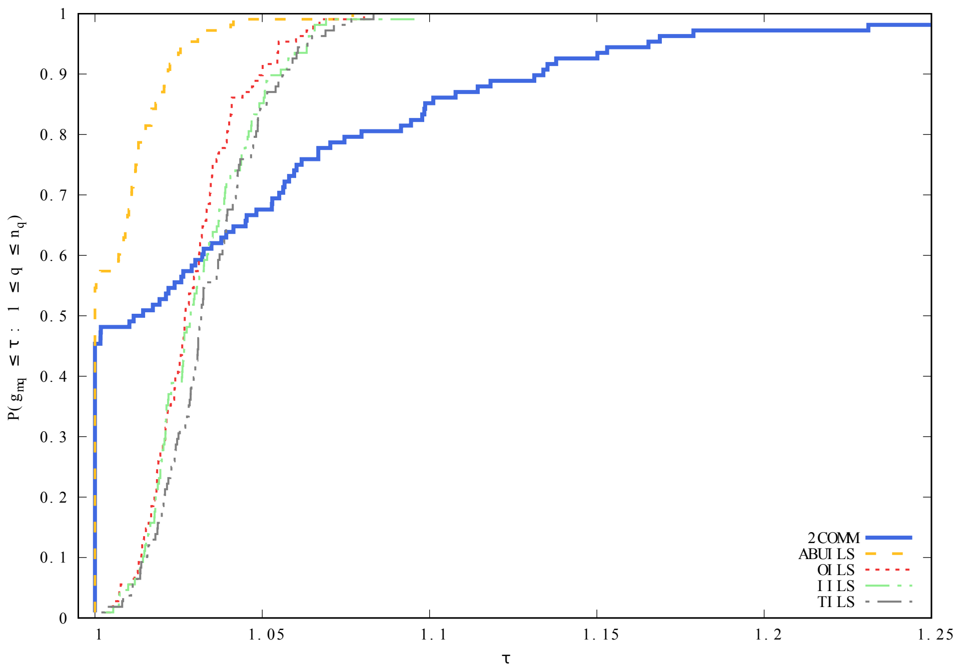
The parametric methodology, ANOVA was chosen, which, even with the premise of normality of the data, works very well even with asymmetric distributions, as long as the sample size is significantly large, which is the case. All the following statistical tests were performed using the *MINITAB® 19*.

The objective function values of the algorithms are studied. The null hypothesis tests if the average objective function value is equal and the alternative hypothesis they are not, with an $\alpha = 0.05$. Equality of variances of the algorithms was assumed for the analysis. As $\alpha < p\!-\!value$ (0.05 $\geq$ 0.05), it rejects the null hypothesis (Table 4.3). Thus, there is sufficient statistical evidence to infer that the four algorithms do not return the same average value for the objective function.

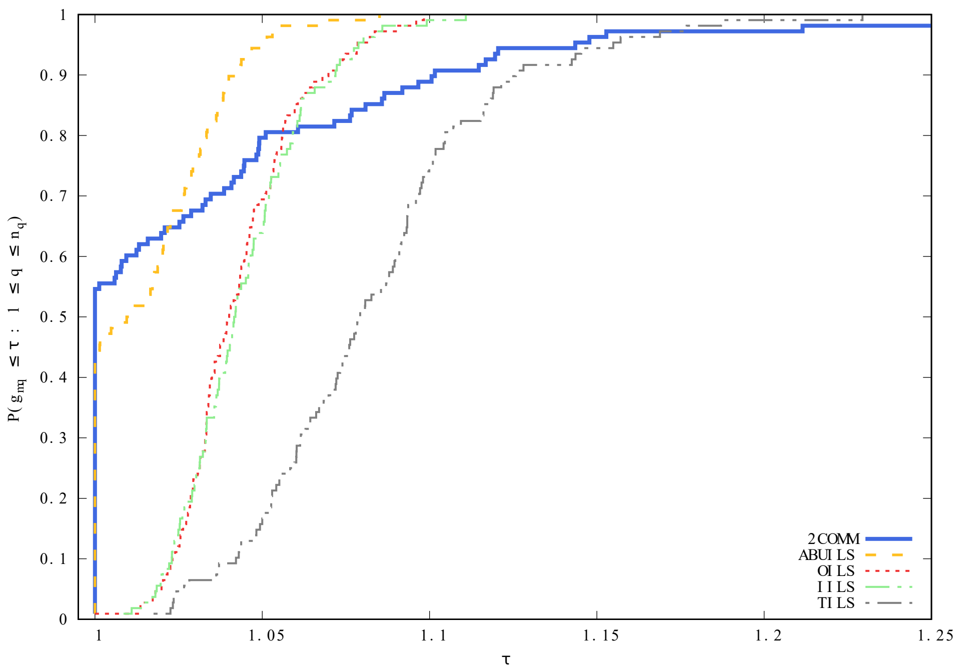
Table 4.3 – ANOVA tests for objective function values.

Source	F	p-value
o.f.	2.61	0.05

The Tukey simultaneous tests with 95% confidence for the average objective function was performed. With the averages presented in Table 4.4, their differences are shown in Table 4.5. Once the confidence interval (CI) difference of ABUILS-TILS does not contain the value 0, the difference between the average objective values can not be 0. That is, when an interval does not contain zero, the corresponding averages will be significantly different. On the other hand, the CI of ABUILS-OILS and ABUILS-IILS contains the value 0, then the average of the objective function value of these methods are not significantly different.



(a) Best known solution plot.



(b) Average objective function plot.

Figure 4.8 – Benchmark profiles comparing the devised ILS methods with 2COMM.

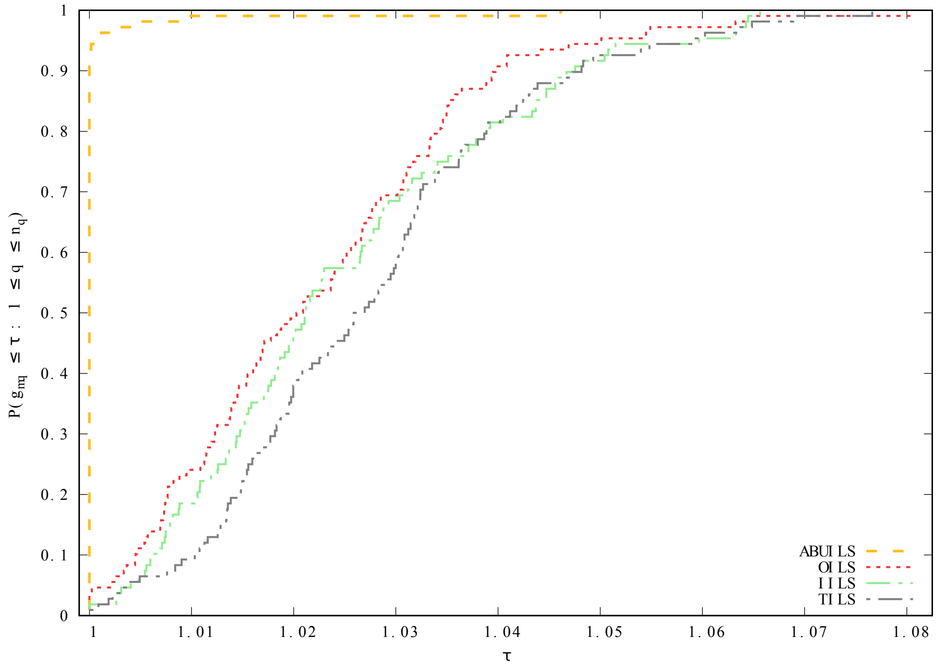


Figure 4.9 – Benchmark profile plot comparing the ABUILS with TDILS variants.

Table 4.4 – Averages and standard deviation for objective function.

Algorithm	N	AVG	SD
TILS	1080	1,031,022	576,228
OILS	1080	995,410	548,640
IILS	1080	1,002,700	551,385
ABUILS	1080	965,101	525,006

AVG - average,SD - standard deviation

Table 4.5 – Tukey simultaneous tests for differences of objective function means.

Level Diff	AVG Diff	SE Diff	CI of 95%	T-value	p-value
TILS - OILS	35612	23719	(-25,268; 96,493)	1.50	0.437
IILS - OILS	7291	23719	(-53,590; 68,171)	0.31	0.990
IILS - TILS	-28322	23719	(-89,203; 32,559)	-1.19	0.631
ABUILS - OILS	-30309	23719	(-91,190; 30,572)	-1.28	0.577
ABUILS - TILS	-65921	23719	(-126,802; -5,040)	-2.78	0.028
ABUILS - IILS	-37599	23719	(-98,480; 23,282)	-1.59	0.387

AVG - average, SE - standard error, Diff - difference, CI - confidence interval

The running time is analyzed considering four different measurements for each algorithm. They are (*i*) the time until the best solution - Opt; (*ii*) the cumulative time spent to solve the PI_{v} and (*iii*) VR_{t}; and (*iv*) the Total running time. The time measure (*ii*) includes the time spent by the production plan operator (PP), while (*iii*) accounts for the time spent with the routing and distribution operator (RR, SR, KK, LX, RD, DL).

A One-way ANOVA analysis is used to detect any differences between each of these times. The null hypothesis is if the average time spent is equal and the alternative hypothesis they are not, with an $\alpha = 0.05$. No equality of variances is assumed for the analysis. The Welch test is significant at a 5% level of significance, that is, there is sufficient statistical evidence to infer that the four algorithms have different Opt, $\mathrm{PI_v}$, $\mathrm{VR_t}$ and the Total average times (Table 4.6), once $\alpha \geq$ p-value, and then the null hypothesis is rejected.

Table 4.6 – Welch tests results.

Source	F	p-value
Opt	63.89	0.00
$\mathrm{PI_v}$	1576.55	0.00
$\mathrm{VR_t}$	145.05	0.00
Total	4434.59	0.00

Table 4.7 – Standard deviation and average of the times.

Time	TILS		OILS		IILS		ABUILS	
	SD	AVG	SD	AVG	SD	AVG	SD	AVG
Opt	654.50	729.60	462.50	621.70	404.98	729.60	1390.63	1229.60
$\mathrm{PI_v}$	1198.70	1482.30	547.40	1085.10	369.18	1018.50	998.98	3213.00
$\mathrm{VR_t}$	162.18	126.81	258.16	204.87	256.42	223.40	854.97	615.80
Total	1082.20	1868.30	456.00	1322.90	219.15	1321.99	684.85	3814.00

SD - standard deviation, AVG - average

Observing the Table 4.7 with the respective standard deviations and averages, it is possible to notice that the ABUILS algorithm has the bigger average times for all the time measurements. Thus, for better clarification, the Games-Howell paired comparison tests are realized to infer the average time difference between all algorithms.

The Games-Howell simultaneous test is significant at a 5% level of significance, that is, there is sufficient statistical evidence to infer that the difference in the average times of the four algorithms (Column AVG Diff, Table 4.8) is different from 0, then, the null hypothesis about the equality of the average times of the four algorithms is rejected ($\alpha \geq$ p-value) in the following pairwise comparisons. For Opt, ABUILS and OILS are different between themselves and from both TILS and IILS. For $\mathrm{PI_v}$ all algorithms are different among themselves. While for $\mathrm{VR_t}$ and Total average times, ABUILS and TILS are different between themselves and from both OILS and IILS.

At last, the performance of the algorithms is examined with a multiple time-to-target (MTTT) plot, an extension of the time-to-target plots (AIEX et al. 2002, 2007) and capable to set multiple instances simultaneously (REYES; RIBEIRO, 2018). Five randomly selected instances from the group of problems with 50 customers had their worst objective function value defined

Table 4.8 – Tukey simultaneous tests for differences of time means.

Time	Level Diff	AVG Diff	SE Diff	CI of 95%	T-value	p-value
Opt	TILS - OILS	107.9	24.4	(50.8; 165.0)	4.42	0.000
	IILS - OILS	107.9	24.4	(50.8; 165.0)	4.42	0.000
	IILS - TILS	0.0	28.2	(-65.9; 65.9)	0.00	1.000
	ABUILS - OILS	607.9	44.6	(493.4; 722.4)	13.63	0.000
	ABUILS - TILS	500.0	46.8	(380.0; 620.1)	10.69	0.000
	ABUILS - IILS	500.0	46.8	(380.0; 620.1)	10.69	0.000
$\mathrm{PI_v}$	TILS - OILS	397.3	40.1	(303.4; 491.1)	9.91	0.000
	IILS - OILS	-66.6	20.1	(-113.6; -19.6)	-3.31	0.003
	IILS - TILS	-463.9	38.2	(-553.2; -374.5)	-12.15	0.000
	ABUILS - OILS	2128.0	34.7	(2039.0; 2217.0)	61.39	0.000
	ABUILS - TILS	1730.7	47.5	(1608.8; 1852.6)	36.45	0.000
	ABUILS - IILS	2194.6	32.4	(2111.4; 2277.8)	67.72	0.000
$\mathrm{VR_t}$	TILS - OILS	-78.06	9.28	(-99.78; -56.35)	-8.41	0.000
	IILS - OILS	18.5	11.1	(-7.4; 44.4)	1.67	0.215
	IILS - TILS	96.59	9.23	(74.98; 118.20)	10.46	0.000
	ABUILS - OILS	411.0	27.2	(341.2; 480.7)	15.12	0.000
	ABUILS - TILS	489.0	26.5	(421.1; 557.0)	18.47	0.000
	ABUILS - IILS	392.4	27.2	(322.7; 462.2)	14.45	0.000
Total	TILS - OILS	545.4	35.7	(461.7; 629.0)	15.26	0.000
	IILS - OILS	-0.9	15.4	(-37.0; 35.1)	-0.06	0.998
	IILS - TILS	-546.3	33.6	(-624.9; -467.7)	-16.26	0.000
	ABUILS - OILS	2491.0	25.0	(2426.8; 2555.3)	99.50	0.000
	ABUILS - TILS	1945.7	39.0	(1845.6; 2045.7)	49.93	0.000
	ABUILS - IILS	2492.0	21.9	(2435.8; 2548.1)	113.89	0.000

as target for each matheuristic. Each matheuristic runs 100 times in each of the selected instances until the target was found or the limit time reached. Figure 4.10 presents the MTTT plot. All the outlined methods reach 54% of the attempted targets within 200 seconds. OILS and IILS (ABUILS and TILS) in 80% (70%) of attempts reach the targets within 400 seconds. But the ABUILS was the only one with 35% within 20 seconds, and guaranteeing 93.5% within 1400 seconds. While the three top-down methods, 90% of attempts need more than 1800 seconds. It suggests that the ABUILS was strongly dependent on the initial solution, but can escape more easily from local optima and to converge faster than the TDILS methods to the targets. In contrast, it is possible to see that the OILS and IILS have a closer performance to ABUILS. In common, OILS, IILS and ABUILS had the perturbation operators at the operational level.

The analysis is extended to the efficiency of the perturbation operators and routing local searches. Observing Table 4.9, it is possible to deduce alone in the TILS variation, the operator PP is not efficient, leading to only 1.43% of improvements. In contrast, when it works together with the routing operators in IILS, its efficiency grows almost 7.5 times. Similar behavior can be observed for the routing operators in OILS and IILS, their performance increase around 1% when coupled with PP. As for the ABUILS that selected the operator adaptively, the most efficient

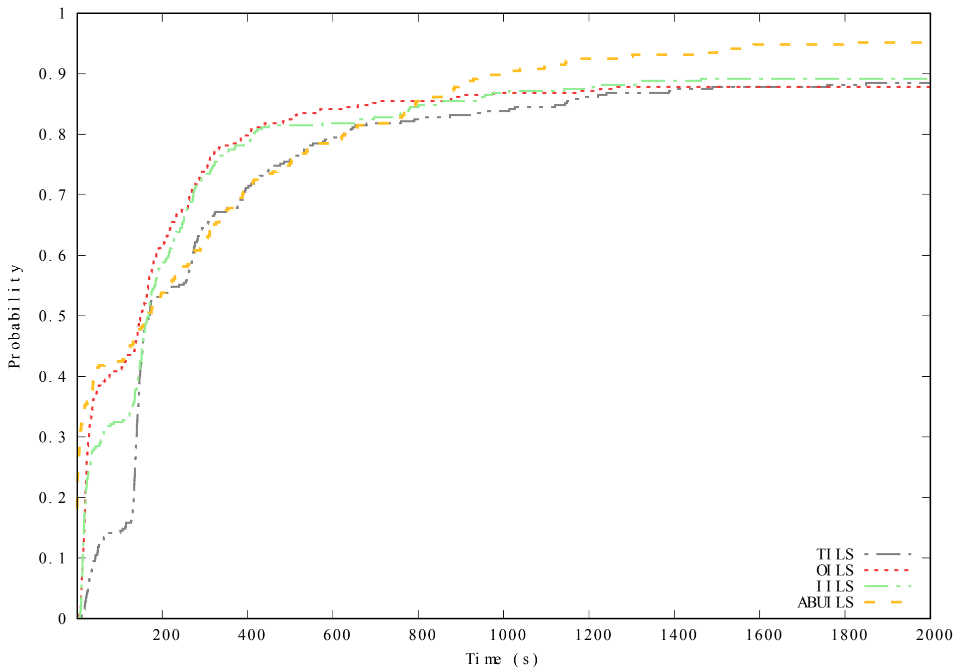


Figure 4.10 – Multiple TTT plot comparing ABUILS with the TDILS variants.

Table 4.9 – Perturbation operators efficiency per matheuristics.

Op.	TILS	OILS	IILS	ABUILS
PP	1,43%	-	10,72%	-
RR	-	4,93%	5,75%	1,86%
LX	-	4,93%	5,84%	2,76%
RD	-	4,95%	5,80%	3,44%
SR	-	4,84%	5,83%	3,82%
KK	-	4,94%	5,78%	4,30%
DL	-	-	-	2,18%

operators are the inter-route (SR and LX).

Table 4.10 summarizes the performance of the routing local searches. Observing the efficiency of the inter-route local searches, Table 4.10a, the three top-down methods reached practically the same performance. In contrast, the ABUILS concentrated the 99.19% of the improvements in four searches, SWAP (2,2), SWAP(2,1), SHIFT (2,0), and VLNS. The latter, in particular, had the best performance, reaching around 39% of the improvements for the TDILS, and 53.99% for the ABUILS. The intra-route methods presented in Table 4.10b, for all outlined matheuristics, had the same performance, once they were randomly selected instead of adaptively.

Table 4.10 – Routing local searches efficiency per matheuristics.

(a) Inter-route searches.

	TILS	OILS	IILS	ABUILS
VLNS	39,42%	38,46%	39,04%	53,99%
SHIFT(2,0)	12,24%	12,45%	12,45%	7,08%
SWAP(2,1)	24,16%	24,31%	24,08%	30,22%
SWAP(2,2)	15,03%	15,33%	15,02%	7,90%
CROSS	1,96%	2,01%	1,99%	0,05%
SHIFT(3)	3,71%	3,79%	3,83%	0,46%
SHIFT(4)	2,11%	2,21%	2,18%	0,19%
SHIFT(5)	1,36%	1,43%	1,40%	0,11%

(b) Intra-route searches.

	TILS	OILS	IILS	ABUILS
1PNT	20,70%	20,72%	20,68%	20,57%
2PNT	20,30%	20,33%	20,29%	19,69%
2OPT	25,85%	25,84%	25,83%	25,97%
3PNT	16,80%	16,78%	16,84%	16,53%
OROPT(3)	7,45%	7,45%	7,51%	7,65%
OROPT(4)	5,38%	5,36%	5,36%	5,63%
OROPT(5)	3,51%	3,52%	3,49%	3,95%

Despite the production, inventory decisions have a tactical character approach, usually occurring before the distribution and routing ones, they cannot be done apart or without at least considers the operational importance. After these analyses, it is clear to infer that operational decisions can strongly affect the performance of integrated problems. This corroborates the thesis that by spending more time in the prospection of the routing component, most of the solutions converge for faster and better results than those found focused on the production-inventory problem. This can be seen in the performance of the algorithms that used operators at the operational level, as ABUILS (best solution values) and OILS (best times until the best solution). It leads to the reflection on whether the decision-making could give more attention to the operational decisions and how they influence the tactical ones.

4.8 Final remarks

Due to the high-level computational challenge offered by the proposed RPRP, this study developed approaches that split it into two following top-down and bottom-up decision manners. A top-tier problem solved the production, inventory, and distribution plans with a commercial solver providing an estimate of the loads to be distributed. The bottom tier, for each period, routed heuristically the operational level.

Embedded in an ILS framework, three top-down algorithms were outlined. The first provoked perturbations at the tactical level, modifying the production plans and affecting directly the inventory, back-order, and distribution variables. While the second explored a set of mechanisms to modify the routing design, changing the operational level. The latter algorithm works by integrating perturbations at both levels. An important and common point in the implementation of the algorithms is the development of an intuitive way to calculate the costs of the delivered loads. This is because, if there is a need to deliver a lot, this lot influences the costs related to production, inventories, and transportation. In this way, during each iteration of the method, these indirect costs are updated with the latest information from the solution. Along with the perturbation mechanisms, it was able to lead to solutions superior to those found by the commercial solver.

The fourth algorithm was based on the bottom-up hierarchical approach, whereas the distribution decision had priority over the production of an inventory. Also ILS-based, it adaptively selected the perturbation operator and inter-route searches. The adaptive principle was a roulette wheel selection, i.e., the selection was proportional to the fitness of the mechanism or search to the problem. The method adopted an operator that perturbed the distribution plan. It was done exchanging lots partially or totally between periods, keeping the solution always feasible. The new routing arrangement Was optimized and fixed in the production-inventory problem, defining for each period in which customers are visited by each vehicle.

Through computational experiments, the methods were compared with the best-proposed formulation, and among themselves. These results are statistically analyzed, showing the importance of our indirect cost and perturbation mechanisms, especially the ones that modified the operational level. Analyzing the bottom-up performance, it was capable to reach equivalent or better solutions in a reasonable time, and with much less computational effort when compared to the formulation. And, according to the statistical analysis, the bottom-up spends more average time than the top-down methods but dominates most of the values of the solution. Besides, the methods that prospect the operational level is capable to bring better and faster solutions than the method that modifies the production plans.

5 A column generation approach for a rich production-routing problem

"Those who can imagine anything, can create the impossible".

Alan Turing

5.1 Introduction

The proposed RPRP grows quickly with the number of retailers, which directly impacts the vehicle-routing component, hindering thus the solution process. This impact was seen on the attained performances of the VINDX and 2COMM formulations and of the devised top-down and bottom-up matheuristics.

Given the possible impacts of the involved decision of the studied RPRP on the costs of a supply chain, it is important to assess the quality of the attained upper bounds by the devised matheuristics. One way to assess these upper bounds is to compare them with the attained linear programming relaxation bounds by formulation 2COMM, for instance. Nonetheless, better lower bounds may be obtained by using a different approach such as a column generation algorithm, the subject of this chapter. We devised a column generation algorithm that uses a compact and equivalent model for the studied RPRP, besides proposing a pricing algorithm, and to price columns heuristically.

Column generation or delayed column generation is an efficient technique for solving linear programs with a large number of variables. For many of these large linear programs, to explicitly consider all of their variables is impractical. Nevertheless, this is not a problem because most of the variables will be non-basic, i.e., they will be equal to zero in an optimal solution. This premise allows us then to consider only a subset of the variables when solving an optimization problem. The column generation technique leverages this idea by seeking to generate only those variables with the potential to improve the objective function. In a minimization problem, this would be translated as generating only those variables with negative reduced costs.

To leverage the idea requires coordinating the solution of two problems: A restricted master problem, which is the original problem but with only a subset of the variables being considered, and a subproblem, which is responsible for generating new variables.

The solution coordination works as follows. The restricted master problem is solved. From this solution, we obtain dual prices for each of the constraints in the master problem. These dual prices are then used in the objective function of the subproblem. The objective function of the subproblem is the reduced cost of the new variable concerning the current dual variables, and subject to the naturally occurring constraints. The subproblem is solved. If its optimal solution is negative, given, without loss of generality, that the original problem is a minimization one, then a variable with negative reduced cost has been found. This variable is then added to the restricted master problem, which is resolved. Re-solving the master problem will generate a new set of dual prices, and the process is repeated until no further negative reduced cost variables are identified. Whenever the subproblem returns a non-negative optimal solution, i.e, the reduced cost is non-negative, we can conclude that the optimal solution to the restricted master problem is optimal to the original problem.

The column generation technique has been originally proposed by Ford and Fulkerson (1958) but formalized by Dantzig and Wolfe (1960). In many cases, the technique has proved to be efficient to solve large linear programs such as the classical cutting stock (GILMORE; GOMORY, 1961; VANCE et al., 1994; VANCE, 1998), the crew scheduling (DESROCHERS; SOUMIS, 1989; BORNDÖRFER et al., 2006), the vehicle routing (PESSOA et al., 2009; BALDACCI et al., 2011; PECIN et al., 2017), and the capacitated p-median (LORENA; SENNE, 2004; CESELLI; RIGHINI, 2005) problems. One particular method in linear programming which uses this kind of approach is the Dantzig-Wolfe decomposition algorithm.

Figure 5.1 illustrates how the original problem is reduced to a restricted problem, and it has its columns added iteratively. Instead of adding only the column with the best-reduced cost (Dantzig's rule, see Le et al. (2013)), it is interesting to add more than one negative reduced cost column by iteration.

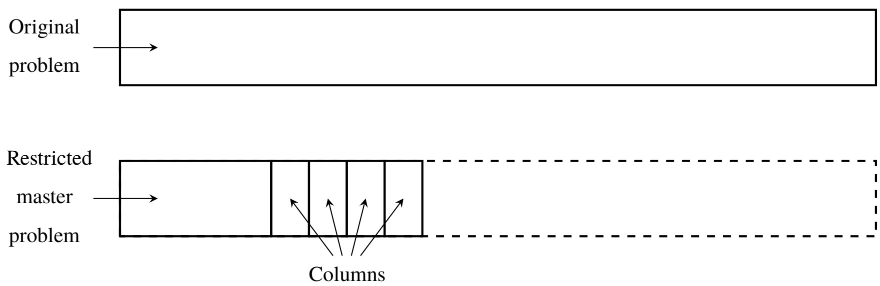


Figure 5.1 – Column generation representation.

5.2 Column generation algorithm

Algorithm 5.1 describes the adopted column generation algorithm. It starts building the restrict master problem (RMP), presented in Section 5.3. The master problem requires an initial set of columns. This set is provided with the procedure illustrated in Algorithm E.1. The current reduced cost $\tilde{r}_c$ receives $-\infty$ (line 3). While the least reduced cost found by the pricing subproblem is less than 0, the following steps occur (lines 4-8). The RMP is solved and its objective function computes a lower bound (LB). The values of dual variables $\boldsymbol{\Pi}$ related to the RMP constraints are retrieved (line 6). In line 7, the chosen pricing subproblem is solved considering the parameterized values of $\mathbf{\Pi}$, please see Section 5.4. In the case of a heterogeneous fleet, it is solved independently for each vehicle $v \in \mathcal{V}$. The columns with negative reduced cost are added to the RMP. If none negative reduced cost column is found ($\tilde{r}_c \geq 0$), the loop is interrupted, and the algorithm returns the attained LB found (line 9).

Algorithm 5.1: Column generation algorithm

Data: Problem information
1 **build** the RMP;
2 **generate** an initial solution; //Alg. E.1
3 $\tilde{r}_c \leftarrow -\infty$;
4 **while** $\tilde{r}_c < 0$ **do**
5   LB $\leftarrow$ **solve** the RMP;
6     $\overline{\mathbf{\Pi}} \leftarrow getDuals(\mathrm{RMP})$;
7     $\tilde{r}_c \leftarrow$ pricing($\overline{\mathbf{\Pi}}$);
8 **end**
9 **return** LB;

5.3 Restricted master problem

An equivalent column generation model for the proposed RPRP is outlined as follows. It is closely related to the set packing problem, as described in formulation (5.1)-(5.8), which is named rich restricted master problem (RRMP). In RRMP, the distribution and routing variables $\mathbf{g}, \mathbf{z}, \mathbf{w}$ and $\mathbf{q}$ are dropped out. The production-inventory variables $\mathbf{y}, \mathbf{p}, \mathbf{I}$, and $\mathbf{b}$ have the same meaning and domains, please see Table 3.2. Variables x_r^t have a new meaning, indicating if a route $r \in \mathcal{R}$ performs some deliveries in the period $t \in \mathcal{T}$. An advantage of the variables x_r^t is the omission of the vehicle index.

$$\min \sum_{t \in \mathcal{T}} \left\{ \sum_{k \in \mathcal{P}} \left[l^k y_k^t + u^k p_k^t + \sum_{i \in \mathcal{N}} \left(h_i^k I_{ik}^t + B_i^k b_{ik}^t \right) \right] + \sum_{r \in \mathcal{R}} c_r x_r^t \right\} \tag{5.1}$$

s.t.: (3.2)

$$I_{0k}^t = I_{0k}^{t-1} + p_k^t - \sum_{i \in \overline{\mathcal{N}}} \sum_{r \in \mathcal{R}} \overline{q}_{ikr}^t x_r^t, \qquad \forall k \in \mathcal{P}, t \in \mathcal{T} (\gamma_k^t) \tag{5.2}$$

$$I_{ik}^t = I_{ik}^{t-1} - d_{ik}^t + b_{ik}^t - b_{ik}^{t-1} + \sum_{r \in \mathcal{R}} \overline{q}_{ikr}^t x_r^t, \qquad \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, t \in \mathcal{T} (\delta_{ik}^t) \tag{5.3}$$

$$\sum_{r \in \mathcal{R}} \overline{z}_{ir}^t x_r^t \leq 1, \qquad \forall i \in \overline{\mathcal{N}}, t \in \mathcal{T} (\pi_i^t) \tag{5.4}$$

$$\sum_{r \in \mathcal{R}} \overline{g}_r^{vt} x_r^t \leq 1, \qquad \forall v \in \mathcal{V}, t \in \mathcal{T} (\phi^{vt}) \tag{5.5}$$

$$0 \leq I_{ik}^t \leq U_i^k, b_{ik}^t \geq 0, \qquad \forall i \in \mathcal{N}, k \in \mathcal{P}, t \in \mathcal{T} \tag{5.6}$$

$$p_k^t \geq 0, y_k^t \in \{0, 1\}, \qquad k \in \mathcal{P}, t \in \mathcal{T} \tag{5.7}$$

$$x_r^t \geq 0 \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{5.8}$$

Each column $x_r^t \geq 0, \forall r \in \mathcal{R}, t \in \mathcal{T}$ is composed by three parameters as follows: $\overline{q}_{ikr}^t \geq 0$ indicates how much of product $k \in \mathcal{P}$ is delivered to the customer $i \in \overline{\mathcal{N}}$; $\overline{z}_{ir}^t \in \mathbb{Z}_0^+$ shows the number of times that the customer $i \in \overline{\mathcal{N}}$ is visited; and $\overline{g}_r^{vt} \in \{0,1\}, \forall v \in \mathcal{V}$, signalizes which vehicle v performed the route. Its cost $c_r = e^v + \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij}^{vt}$, corresponds to the sum of the activation cost of the vehicle v choose to perform the route and the total traveling costs.

Observing RRMP, we can note that constraints (5.2) and (5.3) have the same meaning as (3.3) and (3.4). Constraints (5.4) and (5.5) limit the maximum number of times that a customer is visited and vehicle activation up to one per period. The remaining constraints (5.6)-(5.8) are variable domain related.

As is well known, the potential number of possible routes grows far beyond treatment, then it is impossible, and not interesting, to make explicit all of them. Then, to overcome this trouble, the columns that offer the potential to improve the RRMP are priced as presented in Section 5.4. An interesting characteristic of the column generation approaches involving product(inventory)-routing problem is that the delivered quantities are defined by the pricing problem, which increases its complexity.

5.4 Pricing subproblem

In column generation methods, it is known that the pricing problems are responsible for the major complexity of the algorithms, then they must be capable to generate good columns working as efficient and fast as possible. The first step to designing a pricing algorithm is to define the objective function considering the current dual values. After solving RRMP, dual prices $(\overline{\gamma}_k^t, \overline{\delta}_{ik}^t, \overline{\pi}_i^t, \overline{\phi}^{vt})$ are obtained from the constraints (5.2)-(5.5). The new columns are priced by the subproblem, considering the objective function (5.9). The reduced cost is related to each arc $(i,j) \in \mathcal{A}$ is calculated as shown in (5.10).

$$\min \sum_{t\in\mathcal{T}} \sum_{v\in\mathcal{V}} \left\{ \sum_{(i,j)\in\mathcal{A}} \overline{c}_{ij} x_{ij}^{vt} - \sum_{j\in\overline{\mathcal{N}}} \sum_{k\in\mathcal{P}} (\overline{\gamma}_k^t + \overline{\delta}_{jk}^t) q_{jk}^{vt} \right\} \tag{5.9}$$

$$\overline{c}_{ij} = \begin{cases} c_{0j} - \overline{\phi}^{vt}, & \forall j \in \overline{\mathcal{N}} \\ c_{ij} - \overline{\pi}_i^t, & \forall i \in \overline{\mathcal{N}} \end{cases} \tag{5.10}$$

The proposed RPRP has a multi-product characteristic, which implies that every time a label is extended to a node, it would have to decide not the quantity delivery of one product, but for up to P different items. This significantly increases the complexity of the algorithm which already must be run independently for each vehicle. To pick the combination of products that most contributes to the reduced cost is equivalently to solve the following bounded knapsack problem (BKP)

(5.11)-(5.13) for each visited customer $j \in \overline{\mathcal{N}}$.

$$\max \sum_{k \in \mathcal{P}} (\gamma_k^t + \delta_{jk}^t) q_{jk}^t \tag{5.11}$$

$$\sum_{k \in \mathcal{P}} q_{jk}^t \leq \tilde{Q} \tag{5.12}$$

$$0 \leq q_{jk}^t \leq O_{jk}^t \qquad \forall k \in \mathcal{P} \tag{5.13}$$

The parameter $\tilde{Q}$ is a residual load considering the previous deliveries performed by the vehicle v. Parameter $O_{jk}^t = \min\left\{U_j^k, \max_\tau\left\{\sum_{m=t}^{\tau} d_{jk}^m\right\}, \forall \tau = t+1, \ldots, T : d_{ik}^\tau > 0\right\}$ represents a maximum quantity delivered enough to satisfy the accumulated demand between the current period t and future periods that do have positive demands.

The studies of Engineer et al. (2012), Desaulniers, Rakke and Coelho (2016) and Qiu et al. (2017) developed efficient approaches to inventory/production-routing problems considering single-product and a homogeneous fleet. They worked with the concept of a time-expanded network, where each node corresponded to a pair (i, t) denoting a customer i and period t. The network was exploited with labeling algorithms based on the combination of two problems, the elementary shortest path with resource constraints, and the bounded knapsack.

The following Sections 5.4.1 and 5.4.2 present our proposed *ad hoc* labeling algorithm, considering the required combination of a resource-constrained shortest path and bounded knapsack problems, and the heuristically priced columns procedure, respectively.

5.4.1 Time-oriented simple pricing problem

Dror (1994) stated that the elementary shortest path problem with resource constraints (ESPPRC) belongs to the class of NP-hard problems. Its natural relaxation is the shortest path problem with resource constraints (SPPRC), which accepts the existence of cycles in the solution. If the SPPRC has only one resource, for example, a vehicle load capacity of Q units, it can be extended to the *q-routes* relaxation.

The *q-routes* was proposed by Christofides et al. (1981), and can be defined as a walk that starts at the depot vertex, traverses a sequence of client vertices with total demand at most equal to Q, and returns to the depot. Some vertices may be visited more than once, therefore the set of *q-routes* strictly contains the set of actual routes (PESSOA et al., 2009).

The *q-routes* is a dynamic programming algorithm and it was successfully adapted to the CVRP in the studies of Fukasawa et al. (2006), Pessoa et al. (2009), Contardo and Martinelli (2014) and Pecin et al. (2017), having all of them reached good results with very low computational times. A comparison among the non-elementary and elementary shortest path algorithms, and the *q-routes* can be found in Reis (2015). The study showed a *q-route* algorithm combined with the decremental state-space relaxation (RIGHINI; SALANI, 2008), and the strong degree constraints

(CONTARDO et al., 2015) that outperformed both ESPPRC and SPPRC with very low processing times.

Following all these aforementioned cases applying the *q-routes*, we proposed a variant called *t-routes*, once it is a time-oriented simple pricing problem. Instead of using the vehicles' capacity Q as a resource to limit the path formation, the maximum riding time H is adopted. Let $a_{ij}, \forall(i,j) \in \mathcal{A}$ be the travel time between i and j, and s_i the service time. Then, a *t-route* is a walk that starts at the plant node 0, visit a sequence of customers, and returns to the depot copy $n+1$ with arrival time lesser or equals to the maximum riding time H. Each *t-route* defines a set of indexed arcs $(i,j)^a$, which is started with an arc $(0,j)^a$, where j is the first client visited after leaves the depot and $a = a_{0j}$. Each arc $(h,i)^a$ is followed by an arc $(i,j)^{a+s_i+a_{ij}}$, where j is the client immediately visited after i, and $a \leq H - a_{j,n+1} - s_j$ is the riding time until j. The sequence ends necessarily with an arc $(i, n+1)^a$, where $i \in \overline{\mathcal{N}}$ is the last client before return to the depot $n+1$ and $a = a_i + s_i + a_{i,n+1} \leq H$.

Lets define a matrix R such each position $R(a,j)$ represents the partial minimum-reduced cost of the *t-route* that starts in the plant node 0 with initial time equal to 0, and ends in some customer $j \in \overline{\mathcal{N}}$ with total riding time $a_{0j} \leq a \leq H - s_j - a_{j,n+1}$. The cost $c_{j,n+1}$ of the arc $(j, n+1)$ that returns to the depot are not considered in R. Each entrance $R(a,j)$ has a label $\mathcal{L}$ that consists of two items, the predecessor node i, and the accumulated reduced cost $\overline{r}_c$. The complete path can be recovered from each node j using the predecessor node i, and the respective travel and service times.

Figure 5.2 illustrates an *t-route* example. Let the set of clients be $\overline{\mathcal{N}} = \{1, \ldots, 4\}$, $H = 7$, the service times equal to $s_i = 1, \forall i \in \overline{\mathcal{N}}$, and the travel times between nodes be symmetric and equal to $a_{01} = a_{04} = a_{13} = a_{14} = a_{34} = 1$, $a_{02} = a_{03} = a_{23} = a_{24} = a_{12} = 2$. The infeasible entries of the matrix R are filled with $\times$ symbol. A non-elementary *t-route* is represented in red and dashed arrows. It visits the subset of vertexes $\{2,4\}$ in the sequence $4-2-4$, and would be described by the following arcs: $(0,4)^1$, $(4,2)^3$, $(2,4)^5$. On the other hand, an elementary *t-route* is represented by blue and dotted arrows. It visits the subset of nodes $\{1,2\}$ in the sequence $2-1$, and would be represented by the following arcs: $(0,2)^2$, $(2,1)^5$.

The *t-routes* algorithm can be described in three steps as follows: *(a)* Initialization, *(b)* Filling and *(c)* Retrieving. Algorithm 5.2 summarizes the Initialization step. First, it creates a matrix R of $(H+1) \times (n+1)$ positions. In line 2, the position $R(0,0)$ receives and label with reduced cost $\overline{r}_c = 0$, and predecessor $i = 0$. The entries $R(a_{0j}, j), \forall j \in \overline{\mathcal{N}}$ are filled with a label corresponding of a lonely arc going direct from the plant node 0 to node j, that is, the reduced cost is $\overline{r}_c(R(a_{0j}, j)) = c_{0j}$, and the predecessor is $i(R(a_{0j}, j)) = 0$. The positions $R(a,j)$ such that $a < a_{0j}$ or $a > H - s_j - a_{j,n+1}$ also are initialized with infinite cost labels, because a client can not be reached with a riding time inferior to a_{0j} or superior to $H - s_j - a_{j,n+1}$.

The Filling step is described in Algorithm 5.3, which fills the positions $R(a,j)$ such that $j \in \overline{\mathcal{N}}$, and $a_{0j} < a \leq H - s_j - a_{j,n+1}$. For each $j \in \overline{\mathcal{N}}$ with respective feasible arrival time a,

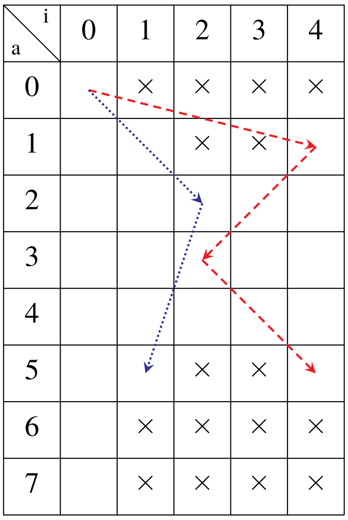


Figure 5.2 – *t-routes* example.

Algorithm 5.2: Initialization

Data: Problem information

create a matrix R with $(H+1) \times (n+1)$ positions;
$R(0,0) \leftarrow (0,0)$;
for $a = 1, ..., H$ **do**
    for $j \in \overline{\mathcal{N}}$ **do**
        if $a = a_{0j}$ **then**
            $\overline{r}_c(R(a,j)) \leftarrow c_{0j}$;
            $i(R(a,j)) \leftarrow 0$;
        else if $a < a_{0j}$ ***or*** $a > H - s_j - a_{j,n+1}$ **then**
            $\overline{r}_c(R(a,j)) \leftarrow +\infty$;
            $i(R(a,j)) \leftarrow -1$;
        end
    end
end
return matrix R;

---

this step chooses the best predecessor node i. To choose the predecessor node i with arrival time $a - s_i - a_{ij}$, the reduced cost until reaches j is computed with Equation (5.14).

$$\overline{r}_c(R(a,j)) = \min_{(i,j)^a \in \Delta_j^{(-)}} \{\overline{r}_c(R(a - s_i - a_{ij}, i)) + \overline{c}_{ij}\} \tag{5.14}$$

Then, for each position of the matrix $R(a,j)$ there is a *t-route* with the minimum reduced cost, and ending in a client $j \in \overline{\mathcal{N}}$ with a total riding time $a = a_{0j}, ..., H - s_j - a_{j,n+1}$, before returning to the plant copy $n+1$.

Algorithm 5.4 explains the Retrieving step. This step has the purpose of selecting for each client $j \in \overline{\mathcal{N}}$ the *t-route* with time a^* which minimizes the reduced cost $(\overline{r}_c(R(a^*,j)) + \overline{c}_{j,n+1})$ to be added to the RRMP.

It is possible to see that the proposed *t-routes* has no elementary guarantee, that is, cyclic

Algorithm 5.3: Filling

```
Data: Problem information, Matrix R
for a = 1, ..., H do
    for j ∈ N̄ do
        if a > a_{0j} and a ≤ H − s_j − a_{j,n+1} then
            c̃ ← +∞;
            ĩ ← −1;
            forall (i, j) ∈ A do
                r̄_c ← r̄_c(R(a − s_i − a_{ij}, i)) + c̄_{ij};
                if r̄_c < c̃ then
                    c̃ ← r̄_c;
                    ĩ ← i;
                end
            end
            r̄_c(R(a, j)) ← c̃;
            i(R(a, j)) ← ĩ;
        end
    end
end
return matrix R;
```

Algorithm 5.4: Retrieving

```
Data: Matrix R
for j ∈ N̄ do
    c̃ ← +∞;
    a* ← −1;
    for a = a_{0j}, ..., H − s_j − a_{j,n+1} do
        if r̄_c(R(a, j)) < c̃ then
            c̃ ← r̄_c(R(a, j));
            a* ← a;
        end
    end
    compute the corresponding t-route that starts in R(a*, j) into a column x_r^t with cost
      c_r ;
    add x_r^t to RRMP;
end
```

paths are allowed. But, observing the RRMP, it is possible to realize that non-elementary columns (that has at least one customer i with $\overline{z}^t_{ir} > 1$) will never be part basic due the presence of constraints (5.4). To get around this condition, the cycles can be progressively forbidden in the pricing algorithm. Here, the decremental state-space relaxation (DSSR) proposed by Righini and Salani (2008) was adopted. It helps the formation of elementary routes considering a new subset of clients, besides not changing the complexity of the pricing. Let's now consider that each entry in matrix $R(a, j)$ has also a has a field $\mathcal{U} \subseteq \overline{\mathcal{N}}$ that stores the set of clients that participates of the *t-route*. Let $\mathcal{E} \subseteq \overline{\mathcal{N}}$ be a subset of clients which their elementary is required. When a *t-route* $\mathcal{L}$ is extended to a node j, the set of unreachable clients that participates of $\mathcal{L}$, must be updated as follows:

$$\mathcal{U}(\mathcal{L}') = \begin{cases} \mathcal{U}(\mathcal{L} \cup \{j\}, & \text{if } j \in \mathcal{E} \setminus \mathcal{U}(\mathcal{L}) \\ \mathcal{U}(\mathcal{L}), & \text{otherwise.} \end{cases} \tag{5.15}$$

Consequently, cycles can occur in any client $j \notin \mathcal{E}$. To seek the elementary, $\mathcal{E}$ is augmented as needed. Typically, the DSSR procedure starts with $\mathcal{E} = \emptyset$. If the path with the smaller reduced cost has a cycle in some client j, this client is added to $\mathcal{E}$. Desaulniers et al. (2008) stated that is interesting to hold the set $\mathcal{E}$ for all iterations during the column generation procedure.

Our proposed *t-routes* is solved individually for each pair vehicle-period $(v, t), \forall \in \mathcal{V}, t \in \mathcal{T}$. Then, in line 6 of Algorithm 5.2, the reduced cost must be computed as $\overline{r}_c(R(a, j)) = c_{0j} - \phi^{vt}$, considering the respective dual price of the vehicle v during period t. As aforementioned, product(inventory)-routing problems solved with column generation approaches have the delivered loads defined by the pricing problem, which increases the algorithm complexity. It augments the amount of information contained in each entry of matrix R. Then, before we proceed to the pricing problems, it is important to bound the possible label extensions as well as to-be-delivered loads, aiming to control the growth in the number labels.

A label $\mathcal{L}_i$ contained in the entry $R(a - s_i - a_{ij}, i)$ can be extended to a node $j \in \overline{\mathcal{N}}$, if and only if, the accumulated load until a predecessor node i more the delivered loads to node j is less than the vehicle's capacity that is performing the route, i.e., $o(R(a - s_i - a_{ij}, i)) + \sum_{k \in \mathcal{P}} q_{jk} \leq Q^v$. The optimal values could be assessed by solving the BKP (5.11)-(5.13) for each visited customer of every possible new label and could lead to very high processing times. Instead, it can be solved with dynamic programming.

The solution of the BKP with dynamic programming works as follows. When a label is extended to a client node $j \in \overline{\mathcal{N}}$, it must select which products and respective quantities are delivered, and compute its respective dual price. Then, in line 7 of Algorithm 5.3 the computation of the reduced cost must be rewritten as $\overline{r}_c = \overline{r}_c(R(a - s_i - a_{ij}, i)) + \overline{c}_{ij} - \sum_{k \in \mathcal{P}} (\gamma^t_k + \delta^t_{jk}) q^t_{jk}$. Only products $k \in \mathcal{P}$ with positive dual values $\left(\gamma^t_k + \delta^t_{jk}\right)$ are selected to be delivered, once the BKP aims to maximize its objective function. The selected dual prices are decreasingly ordered. The product with the larger dual price is added to the vehicle assuming one of the following possible values $q^t_{jk} = \min\{\tilde{Q}, \{0, d^t_{jk}, O^t_{jk}\}\}$. It is done to bounds the possible number of labels, once variables q_{jk} are non-negatives, and would lead to an intractable number of possible extensions. For the selected product k, the residual load is updated as $\tilde{Q} = \tilde{Q} - q^t_{jk}$. This step is repeated until no longer products can be delivered to customer j.

Let's define that each entry in matrix $R(a, j)$ contains no longer a single label, but a bucket of labels. A bucket $R(a, j)$ represents not only the cheapest *t-route* with riding time a that reaches j, but also alternative *t-routes* that ensure that all possible extensions from the plant node 0 to j are considered. The number of labels in each bucket is limited by eliminating all dominated labels.

Each label $\mathcal{L}(a, j) = (\overline{r}_c, \mathcal{U}, i, q_{ik}, o)$ reaching the client $j \in \overline{\mathcal{N}}$ with riding time a, corresponds to the following information: $\overline{r}_c$ is the accumulated reduced cost, $\mathcal{U}$ is the set of clients that compose the path, i is the predecessor node, q_{ik} is the quantity delivered for the client $i \in \mathcal{U}$ of the product $k \in \mathcal{P}$, and $o = \sum_{(i,k)\in\{i\in\mathcal{U},k\in\mathcal{P}\}}$ is the total carried load by the vehicle. Lets $q_k = \sum_{i\in\mathcal{U}}, q_j = \sum_{k\in\mathcal{P}}, q_{jk}$ be the total quantities delivered by the label $\mathcal{L}(a, j)$ related to a product $k \in \mathcal{P}$, to a client $j \in \overline{\mathcal{N}}$, and the individual quantities of product k to customer j, respectively. Every time a label is extended, a pairwise dominance rule is adopted. These dominance rules were derived from the work of Desaulniers (2010). Given two labels $\mathcal{L}^*$ and $\mathcal{L}^{'}$, that reaches j with riding time a, $\mathcal{L}^*_j$ dominates $\mathcal{L}^{'}_j$ ($\mathcal{L}^*_j \succeq \mathcal{L}^{'}_j$), if and only if, the following conditions are satisfied:

1. $o_1 \geq o_2$;
2. $\overline{c}^1_r - q^1_{jk}(\gamma_k + \delta_{jk}) \leq \overline{c}^2_r - q^2_{jk}(\gamma_k + \delta_{jk})$;
3. $\overline{c}^1_r - (q^2_k - q^1_k)(\gamma_k + \delta_{jk}) \leq \overline{c}^2_r$;
4. $\overline{c}^1_r - (q^2_k + q^2_{jk} - q^1_k)(\gamma_k + \delta_{jk}) \leq \overline{c}^2_r - q^2_{jk}(\gamma_k + \delta_{jk})$.

If a new $\mathcal{L}^{'}$ is not dominated, then it is added to its respective bucket. The dominance rule test replaces the reduced cost test in line 8 of Algorithm 5.3, and provokes that for every entry in matrix $R(a, j)$, Algorithm 5.4 would have to select the *t-routes* with negative reduced cost from the bucket.

Accordingly to Pessoa et al. (2009), the pricing subproblem of finding the *q-routes* yielding a variable with minimum reduced cost is NP-hard, but can be solved in pseudopolynomial $O(mC)$ time, where $m = |\mathcal{A}|$, and C is the vehicle capacity. Once the proposed *t-routes* relies on the same concept, we can state as follows.

The matrix $R(a, j)$ have $(n+1)\times(H+1)$ entries. Moreover, the reduced cost of each of them is accessed exactly once. For each accessed entry, the procedure takes $O(1)$ time to update the matrix. So, the total running time is $O(nH)$. Considering that the proposed pricing runs for each pair period-vehicle (t, v), the total number of entries updated is $T\times V\times(n+1)\times(H+1)$. For example, the minor (larger) proposed instance has $n = 20, T = 5, V = 7$ ($n = 50, T = 15, V =$

11), and $H = 6000$, soon, there are 4,410,735 (50,498,415) entries to be updated during every pricing execution. Updating an entry also requires the solution of a bounded knapsack problem. Whereas, the proposed pricing presented impracticable computational times. To get around this situation and allow us to study the RRMP attained bounds, the columns were heuristically priced, as shown in the next section.

5.4.2 Pricing columns heuristically

The pricing proposed in the previous section is executed for every pair of vehicle-period (v, t). It leads to elevated computational times, even for the smallest instances proposed. To offer an alternative for generating negative reduced cost columns, the following procedure is based on the refinement heuristics and perturbation operators presented in Chapter 4. First, the proposed problems $\mathrm{PI_v}$ and $\mathrm{VR_t}$ are built and solved, accordingly to the Algorithm 4.1. Next, the part of the solution about the delivery lots $\mathbf{q} = \{q_{ik}^{vt}, \forall i \in \overline{\mathcal{N}}, k \in \mathcal{P}, v \in \mathcal{V}, t \in \mathcal{T}\}$, and the T routing problems $\mathrm{VR_t}$ are retained as a delivery pattern.

The procedure PCH (Alg. 5.5) works with the obtained dual prices $(\overline{\gamma}_k^t, \overline{\delta}_{ik}^t, \overline{\pi}_i^t, \overline{\phi}^{vt})$, the delivery pattern information $\mathbf{q}$, and routing solution $\mathrm{VR_t}$. For each period, it perturbs the current vehicle routing solution vr_t with the selected perturbation operator r^κ (line 2). With vr_t perturbed by r^κ, it is optimized with a modified version of the NRS (Alg. 4.3) procedure (line 3). Every optimized route by NRS is tested if its reduced cost is negative, considering the dual prices $(\overline{\gamma}_k^t, \overline{\delta}_{ik}^t, \overline{\pi}_i^t, \overline{\phi}^{vt})$, and delivery pattern $\mathbf{q}$. The reduced cost is computed with Equations(5.9) and (5.10), and if negative, this route is added to the RRMP. In line 4, if a better routing solution was found, it is stored for the next iterations. Also, in the case that the perturbation operator is DL, the new delivery pattern is updated.

Algorithm 5.5: Pricing columns heuristically (PCH)

Data: $(\mathbf{q}, \mathrm{VR_t})$, $(\overline{\gamma}_k^t, \overline{\delta}_{ik}^t, \overline{\pi}_i^t, \overline{\phi}^{vt}), r^\kappa$

1 **forall** $t \in \mathcal{T}$ **do**
2 $\quad vr_t' \leftarrow Perturb(vr_t, r^\kappa)$;
3 $\quad vr_t'' \leftarrow NRS(vr_t', \mathbf{q}, \overline{\gamma}_k^t, \overline{\delta}_{ik}^t, \overline{\pi}_i^t, \overline{\phi}^{vt})$; //Alg. 4.3, Eq. (5.9), Eq. (5.10)
4 $\quad$ **if** $f(vr_t'') < f(vr_t)$ **then**
5 $\quad\quad vr_t \leftarrow vr_t''$;
6 $\quad\quad$ **if** $\mathbf{q}$ was modified **then** updade $\mathbf{q}$;
7 $\quad$ **end**
8 **end**

To induce the formation of columns with negative reduced costs, a small modification is realized over the perturbation operator DL, please see Section 4.4.3. When a move $m_{ik}^{tt'}$ is identified with Equations 4.6 or 4.7, it is added to the list of possible moves if, and only if, the dual prices of the period t' are bigger than the ones of period t ($\overline{\gamma}_k^{t'} + \overline{\delta}_{ik}^{t'} > \overline{\gamma}_k^t + \overline{\delta}_{ik}^t$). Here, this modification relies on the idea of maximizing the contribution of the delivery to the reduced cost of the route, as done in problem (5.11)-(5.13). After applying DL on delivery pattern $\mathbf{q}$, the routing solution vr_t is rebuilt.

5.5 Computational results and analysis

This section presents the attained bounds by the column generation approach. The adoption of the proposed *t-routes* pricing algorithm led to a prohibitive number of labels and computational times, even for the smaller instances. To get around this situation, we proposed to generate heuristically negative reduced cost columns, and to study the bounds reached.

As the variables are heuristically priced, different columns are generated during each run of the method. Due to this, the method runs 5 rounds for at most 20 minutes. The complete data are available in Tables E.1-E.3.

As shown in Section 3.4, the proposed 2COMM model offers better lower bounds than VINDX. Thus, the lower bounds (LB) reached by the Algorithm 5.1 are compared to them. This comparison adopts the linear programming gap (LP GAP), computed as shown in Equation 3.41, but considering the upper bounds (UB) as the best-known solution value found between the 2COMM, ABUILS, and the TDILS methods.

To compare, we selected within the 5 rounds the best bounds found for each instance. Because it would be closer to the attained bounds of non-heuristic pricing since non-heuristic pricing would generate the best columns guided only by the dual prices and without a random characteristic as the one present in the PCH procedure.

Figure 5.3 presents the attained linear programming gaps for RRMP (gray boxes), and 2COMM (blue boxes). It is possible to see that RRMP provided tighter bounds in 100% of the instances. Considering the LP GAP reached, RRMP (2COMM) had an average value equals to 11.14% (27.10%), worst value equals 23.38% (42.46%), and best value equals 2.47% (13.33%). Besides, the average percentage difference showed that RRMP performed 15.96% better than 2COMM. The complete data are available in Tables E.4-E.6. The columns represent the index of the instance (#), the name, best known solution (UB), lower bounds (LB) and linear programming gaps (LP GAP) for 2COMM and RRMP, and the percentage difference (Diff) between the attained LP GAP.

The proposed RRMP formulation, even with heuristically priced columns, is capable to return better lower bounds than the ones found by 2COMM. To tight these attained lower bounds, improvements can be realized with the addition of strengthening cuts, and on the proposed *t-routes* pricing algorithm, to generate high-quality columns within low computational times, instead to price them heuristically.

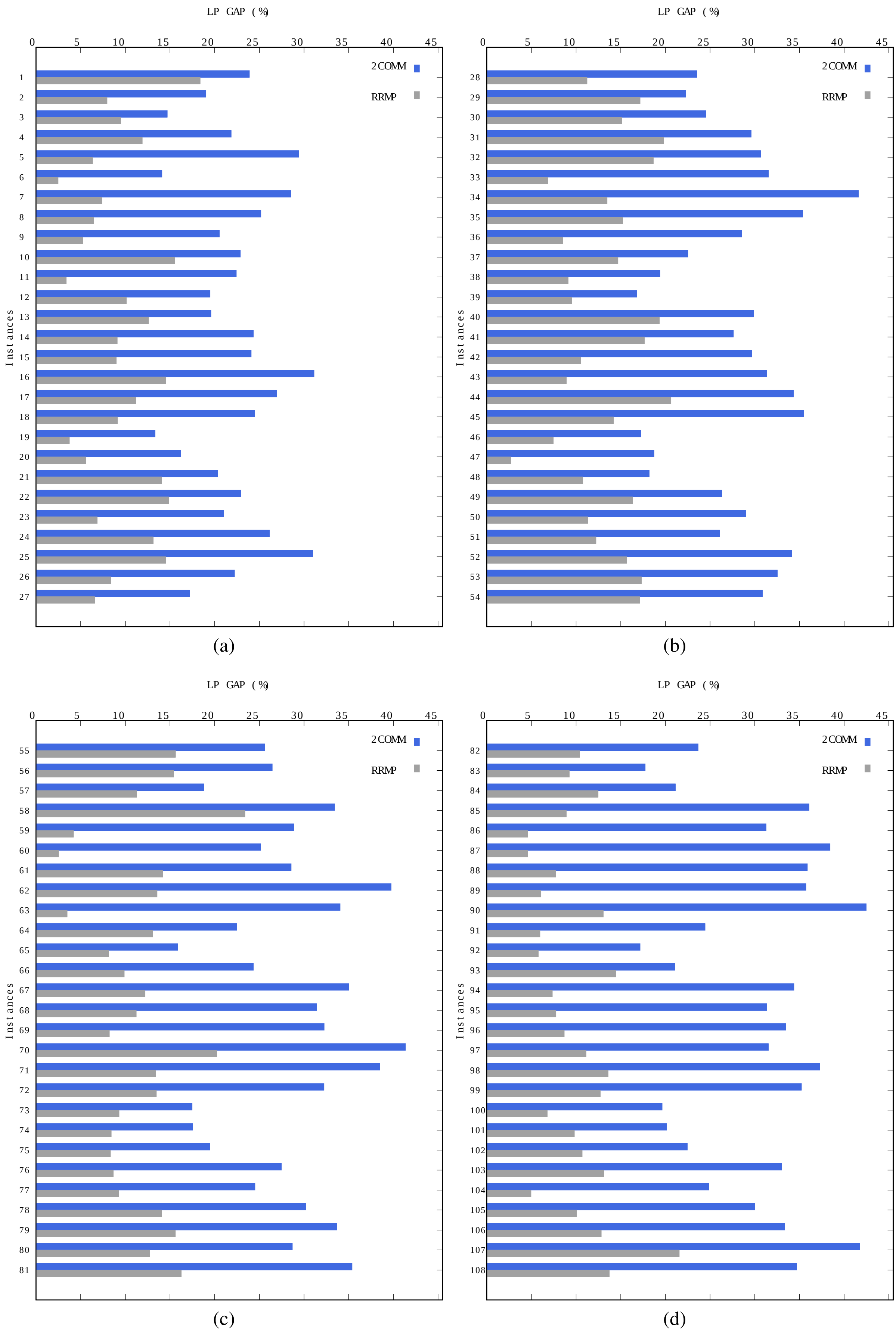


Figure 5.3 – Comparison of the LP GAP for RRMP and 2COMM.

5.6 Final remarks

The proposed RPRP has proved itself a huge challenge. Aiming to provide lower bounds to assess the quality of the attained upper bounds, this chapter focused on the development of column generation approach. A column generation is an efficient method for solving large linear problems, and provides better lower bounds than linear programming relaxation. This method splits the problem into a master problem, considering a subset of variables, and a pricing problem that generates new ones.

The chosen master problem was a compact and equivalent formulation for the RPRP, called RRMP. Besides, a concept of a pricing algorithm was proposed. The proposed pricing, named *t-routes*, relied in the same ground of the well known *q-routes* algorithm largely adopted for capacitated VRP in the literature. This pricing worked with time-indexed arcs, and every label extension solved a multi-product bounded knapsack problem, which contributed to increase the complexity of an already complex algorithm, leading to a elevated computational processing times.

To get around this situation, and to be able to study the quality of the lower bounds found by the compact formulation and the known upper bounds, the columns were generated heuristically. With a delivery pattern and routing design found by the initial solution procedure from the adaptive bottom-up method from Chapter 4, the delivery plan and the arrangement of routes were modified with the operational-level perturbation operators and optimized with the routing local searches. Every column with a negative reduced cost were added to the master problem, which is solved again.

The computational experiments showed that the proposed formulation was capable to provide lower limits tighter than the ones found by the 2COMM formulation. Future researches encompass the improvement of the proposed *t-routes* pricing subproblem making it capable to generate quality columns in reasonable computational time and effort, also, to develop cuts to strengthen the bounds.

6 Conclusions and future researches

"One is always a long way from solving a problem until one actually has the answer".

Stephen W. Hawking

In this thesis, a rich production-routing problem (RPRP) was proposed, bringing interesting features like the possibility of back-order the demand, the production of several items and to carry mixed loads using a heterogeneous fleet that must travel within a time limit. Two formulations were proposed, modeling the routing component of the problem considering vehicle-indexes or a two-commodity flow pattern. With the increase in the number of clients, products, vehicles and, periods, the problem delivered a high-level computational challenge. To provide solutions and assess their quality, hybrid, and lower bound approaches were developed to treat the RPRP.

The first three matheuristic approaches were based on the top-down hierarchical two-level decision. The methods divided the problem into two. A top tier, treating the production-inventory and distribution decisions, and a bottom one focused in the routing ones. The top tier was solved using a commercial solver, as the operational was routed and improved with heuristics independently for each period. With an iterated-local search framework, perturbations occurred changing the production and routing plans. The production plan was modified increasing or decreasing the production levels. Whereas the routing design was changed in an intuitive way to price the costs of the delivered lots is proposed to feedback the production-inventory level with information about the routing, which helps to lead to faster and improved solutions. The methods that considered the perturbations over the routing lead to the best solutions, suggesting that approaches giving first attention to the routing-distribution could achieve better results.

A bottom-up and hybrid approach was proposed to allow an analysis of a counterpoint in decision making. Still, with the problem divided into two, it focused in to modify the distribution and routing plans. The distribution plans were modified exchanging delivery lots between different periods. The perturbation operators were adaptively selected, guaranteeing that the ones with more fitness to the problem had more opportunities. These decisions were fixed in the top tier problem, which was reoptimized. The solutions found, despite needing more time than the top-down approaches, achieved better objective function values.

The last approach proposed was a column generation algorithm. It had the purpose to provide better lower bounds than the ones found by the proposed models, and to assess the quality of the best-known solutions. An equivalent formulation for the RPRP was developed, dropping out the distribution-routing variables and constraints. This model worked as a restricted master problem having the routing-distribution columns added iteratively. Column generation approaches are known to have their complexity residing in the pricing subproblems. As done in the close related studies found in the literature, the delivered quantities are defined in the pricing subproblem. This increases the complexity, once the routing component is usually solved with the shortest path problem with resource constraints, and also solves a bounded knapsack problem to define the size of the lots delivered. A pricing approach named *t-routes* is proposed. However, even with dominance policies, the number of entries (labels) became intractable. To overcome this situation, we decided to generate the columns heuristically. With a routing and distribution pattern provided by the initial solution of the matheuristic, it had its distribution plans and routing design perturbed and optimized. The columns with negative reduced costs were added to the master problem. The method was capable to reach tighter lower bounds than the ones found by 2COMM with the columns heuristically priced.

Future researches have several opportunities that would turn the problem richer and still more challenging. It could be done with the inclusion of some factors, such as allowing partial deliveries or lost sales, perishable products, adoption of periodic time windows with demands being served within a given range of periods, heterogeneous and multiple fleet, and proportional service time. This last would turn the problem non-linear, once the parameter service time becoming a variable proportional to the size of the lot. Also, the inclusion of more plants and permanent or rented distribution center, creating a multi-echelon network. The demands and travel times could be considered under uncertainty. Part of these proposals is treatable with branch-and-price approaches, strongly dependent on the pricing subproblem.

Bibliography

ABSI, N. et al. A two-phase iterative heuristic approach for the production routing problem. *Transportation Science*, INFORMS, v. 49, n. 4, p. 784–795, 2014. 11, 21

ADULYASAK, Y. et al. Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, INFORMS, v. 26, n. 1, p. 103–120, 2014. 17, 22

ADULYASAK, Y. et al. Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science*, INFORMS, v. 48, n. 1, p. 20–45, 2014. 4, 5, 10, 11, 13, 17, 22

ADULYASAK, Y. et al. Benders decomposition for production routing under demand uncertainty. *Operations Research*, INFORMS, v. 63, n. 4, p. 851–867, 2015. 17, 22

ADULYASAK, Y. et al. The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, Elsevier, v. 55, p. 141–152, 2015. 1, 2, 4, 6, 9, 20

AHUJA, R. K. *Network flows: theory, algorithms, and applications*. [S.l.]: Pearson Education, 2017. 42

AHUJA, R. K. et al. New neighborhood search structures for the capacitated minimum spanning tree problem. Sloan School of Management, Massachusetts Institute of Technology, 1998. 42

AHUJA, R. K. et al. Very large-scale neighborhood search. *International Transactions in Operational Research*, Wiley Online Library, v. 7, n. 4-5, p. 301–317, 2000. 40, 42

AIEX, R. M.; RESENDE, M. G.; RIBEIRO, C. C. Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, Springer, v. 8, n. 3, p. 343–373, 2002. 56

AIEX, R. M.; RESENDE, M. G.; RIBEIRO, C. C. Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, Springer, v. 1, n. 4, p. 355–366, 2007. 56

ALMEIDA, J. F. d. F. *Otimização do planejamento tático da cadeia de suprimentos: formulações e métodos*. Phd Thesis (PhD Thesis) — Federal University of Minas Gerais (UFMG), 2015. 1

ARCHETTI, C. et al. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, INFORMS, v. 41, n. 3, p. 382–391, 2007. 12, 15

ARCHETTI, C. et al. Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research*, Elsevier, v. 38, n. 12, p. 1731–1746, 2011. 6, 11, 14, 15, 21

ARMENTANO, V. A. et al. Tabu search with path relinking for an integrated production–distribution problem. *Computers & Operations Research*, Elsevier, v. 38, n. 8, p. 1199–1209, 2011. 13, 16, 21, 45

BALDACCI, R. et al. Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs. *Networks: An International Journal*, Wiley Online Library, v. 54, n. 4, p. 178–189, 2009. 26

BALDACCI, R. et al. New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, INFORMS, v. 59, n. 5, p. 1269–1283, 2011. 62

BARANY, I. et al. Strong formulations for multi-item capacitated lot sizing. *Management Science*, INFORMS, v. 30, n. 10, p. 1255–1261, 1984. 9

BARD, J. F.; NANANUKUL, N. Heuristics for a multiperiod inventory routing problem with production decisions. *Computers & Industrial Engineering*, Elsevier, v. 57, n. 3, p. 713–723, 2009. 7, 12, 21

BARD, J. F.; NANANUKUL, N. The integrated production–inventory–distribution–routing problem. *Journal of Scheduling*, Springer, v. 12, n. 3, p. 257–280, 2009. 10, 12, 13, 21

BARD, J. F.; NANANUKUL, N. A branch-and-price algorithm for an integrated production and inventory routing problem. *Computers & Operations Research*, Elsevier, v. 37, n. 12, p. 2202–2217, 2010. 4, 14, 16, 21

BELLMAN, R. On a routing problem. *Quarterly of applied mathematics*, v. 16, n. 1, p. 87–90, 1958. 41

BELO-FILHO, M. et al. An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. *International Journal of Production Research*, Taylor & Francis, v. 53, n. 20, p. 6040–6058, 2015. 14, 22

BENDERS, J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, Springer, v. 4, n. 1, p. 238–252, 1962. 17

BERTAZZI, L. et al. Minimizing the total cost in an integrated vendor—managed inventory system. *Journal of heuristics*, Springer, v. 11, n. 5, p. 393–419, 2005. 9, 21

BIANCHESSI, N. et al. A branch-and-cut algorithm for the Team Orienteering Problem. *International Transactions in Operational Research*, Wiley Online Library, v. 25, n. 2, p. 627–635, 2018. 26

BITRAN, G. R.; TIRUPATI, D. Hierarchical production planning. *Handbooks in operations research and management science*, Elsevier, v. 4, p. 523–568, 1993. 46

BORNDÖRFER, R. et al. A column generation approach to airline crew scheduling. In: *Operations Research Proceedings 2005*. [S.l.]: Springer, 2006. p. 343–348. 62

BOUDIA, M. et al. Combined optimization of production and distribution. In: *CD-ROM proceedings of the international conference on industrial engineering and systems management, IESM*. [S.l.: s.n.], 2005. v. 5. 10, 14, 15, 21

BOUDIA, M. et al. A reactive GRASP and path relinking for a combined production–distribution problem. *Computers & Operations Research*, Elsevier, v. 34, n. 11, p. 3402–3419, 2007. 6, 8, 10, 12, 13, 14, 15, 17, 21

BOUDIA, M. et al. Fast heuristics for a combined production planning and vehicle routing problem. *Production Planning and Control*, Taylor & Francis, v. 19, n. 2, p. 85–96, 2008. 8, 13, 21

BOUDIA, M.; PRINS, C. A memetic algorithm with dynamic population management for an integrated production–distribution problem. *European Journal of Operational Research*, Elsevier, v. 195, n. 3, p. 703–715, 2009. 10, 14, 21

BRAHIMI, N.; AOUAM, T. Multi-item production routing problem with backordering: a MILP approach. *International Journal of Production Research*, Taylor & Francis, v. 54, n. 4, p. 1076–1093, 2016. 1, 15, 22, 26, 30

BROWN, G. et al. The kellogg company optimizes production, inventory, and distribution. *Interfaces*, Informs, v. 31, n. 6, p. 1–15, 2001. 2

CALVETE, H. I. et al. Bilevel model for production–distribution planning solved by using ant colony optimization. *Computers & operations research*, Elsevier, v. 38, n. 1, p. 320–327, 2011. 14, 21

CASERTA, M.; VOSS, S. Metaheuristics: intelligent problem solving. In: *Matheuristics*. [S.l.]: Springer, 2009. p. 1–38. 35

CESELLI, A.; RIGHINI, G. A branch-and-price algorithm for the capacitated p-median problem. *Networks: An International Journal*, Wiley Online Library, v. 45, n. 3, p. 125–142, 2005. 62

ÇETINKAYA, S. et al. An integrated outbound logistics model for frito-lay: Coordinating aggregate-level production and distribution decisions. *Interfaces*, Informs, v. 39, n. 5, p. 460–475, 2009. 2

CHANDRA, P. On coordination of production and distribution decisions. 1991. 9

CHANDRA, P. A dynamic distribution model with warehouse and customer replenishment requirements. *Journal of the Operational Research Society*, JSTOR, p. 681–692, 1993. 1, 6, 9, 21

CHANDRA, P.; FISHER, M. L. Coordination of production and distribution planning. *European Journal of Operational Research*, Elsevier, v. 72, n. 3, p. 503–517, 1994. 1, 9, 21

CHEN, H.-K. et al. Production scheduling and vehicle routing with time windows for perishable food products. *Computers & operations research*, Elsevier, v. 36, n. 7, p. 2311–2319, 2009. 10, 21

CHRISTOFIDES, N. et al. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, Springer, v. 20, n. 1, p. 255–282, 1981. 65

CLARKE, G. u.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, INFORMS, v. 12, n. 4, p. 568–581, 1964. 11, 12, 13, 36

COELHO, L. C. et al. Thirty years of inventory routing. *Transportation Science*, INFORMS, v. 48, n. 1, p. 1–19, 2013. 1, 5, 17, 20

COELHO, L. C.; LAPORTE, G. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, Taylor & Francis, v. 51, n. 23-24, p. 7156–7169, 2013. 30

CONTARDO, C. et al. Reaching the elementary lower bound in the vehicle routing problem with time windows. *Networks*, Wiley Online Library, 2015. 66

CONTARDO, C.; MARTINELLI, R. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, Elsevier, v. 12, p. 129–146, 2014. 65

DANTZIG, G. B.; WOLFE, P. Decomposition principle for linear programs. *Operations research*, v. 8, n. 1, p. 101–111, 1960. 18, 62

DARVISH, M.; ARCHETTI, C.; COELHO, L. C. Trade-offs between environmental and economic performance in production and inventory-routing problems. *International Journal of Production Economics*, Elsevier, 2018. 19

DARVISH, M. et al. Minimizing Emissions in Integrated Distribution Problems. 2017. 22

DARVISH, M.; COELHO, L. C. Sequential versus integrated optimization: production, location, inventory control, and distribution. *European Journal of Operational Research*, p. –, 2018. ISSN 0377-2217. Available at: <https://www.sciencedirect.com/science/article/pii/S0377221718300493>. 48

DARVISH, M. et al. A dynamic multi-plant lot-sizing and distribution problem. *International Journal of Production Research*, Taylor & Francis, v. 54, n. 22, p. 6707–6717, 2016. 18, 22

DESAULNIERS, G. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations research*, INFORMS, v. 58, n. 1, p. 179–192, 2010. 70

DESAULNIERS, G. et al. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, INFORMS, v. 42, n. 3, p. 387–404, 2008. 69

DESAULNIERS, G.; RAKKE, J. G.; COELHO, L. C. A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, INFORMS, v. 50, n. 3, p. 1060–1076, 2016. 65

DESROCHERS, M.; SOUMIS, F. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, INFORMS, v. 23, n. 1, p. 1–13, 1989. 62

DÍAZ-MADROÑERO, M. et al. A review of tactical optimization models for integrated production and transport routing planning decisions. *Computers & Industrial Engineering*, Elsevier, v. 88, p. 518–535, 2015. 1, 5, 20

DOLAN, E. D.; MORÉ, J. J. Benchmarking optimization software with performance profiles. *Mathematical programming*, Springer, v. 91, n. 2, p. 201–213, 2002. 31, 52

DROR, M. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, INFORMS, v. 42, n. 5, p. 977–978, 1994. 65

ENGINEER, F. G. et al. A branch-price-and-cut algorithm for single-product maritime inventory routing. *Operations Research*, INFORMS, v. 60, n. 1, p. 106–122, 2012. 65

FANG, X. et al. Reducing carbon emissions in a closed-loop production routing problem with simultaneous pickups and deliveries under carbon cap-and-trade. *Sustainability*, Multidisciplinary Digital Publishing Institute, v. 9, n. 12, p. 2198, 2017. 18, 22, 34

FEILLET, D. et al. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, Wiley Online Library, v. 44, n. 3, p. 216–229, 2004. 19

FEO, T. A.; RESENDE, M. G. C. A probabilistic heuristic for a computationally dificult set covering problem. *Operations research letters*, v. 8, n. 2, p. 67–71, 1989. 13

FISHER, M. L.; JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. *Networks*, Wiley Online Library, v. 11, n. 2, p. 109–124, 1981. 16

Ford Jr., L. R. *Network flow theory*. [S.l.], 1956. 41

FORD, L. R.; FULKERSON, D. R. A suggested computation for maximal multi-commodity network flows. *Management Science*, INFORMS, v. 5, n. 1, p. 97–101, 1958. 62

FUKASAWA, R. et al. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, Springer, v. 106, n. 3, p. 491–511, 2006. 65

FUMERO, F.; VERCELLIS, C. Synchronized development of production, inventory, and distribution schedules. *Transportation science*, INFORMS, v. 33, n. 3, p. 330–340, 1999. 19, 21

GILLETT, B. E.; MILLER, L. R. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, INFORMS, v. 22, n. 2, p. 340–349, 1974. 9

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. *Operations research*, INFORMS, v. 9, n. 6, p. 849–859, 1961. 62

GLOCK, C. H. et al. The lot sizing problem: A tertiary study. *International Journal of Production Economics*, Elsevier, v. 155, p. 39–51, 2014. 5

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, Elsevier, n. 5, p. 533–549, 1986. 12

GOLDENBERG, D. E. *Genetic algorithms in search, optimization and machine learning*. [S.l.]: Addison Wesley, Reading: MA, 1989. 102

GROËR, C. et al. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, Springer, v. 2, n. 2, p. 79–101, 2010. 38

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European journal of operational research*, Elsevier, v. 130, n. 3, p. 449–467, 2001. 12, 47

HANSEN, P. et al. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, Springer, v. 5, n. 3, p. 423–454, 2017. 47

HEIN, F.; ALMEDER, C. Quantitative insights into the integrated supply vehicle routing and production planning problem. *International Journal of Production Economics*, Elsevier, v. 177, p. 66–76, 2016. 11, 22

JOLAYEMI, J. K.; OLORUNNIWO, F. O. A deterministic model for planning production quantities in a multi-plant, multi-warehouse environment with extensible capacities. *International Journal of Production Economics*, Elsevier, v. 87, n. 2, p. 99–113, 2004. 16, 21

KUMAR, R. S. et al. Multi-objective modeling of production and pollution routing problem with time window: A self-learning particle swarm optimization approach. *Computers & Industrial Engineering*, Elsevier, v. 99, p. 29–40, 2016. 14, 22

KYTÖJOKI, J. et al. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research*, Elsevier, v. 34, n. 9, p. 2743–2757, 2007. 38

LARRAIN, H.; COELHO, L. C.; CATALDO, A. A variable mip neighborhood descent algorithm for managing inventory and distribution of cash in automated teller machines. *Computers & Operations Research*, Elsevier, v. 85, p. 22–31, 2017. 19

LE, T. et al. A column generation-based heuristic algorithm for an inventory routing problem with perishable goods. *Optimization Letters*, Springer, v. 7, n. 7, p. 1481–1502, 2013. 62

LEE, H. L.; NAHMIAS, S. Single-product, single-location models. *Handbooks in operations research and management science*, Elsevier, v. 4, p. 3–55, 1993. 9

LEI, L. et al. On the integrated production, inventory, and distribution routing problem. *IIE Transactions*, Taylor & Francis, v. 38, n. 11, p. 955–970, 2006. 10, 21

LEUNG, J. M. et al. Facets and algorithms for capacitated lot sizing. *Mathematical programming*, Springer, v. 45, n. 1, p. 331–359, 1989. 9

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, INFORMS, v. 21, n. 2, p. 498–516, 1973. 10

LIPOWSKI, A.; LIPOWSKA, D. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 391, n. 6, p. 2193–2196, 2012. 102

LÓPEZ-IBÁÑEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, Elsevier, v. 3, p. 43–58, 2016. 50

LORENA, L. A.; SENNE, E. L. A column generation approach to capacitated p-median problems. *Computers & Operations Research*, Elsevier, v. 31, n. 6, p. 863–876, 2004. 62

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2003. p. 320–353. 35

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search: Framework and applications. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2019. p. 129–168. 42, 46

LYSGAARD, J. Clarke & Wright's Savings Algorithm. *Department of Management Science and Logistics, The Aarhus School of Business*, 1997. 36

MELO, R. A.; WOLSEY, L. A. MIP formulations and heuristics for two-level production-transportation problems. *Computers & Operations Research*, Elsevier, v. 39, n. 11, p. 2776–2786, 2012. 15, 21

METTERS, R. D. Interdependent transportation and production activity at the United States postal service. *Journal of the Operational Research Society*, Springer, v. 47, n. 1, p. 27–37, 1996. 16, 21

MILLER, C. E.; TUCKER, A. W.; ZEMLIN, R. A. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, ACM, v. 7, n. 4, p. 326–329, 1960. 17

MIRANDA, P. L. et al. Optimization model for a production, inventory, distribution and routing problem in small furniture companies. *TOP*, Springer, v. 26, n. 1, p. 30–67, 2018. 11, 12, 22

MOSTAFA, N. A.; ELTAWIL, A. B. The production-inventory-distribution-routing problem: An integrated formulation and solution framework. In: IEEE. *Industrial Engineering and Operations Management (IEOM), 2015 International Conference on*. [S.l.], 2015. p. 1–10. 2, 20

NEVES-MOREIRA, F. et al. Solving a large multi-product production-routing problem with delivery time windows. *Omega*, Elsevier, v. 86, p. 154–172, 2019. 16, 22

OR, I. TRAVELING SALESMAN TYPE COMBINATORIAL PROBLEMS AND THEIR RELATION TO THE LOGISTICS OF REGIONAL BLOOD BANKING. 1977. 10

PARK, Y. An integrated approach for production and distribution planning in supply chain management. *International Journal of Production Research*, Taylor & Francis, v. 43, n. 6, p. 1205–1224, 2005. 9, 21

PECIN, D. et al. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, Springer, v. 9, n. 1, p. 61–100, 2017. 62, 65

PENNA, P. H. V.; SUBRAMANIAN, A.; OCHI, L. S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, Springer, v. 19, n. 2, p. 201–232, 2013. 38, 39

PESSOA, A. et al. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks*, Wiley Online Library, v. 54, n. 4, p. 167–177, 2009. 62, 65, 70

PIEWTHONGNGAM, K. et al. An interactive approach to optimize production–distribution planning for an integrated feed swinecompany. *International Journal of Production Economics*, Elsevier, v. 142, n. 2, p. 290–301, 2013. 10, 21

PISINGER, D.; ROPKE, S. Large neighborhood search. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). *Handbook of metaheuristics*. Springer New York Dordrecht Heidelberg London: Springer, 2010, (International Series in Operations Research & Management Science, 146). p. 399–420. 40

POCHET, Y.; WOLSEY, L. A. *Production planning by mixed integer programming*. [S.l.]: Springer Science & Business Media, 2006. 15, 47

QIU, Y. et al. Production routing problems with reverse logistics and remanufacturing. *Transportation Research Part E: Logistics and Transportation Review*, Elsevier, v. 111, p. 87–100, 2018. 18, 22, 34

QIU, Y. et al. A branch-and-price algorithm for production routing problems with carbon cap-and-trade. *Omega*, Elsevier, v. 68, p. 49–61, 2017. 18, 22, 34, 65

QIU, Y. et al. Formulations and branch-and-cut algorithms for production routing problems with time windows. *Transportmetrica A: Transport Science*, Taylor & Francis, p. 1–22, 2018. 17, 22

QIU, Y. et al. A variable neighborhood search heuristic algorithm for production routing problems. *Applied Soft Computing*, Elsevier, v. 66, p. 311–318, 2018. ix, x, 11, 12, 22, 35, 44, 46, 51, 52, 103

REIMANN, M. et al. Joint optimization of production planning and vehicle routing problems: A review of existing strategies. *Pesquisa Operacional*, SciELO Brasil, v. 34, n. 2, p. 189–214, 2014. 20

REIS, A. F. d. S. *O problema do transporte escolar rural: uma abordagem Column-and-cut para o problema de roteamento de veículos capacitado*. Master's Thesis (Master's Thesis) — Federal University of Minas Gerais (UFMG), 2015. 65

REYES, A.; RIBEIRO, C. C. Extending time-to-target plots to multiple instances. *International Transactions in Operational Research*, Wiley Online Library, v. 25, n. 5, p. 1515–1536, 2018. 56

RIGHINI, G.; SALANI, M. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, Wiley Online Library, v. 51, n. 3, p. 155–170, 2008. 65, 69

ROPKE, S.; PISINGER, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, Informs, v. 40, n. 4, p. 455–472, 2006. 13

ROSENCRANTZ, D.; STEARNS, R.; LEWIS, P. Approximate algorithms for the tsp. In: *Proceedings 15th IEEE Symposium on Switching and Automata Theory*. [S.l.: s.n.], 1974. p. 33–42. 9

RUOKOKOSKI, M. et al. Efficient formulations and a branch-and-cut algorithm for a production-routing problem. *GERAD Technical Report G-2010-66*, HEC, 2010. 17, 21

RUSSELL, R. A. Mathematical programming heuristics for the production routing problem. *International Journal of Production Economics*, Elsevier, v. 193, p. 40–49, 2017. 11, 22

SENOUSSI, A. et al. Modeling and solving a one-supplier multi-vehicle production-inventory-distribution problem with clustered retailers. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 85, n. 5-8, p. 971–989, 2016. 18, 22

SHAW, P. Using constraint programming and local search methods to solve vehicle routing problems. In: SPRINGER. *International conference on principles and practice of constraint programming*. [S.l.], 1998. p. 417–431. 40

SHIGUEMOTO, A. L.; ARMENTANO, V. A. A tabu search procedure for coordinating production, inventory and distribution routing problems. *International Transactions in Operational Research*, Wiley Online Library, v. 17, n. 2, p. 179–195, 2010. 13, 21

SILVA, G. P.; REIS, A. F. d. S. A study of different metaheuristics to solve the urban transit crew scheduling problem. *Journal of Transport Literature*, SciELO Brasil, v. 8, n. 4, p. 227–251, 2014. 42

SIMCHI-LEVI, D. et al. *The Logic of Logistics: Algorithms, and Applications for Logistics and Supply Chain Management, Theory*. [S.l.]: Springer, 2014. (Springer Series in Operations Research and Financial Engineering). 1

SOLYALI, O.; SÜRAL, H. A relaxation based solution approach for the inventory control and vehicle routing problem in vendor managed systems. In: *Modeling, computation and Optimization*. [S.l.]: World Scientific, 2009. p. 171–189. 19, 21

SOLYALI, O.; SÜRAL, H. A multi-phase heuristic for the production routing problem. *Computers & Operations Research*, Elsevier, v. 87, p. 114–124, 2017. 11, 22

SOUZA, M. J. et al. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, Elsevier, v. 207, n. 2, p. 1041–1051, 2010. 38

THOMPSON, P. M.; ORLIN, J. B. The theory of cyclic transfers. In: *OPERATIONS RESEARCH CENTER WORKING PAPER, MIT*. [S.l.: s.n.], 1989. 41

TOTH, P.; VIGO, D. *Vehicle routing: problems, methods, and applications*. [S.l.]: SIAM, 2014. 5

Van Buer, M. G. et al. Solving the medium newspaper production/distribution problem. *European Journal of Operational Research*, Elsevier, v. 115, n. 2, p. 237–253, 1999. 12, 21

VANCE, P. H. Branch-and-price algorithms for the one-dimensional cutting stock problem. *Computational optimization and applications*, Springer, v. 9, n. 3, p. 211–228, 1998. 62

VANCE, P. H. et al. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational optimization and applications*, Springer, v. 3, n. 2, p. 111–130, 1994. 62

WAGNER, H. M.; WHITIN, T. M. Dynamic version of the economic lot size model. *Management science*, INFORMS, v. 5, n. 1, p. 89–96, 1958. 10, 13, 14

WATANABE, H. et al. Solution Methods for the Integrated Production Routing Problem. *Salesian Journal on Information Systems*, Directory of Open Access Journals, v. 1, n. 19, p. 53–61, 2017. 16, 22

Appendix

APPENDIX A – Generating and reading RPRP instances

A.1 C++ code to generate instances

```
#include <iostream>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <string.h>
#include <random>
#include <algorithm>

using namespace std;

typedef struct{
    char type; //tipo, (u)niforme
    int n;  //numero de clientes
    int P;  //numero de produtos
    int T;  //numero de periodos;
    int V;  //numero de veiculos

    //informacoes sobre a planta produtiva
    std::vector<int>  C;//capacidade de produção de cada linha de cada item
    std::vector<double> l;              //custo de setup/ativacao de cada item em cada linha
    std::vector<double> u;              //custo unitário de produção de cada item em cada lkinha;

    //informacoes dos clientes
    std::vector<std::vector<std::vector<int> > > d;    //demanda periodica de cada item por cada cliente
    std::vector<std::vector<int> > a; //tempo de viagem de i para j
    std::vector<int> s;              //tempo de servico de i
    std::vector<std::vector<double> > B;              //custo de atrasar o produto p para o cliente i

    //informacoes da frota
    std::vector<int> Q; //capacidade maxima de transporte de determinado veiculo
    std::vector<int> e; //custo de setup/ativação de cada veiculo
    int H;              //horizonte de tempo para entrega

    //informacoes gerais
    std::vector<std::vector<double> > c; //custo de transporte de i para j
    std::vector<double> x;               //coordenada x do no i
    std::vector<double> y;               //coordenada y do no i
    std::vector<std::vector<double> > h; //custo de estocagem do produto p no no i
    std::vector<std::vector<int> > U;   //limite superior de estocagem de cada item na planta produtiva
    std::vector<std::vector<int> > I0;   //inventario inicial em cada no
} DATA;
void help();
void vSort(std::vector<int> &V);
void print(DATA d);
void generate(DATA &d);
```

```
void help(){
    std::cout << std::endl << std::endl << "exec [data file] \n " << std::endl;
    exit(1);
}
void vSort(std::vector<int> &V) {
        for (int fixo = 0; fixo < V.size() - 1; fixo++) {
                int menor = fixo;
                for (int i = menor + 1; i < V.size(); i++) if (V[i] < V[menor]) menor = i;
                if (menor != fixo) {
                        int sV = V[fixo];
                        V[fixo] = V[menor];
                        V[menor] = sV;
                }
        }
}
void print(DATA d){
    ofstream arq;

    string n = std::to_string(d.n);
    string T = std::to_string(d.T);
    string P = std::to_string(d.P);
    string V = std::to_string(d.V);

    string name = "P" + n + "n" + P + "p" + T + "t" + V + "v" + ".dat";
    string outputfile = "..." + name; //"..." equivale ao caminho para salvar os arquivos

    //convertendo string para char
    char *cName = new char[outputfile.length()+1];
    memcpy(cName, outputfile.c_str(), outputfile.length() + 1);
    arq.open(cName);

    if (!arq.is_open()) help();

    arq << d.n;
    arq << "\n" << d.T;
    arq << "\n" << d.P;
    arq << "\n" << d.V;
    arq << "\n" << d.H;

    for (int v = 1; v <= d.V; v++)
        arq << "\n" << d.Q[v] << "\t" << d.e[v];

    for (int i = 0; i <= d.n; i++){
        arq << "\n" << fixed << setprecision(1) << d.x[i] << "\t" << d.y[i];
        if (i > 0){
            arq << "\t" << d.s[i];
            for (int p = 1; p <= d.P; p++)
                arq << "\t" << d.I0[p][i] << "\t" << d.h[p][i] << "\t" << d.U[p][i] << "\t" << d.B[p][i];
        }else if (i == 0){
            for (int p = 1; p <= d.P; p++)
                arq << "\t" << d.I0[p][i] << "\t" << d.h[p][i] << "\t" << d.U[p][i] << "\t" << d.C[p] << "\t" << d.l[p]<<
        }
    }

    for (int p = 1; p <= d.P; p++){
        for (int i = 1; i <= d.n; i++){
            arq << "\n";
            for (int t = 1; t <= d.T; t++) arq << d.d[t][p][i] << "\t";
        }
```

```
    }
    arq.close();
}
void generate(DATA &d){
    std::random_device rd;  //Will be used to obtain a seed for the random number engine
    std::mt19937_64 gen(rd()); //Standard mersenne_twister_engine seeded with rd()

    //gerando as coordenadas dos pontos
    std::uniform_real_distribution<> pos(0,1000); //coelho
    //std::normal_distribution<double> pos(0.0,100.0);
    d.x = std::vector<double> (d.n+2,0.0);  //coordenada x do no i
    d.y = std::vector<double> (d.n+2,0.0);
    //reservar as posições 0 e n+1 para os nós artificiais
    for (int i = 0; i <= d.n; i++){
        d.x[i] = pos(gen);
        d.y[i] = pos(gen);
        if (i == 0){
            d.x[d.n+1] = d.x[0];
            d.y[d.n+1] = d.y[0];
        }
        //std::cout << "\ni: " << i << " -> x: " << d.x[i] << " -> y: " << d.y[i];
    }

    std::uniform_int_distribution<> demLow(5,25);
    std::uniform_int_distribution<> demHigh(30,55);
    std::uniform_int_distribution<> LowHigh(0,1);
    d.d = std::vector<std::vector<std::vector<int> > >
    (d.T + 1,std::vector<std::vector <int> > (d.P + 1, std::vector<int>(d.n + 1, 0)));
    int demandTotal = 0;
    for (int t = 1; t <= d.T; t++){
        //std::cout << "\n\tt = " << t;
        for (int p = 1; p <= d.P; p++){
            //std::cout << "\n\t\tk = " << p;
            for (int i = 1; i <= d.n; i++){
                if (LowHigh(gen) == 0)
                    d.d[t][p][i] = demLow(gen);
                else if (LowHigh(gen) == 1)
                    d.d[t][p][i] = demHigh(gen);

                demandTotal += d.d[t][p][i];
                //std::cout << "\ti = " << i << " -> d: " << d.d[t][p][i];
            }
        }
    }

    //custos de estocagem
    std::uniform_int_distribution<> h0(1,5);
    std::uniform_int_distribution<> hi(6,10);
    d.h = std::vector<std::vector<double> > (d.P + 1, std::vector<double> (d.n+1,0.0));
    for (int p = 1; p <= d.P; p++){
        for (int i = 0; i <= d.n; i++){
            if (i > 0) d.h[p][i] = hi(gen);
            else d.h[p][i] = h0(gen);
        }
    }

    std::uniform_int_distribution<> backorder(8,12);
    d.B = std::vector<std::vector<double> > (d.P + 1, std::vector<double> (d.n+1,0.0));
    for (int p = 1; p <= d.P; p++)
        for (int i = 1; i <= d.n; i++)
```

```
            d.B[p][i] = (d.h[p][i])*backorder(gen);

    std::uniform_int_distribution<> holdCapacity(140,190);
    std::uniform_int_distribution<> g(2,4);
    d.U = std::vector<std::vector<int> > (d.P + 1, std::vector<int> (d.n + 1, 0));
    for (int i = 0; i <= d.n; i++)
        for (int p = 1; p <= d.P; p++) d.U[p][i] = g(gen)*holdCapacity(gen);

    //as capacidades de produção por produto
    d.C = std::vector<int> (d.P + 1, 0);
    d.l = std::vector<double> (d.P + 1,0); //custo de preparação de produção (setup)
    d.u = std::vector<double> (d.P + 1,0.0); //custo de producao unitario
    std::uniform_int_distribution<> prodCost(2,8);
    std::uniform_int_distribution<> prodCapacity(50,140);
    for(int p = 1; p <= d.P; p++) {
        d.C[p] = d.n*prodCapacity(gen); //leandro coelho mmirp
        d.u[p] =  prodCost(gen);
        d.l[p] = 10000*d.u[p];
    }

    //definindo o I0 igual a demanda do primeiro período para cada cliente
    d.I0 = std::vector<std::vector<int> > (d.P + 1, std::vector<int> (d.n + 1,0));
    std::uniform_int_distribution<> i0(100,150);  //coelho
    for (int p = 1; p <= d.P; p++){
        d.I0[p][0] = i0(gen); //coelho
        for (int i = 1; i <= d.n; i++){
            d.I0[p][i] = d.d[1][p][i];
        }
    }

    d.Q = std::vector<int> (d.V + 1, 0);
    d.e = std::vector<int> (d.V + 1, 0);
    std::uniform_int_distribution<>
        maxVehicleLoad(0.8*2*floor(demandTotal/(d.T*d.V)),2*floor(demandTotal/(d.T*d.V)));
    std::uniform_int_distribution<> setupVehicle(500,1000);
    for (int v = 1; v <= d.V; v++){
        d.Q[v] = maxVehicleLoad(gen);
        d.e[v] = setupVehicle(gen);
    }
    //ordenando os custos e capacidades
    vSort(d.Q);
    vSort(d.e);

    d.H = 6000;
    d.s = std::vector<int> (d.n + 1, 50);

    //print(d,type);
}
int main(){
    //Fortes
    //n = 20, 30, 40, 50
    //P = 3,4,5
    //T = 5,10,15
    //V = 7,9,11
    int inst = 0;

    for (int c = 2; c <= 5; c++){
        DATA d;
        d.n = 10*c;
        for (int p = 3; p <= 5; p++){
```

```
            d.P = p;
            for (int t = 5; t <= 15; t+=5){
                d.T = t;
                for (int v = 7; v <= 11; v+=2){
                    d.V = v;
                    generate(d);
                    inst++;
                }
                }
        }
        }
    std::cout << std::endl << inst;
    //getchar();
    return 0;
}
```

A.2 C++ code to read instances

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <cmath>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <algorithm>
#include <time.h>
#include <string>
#include <string.h>
#include <deque>

//instance data
typedef struct{
    int n;      //numero de clientes
    int P;      //numeto de produtos
    int T;      //numero de periodos;
    int V;      //numero de veiculos

    //informacoes sobre a planta produtiva
    std::vector<int> C;     //capacidade de produção de cada fabrica de cada item
    std::vector<double> l;  //custo de setup/ativacao de cada item em cada fabrica
    std::vector<double> u;  //custo unitário de produção de cada item em cada fabrica;

    //informacoes dos clientes
    std::vector<std::vector<std::vector<int> > >d;    //demanda periodica de cada item por cada cliente
    std::vector<std::vector<double> > a;    //tempo de viagem de i para j
    std::vector<int> s;                 //tempo de servico de i
    std::vector<std::vector<double> > B;//custo de atrasar o produto p para o cliente i

    //informacoes da frota
    std::vector<int> Q;                 //capacidade maxima de transporte de determinado veiculo
    std::vector<int> e;                 //custo de setup/ativação de cada veiculo
    int H;                              //horizonte de tempo para entrega, de 12 horas (6h - 18h)

    //informacoes gerais
    std::vector<std::vector<double> > c; //custo de transporte de i para j
    std::vector<double> x;               //coordenada x do no i
```

```
    std::vector<double> y;               //coordenada y do no i
    std::vector<std::vector<double> > h; //custo de estocagem do produto p no no i
    std::vector<std::vector<int> > U;    //limite superior de estocagem no item de cada planta
    std::vector<std::vector<int> > I0;   //inventario inicial em cada no, deve satisfazer a demanda do periodo

    string name;
    string pathIn;
}data;

//Erro de leitura de dados
void help(){
  std::cout << std::endl << std::endl << "exec [data file] \n " << std::endl;
  exit(1);
}
void read(data &d){
    string sPath = d.pathIn + d.name;
    //std::cout << "\n\tPath = " << sPath;
    char *cPath = new char[sPath.length()+1];
    memcpy(cPath, sPath.c_str(), sPath.length() + 1);
    //std::cout << "\n\tPath = " << cPath;
    //fim conversao string para char

    //abertura do arquivo de dados
    ifstream arq(cPath);
    if (!arq.is_open()) help();

    //iniciando leitura dos parametros
    arq >> d.n;      //number of clients
    arq >> d.T;      //number of periods
    arq >> d.P;      //number of commodities
    arq >> d.V;      //number of vehicles
    arq >> d.H;      //maximum time route size

    //std::cout << "\nV = " << d.V << "\nP = " << d.P;

    //reservando posicoes 0 e n+1 para nos artificiais
    d.x = std::vector<double> (d.n + 2,0.0);  //coordenada x do no i
    d.y = std::vector<double> (d.n + 2,0.0);  //coordenada x do no i
    d.c = std::vector<std::vector<double> > (d.n + 2, std::vector<double> (d.n + 2, 0.0));  //custo (i,j)
    d.a = std::vector<std::vector<double> > (d.n + 2, std::vector<double> (d.n + 2, 0.0));  //tempo (i,j)
    d.s = std::vector<int> (d.n + 1, 10); //tempo de atendimento

    //estoque inicial
    d.I0 = std::vector<std::vector<int> > (d.P + 1, std::vector<int> (d.n + 1,0));

    //demanda periodica e acumulada
    d.d = std::vector<std::vector<std::vector<int> > >
            (d.T + 1, std::vector<std::vector <int> > (d.P + 1, std::vector<int>(d.n + 1, 0)));

    //informacoes sobre producao e fabrica
    d.u = std::vector<double> (d.P + 1,0); //custo de producao unitario
    d.C = std::vector<int> (d.P + 1,0);       //capacidade de cada linha de produção
    d.l = std::vector<double> (d.P + 1,0); //custo de preparação de produção (setup)

    //informacoes sobre os veiculos
    d.Q = std::vector<int> (d.V + 1, 0);        //carga maxima carregada
    d.e = std::vector<int> (d.V + 1, 0);        //custo de ativação

    //custos e limites de estocagem e custo de atraso
    d.h = std::vector<std::vector<double> > (d.P + 1, std::vector<double> (d.n + 1,0.0));
```

```
    d.U = std::vector<std::vector<int> > (d.P + 1, std::vector<int> (d.n + 1, 0));
    d.B = std::vector<std::vector<double> > (d.P + 1, std::vector<double> (d.n + 1,0.0));

    //informacoes sobre os veiculos
    for (int v = 1; v <= d.V; v++){
        arq >> d.Q[v];
        arq >> d.e[v];
    }

    //lendo informacoes dos nos clientes e plantas
    for (int i = 0; i <= d.n; i++){
        arq >> d.x[i];
        arq >> d.y[i];
        if (i == 0){
            for (int p = 1; p <= d.P; p++){
                arq >> d.IO[p][i];
                arq >> d.h[p][i];
                arq >> d.U[p][i];
                arq >> d.C[p];
                arq >> d.l[p];
                arq >> d.u[p];
            }
            d.x[d.n+1] = d.x[i];
            d.y[d.n+1] = d.y[i];
        }else if (i > 0){
            arq >> d.s[i];
            for (int p = 1; p <= d.P; p++){
                arq >> d.IO[p][i];
                arq >> d.h[p][i];
                arq >> d.U[p][i];
                arq >> d.B[p][i];
            }
        }
    }

    //demandas de cada cliente por cada produto em cada periodo
    for (int p = 1; p <= d.P; p++)
        for (int i = 1; i <= d.n; i++)
            for (int t = 1; t <= d.T; t++) arq >> d.d[t][p][i];

    for (int i = 0; i <= d.n + 1; i++)
        for (int j = 0; j <= d.n + 1; j++)
            d.c[i][j] = round(sqrt((d.x[i] - d.x[j])*(d.x[i] - d.x[j]) + (d.y[i] - d.y[j])*(d.y[i] - d.y[j])));

    d.a = std::vector<std::vector<double> > (d.c);
    arq.close();
}

int main(int argc, char *argv[]){
    data d;
    d.name = argv[1];
    d.pathIn = "...";       //caminho onde estao as instancias
    read(d);
    return 0;
}
```

APPENDIX B – Data from Chapter 3

B.1 Instance labeling

Table B.1 – Instance labels

#	INSTANCE	#	INSTANCE	#	INSTANCE
1	P20n3p5t7v	*37*	P30n4p5t7v	*73*	P40n5p5t7v
2	P20n3p5t9v	*38*	P30n4p5t9v	*74*	P40n5p5t9v
3	P20n3p5t11v	*39*	P30n4p5t11v	*75*	P40n5p5t11v
4	P20n3p10t7v	*40*	P30n4p10t7v	*76*	P40n5p10t7v
5	P20n3p10t9v	*41*	P30n4p10t9v	*77*	P40n5p10t9v
6	P20n3p10t11v	*42*	P30n4p10t11v	*78*	P40n5p10t11v
7	P20n3p15t7v	*43*	P30n4p15t7v	*79*	P40n5p15t7v
8	P20n3p15t9v	*44*	P30n4p15t9v	*80*	P40n5p15t9v
9	P20n3p15t11v	*45*	P30n4p15t11v	*81*	P40n5p15t11v
10	P20n4p5t7v	*46*	P30n5p5t7v	*82*	P50n3p5t7v
11	P20n4p5t9v	*47*	P30n5p5t9v	*83*	P50n3p5t9v
12	P20n4p5t11v	*48*	P30n5p5t11v	*84*	P50n3p5t11v
13	P20n4p10t7v	*49*	P30n5p10t7v	*85*	P50n3p10t7v
14	P20n4p10t9v	*50*	P30n5p10t9v	*86*	P50n3p10t9v
15	P20n4p10t11v	*51*	P30n5p10t11v	*87*	P50n3p10t11v
16	P20n4p15t7v	*52*	P30n5p15t7v	*88*	P50n3p15t7v
17	P20n4p15t9v	*53*	P30n5p15t9v	*89*	P50n3p15t9v
18	P20n4p15t11v	*54*	P30n5p15t11v	*90*	P50n3p15t11v
19	P20n5p5t7v	*55*	P40n3p5t7v	*91*	P50n4p5t7v
20	P20n5p5t9v	*56*	P40n3p5t9v	*92*	P50n4p5t9v
21	P20n5p5t11v	*57*	P40n3p5t11v	*93*	P50n4p5t11v
22	P20n5p10t7v	*58*	P40n3p10t7v	*94*	P50n4p10t7v
23	P20n5p10t9v	*59*	P40n3p10t9v	*95*	P50n4p10t9v
24	P20n5p10t11v	*60*	P40n3p10t11v	*96*	P50n4p10t11v
25	P20n5p15t7v	*61*	P40n3p15t7v	*97*	P50n4p15t7v
26	P20n5p15t9v	*62*	P40n3p15t9v	*98*	P50n4p15t9v
27	P20n5p15t11v	*63*	P40n3p15t11v	*99*	P50n4p15t11v
28	P30n3p5t7v	*64*	P40n4p5t7v	*100*	P50n5p5t7v
29	P30n3p5t9v	*65*	P40n4p5t9v	*101*	P50n5p5t9v
30	P30n3p5t11v	*66*	P40n4p5t11v	*102*	P50n5p5t11v
31	P30n3p10t7v	*67*	P40n4p10t7v	*103*	P50n5p10t7v
32	P30n3p10t9v	*68*	P40n4p10t9v	*104*	P50n5p10t9v
33	P30n3p10t11v	*69*	P40n4p10t11v	*105*	P50n5p10t11v
34	P30n3p15t7v	*70*	P40n4p15t7v	*106*	P50n5p15t7v
35	P30n3p15t9v	*71*	P40n4p15t9v	*107*	P50n5p15t9v
36	P30n3p15t11v	*72*	P40n4p15t11v	*108*	P50n5p15t11v

B.2 Data from Section 3.4

Table B.2 – VINDX formulation after 6 hours running.

INSTANCE	VINDX					
	UB	LB	LP GAP (%)	GAP(%)	B&B Nodes	Nodes left
P20n3p5t7v	393.771	295.889	24,86	3,18	32878	22032
P20n3p5t9v	223.426	161.474	27,73	20,62	21769	16235
P20n3p5t11v	292.348	244.798	16,26	11,04	18135	10874
P20n3p10t7v	528.468	379.945	28,10	15,47	2747	617
P20n3p10t9v	482.892	304.049	37,04	33,13	1762	0
P20n3p10t11v	720.290	588.051	18,36	14,98	1211	1076
P20n3p15t7v	857.569	534.081	37,72	31,30	418	413
P20n3p15t9v	1.191.080	852.533	28,42	21,18	39	40
P20n3p15t11v	945.068	707.484	25,14	19,15	0	1
P20n4p5t7v	375.892	285.778	23,97	4,25	32807	24075
P20n4p5t9v	347.962	264.463	24,00	19,29	14073	10800
P20n4p5t11v	289.224	215.791	25,39	16,67	8792	6674
P20n4p10t7v	829.422	627.574	24,34	13,49	2531	2284
P20n4p10t9v	707.403	497.865	29,62	23,56	1945	1915
P20n4p10t11v	959.711	684.246	28,70	20,32	60	61
P20n4p15t7v	1.156.740	755.195	34,71	28,51	76	77
P20n4p15t9v	1.247.360	785.504	37,03	31,61	0	1
P20n4p15t11v	1.312.450	927.767	29,31	24,96	0	1
P20n5p5t7v	340.144	294.378	13,45	2,49	39251	24981
P20n5p5t9v	394.751	324.860	17,71	6,66	15153	9681
P20n5p5t11v	494.888	383.946	22,42	8,37	6658	2996
P20n5p10t7v	990.685	723.308	26,99	17,32	2893	2574
P20n5p10t9v	821.995	606.059	26,27	18,18	472	456
P20n5p10t11v	770.823	527.693	31,54	28,42	92	93
P20n5p15t7v	1.387.800	868.433	37,42	31,54	46	39
P20n5p15t9v	1.577.950	1.150.470	27,09	21,81	0	1
P20n5p15t11v	1.851.200	1.441.040	22,16	18,43	0	1
P30n3p5t7v	245.963	162.438	33,96	27,36	7081	3358
P30n3p5t9v	431.912	301.409	30,22	14,66	3218	2208
P30n3p5t11v	384.292	274.973	28,45	25,31	3493	2935
P30n3p10t7v	1.201.720	481.578	59,93	54,20	40	41
P30n3p10t9v	715.053	460.111	35,65	29,03	0	1
P30n3p10t11v	709.858	444.824	37,34	33,74	0	1
P30n3p15t7v	2.442.140	510.576	79,09	76,27	0	1
P30n3p15t9v	2.145.500	655.632	69,44	65,42	0	1
P30n3p15t11v	1.537.150	840.268	45,34	38,59	0	1

Table B.3 – VINDX formulation after 6 hours running.

INSTANCE	VINDX					
	UB	LB	LP GAP (%)	GAP(%)	B&B Nodes	Nodes left
P30n4p5t7v	434.697	316.854	27,11	11,72	4486	1099
P30n4p5t9v	558.914	431.259	22,84	9,38	4189	799
P30n4p5t11v	549.718	410.156	25,39	18,75	2325	2121
P30n4p10t7v	750.017	481.951	35,74	23,43	37	38
P30n4p10t9v	815.512	530.540	34,94	30,82	0	1
P30n4p10t11v	896.933	575.316	35,86	31,55	0	1
P30n4p15t7v	3.350.600	803.950	76,01	72,51	0	1
P30n4p15t9v	1.330.540	772.189	41,96	36,65	0	1
P30n4p15t11v	1.342.350	824.193	38,60	34,68	0	1
P30n5p5t7v	456.627	341.105	25,30	19,72	3414	3079
P30n5p5t9v	426.722	320.281	24,94	22,92	160	134
P30n5p5t11v	504.405	392.082	22,27	21,11	186	154
P30n5p10t7v	2.759.440	755.767	72,61	69,46	0	1
P30n5p10t9v	2.649.030	675.739	74,49	72,30	0	1
P30n5p10t11v	1.115.330	755.193	32,29	26,10	0	1
P30n5p15t7v	1.507.690	875.531	41,93	37,25	0	1
P30n5p15t9v	2.546.510	1.220.080	52,09	48,23	0	1
P30n5p15t11v	4.297.030	1.152.160	73,19	71,93	0	1
P40n3p5t7v	505.832	329.084	34,94	33,43	1708	1336
P40n3p5t9v	296.602	182.776	38,38	31,71	0	1
P40n3p5t11v	408.951	307.084	24,91	17,42	0	1
P40n3p10t7v	1.836.600	484.415	73,62	70,03	0	1
P40n3p10t9v	1.851.760	704.946	61,93	60,23	0	1
P40n3p10t11v	1.264.500	685.990	45,75	42,92	0	1
P40n3p15t7v	3.140.920	872.886	72,21	70,59	0	1
P40n3p15t9v	1.728.290	466.618	73,00	68,40	0	1
P40n3p15t11v	1.895.240	682.968	63,96	60,93	0	1
P40n4p5t7v	453.213	245.731	45,78	41,09	218	212
P40n4p5t9v	430.170	319.420	25,75	22,08	18	17
P40n4p5t11v	515.293	356.201	30,87	23,80	0	1
P40n4p10t7v	-	515.181	-	-	-	-
P40n4p10t9v	2.434.040	688.190	71,73	71,05	0	1
P40n4p10t11v	-	662.319	-	-	-	-
P40n4p15t7v	-	907.440	-	-	-	-
P40n4p15t9v	3.358.250	815.336	75,72	73,37	0	1
P40n4p15t11v	4.390.560	1.161.270	73,55	72,45	0	1

Table B.4 – VINDX formulation after 6 hours running.

INSTANCE	VINDX					
	UB	LB	LP GAP (%)	GAP(%)	B&B Nodes	Nodes left
P40n5p5t7v	518.105	345.830	33,25	31,77	866	802
P40n5p5t9v	565.973	431.552	23,75	20,60	0	1
P40n5p5t11v	611.745	417.736	31,71	30,74	0	1
P40n5p10t7v	3.040.470	898.987	70,43	68,52	0	1
P40n5p10t9v	2.210.840	1.169.390	47,11	43,38	0	1
P40n5p10t11v	1.779.260	934.598	47,47	46,04	0	1
P40n5p15t7v	-	1.290.700	-	-	-	-
P40n5p15t9v	5.104.740	1.677.770	67,13	65,71	0	1
P40n5p15t11v	-	1.364.410	-	-	-	-
P50n3p5t7v	1.006.910	304.794	69,73	68,55	36	37
P50n3p5t9v	713.850	429.257	39,87	39,04	0	1
P50n3p5t11v	1.429.730	444.786	68,89	65,31	0	1
P50n3p10t7v	-	410.006	-	-	-	-
P50n3p10t9v	-	771.997	-	-	-	-
P50n3p10t11v	-	488.024	-	-	-	-
P50n3p15t7v	-	858.541	-	-	-	-
P50n3p15t9v	3.608.090	946.907	73,76	72,11	0	1
P50n3p15t11v	-	800.089	-	-	-	-
P50n4p5t7v	950.456	332.052	65,06	64,02	0	1
P50n4p5t9v	-	492.561	-	-	-	-
P50n4p5t11v	747.986	473.114	36,75	36,51	0	1
P50n4p10t7v	-	624.403	-	-	-	-
P50n4p10t9v	2.363.800	898.420	61,99	61,47	0	1
P50n4p10t11v	-	790.445	-	-	-	-
P50n4p15t7v	-	1.369.540	-	-	-	-
P50n4p15t9v	-	1.049.610	-	-	-	-
P50n4p15t11v	-	1.295.940	-	-	-	-
P50n5p5t7v	841.087	481.136	42,80	41,70	0	1
P50n5p5t9v	1.134.560	553.521	51,21	50,77	30	29
P50n5p5t11v	865.317	526.570	39,15	34,25	0	1
P50n5p10t7v	-	961.170	-	-	-	-
P50n5p10t9v	-	1.104.200	-	-	-	-
P50n5p10t11v	2.538.630	911.456	64,10	62,23	0	1
P50n5p15t7v	-	1.405.500	-	-	-	-
P50n5p15t9v	-	1.224.080	-	-	-	-
P50n5p15t11v	-	1.334.770	-	-	-	-

Table B.5 – 2COMM formulation after 6 hours running.

INSTANCE	2COMM					
	UB	LB	LP GAP (%)	GAP(%)	B&B Nodes	Nodes left
P20n3p5t7v	391.528	297.990	23,89	2,45	23259	13026
P20n3p5t9v	202.915	164.319	19,02	2,50	28429	14195
P20n3p5t11v	288.668	246.252	14,69	2,42	22671	17321
P20n3p10t7v	489.465	382.494	21,85	2,50	55613	32084
P20n3p10t9v	436.894	308.409	29,41	7,87	19768	11236
P20n3p10t11v	688.757	591.686	14,09	2,93	84234	73302
P20n3p15t7v	755.877	540.318	28,52	16,33	35474	26928
P20n3p15t9v	1.182.860	859.310	27,35	14,91	33916	29562
P20n3p15t11v	898.279	713.910	20,52	9,32	20472	18164
P20n4p5t7v	372.428	287.229	22,88	2,44	22031	13081
P20n4p5t9v	344.193	266.998	22,43	2,49	5856	3684
P20n4p5t11v	273.274	220.017	19,49	2,49	36742	19435
P20n4p10t7v	782.581	629.284	19,59	3,62	18649	5998
P20n4p10t9v	660.923	500.170	24,32	8,57	38187	24094
P20n4p10t11v	906.773	688.348	24,09	12,70	22271	15124
P20n4p15t7v	1.108.540	763.501	31,13	15,70	27059	20407
P20n4p15t9v	1.083.300	791.510	26,94	12,97	16280	13455
P20n4p15t11v	1.240.280	936.779	24,47	12,71	8876	6192
P20n5p5t7v	341.003	295.544	13,33	2,43	10178	3740
P20n5p5t9v	389.873	326.648	16,22	2,50	28718	17710
P20n5p5t11v	485.908	387.007	20,35	2,22	8784	5477
P20n5p10t7v	944.844	728.208	22,93	7,80	39916	29811
P20n5p10t9v	775.047	612.165	21,02	6,55	24806	18871
P20n5p10t11v	717.215	529.836	26,13	4,46	25916	18095
P20n5p15t7v	1.284.820	874.308	31,95	18,61	6600	3891
P20n5p15t9v	1.494.610	1.162.500	22,22	11,27	13722	8267
P20n5p15t11v	1.750.670	1.449.880	17,18	8,36	13034	10287
P30n3p5t7v	213.712	163.465	23,51	7,49	28443	11805
P30n3p5t9v	391.676	304.526	22,25	2,50	43041	27849
P30n3p5t11v	367.768	277.559	24,53	2,50	22404	11863
P30n3p10t7v	688.428	484.644	29,60	16,10	24871	19469
P30n3p10t9v	667.572	463.102	30,63	10,82	16486	8420
P30n3p10t11v	655.854	448.232	31,66	16,54	18263	15412
P30n3p15t7v	903.626	514.078	43,11	25,03	4218	172
P30n3p15t9v	1.080.670	661.370	38,80	24,56	3887	2222
P30n3p15t11v	1.196.560	846.593	29,25	18,18	2105	275

Table B.6 – 2COMM formulation after 6 hours running.

INSTANCE	2COMM					
	UB	LB	LP GAP (%)	GAP(%)	B&B Nodes	Nodes left
P30n4p5t7v	410.940	318.423	22,51	1,64	108440	82807
P30n4p5t9v	537.101	432.958	19,39	0,62	224435	181648
P30n4p5t11v	495.875	412.699	16,77	0,64	137388	99115
P30n4p10t7v	692.268	484.873	29,96	13,07	18877	13407
P30n4p10t9v	739.013	534.998	27,61	10,68	21103	16141
P30n4p10t11v	853.287	580.149	32,01	17,68	7804	4585
P30n4p15t7v	1.221.400	806.908	33,94	19,29	3416	2213
P30n4p15t9v	1.263.490	775.141	38,65	23,02	5052	3275
P30n4p15t11v	1.352.240	834.354	38,30	24,07	4756	4696
P30n5p5t7v	415.180	343.669	17,22	1,30	171274	151919
P30n5p5t9v	396.522	322.253	18,73	1,05	83896	53145
P30n5p5t11v	481.502	393.899	18,19	0,62	139601	88833
P30n5p10t7v	1.030.250	759.197	26,31	12,36	14600	10310
P30n5p10t9v	975.333	680.593	30,22	13,70	3395	1385
P30n5p10t11v	1.040.080	760.388	26,89	13,45	5769	3498
P30n5p15t7v	1.364.230	881.433	35,39	18,39	5651	3859
P30n5p15t9v	1.884.860	1.225.560	34,98	23,55	5434	2641
P30n5p15t11v	1.719.730	1.163.680	32,33	20,02	3710	2602
P40n3p5t7v	444.187	330.565	25,58	2,98	39841	7505
P40n3p5t9v	249.859	183.788	26,44	3,25	37128	13985
P40n3p5t11v	380.325	308.907	18,78	3,08	20230	13699
P40n3p10t7v	748.607	486.765	34,98	19,12	4002	228
P40n3p10t9v	1.039.380	707.615	31,92	20,34	4095	1733
P40n3p10t11v	978.994	690.991	29,42	19,67	2979	637
P40n3p15t7v	1.391.770	876.775	37,00	27,97	570	570
P40n3p15t9v	860.608	470.927	45,28	25,67	1763	1175
P40n3p15t11v	1.182.990	689.736	41,70	29,65	571	572
P40n4p5t7v	318.810	247.166	22,47	7,59	19571	7896
P40n4p5t9v	380.047	319.847	15,84	1,51	39594	20812
P40n4p5t11v	474.411	358.988	24,33	8,98	22097	17776
P40n4p10t7v	820.170	516.326	37,05	16,82	6040	4597
P40n4p10t9v	1.067.600	691.271	35,25	20,55	2512	730
P40n4p10t11v	1.058.480	672.186	36,50	22,67	3552	3339
P40n4p15t7v	1.722.080	911.707	47,06	32,98	273	274
P40n4p15t9v	1.438.880	823.494	42,77	28,12	224	217
P40n4p15t11v	1.829.910	1.173.510	35,87	25,78	813	814
P40n5p5t7v	420.504	347.023	17,47	2,73	24031	9245

Table B.7 – 2COMM formulation after 6 hours running.

INSTANCE	2COMM					
	UB	LB	LP GAP (%)	GAP(%)	B&B Nodes	Nodes left
P40n5p5t9v	524.461	432.411	17,55	5,85	24326	15164
P40n5p5t11v	522.045	420.379	19,47	10,43	11238	6260
P40n5p10t7v	1.244.820	902.930	27,47	14,73	3569	2913
P40n5p10t9v	1.640.770	1.173.900	28,45	17,73	4373	3943
P40n5p10t11v	1.377.760	937.402	31,96	20,22	4545	2195
P40n5p15t7v	2.013.740	1.294.080	35,74	22,24	538	539
P40n5p15t9v	2.549.640	1.683.920	33,95	25,06	560	559
P40n5p15t11v	2.417.880	1.373.290	43,20	32,47	343	344
P50n3p5t7v	401.351	305.876	23,79	13,23	20588	11138
P50n3p5t9v	524.467	431.451	17,74	11,12	21980	14474
P50n3p5t11v	598.881	448.696	25,08	13,21	14786	9494
P50n3p10t7v	710.314	413.584	41,77	25,16	253	254
P50n3p10t9v	-	776.136	-	-	-	-
P50n3p10t11v	879.953	493.255	43,95	27,73	393	394
P50n3p15t7v	1.399.850	861.784	38,44	24,66	0	1
P50n3p15t9v	1.688.620	955.951	43,39	32,00	0	1
P50n3p15t11v	1.727.050	807.048	53,27	41,25	0	1
P50n4p5t7v	468.341	333.285	28,84	16,59	6286	219
P50n4p5t9v	656.003	494.613	24,60	17,67	3242	1151
P50n4p5t11v	614.873	475.263	22,71	14,95	3579	1562
P50n4p10t7v	1.069.130	627.398	41,32	24,97	0	1
P50n4p10t9v	1.547.480	901.283	41,76	31,54	0	1
P50n4p10t11v	1.380.320	796.470	42,30	29,34	0	1
P50n4p15t7v	2.190.220	1.374.240	37,26	26,86	0	1
P50n4p15t9v	1.961.430	1.055.440	46,19	33,66	0	1
P50n4p15t11v	2.245.050	1.305.000	41,87	33,30	0	1
P50n5p5t7v	629.287	482.461	23,33	15,83	2186	1722
P50n5p5t9v	715.774	555.604	22,38	16,74	2204	922
P50n5p5t11v	717.739	528.628	26,35	20,83	2142	1853
P50n5p10t7v	1.519.620	962.170	36,68	21,25	0	1
P50n5p10t9v	1.724.700	1.108.840	35,71	25,09	0	1
P50n5p10t11v	1.503.930	915.536	39,12	25,73	0	1
P50n5p15t7v	2.360.440	1.409.850	40,27	27,90	0	1
P50n5p15t9v	-	1.228.240	-	-	-	-
P50n5p15t11v	2.188.780	1.339.930	38,78	25,04	0	1

B.3 Data from the Section 3.5

Table B.8 presents the data attained after the solving the instance P20n3p10t9v with the 2COMM formulation considering of four cases: *(a)* no back-ordering and homogeneous fleet, *(b)* no back-ordering and heterogeneous fleet, *(c)* back-ordering and homogeneous fleet, and the proposed rich PRP considering *(d)* back-ordering and heterogeneous fleet. For the tested cases,

Table B.8 – Metrics of the solutions attained by the cases *a, b* and *c*.

	Cases			
Metrics	*(a)*	*(b)*	*(c)*	*(d)*
Holding costs	97,101	94,922	79,209	77,328
Back-ordering costs	0	0	0	7,028
Traveling costs	50,144	48,567	52,576	47,742
Vehicle activation costs	26,367	32,788	27,401	32,160
Total costs	446,248	448,913	431,822	436,894
number of vehicles	51	49	53	49

all of them obtained the same sum of setup, and production costs, which are equal to 240,000, and 32,636, respectively.

APPENDIX C – Adaptive selection

The ABUILS method, presented in Section 4.6.2, used the following adaptive criterion to select perturbation operators and inter-route local searches.

The roulette wheel or fitness proportionate selection (GOLDENBERG, 1989) is a frequently used method in genetic and evolutionary algorithms which assumes that the probability of selection is proportional to the fitness of an individual (LIPOWSKI; LIPOWSKA, 2012).

Based on this concept and to guarantee greater chances of improving the solutions found by the ABUILS approach, the roulette wheel procedure is used in two moments of the Algorithm 4.5. First to select the perturbation mechanism (line 9) and second when the problems $\mathrm{VR_t}$ are optimized with the inter-route search algorithm 4.3 (line 10). In both cases, before picking an perturbation operator or inter-route search and apply it to the solution, the fitness is calculated with Algorithm C.1. Vector $\mathbb{S}$ registers the number of times that an operator or an inter-route search κ lead to some solution improvement. Parameter $\mathbf{S}$ accounts the total of improvements achieved, and $\mathcal{K}$ corresponds to the respective cardinality of the sets of perturbations (R) or searches (RS). Then, with the Algorithm C.2 a number $\rho \in [0, 1]$ is drawn, and accordingly with the proportional fitness, the correspondent item κ is selected and applied to the solution.

Algorithm C.1: Roulette wheel

Data: vector $\mathbb{S}$, $\mathbf{S}$, $\mathcal{K}$

for $\kappa = 1, \ldots, \mathcal{K}$ **do**
    $\mathrm{RW}[\kappa] \leftarrow (\mathrm{RW}[\kappa - 1] + \mathbb{S}[\kappa])/\mathbf{S}$;
end
return RW;

Algorithm C.2: Adaptive selection

Data: vector RW, $\mathcal{K}$

$\rho \leftarrow [0, 1]$;
for $\kappa = 1, \ldots, \mathcal{K}$ **do**
    if $\mathrm{RW}[\kappa - 1] < \rho \leq \mathrm{RW}[\kappa]$ **then** choose item κ;
end
return κ;

APPENDIX D – Data from Chapter 4

D.1 Δ comparison

Table D.1 compares the results obtained by each of the three Top-down ILS matheuristics when they adopt the new proposed Δ_{ik}^{vt} or the Δ_i proposed by Qiu et al. (2018b).

Table D.1 – Results found for each algorithm tested with different $\boldsymbol{\Delta}$.

INSTANCE	TILS		OILS		IILS	
	Δ_{ik}^{vt}	Δ_i	Δ_{ik}^{vt}	Δ_i	Δ_{ik}^{vt}	Δ_i
P20n5p15t7v	1.292.997	1.367.683	1.298.490	1.356.261	1.291.880	1.357.766
P20n5p15t9v	1.536.466	1.602.247	1.530.284	1.575.562	1.540.208	1.559.412
P20n5p15t11v	1.805.196	1.870.957	1.798.096	1.868.928	1.807.442	1.836.693
P30n5p15t7v	1.380.956	1.483.223	1.385.304	1.441.097	1.375.116	1.477.505
P30n5p15t9v	1.882.327	1.913.187	1.874.242	1.906.038	1.883.683	1.875.127
P30n5p15t11v	1.714.216	1.758.211	1.712.711	1.712.553	1.715.671	1.733.160
P40n5p15t7v	2.008.522	2.146.106	2.010.339	2.124.787	1.991.491	2.130.201
P40n5p15t9v	2.421.443	2.484.539	2.424.660	2.401.548	2.438.379	2.478.534
P40n5p15t11v	2.190.505	2.257.174	2.175.225	2.216.623	2.226.000	2.175.248
P50n5p15t7v	2.163.148	2.383.223	2.211.989	2.369.234	2.217.961	2.389.641
P50n5p15t9v	2.176.566	2.215.536	2.164.321	-	2.190.110	2.203.621
P50n5p15t11v	2.172.959	2.157.400	2.122.353	2.125.158	2.151.538	2.171.415

D.2 Parameters tuning

The set of instances tested in the parameters tuning is composed by: X20C3P7T9V, X20C3P7T11V, X20C3P10T9V, X20C3P10T11V, X20C3P14T9V, X20C3P14T11V, X20C4P7T11V, X20C4P10T7V, X20C4P10T11V, X20C4P14T9V, X20C4P14T11V, X20C5P10T9V, X20C5P10T11V, X20C5P14T11V, X20C6P7T9V, X20C6P10T11V, X20C6P14T9V, X20C6P14T11V, X20C7P7T9V, X20C7P10T9V, X20C7P14T9V, X20C7P14T11V, X30C3P7T7V, X30C3P7T9V, X30C3P7T11V, X30C3P10T11V, X30C3P14T7V, X30C3P14T9V, X30C3P14T11V, X30C4P7T7V, X30C4P7T9V, X30C4P7T11V, X30C4P10T9V, X30C4P10T11V, X30C4P14T7V, X30C4P14T9V, X30C4P14T11V, X30C5P7T7V, X30C5P7T9V, X30C5P10T9V, X30C5P10T11V, X30C5P14T9V, X30C5P14T11V, X30C6P7T9V, X30C6P7T11V, X30C6P14T11V, X30C7P7T11V, X30C7P10T11V, X30C7P14T7V, X30C7P14T9V, X30C7P14T11V, X40C3P7T9V, X40C3P7T11V, X40C3P10T7V, X40C3P10T9V, X40C3P10T11V, X40C3P14T11V, X40C4P3T9V, X40C4P10T7V, X40C4P10T11V, X40C4P14T11V, X40C5P7T9V, X40C5P14T11V, X40C6P7T9V, X40C6P7T11V, X40C6P10T7V, X40C6P10T9V, X40C6P10T11V, X40C7P3T9V, X40C7P7T11V, X40C7P10T9V, X40C7P10T11V, X40C7P14T7V, X40C7P14T9V.

Table D.2 – Parameters values calibrated.

Parameter	**Possible values**
α	(10%, 15%, 20%)
$maxWeight$	(5,10)
$maxIter$	(200, 300, 500, 1000)
$maxPertPP$	(5,10,15,20)
$maxPertRD$	(3,5,7)
$maxPertLX$	(3,5,7)
$maxPertKK$	(5,10,15)
$maxPertDL$	(3,5,7)

```
TILS irace report
# Best configurations (first number is the configuration ID; listed from best to
worst according to the sum of ranks):
   maxweight maxiterils maxpertP maxiter alpha
14         5         15        3     300   0.1
19         5         10        3     200  0.15
3          5         15        3     200  0.15
25         5         15        3     200  0.15
# Best configurations as commandlines (first number is the configuration ID; same
order as above):
14  --MAXWEIGHT 5 --MAXITERILS 15 --MAXPERTP 3 --MAXITER 300 --ALPHA 0.1
19  --MAXWEIGHT 5 --MAXITERILS 10 --MAXPERTP 3 --MAXITER 200 --ALPHA 0.15
3   --MAXWEIGHT 5 --MAXITERILS 15 --MAXPERTP 3 --MAXITER 200 --ALPHA 0.15
25  --MAXWEIGHT 5 --MAXITERILS 15 --MAXPERTP 3 --MAXITER 200 --ALPHA 0.15

OILS irace report
# Best configurations (first number is the configuration ID; listed from best to
worst according to the sum of ranks):
   maxweight maxpertkk maxpertrd maxpertlx maxiter alpha
19         5        15         3         7     300   0.2
25         5        10         3         7     300   0.2
26         5        15         3         3     200  0.15
5          5        10         3         7     300   0.2
# Best configurations as commandlines (first number is the configuration ID; same
order as above):
19  --MAXWEIGHT 5 --MAXPERTKK 15 --MAXPERTRD 3 --MAXPERTLX 7 --MAXITER 300 --ALPHA
0.2
25  --MAXWEIGHT 5 --MAXPERTKK 10 --MAXPERTRD 3 --MAXPERTLX 7 --MAXITER 300 --ALPHA
0.2
26  --MAXWEIGHT 5 --MAXPERTKK 15 --MAXPERTRD 3 --MAXPERTLX 3 --MAXITER 200 --ALPHA
0.15
5   --MAXWEIGHT 5 --MAXPERTKK 10 --MAXPERTRD 3 --MAXPERTLX 7 --MAXITER 300 --ALPHA
0.2


IILS irace report
# Best configurations (first number is the configuration ID; listed from best to
worst according to the sum of ranks):
   maxweight maxiterils maxpertP maxiter alpha maxpertkk maxpertrd maxpertlx
13         5         15        5     300  0.15        10         7         3
26         5         10        3     200  0.15        10         5         3
33         5         15        3     200  0.1         15         7         5
37         5         15        3     300  0.15        10         7         7

# Best configurations as commandlines (first number is the configuration ID; same
order as above):
13  --MAXWEIGHT 5 --MAXITERILS 15 --MAXPERTP 5 --MAXITER 300 --ALPHA 0.15   --
MAXPERTKK 10  --MAXPERTRD 7   --MAXPERTLX 3
26  --MAXWEIGHT 5 --MAXITERILS 10 --MAXPERTP 3 --MAXITER 200 --ALPHA 0.15   --
MAXPERTKK 10  --MAXPERTRD 5   --MAXPERTLX 3
33  --MAXWEIGHT 5 --MAXITERILS 15 --MAXPERTP 3 --MAXITER 200 --ALPHA 0.1    --
MAXPERTKK 15  --MAXPERTRD 7   --MAXPERTLX 5
37  --MAXWEIGHT 5 --MAXITERILS 15 --MAXPERTP 3 --MAXITER 200 --ALPHA 0.15   --
MAXPERTKK 10  --MAXPERTRD 7   --MAXPERTLX 7
```

D.3 Percentage gaps

Table D.3 – Percentage gap (%) of the methods w.r.t the best known solution and average.

INSTANCE	GAP (%)									
	2COMM		OILS		TILS		IILS		ABUILS	
	OPT	BKS	BKS	AVG	BKS	AVG	BKS	AVG	BKS	AVG
P20n3p5t7v	2,45		1,85	3,15	2,19	5,51	2,08	3,28	0,70	2,25
P20n3p5t9v	2,50		2,95	9,83	3,19	14,53	9,57	11,08	7,70	8,50
P20n3p5t11v	2,42		1,95	2,63	1,68	6,13	1,96	3,30	2,16	2,58
P20n3p10t7v	2,50		5,47	7,49	7,14	8,95	6,39	7,74	3,13	4,22
P20n3p10t9v	7,87		6,71	9,60	8,32	15,49	6,90	9,91	4,14	6,70
P20n3p10t11v	2,93		4,08	5,75	4,86	11,83	4,91	5,53	2,07	3,77
P20n3p15t7v	16,33		6,01	8,36	5,85	15,71	5,55	7,67	4,05	4,62
P20n3p15t9v	14,91	0,38	0,89	3,86	0,45	2,78		2,03		1,74
P20n3p15t11v	9,32		5,01	6,43	6,06	14,25	5,77	8,59	2,55	3,88
P20n4p5t7v	2,44		1,83	2,57	2,38	2,82	1,88	2,49	1,46	1,99
P20n4p5t9v	2,49		1,85	6,80	3,92	10,71	2,64	7,87	1,10	2,88
P20n4p5t11v	2,49		3,21	4,19	5,60	10,42	3,26	5,10	2,49	4,35
P20n4p10t7v	3,62		3,31	4,31	3,96	7,84	2,65	4,10	0,92	3,09
P20n4p10t9v	8,57		2,68	4,14	3,83	11,91	3,90	4,74	1,22	2,73
P20n4p10t11v	12,70		2,64	3,74	2,88	7,79	3,38	4,26	1,09	2,03
P20n4p15t7v	15,70		1,64	8,21	4,26	10,18	5,07	6,99	2,57	4,37
P20n4p15t9v	12,97		4,76	9,03	5,13	8,07	4,19	7,19	1,85	3,21
P20n4p15t11v	12,71		4,05	5,41	4,77	9,76	4,68	6,30	3,07	4,00
P20n5p5t7v	2,43		1,86	4,53	2,80	4,30	1,77	2,54	1,29	2,09
P20n5p5t9v	2,50		2,12	3,30	2,15	4,34	1,65	4,58	0,86	1,66
P20n5p5t11v	2,22		1,75	2,79	2,32	7,92	2,04	2,45	0,97	2,17
P20n5p10t7v	7,80		1,52	3,06	2,09	6,29	1,80	3,13	1,05	3,15
P20n5p10t9v	6,55		2,11	3,94	3,81	4,78	2,67	3,96	0,90	3,43
P20n5p10t11v	4,46		5,45	6,51	3,92	9,09	3,64	5,91	2,07	3,35
P20n5p15t7v	18,61		1,06	4,37	0,64	6,70	0,55	3,31		2,56
P20n5p15t9v	11,27		2,39	3,40	2,80	7,21	3,05	4,59	0,74	1,72
P20n5p15t11v	8,36		2,71	5,58	3,11	7,33	3,24	5,04	1,14	2,67
P30n3p5t7v	7,49		2,37	6,06	2,41	10,10	2,62	6,11	1,91	4,69
P30n3p5t9v	2,50		1,34	2,92	1,81	2,66	0,99	2,97	0,12	2,12
P30n3p5t11v	2,50		1,89	3,74	2,05	7,53	1,61	3,61	0,52	2,04
P30n3p10t7v	16,10		3,11	4,75	4,87	16,89	2,19	4,59	0,03	3,83
P30n3p10t9v	10,82		4,92	7,85	5,74	12,81	5,12	8,36	1,29	5,14
P30n3p10t11v	16,54		0,59	5,12	3,69	7,10	1,74	4,87		2,20
P30n3p15t7v	25,03	0,76	0,82	5,08	0,15	9,32		4,35		1,85
P30n3p15t9v	24,56	0,17		4,24	0,98	8,33	1,06	4,84		2,34
P30n3p15t11v	18,18		2,40	4,89	3,61	10,47	2,87	5,96		2,92

Table D.4 – Percentage gap (%) of the methods w.r.t the best known solution and average.

INSTANCE	**GAP (%)**									
	2COMM		**OILS**		**TILS**		**IILS**		**ABUILS**	
	OPT	**BKS**	**BKS**	**AVG**	**BKS**	**AVG**	**BKS**	**AVG**	**BKS**	**AVG**
P30n4p5t7v	1,64		2,10	3,49	3,68	7,25	2,67	3,72	2,18	3,62
P30n4p5t9v	0,62		1,90	2,94	2,86	8,55	2,29	3,00	1,68	2,68
P30n4p5t11v	0,64		2,92	4,01	3,66	10,14	2,66	4,11	1,45	2,26
P30n4p10t7v	13,07		0,56	3,57	3,95	8,79	1,96	4,20		1,11
P30n4p10t9v	10,68		4,01	6,19	4,87	10,46	5,73	7,15	1,21	3,31
P30n4p10t11v	17,68	0,85		4,13	0,50	7,22	1,68	3,35		2,83
P30n4p15t7v	19,29	1,61	1,19	2,85	0,66	5,72		2,26		2,57
P30n4p15t9v	23,02	4,43	1,86	3,89		3,18	1,83	4,20		2,05
P30n4p15t11v	24,07	0,54		3,04	0,31	5,80	1,06	2,81		1,61
P30n5p5t7v	1,30		2,53	7,19	3,72	9,69	2,97	6,57	1,69	3,86
P30n5p5t9v	1,05		3,71	5,68	3,94	9,92	3,35	6,15	2,45	5,30
P30n5p5t11v	0,62		3,14	4,45	4,26	10,94	3,52	4,74	2,24	3,33
P30n5p10t7v	12,36		2,25	3,67	2,75	9,34	1,19	3,22	0,16	1,20
P30n5p10t9v	13,70	0,05		1,29	0,32	1,67	0,34	1,35		1,60
P30n5p10t11v	13,45		0,44	2,40	1,69	8,97	1,02	2,29		1,64
P30n5p15t7v	18,39		1,54	3,34	1,23	9,24	0,80	2,76		2,07
P30n5p15t9v	23,55	0,57		1,16	0,43	9,48	0,50	2,04		1,22
P30n5p15t11v	20,02	0,41		7,26	0,09	5,88	0,17	2,60		1,25
P40n3p5t7v	2,98		3,63	4,54	4,26	7,19	3,75	5,01	0,84	5,01
P40n3p5t9v	3,25		3,44	4,63	5,15	6,42	4,50	6,05	1,69	5,45
P40n3p5t11v	3,08		2,60	3,43	3,06	7,44	2,95	4,31	1,21	2,87
P40n3p10t7v	19,12		1,67	4,72	0,44	8,77	2,60	3,94		3,90
P40n3p10t9v	20,34	4,23	1,57	4,45	0,81	4,01		3,56		1,81
P40n3p10t11v	19,67	3,01	0,39	1,69	0,29	8,29		1,89		1,51
P40n3p15t7v	27,97	10,45	0,95	4,33	0,46	3,26		3,26		1,73
P40n3p15t9v	25,67	3,56	0,18	1,71	0,02	6,30		1,55		2,76
P40n3p15t11v	29,65	8,92	2,37	4,24	1,56	3,98		4,74		2,79
P40n4p5t7v	7,59		2,17	4,47	4,67	7,60	2,10	4,21	1,01	3,66
P40n4p5t9v	1,51		2,96	6,98	4,21	18,71	3,69	7,32	1,81	3,11
P40n4p5t11v	8,98		1,18	3,39	1,56	17,68	1,50	3,34	1,30	2,67
P40n4p10t7v	16,82	1,40	1,21	3,42	1,28	8,90		2,26		3,36
P40n4p10t9v	20,55	4,06	0,85	2,63	1,40	5,44		2,44		2,41
P40n4p10t11v	22,67	4,17	0,08	1,96		5,85	0,62	3,42		2,11
P40n4p15t7v	32,98	7,20		2,73	1,99	8,32	0,36	3,87		2,03
P40n4p15t9v	28,12	4,06	1,69	4,77		11,54	1,17	5,29		2,43
P40n4p15t11v	25,78	2,47		2,72	1,76	7,67	1,97	3,90		1,52

Table D.5 – Percentage gap (%) of the methods w.r.t the best known solution and average.

INSTANCE	GAP (%)									
	2COMM		OILS		TILS		IILS		ABUILS	
	OPT	BKS	BKS	AVG	BKS	AVG	BKS	AVG	BKS	AVG
P40n5p5t7v	2,73		2,27	4,55	2,29	7,84	2,12	3,71	1,51	3,83
P40n5p5t9v	5,85		2,81	3,93	2,45	12,58	2,59	4,16	1,10	1,77
P40n5p5t11v	10,43		1,31	5,97	5,13	22,94	1,28	4,51	1,30	2,28
P40n5p10t7v	14,73		1,46	3,29	0,81	5,27	0,72	4,07	0,95	1,62
P40n5p10t9v	17,73	3,86	0,10	1,05	0,01	3,42		1,52		1,30
P40n5p10t11v	20,22	2,03		3,30	1,33	6,49	1,27	3,80		0,99
P40n5p15t7v	22,24	1,12	0,95	2,81	0,86	9,94		3,23		2,43
P40n5p15t9v	25,06	5,29	0,13	2,57		5,26	0,70	2,81		1,79
P40n5p15t11v	32,47	11,16		3,10	0,70	8,35	2,33	4,51		1,63
P50n3p5t7v	13,23		0,17	3,43	2,53	6,02	1,27	5,07		3,10
P50n3p5t9v	11,12		1,28	3,48	3,05	11,52	1,53	3,56	0,70	1,77
P50n3p5t11v	13,21	4,83		1,48	0,39	10,51	0,26	3,88		2,20
P50n3p10t7v	25,16	3,13	1,50	4,26	1,14	4,25		4,03		4,73
P50n3p10t9v		-		2,73	2,16	8,11	0,96	3,07		1,58
P50n3p10t11v	27,73	7,57	1,20	3,64	1,04	11,03		3,11		1,10
P50n3p15t7v	24,66		0,53	4,61	2,64	9,31	2,11	5,49		4,68
P50n3p15t9v	32,00	10,86		4,83	1,80	4,23	0,42	5,40		1,65
P50n3p15t11v	41,25	18,28		2,86	0,09	2,80	1,80	4,69		1,62
P50n4p5t7v	16,59	5,97	0,51	5,55		6,54	0,55	2,65		1,73
P50n4p5t9v	17,67	8,40	0,04	4,33	0,89	12,70		4,42		2,09
P50n4p5t11v	14,95	1,02		2,76	0,26	5,70	1,04	2,47		1,53
P50n4p10t7v	24,97	9,68	1,13	3,39		5,74	0,89	2,99		2,42
P50n4p10t9v	31,54	13,94		1,40	0,21	6,06	0,57	1,87		2,71
P50n4p10t11v	29,34	11,76		3,11	0,71	4,10	0,23	3,65		2,91
P50n4p15t7v	26,86	7,79	1,53	3,34		5,95	2,23	4,01		1,37
P50n4p15t9v	33,66	11,42	0,86	2,57	0,21	6,53		2,41		1,91
P50n4p15t11v	33,30	7,22		3,22	2,01	11,81	0,70	3,54		1,62
P50n5p5t7v	15,83	3,39	0,63	3,95		9,24	0,56	3,72		2,79
P50n5p5t9v	16,74	1,74	0,33	1,38		6,25	0,34	1,97		0,79
P50n5p5t11v	20,83	3,90	0,08	1,84	0,02	8,12		1,76		1,37
P50n5p10t7v	21,25	5,23	1,54	3,04	1,01	2,57		2,98		0,85
P50n5p10t9v	25,09	13,48	0,92	1,71		7,45	1,34	3,07		1,39
P50n5p10t11v	25,73	10,71		2,68	0,81	2,87	0,65	2,73		4,41
P50n5p15t7v	27,90	9,12	2,26	3,76		3,31	2,53	4,01		3,22
P50n5p15t9v		-		2,53	0,57	6,79	1,19	4,42		1,86
P50n5p15t11v	25,04	3,13		2,33	2,38	3,59	1,38	3,50		1,50

APPENDIX E – Data from Chapter 5

E.1 Generating feasible routes for the RRMP

To first solve the RRMP, a set of routes must be provided. This work generate them with the Algorithm E.1. It explores the vehicles from the biggest to the smallest capacities.

Algorithm E.1: Feasible routes

Data: Problem information

forall $t \in \mathcal{T}$ **do**
  $C \leftarrow \emptyset$;
  forall $i \in \overline{\mathcal{N}}$ **do**
    $C \leftarrow C \cup \{i\}$;
    forall $k \in \mathcal{P}$ **do**
      $D_i^t \leftarrow D_i^t + d_{ik}t$;
    end
  end
  sort the set of customers C decreasingly, from highest to lowest accumulated periodic demand D_i^t;
  for $v = V, \ldots, 1$ **do**
    $load \leftarrow Q^v$; //residual load
    $ttt \leftarrow 0$; //accumulated travel time
    $h \leftarrow 0$; //predecessor node
    $r \leftarrow r \cup \{h\}$;
    while *is feasible to add some customer to* r **do**
      $i \leftarrow first(C)$;
      if i *is not visited by some route* **then**
        if $D_i^t \leq load$ ***and*** $ttt - a_{hi} + s_i + a_{i,n+1} \leq H$ **then**
          $C \leftarrow C \setminus \{i\}$;
          $r \leftarrow r \cup \{i\}$;
          $load \leftarrow load - D_i^t$;
          $ttt \leftarrow ttt - a_{hi} + s_i + a_{i,n+1}$;
          $h \leftarrow i$;
        end
      end
    end
    $r \leftarrow r \cup \{n+1\}$;
    add r with correspondent cost $c_r = e^v + \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij}^{vt}$ to the RRMP;
  end
end

But, if it is not possible to generated feasible routes for a certain period, an artificial routing solution is easily provided considering an infinite-cost path leaving the plant, visiting all customers while deliveries their respective demands, and returning to the plant.

E.2 Lower bounds found with the heuristically priced columns

Table E.1 – Upper, lower and column generation bounds.

INSTANCE	UB	LB	Column generation					
			0	1	2	3	4	5
P20n3p5t7v	391.528	297.990	378.221	316.230	319.557	318.831	315.452	316.800
P20n3p5t9v	202.915	164.319	220.900	186.772	184.161	184.510	183.052	180.439
P20n3p5t11v	288.668	246.252	304.429	260.695	260.303	261.297	304.429	304.429
P20n3p10t7v	489.465	382.494	483.251	431.225	419.215	418.799	412.041	418.072
P20n3p10t9v	436.894	308.409	409.192	349.341	354.851	346.981	409.192	331.243
P20n3p10t11v	688.757	591.686	692.487	692.487	671.707	646.115	692.487	692.487
P20n3p15t7v	755.877	540.318	700.129	700.129	619.669	609.340	700.129	700.129
P20n3p15t9v	1.148.420	859.310	1.074.280	1.074.280	1.062.760	1.057.490	1.074.280	1.074.280
P20n3p15t11v	898.279	713.910	850.924	850.924	833.652	831.190	765.745	761.514
P20n4p5t7v	372.428	287.229	347.784	314.098	314.695	304.784	301.133	301.063
P20n4p5t9v	344.193	266.998	332.538	297.398	282.628	281.045	274.974	332.538
P20n4p5t11v	273.274	220.017	274.783	274.783	245.647	240.508	236.172	236.384
P20n4p10t7v	782.581	629.284	744.194	683.940	670.631	672.940	657.359	658.110
P20n4p10t9v	660.923	500.170	600.768	578.363	547.785	544.008	525.107	600.768
P20n4p10t11v	906.773	688.348	825.281	825.281	798.170	800.098	729.597	726.847
P20n4p15t7v	1.108.540	763.501	949.640	869.340	947.241	936.717	926.796	876.052
P20n4p15t9v	1.083.300	791.510	962.342	962.342	882.050	901.974	863.345	880.040
P20n4p15t11v	1.240.280	936.779	1.127.250	1.127.250	1.123.070	1.051.580	1.029.100	1.127.250
P20n5p5t7v	341.003	295.544	355.497	328.256	355.497	315.052	306.466	308.231
P20n5p5t9v	389.873	326.648	397.985	351.366	397.985	353.189	334.757	368.179
P20n5p5t11v	485.908	387.007	474.204	417.481	406.505	408.698	402.268	402.897
P20n5p10t7v	944.844	728.208	875.819	804.639	802.722	789.960	769.896	766.112
P20n5p10t9v	775.047	612.165	721.939	674.156	669.181	698.869	721.939	651.315
P20n5p10t11v	717.215	529.836	623.073	614.319	580.625	623.073	582.273	582.463
P20n5p15t7v	1.266.602	874.308	1.104.140	1.014.720	1.020.600	1.015.930	1.082.560	986.541
P20n5p15t9v	1.494.610	1.162.500	1.369.590	1.272.360	1.264.210	1.369.590	1.283.020	1.272.140
P20n5p15t11v	1.750.670	1.449.880	1.635.030	1.543.590	1.539.530	1.635.030	1.560.030	1.530.490
P30n3p5t7v	213.712	163.465	229.878	189.713	184.696	180.980	181.170	182.609
P30n3p5t9v	391.676	304.526	389.140	324.450	323.454	320.157	321.432	319.081
P30n3p5t11v	367.768	277.559	374.909	303.986	294.451	290.560	290.451	312.275
P30n3p10t7v	688.428	484.644	662.270	552.102	546.392	530.212	528.101	532.785
P30n3p10t9v	667.572	463.102	642.806	530.126	543.126	508.292	501.700	505.191
P30n3p10t11v	654.725	448.232	609.833	503.163	502.056	609.833	485.869	489.356
P30n3p15t7v	880.395	514.078	775.153	617.246	761.807	659.923	624.919	612.543
P30n3p15t9v	1.023.110	661.370	912.183	762.005	750.084	770.349	867.381	786.251
P30n3p15t11v	1.184.380	846.593	1.083.730	1.083.730	992.255	922.044	1.083.730	1.083.730

Table E.2 – Upper, lower and column generation bounds.

INSTANCE	**UB**	**LB**	**Column generation**					
			0	**1**	**2**	**3**	**4**	**5**
P30n4p5t7v	410.940	318.423	406.527	347.910	350.592	339.147	334.691	339.692
P30n4p5t9v	537.101	432.958	541.255	488.135	541.255	448.291	448.549	448.458
P30n4p5t11v	495.875	412.699	531.912	448.750	434.420	429.181	429.001	428.625
P30n4p10t7v	691.145	484.873	643.588	548.043	557.528	538.815	540.460	534.780
P30n4p10t9v	739.013	534.998	705.602	608.693	606.143	601.251	596.706	590.572
P30n4p10t11v	824.586	580.149	737.904	737.904	692.715	640.822	659.953	686.876
P30n4p15t7v	1.175.321	806.908	1.070.500	1.070.500	909.590	953.337	958.318	977.888
P30n4p15t9v	1.180.465	775.141	1.007.620	880.521	902.890	937.109	893.888	918.400
P30n4p15t11v	1.293.592	834.354	1.109.930	1.109.930	955.443	1.109.930	1.094.040	982.480
P30n5p5t7v	415.180	343.669	429.630	384.213	366.076	429.630	358.895	361.367
P30n5p5t9v	396.522	322.253	409.511	355.587	385.793	335.971	409.511	339.658
P30n5p5t11v	481.502	393.899	484.450	421.994	429.706	408.827	411.910	408.810
P30n5p10t7v	1.030.250	759.197	941.360	843.486	852.569	861.994	823.940	825.892
P30n5p10t9v	958.748	680.593	850.361	850.361	768.534	745.159	850.361	758.686
P30n5p10t11v	1.028.217	760.388	917.701	856.308	835.270	824.262	820.004	902.479
P30n5p15t7v	1.338.394	881.433	1.128.950	1.128.950	1.017.510	1.031.370	1.128.950	1.128.950
P30n5p15t9v	1.816.277	1.225.560	1.502.160	1.502.160	1.390.000	1.376.730	1.381.740	1.377.720
P30n5p15t11v	1.682.814	1.163.680	1.417.660	1.355.320	1.305.250	1.295.550	1.309.390	1.394.880
P40n3p5t7v	444.187	330.565	441.780	362.419	367.291	374.819	354.182	358.254
P40n3p5t9v	249.859	183.788	272.570	208.381	204.372	204.548	205.757	211.336
P40n3p5t11v	380.325	308.907	415.983	337.528	336.593	330.847	335.385	332.868
P40n3p10t7v	731.204	486.765	704.410	558.866	560.186	546.776	553.904	548.079
P40n3p10t9v	994.609	707.615	952.824	952.824	796.461	770.501	952.824	773.910
P40n3p10t11v	923.388	690.991	899.978	758.946	756.885	899.978	755.321	761.380
P40n3p15t7v	1.227.324	876.775	1.179.120	1.013.280	1.006.820	1.045.300	1.053.510	1.028.710
P40n3p15t9v	781.594	470.927	789.579	605.909	589.054	613.683	675.677	623.328
P40n3p15t11v	1.045.710	689.736	1.009.240	1.009.240	786.124	812.433	828.758	847.537
P40n4p5t7v	318.810	247.166	337.967	271.306	277.104	267.403	269.636	264.626
P40n4p5t9v	380.047	319.847	418.788	349.184	339.045	418.788	418.788	331.132
P40n4p5t11v	474.411	358.988	492.036	397.246	427.550	394.432	384.813	381.860
P40n4p10t7v	794.697	516.326	697.640	697.640	586.091	661.526	643.614	604.643
P40n4p10t9v	1.007.635	691.271	894.533	796.293	773.223	783.479	894.533	765.742
P40n4p10t11v	992.252	672.186	910.682	910.682	756.053	773.302	770.770	790.978
P40n4p15t7v	1.554.534	911.707	1.239.890	1.239.890	1.239.890	1.174.100	1.239.890	1.239.890
P40n4p15t9v	1.339.051	823.494	1.159.650	1.159.650	1.159.650	1.141.460	1.017.860	1.056.830
P40n4p15t11v	1.731.838	1.173.510	1.498.500	1.498.500	1.498.500	1.414.560	1.447.250	1.498.500

Table E.3 – Upper, lower and column generation bounds.

INSTANCE	UB	LB	Column generation					
			0	1	2	3	4	5
P40n5p5t7v	420.504	347.023	473.455	380.507	381.398	366.176	372.585	378.508
P40n5p5t9v	524.461	432.411	559.977	480.258	471.445	453.328	452.651	453.601
P40n5p5t11v	522.045	420.379	535.874	478.495	463.675	441.719	441.472	442.793
P40n5p10t7v	1.244.820	902.930	1.137.100	1.137.100	1.045.060	1.018.560	1.057.210	1.032.430
P40n5p10t9v	1.555.132	1.173.900	1.411.580	1.411.580	1.299.510	1.375.790	1.275.660	1.283.350
P40n5p10t11v	1.343.143	937.402	1.154.420	1.154.420	1.064.940	1.084.140	1.081.670	1.154.420
P40n5p15t7v	1.950.402	1.294.080	1.646.040	1.646.040	1.646.040	1.619.110	1.627.930	1.646.040
P40n5p15t9v	2.361.557	1.683.920	2.061.400	2.061.400	1.868.700	1.959.840	1.940.190	1.998.010
P40n5p15t11v	2.124.965	1.373.290	1.779.400	1.779.400	1.561.020	1.664.890	1.779.400	1.779.400
P50n3p5t7v	400.692	305.876	438.872	343.882	359.001	331.275	339.089	337.613
P50n3p5t9v	524.467	431.451	576.475	473.529	468.345	474.539	476.074	473.676
P50n3p5t11v	568.760	448.696	625.108	488.305	489.755	497.803	497.064	484.433
P50n3p10t7v	647.029	413.584	669.782	668.185	589.366	519.107	508.348	514.674
P50n3p10t9v	1.129.270		1.102.400	897.255	860.366	922.514	869.434	1.077.380
P50n3p10t11v	801.108	493.255	764.509	764.509	628.297	591.580	586.394	625.741
P50n3p15t7v	1.344.246	861.784	1.240.580	1.240.580	1.240.580	1.220.380	1.212.200	1.148.840
P50n3p15t9v	1.487.444	955.951	1.397.120	1.397.120	1.397.120	1.375.610	1.156.380	1.146.280
P50n3p15t11v	1.402.785	807.048	1.219.830	1.219.830	1.219.830	1.219.830	1.137.120	1.042.100
P50n4p5t7v	441.109	333.285	460.644	414.822	366.419	374.261	375.134	358.623
P50n4p5t9v	597.091	494.613	681.331	541.139	562.592	533.143	531.276	533.884
P50n4p5t11v	602.110	475.263	635.962	635.962	515.059	505.148	513.242	509.042
P50n4p10t7v	956.102	627.398	885.836	885.836	885.836	770.687	827.100	781.330
P50n4p10t9v	1.312.692	901.283	1.211.010	1.211.010	1.029.850	1.074.910	1.140.020	1.129.040
P50n4p10t11v	1.197.104	796.470	1.093.160	1.093.160	911.582	970.951	1.093.160	1.089.900
P50n4p15t7v	2.006.724	1.374.240	1.783.740	1.783.740	1.783.740	1.777.820	1.775.850	1.686.410
P50n4p15t9v	1.683.038	1.055.440	1.454.250	1.454.250	1.215.740	1.454.250	1.454.110	1.384.980
P50n4p15t11v	2.014.575	1.305.000	1.758.750	1.758.750	1.758.750	1.649.150	1.758.750	1.555.740
P50n5p5t7v	600.330	482.461	644.828	559.732	526.620	510.148	511.067	514.125
P50n5p5t9v	695.681	555.604	751.063	627.399	751.063	597.440	584.614	751.063
P50n5p5t11v	681.704	528.628	698.492	608.841	586.457	562.254	561.621	698.492
P50n5p10t7v	1.436.290	962.170	1.247.600	1.247.600	1.247.600	1.247.600	1.179.920	1.162.090
P50n5p10t9v	1.475.592	1.108.840	1.402.430	1.402.430	1.402.430	1.402.430	1.302.360	1.301.520
P50n5p10t11v	1.307.607	915.536	1.175.980	1.175.980	1.047.630	1.153.290	1.175.980	1.125.390
P50n5p15t7v	2.115.728	1.409.850	1.844.430	1.844.430	1.844.430	1.815.260	1.844.430	1.844.430
P50n5p15t9v	2.108.270		1.653.910	1.653.910	1.653.910	1.653.910	1.653.470	1.652.020
P50n5p15t11v	2.051.843	1.339.930	1.770.250	1.770.250	1.770.250	1.769.180	1.770.250	1.770.250

Table E.4 – Linear programming gaps between 2COMM and RRMP.

#	INSTANCE	UB	LB		LP GAP (%)		Diff (%)
			2COMM	RRMP	RRMP	2COMM	
1	**P20n3p5t7v**	391.528	297.990	319.557	18,38	23,89	5,51
2	**P20n3p5t9v**	202.915	164.319	186.772	7,96	19,02	11,07
3	**P20n3p5t11v**	288.668	246.252	261.297	9,48	14,69	5,21
4	**P20n3p10t7v**	489.465	382.494	431.225	11,90	21,85	9,96
5	**P20n3p10t9v**	436.894	308.409	409.192	6,34	29,41	23,07
6	**P20n3p10t11v**	688.757	591.686	671.707	2,48	14,09	11,62
7	**P20n3p15t7v**	755.877	540.318	700.129	7,38	28,52	21,14
8	**P20n3p15t9v**	1.148.420	859.310	1.074.280	6,46	25,17	18,72
9	**P20n3p15t11v**	898.279	713.910	850.924	5,27	20,52	15,25
10	**P20n4p5t7v**	372.428	287.229	314.695	15,50	22,88	7,37
11	**P20n4p5t9v**	344.193	266.998	332.538	3,39	22,43	19,04
12	**P20n4p5t11v**	273.274	220.017	245.647	10,11	19,49	9,38
13	**P20n4p10t7v**	782.581	629.284	683.940	12,60	19,59	6,98
14	**P20n4p10t9v**	660.923	500.170	600.768	9,10	24,32	15,22
15	**P20n4p10t11v**	906.773	688.348	825.281	8,99	24,09	15,10
16	**P20n4p15t7v**	1.108.540	763.501	947.241	14,55	31,13	16,57
17	**P20n4p15t9v**	1.083.300	791.510	962.342	11,17	26,94	15,77
18	**P20n4p15t11v**	1.240.280	936.779	1.127.250	9,11	24,47	15,36
19	**P20n5p5t7v**	341.003	295.544	328.256	3,74	13,33	9,59
20	**P20n5p5t9v**	389.873	326.648	368.179	5,56	16,22	10,65
21	**P20n5p5t11v**	485.908	387.007	417.481	14,08	20,35	6,27
22	**P20n5p10t7v**	944.844	728.208	804.639	14,84	22,93	8,09
23	**P20n5p10t9v**	775.047	612.165	721.939	6,85	21,02	14,16
24	**P20n5p10t11v**	717.215	529.836	623.073	13,13	26,13	13,00
25	**P20n5p15t7v**	1.266.602	874.308	1.082.560	14,53	30,97	16,44
26	**P20n5p15t9v**	1.494.610	1.162.500	1.369.590	8,36	22,22	13,86
27	**P20n5p15t11v**	1.750.670	1.449.880	1.635.030	6,61	17,18	10,58
28	**P30n3p5t7v**	213.712	163.465	189.713	11,23	23,51	12,28
29	**P30n3p5t9v**	391.676	304.526	324.450	17,16	22,25	5,09
30	**P30n3p5t11v**	367.768	277.559	312.275	15,09	24,53	9,44
31	**P30n3p10t7v**	688.428	484.644	552.102	19,80	29,60	9,80
32	**P30n3p10t9v**	667.572	463.102	543.126	18,64	30,63	11,99
33	**P30n3p10t11v**	654.725	448.232	609.833	6,86	31,54	24,68
34	**P30n3p15t7v**	880.395	514.078	761.807	13,47	41,61	28,14
35	**P30n3p15t9v**	1.023.110	661.370	867.381	15,22	35,36	20,14
36	**P30n3p15t11v**	1.184.380	846.593	1.083.730	8,50	28,52	20,02

Table E.5 – Linear programming gaps between 2COMM and RRMP.

#	INSTANCE	UB	LB		LP GAP (%)		Diff (%)
			2COMM	RRMP	RRMP	2COMM	
37	**P30n4p5t7v**	410.940	318.423	350.592	14,69	22,51	7,83
38	**P30n4p5t9v**	537.101	432.958	488.135	9,12	19,39	10,27
39	**P30n4p5t11v**	495.875	412.699	448.750	9,50	16,77	7,27
40	**P30n4p10t7v**	691.145	484.873	557.528	19,33	29,84	10,51
41	**P30n4p10t9v**	739.013	534.998	608.693	17,63	27,61	9,97
42	**P30n4p10t11v**	824.586	580.149	737.904	10,51	29,64	19,13
43	**P30n4p15t7v**	1.175.321	806.908	1.070.500	8,92	31,35	22,43
44	**P30n4p15t9v**	1.180.465	775.141	937.109	20,62	34,34	13,72
45	**P30n4p15t11v**	1.293.592	834.354	1.109.930	14,20	35,50	21,30
46	**P30n5p5t7v**	415.180	343.669	384.213	7,46	17,22	9,77
47	**P30n5p5t9v**	396.522	322.253	385.793	2,71	18,73	16,02
48	**P30n5p5t11v**	481.502	393.899	429.706	10,76	18,19	7,44
49	**P30n5p10t7v**	1.030.250	759.197	861.994	16,33	26,31	9,98
50	**P30n5p10t9v**	958.748	680.593	850.361	11,31	29,01	17,71
51	**P30n5p10t11v**	1.028.217	760.388	902.479	12,23	26,05	13,82
52	**P30n5p15t7v**	1.338.394	881.433	1.128.950	15,65	34,14	18,49
53	**P30n5p15t9v**	1.816.277	1.225.560	1.502.160	17,29	32,52	15,23
54	**P30n5p15t11v**	1.682.814	1.163.680	1.394.880	17,11	30,85	13,74
55	**P40n3p5t7v**	444.187	330.565	374.819	15,62	25,58	9,96
56	**P40n3p5t9v**	249.859	183.788	211.336	15,42	26,44	11,03
57	**P40n3p5t11v**	380.325	308.907	337.528	11,25	18,78	7,53
58	**P40n3p10t7v**	731.204	486.765	560.186	23,39	33,43	10,04
59	**P40n3p10t9v**	994.609	707.615	952.824	4,20	28,85	24,65
60	**P40n3p10t11v**	923.388	690.991	899.978	2,54	25,17	22,63
61	**P40n3p15t7v**	1.227.324	876.775	1.053.510	14,16	28,56	14,40
62	**P40n3p15t9v**	781.594	470.927	675.677	13,55	39,75	26,20
63	**P40n3p15t11v**	1.045.710	689.736	1.009.240	3,49	34,04	30,55
64	**P40n4p5t7v**	318.810	247.166	277.104	13,08	22,47	9,39
65	**P40n4p5t9v**	380.047	319.847	349.184	8,12	15,84	7,72
66	**P40n4p5t11v**	474.411	358.988	427.550	9,88	24,33	14,45
67	**P40n4p10t7v**	794.697	516.326	697.640	12,21	35,03	22,82
68	**P40n4p10t9v**	1.007.635	691.271	894.533	11,22	31,40	20,17
69	**P40n4p10t11v**	992.252	672.186	910.682	8,22	32,26	24,04
70	**P40n4p15t7v**	1.554.534	911.707	1.239.890	20,24	41,35	21,11
71	**P40n4p15t9v**	1.339.051	823.494	1.159.650	13,40	38,50	25,10
72	**P40n4p15t11v**	1.731.838	1.173.510	1.498.500	13,47	32,24	18,77

Table E.6 – Linear programming gaps between 2COMM and RRMP.

#	INSTANCE	UB	LB		LP GAP (%)		Diff (%)
			2COMM	RRMP	RRMP	2COMM	
73	**P40n5p5t7v**	420.504	347.023	381.398	9,30	17,47	8,17
74	**P40n5p5t9v**	524.461	432.411	480.258	8,43	17,55	9,12
75	**P40n5p5t11v**	522.045	420.379	478.495	8,34	19,47	11,13
76	**P40n5p10t7v**	1.244.820	902.930	1.137.100	8,65	27,47	18,81
77	**P40n5p10t9v**	1.555.132	1.173.900	1.411.580	9,23	24,51	15,28
78	**P40n5p10t11v**	1.343.143	937.402	1.154.420	14,05	30,21	16,16
79	**P40n5p15t7v**	1.950.402	1.294.080	1.646.040	15,61	33,65	18,05
80	**P40n5p15t9v**	2.361.557	1.683.920	2.061.400	12,71	28,69	15,98
81	**P40n5p15t11v**	2.124.965	1.373.290	1.779.400	16,26	35,37	19,11
82	**P50n3p5t7v**	400.692	305.876	359.001	10,40	23,66	13,26
83	**P50n3p5t9v**	524.467	431.451	476.074	9,23	17,74	8,51
84	**P50n3p5t11v**	568.760	448.696	497.803	12,48	21,11	8,63
85	**P50n3p10t7v**	647.029	413.584	589.366	8,91	36,08	27,17
86	**P50n3p10t9v**	1.129.270	776.136	1.077.380	4,60	31,27	26,68
87	**P50n3p10t11v**	801.108	493.255	764.509	4,57	38,43	33,86
88	**P50n3p15t7v**	1.344.246	861.784	1.240.580	7,71	35,89	28,18
89	**P50n3p15t9v**	1.487.444	955.951	1.397.120	6,07	35,73	29,66
90	**P50n3p15t11v**	1.402.785	807.048	1.219.830	13,04	42,47	29,43
91	**P50n4p5t7v**	441.109	333.285	414.822	5,96	24,44	18,48
92	**P50n4p5t9v**	597.091	494.613	562.592	5,78	17,16	11,39
93	**P50n4p5t11v**	602.110	475.263	515.059	14,46	21,07	6,61
94	**P50n4p10t7v**	956.102	627.398	885.836	7,35	34,38	27,03
95	**P50n4p10t9v**	1.312.692	901.283	1.211.010	7,75	31,34	23,59
96	**P50n4p10t11v**	1.197.104	796.470	1.093.160	8,68	33,47	24,78
97	**P50n4p15t7v**	2.006.724	1.374.240	1.783.740	11,11	31,52	20,41
98	**P50n4p15t9v**	1.683.038	1.055.440	1.454.250	13,59	37,29	23,70
99	**P50n4p15t11v**	2.014.575	1.305.000	1.758.750	12,70	35,22	22,52
100	**P50n5p5t7v**	600.330	482.461	559.732	6,76	19,63	12,87
101	**P50n5p5t9v**	695.681	555.604	627.399	9,82	20,14	10,32
102	**P50n5p5t11v**	681.704	528.628	608.841	10,69	22,45	11,77
103	**P50n5p10t7v**	1.436.290	962.170	1.247.600	13,14	33,01	19,87
104	**P50n5p10t9v**	1.475.592	1.108.840	1.402.430	4,96	24,85	19,90
105	**P50n5p10t11v**	1.307.607	915.536	1.175.980	10,07	29,98	19,92
106	**P50n5p15t7v**	2.115.728	1.409.850	1.844.430	12,82	33,36	20,54
107	**P50n5p15t9v**	2.108.270	1.228.240	1.653.910	21,55	41,74	20,19
108	**P50n5p15t11v**	2.051.843	1.339.930	1.770.250	13,72	34,70	20,97