CIRRELT

CIRRELT-2022-23

A Three-Front Parallel Branch-and-Cut Algorithm for Production and Inventory Routing Problems

Cleder M. Schenekemberg
Thiago A. Guimarães
Antonio A. Chaves
Leandro C. Coelho

August 2022

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2022-007.

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

A Three-Front Parallel Branch-and-Cut Algorithm for Production and Inventory Routing Problems

Cleder M. Schenekemberg[1, *], Thiago A. Guimarães[2], Antonio A. Chaves[3], Leandro C. Coelho[4]

1. Federal University of São Paulo (UNIFESP), São José dos Campos, Brazil, Aeronautics Institute of Technology (ITA), São José dos Campos, Brazil
2. Research Group of Technology Applied to Optimization (GTAO), Federal University of Paraná (UFPR), Curitiba, Brazil
3. Federal University of São Paulo (UNIFESP), São José dos Campos, Brazil
4. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Canada Research Chair in Integrated Logistics

Abstract. Production and inventory routing problems consider a single-product supply chain operating under a vendor-managed inventory system. A plant creates a production plan and vehicle routes over a planning horizon to replenish its customers at minimum cost. In this paper, we present two- and three-index formulations, implement a branch-and-cut algorithm based on each formulation, and introduce a local search matheuristic-based algorithm to solve the problem. In order to combine all benefits of each algorithm, we design a parallel framework to integrate all three fronts, called the three-front parallel branch-and-cut algorithm (3FP-B&C). We assess the performance of our method on well-known benchmark instances of the inventory-routing problem (IRP) and the production-routing problem (PRP). The results show that our 3FP-B&C outperforms by far other approaches from the literature. For the 956 feasible small-size IRP instances, our method has proved optimality for 769, being the first exact algorithm to solve all instances with up to two vehicles. 3FP-B&C has found 949 best-known solutions (BKS), with 139 new BKS (NBKS). For the large-size set, our method provides two new optimal solutions (OPT), and found 82% of BKS, being 70% of NBKS for instances with up to 5 vehicles. This result is more than twice the number of BKS considering all heuristic methods from the literature combined. Finally, our 3FP-B&C found the best lower bounds (BLB) for 1169/1316 instances, outperforming all previous exact algorithms. On the PRP, our method obtained 278 OPT out of the 336 instances of Adulyasak et al. (2014) being 19 new ones, in addition to 335 BKS (74 NBKS) and 313 BLB (52 new ones). On the PRP set of Archetti et al. (2011), our algorithm finds 1105 BKS out of 1440 instances, with 584 NBKS. Besides that, our 3FP-B&C is the first exact algorithm to solve the instances with an unlimited fleet, providing the first lower bounds for this subset, with an average optimality gap of 0.61%. We also address the very large-size instance set of Boudia et al. (2007), the second exact algorithm to address this set, outperforming the previous approach by far. Finally, a comparative analysis of each front shows the gains of the integrated approach.

Keywords: Production-routing, inventory-routing, branch-and-cut, matheuristic, parallel processing.

Acknowledgements. Cleder M. Schenekemberg was supported by the São Paulo Research Foundation (FAPESP) under grant 2020/07145-8. Antonio A. Chaves was supported by FAPESP under grants 2018/15417-8 and 2016/01860-1, and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under grants 312747/2021-7 and 405702/2021-3. Leandro C. Coelho was supported by the Natural Sciences and Engineering Council of Canada (NSERC) under grant 2019-00094. We also thank Compute Canada for providing parallel computing facilities.

---

* Corresponding author: cledercms@hotmail.com

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2022

1. Introduction

Supply chain coordination is central to cost reduction and performance improvement. From an operational perspective, it requires joint production, inventory, and distribution planning, and may involve collaboration among different companies. In this context, vendor-managed inventory (VMI) is a successful practice to enable coordination [27]. The inventory-routing problem (IRP) optimizes the VMI operations, while the production-routing problem (PRP) is more general and includes a lot-sizing problem in the IRP scope. Due to their practical relevance, both problems have attracted wide literature attention over the last decades, and a myriad of solution algorithms and applications are reported [2, 27].

In this paper, we address the standard IRP and PRP [27]. Despite numerous variants [7], the basic IRP and PRP remain methodologically challenging. Their strong combinatorial nature is a big obstacle for exact methods, which often cannot find feasible solutions for large instances in reasonable time. On the other hand, the fact that the vast majority of IRP and PRP instances remain open motivates the research toward exact algorithms development.

In a recent paper, Manousakis et al. [34] proposed a two-commodity flow formulation and a branch-and-cut (B&C) for the IRP. As this formulation is not affected by the number of vehicles, the B&C was able to handle larger and more complicated instances [9, 10], providing several improvements in terms of upper and lower bounds. However, a comparison with the three-index B&C from Guimarães et al. [31] does not establish a dominance; in some cases, the three-index B&C performed better than the two-index B&C.

A promising alternative to purely exact algorithms explores parallel processing of exact and heuristic algorithms, where the first one is more focused on lower bounds, while the second one is dedicated to improving upper bounds. This technique has been successfully employed in related problems by Adulyasak et al. [4] and Schenekemberg et al. [39, 40]. In this sense, some advantages can be gained when two formulations are combined. At the same time, parallel processing with heuristic algorithms helps B&C find better solutions.

This paper introduces a three-front parallel B&C (3FP-B&C) algorithm to solve the IRP and the PRP. In our 3FP-B&C, a two- and a three-index B&C (2I-B&C and 3I-B&C) and a matheuristic front run independently, all sharing information and stopping criteria. We assess the performance of our 3FP-B&C by solving the small- and large-size IRP instances from Archetti et al. [9] and Archetti et al. [10], respectively. For the PRP, we consider the small-size set of Adulyasak et al. [3], the medium- and large-size set of Archetti et al. [8], and the very large-size set of Boudia et al. [22]. We extensively compare our 3FP-B&C with all state-of-the-art exact and heuristic algorithms from the literature. The scientific contributions of this paper are to:

1. introduce two- and three-index formulations for the problems and to design a 2I-B&C and a 3I-B&C algorithm.

2. design a local search matheuristic (LSM) in two steps. The first one repairs and polishes solutions, even partial ones, provided by both 2I-B&C and 3I-B&C. The second step intensifies the improvement efforts by recursively applying a mixed-integer programming (MIP) procedure which reorganizes the delivery routes and optimizes inventory and production decisions.
3. develop a parallel framework so that 3FP-B&C benefits from the strengths of each front. It integrates 2I-B&C, 3I-B&C, and LSM by sharing to all fronts any solution improvement as soon as it is found. To our knowledge, no such approach was previously introduced in the literature.
4. provide extensive computational comparisons. Our 3FP-B&C is the first method to solve the PRP instances of Archetti et al. [8] exactly when the fleet size is unlimited, providing the first lower bounds for this subset. Computational experiments also provide several new optimal solutions for open instances, both for the IRP and the PRP, besides finding numerous best lower and upper bounds, outperforming by far all other approaches from the literature.

The remainder of the paper is organized as follows. Section 2 provides a relevant literature revision, while Section 3 formally describes the problems and introduces the formulations. Section 4 details the 2I-B&C, 3I-B&C, LSM, and the general framework of our method. In Section 5, we discuss the results of extensive computational experiments performed to assess the quality of our 3FP-B&C, while Section 6 presents our conclusions.

2. Literature review

Since the IRP was introduced by Bell et al. [19] and PRP was first discussed by Chandra and Fisher [23], their challenging nature provided ample space for the development of exact and many approximate methods.

Among the heuristics for the IRP, Archetti et al. [10] introduced the first matheuristic for the single-vehicle IRP by combining tabu search (TS) with MIP-based procedures. The authors proposed a large set of benchmark instances for the problem. Another matheuristic was proposed by Adulyasak et al. [3] based on an adaptive large neighborhood search (ALNS), considering both two- and three-index formulations for the IRP. Later, Archetti et al. [11] extended the algorithm from [10] for the multi-vehicle IRP, obtaining 92% of the best-known solutions (BKS) for the large instance set.

Alvarez et al. [5] designed an iterated local search (ILS) and a simulated annealing (SA) algorithm, reporting results for both small and large instance sets. Chitsaz et al. [24] proposed a three-phase decomposition matheuristic for the assembly routing problem, a generalization of the IRP and PRP. This algorithm was particularly successful in large-scale multi-vehicle instances. The ILS from [5] was updated by Alvarez et al. [6] for an IRP with perishable products. The authors enhanced the ILS with two MIPs, providing a few

BKS for small IRP instances. A similar idea was employed by Diniz et al. [30], by proposing an ILS with a randomized variable neighborhood descent, and the method found several BKS for the small instances. In the same line as their previous matheuristic, Archetti et al. [12] proposed a kernel search embedded in a sequence of MIP subproblems, each restricted to a few variables. The results outperform other heuristic algorithms in terms of average solution quality.

Generally, the previous matheuristics solve several MIPs of a restricted version of the IRP. Differing from this scheme, Solyalı and Süral [42], introduce a new strategy where the matheuristic relies on a sequential solution of three MIPs only once, followed by routing procedures. The first two MIPs were adapted from [43] and construct a good feasible solution for the IRP. The third one uses actual transportation costs to assign customers to vehicle routes. The matheuristic found many new BKS for both small and large instances.

Recently, a new matheuristic was introduced by Vadseth et al. [44] that iterates between routing operators and a path-flow formulation to solve the associated inventory problem. The results were remarkable, especially on large instances, having found 178 new BKS in a set of 240 tested instances. The latest improvements were made by Vadseth et al. [45], who address a multi-start route improving matheuristic for the IRP and the PRP. A valid solution to the problem is constructed in each restart, and its routes are used on a novel path-flow model, which is solved for several iterations, performing some route changes. The matheuristic provided 93 BKS out of 240 large instances.

Regarding exact algorithms for the IRP, Archetti et al. [9] introduced the first B&C algorithm for the single-vehicle case. Later, Coelho and Laporte [25] adapted this instance set to the multi-vehicle IRP and proposed a new B&C algorithm based on a three-index formulation. Later on, Desaulniers et al. [29] proposed a branch-and-price-and-cut (B&PC) algorithm, introducing new cuts and providing good results for small instances, especially for those with four and five vehicles. Some modifications to the three-index formulation were introduced by Avella et al. [16, 17], but these algorithms were limited to solving only small to medium size instances of Archetti et al. [9], and have not been tested on the large instance set of Archetti et al. [10]. More recently, Guimarães et al. [31] exploited a three-index formulation improving the traditional B&C scheme with two mechanisms to recover and improve primal solutions. Their B&C found 129 BKS for the small set [9], and proved optimality for the first time for an instance with 100 customers and six periods of the large set [10]. Finally, Manousakis et al. [34] proposed a B&C based on a two-commodity flow formulation for the IRP. The authors also employ a TS to provide upper bounds used as MIP starts. The method improved 139 BKS for the large set of 300 instances [10].

The inherent difficulties of the PRP prevent the development of exact algorithms. Archetti et al. [8] proposed an algorithm for the single-vehicle case. Adulyasak et al. [3] designed B&C algorithms from two- and three-index formulations, where an ALNS provides an initial solution. More recently, Zhang et al. [48] proposed

a Benders decomposition approach, and finally, Schenekemberg et al. [40] designed a parallel B&C able to deal with broader features. Three sets of instances are available in the literature. A set of medium and large instances with an unlimited fleet was proposed by Archetti et al. [8], but no exact methods solve it with two or more vehicles. Adulyasak et al. [3] proposed a set of small and medium-sized ones derived from [8, 9], also solved by Schenekemberg et al. [40] and Zhang et al. [48]. Finally, the oldest and most challenging set was proposed by Boudia et al. [22], which is classified as very large instances. To the best of our knowledge, only Schenekemberg et al. [40] have solved it exactly, providing the first known lower bounds for instances with up to 100 customers.

Regarding approximate algorithms, several metaheuristics have been designed to solve the PRP, and we highlight the memetic algorithm of Boudia and Prins [21] and the variable neighborhood search heuristic of Qiu et al. [37]. However, matheuristics algorithms perform better and usually decompose the problem in a production subproblem (PS) and a routing subproblem (RS). This approach is employed by Archetti et al. [8] and Absi et al. [1], where a PS provides production and distribution schedules, and an RS is responsible for creating the vehicle routes. Solyalı and Süral [43] also used the decomposition approach in five MIP phases. The first two phases solve the PS, the third phase solves the RS, and the fourth and fifth phases repair and improve infeasible solutions generated after solving the RS. Later, Avci and Yildiz [15] introduced a multi-start matheuristic, where at each re-start, a PS is solved by creating a random distribution schedule and an associated production schedule. Routes are then heuristically generated according to the latest schedules. Li et al. [33] proposed a three-level matheuristic, where PS and RS are solved in the first and second levels, while the solution is improved in the third level by a fix-and-optimized scheme. More recently, Manousakis et al. [35] developed a two-phase metaheuristic, where a PS is solved in the first phase while an RS is solved in the second phase by a local search procedure, which oscillates between the feasible and the infeasible solution space. The matheuristics of Adulyasak et al. [3] and Chitsaz et al. [24] have solved the instances of Archetti et al. [8] and Boudia et al. [22]. So far, the best results for the PRP instances were obtained by the matheuristic of Vadseth et al. [45]. The authors reported 516 BKS out of 960 for the small instances of [8] and 75 BKS out of 90 instances for the large set [22].

3. Problem description and mathematical formulations

The network is defined by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The vertex set is $\mathcal{V} = \{0, 1, ..., n, n+1\}$, where vertices 0 and $n+1$ represent the plant, and $\mathcal{V}' = \{1, ..., n\}$ denotes n customers geographically dispersed. The edge set is given by $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}, i < j\}$. Each customer $i \in \mathcal{V}'$ has an inventory capacity U_i and a constant demand d_i^t over a discrete and finite planning horizon $t \in \mathcal{T} = \{1, ..., \rho\}$, which must be satisfied by its inventory. In $t = 0$, both the plant and the customers have an initial inventory $I_i^0, i \in \mathcal{V}\backslash\{n+1\}$. A

finite and homogeneous fleet $\mathcal{K}$ with capacity Q is housed at the plant and is available to perform deliveries to the customers. Vehicle routes start and end at the plant, and incur transportation costs c_{ij} proportional to the edge distances $(i, j) \in \mathcal{E}$. For the IRP, at each period t, a certain amount of product r^t is made available at the plant, which has to decide which customers to serve, how much to deliver to them, and how to combine deliveries into vehicle routes. The product can be stored either at the plant, limited by U_0, or at the customers, and the remaining inventory I_i^t at the end of a period incurs a unit holding cost h_i (with $U_{n+1} = h_{n+1} = I_{n+1}^0 = 0$). In the case of the PRP, one decides how much to produce at each period (p^t). The production capacity is C, and if a production batch takes place, fixed setup γ and unit production costs μ are incurred. Regarding customer replenishment, we consider both maximum level (ML) and order-up-to level (OU) policies. Following ML, the plant is free to schedule any non-capacity-exceeding delivery to the customers. Under the OU policy, the plant has to fill the inventory capacity whenever a customer is served.

3.1. Two-index formulation

We now present the mathematical formulation used by our 2I-B&C to solve the problems. We generalize the models of Manousakis et al. [34, 35] for the IRP and PRP. The continuous variables are:

- I_i^t: inventory level at the end of period $t \in \mathcal{T} \cup \{0\}$ for each vertex $i \in \mathcal{V}$.
- q_i^t: quantity delivered to customer $i \in \mathcal{V}'$ in $t \in \mathcal{T}$.
- p^t: quantity produced in $t \in \mathcal{T}$.
- f_{ij}^t: flow variable indicating the vehicle load traversing edge (i, j) in period $t \in \mathcal{T}$.
- f_{ji}^t: flow variable indicating the residual vehicle capacity traversing edge (i, j) in period $t \in \mathcal{T}$.

In addition, the following binary variables are used:

- $y^t = 1$, if a production batch occurs in period $t \in \mathcal{T}$, 0 otherwise.
- $z_i^t = 1$, if customer $i \in \mathcal{V}'$ is visited in $t \in \mathcal{T}$, 0 otherwise.
- $x_{ij}^t = 1$, if a vehicle travels between vertices i and j in $t \in \mathcal{T}$, 0 otherwise.

The model is defined by (1)−(20).

$$\min \sum_{i \in \mathcal{V}} \sum_{t \in \mathcal{T} \cup \{0\}} h_i \cdot I_i^t + \sum_{t \in \mathcal{T}} \mu \cdot p^t + \sum_{t \in \mathcal{T}} \gamma \cdot y^t + \sum_{t \in \mathcal{T}} \sum_{(i,j) \in \mathcal{E}} c_{ij} \cdot x_{ij}^t \tag{1}$$

subject to

$$I_0^t = I_0^{t-1} + p^t - \sum_{i \in \mathcal{V}'} q_i^t \qquad t \in \mathcal{T} \tag{2}$$

$$I_i^t = I_i^{t-1} + q_i^t - d_i^t \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{3}$$

$$q_i^t \leq U_i - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{4}$$

$$q_i^t \leq \min \left\{ Q, U_i, \sum_{t'=t}^{\rho} d_i^{t'} \right\} \cdot z_i^t \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{5}$$

$$p^t \leq \min \left\{ C, \sum_{i \in \mathcal{V}'} \sum_{t'=t}^{\rho} d_i^{t'} \right\} \cdot y^t \qquad t \in \mathcal{T} \tag{6}$$

$$\sum_{j \in \mathcal{V}'} x_{0j}^t = \sum_{i \in \mathcal{V}'} x_{i,n+1}^t \qquad t \in \mathcal{T} \tag{7}$$

$$\sum_{j \in \mathcal{V}'} x_{0j}^t \leq |\mathcal{K}| \qquad t \in \mathcal{T} \tag{8}$$

$$\sum_{\substack{j \in \mathcal{V} \\ i<j}} x_{ij}^t + \sum_{\substack{j \in \mathcal{V} \\ i>j}} x_{ji}^t = 2 \cdot z_i^t \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{9}$$

$$f_{ij}^t + f_{ji}^t = Q \cdot x_{ij}^t \qquad (i,j) \in \mathcal{E}, t \in \mathcal{T} \tag{10}$$

$$\sum_{\substack{j \in \mathcal{V} \\ i \neq j}} f_{ij}^t = Q \cdot z_i^t - q_i^t \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{11}$$

$$\sum_{j \in \mathcal{V}'} f_{0j}^t = \sum_{i \in \mathcal{V}'} q_i^t \qquad t \in \mathcal{T} \tag{12}$$

$$\sum_{i \in \mathcal{V}'} f_{i,n+1}^t = 0 \qquad t \in \mathcal{T} \tag{13}$$

$$0 \leq I_i^t \leq U_i \qquad i \in \mathcal{V}, t \in \mathcal{T} \cup \{0\} \tag{14}$$

$$z_i^t \in \{0,1\} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{15}$$

$$x_{ij}^t \in \{0,1\} \qquad (i,j) \in \mathcal{E}, t \in \mathcal{T} \tag{16}$$

$$0 \leq q_i^t \leq \min \left\{ Q, U_i, \sum_{t'=t}^{\rho} d_i^{t'} \right\} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{17}$$

$$0 \leq f_{ij}^t \leq Q \qquad i \neq j \in \mathcal{V}, t \in \mathcal{T} \tag{18}$$

$$0 \leq p^t \leq \min \left\{ C, \sum_{i \in \mathcal{V}'} \sum_{t'=t}^{\rho} d_i^{t'} \right\} \qquad t \in \mathcal{T} \tag{19}$$

$$y^t \in \{0,1\} \qquad t \in \mathcal{T}. \tag{20}$$

The objective function (1) minimizes the total costs given by inventory costs at the plant and at the cus-

tomers, fixed and variable production costs, and transportation costs. Constraints (2) and (3) balance the inventory flow at the plant and the customers, while (4) formulates the ML policy for the customers, considering that deliveries occur before the demand takes place. Constraints (5) link visit and delivery variables for each customer $i \in \mathcal{V}'$ and period $t \in \mathcal{T}$, while constraints (6) do the same for production setup and production quantity variables. Constraints (7) and (8) ensure that the number of vehicles leaving and returning to the plant does not exceed the fleet size, while (9) are customer degree constraints. Constraints (10) balance the load and residual capacity of the vehicle if it travels between the vertices i and j. Constraints (11) impose that the delivered quantity to customer i must equal the sum of flows originating from node j. Constraints (12) compute the total amount of product leaving the plant in $t \in \mathcal{T}$, while constraints (13) ensure that vehicles return empty to the plant, and both the constraints (10)–(13) eliminate subtours. The variables domain is defined by constraints (14)−(20).

Formulation (1)−(20) address the PRP instances of Adulyasak et al. [3] under the ML policy. In order to handle the OU policy, constraints (21) are added, and constraints (5),(6), (17), and (19) are replaced by (22)−(25).

$$q_i^t \geq U_i \cdot z_i^t - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{21}$$

$$q_i^t \leq \min\{Q, U_i\} \cdot z_i^t \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{22}$$

$$p^t \leq C \cdot y^t \qquad t \in \mathcal{T} \tag{23}$$

$$0 \leq q_i^t \leq \min\{Q, U_i\} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{24}$$

$$0 \leq p^t \leq C \qquad t \in \mathcal{T}. \tag{25}$$

In particular, constraints (22)−(25) are also required under the ML policy when the inventory cost at the plant is higher than at the customers since keeping inventory at the plant at the end of the planning horizon corresponds to a sub-optimal solution.

To solve the IRP instances of [9, 10] under the ML policy, the production setup variables y^t must be removed, the quantity produced becomes $p^t = r^t$, and constraints (22)−(24) be added. Moreover, Archetti et al. [8] and Boudia et al. [22] consider that customers are served at the end of the period, after their demand takes place. Thus, constraints (4) are replaced by (26) and the OU policy formulated with (27):

$$q_i^t \leq U_i + d_i^t - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{26}$$

$$q_i^t \geq U_i \cdot z_i^t + d_i^t - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T}. \tag{27}$$

Additionally, Boudia et al. [22] consider that the quantity produced in t is made available for delivery from the next period $t+1$. Thus, the instance set introduced by the authors requires that all demands in $t=1$ be satisfied from I_0^0, and this initial inventory is precisely the sum of all customer demands in $t=1$. As pointed out by Adulyasak et al. [3] and Manousakis et al. [35], no production can occur in $t=1$, setting $y^1 = p^1 = 0$.

Several valid inequalities (VI) can be used to reinforce this formulation. For the sake of conciseness, they are presented and described in Appendix A.

3.2. Three-index formulation

We now describe the three-index mathematical formulation used by our 3I-B&C to solve the problems. We generalize the multi-vehicle IRP models of Coelho and Laporte [25, 26], and the multi-vehicle PRP model of Adulyasak et al. [3]. In the formulation presented below, $\mathcal{V} = \{0, 1, ..., n\}$ defines the set of vertices, where the plant is represented by vertex 0, while customers are defined by the subset $\mathcal{V}' = \{1, ..., n\}$. The new variables are:

- q_i^{kt}: quantity delivered to customer $i \in \mathcal{V}'$ by vehicle $k \in \mathcal{K}$ in period $t \in \mathcal{T}$.
- $z_i^{kt} = 1$ if customer $i \in \mathcal{V}'$ is visited by vehicle $k \in \mathcal{K}$ in $t \in \mathcal{T}$, 0 otherwise.
- $x_{ij}^{kt} = 1$ if vehicle k travels between customers i and j in $t \in \mathcal{T}$, 0 otherwise.
- $x_{0j}^{kt} \in \{0, 1, 2\}$ where $x_{0j}^{kt} = 1$ indicates that vehicle k travels between vertices 0 and j in $t \in \mathcal{T}$. If $x_{0j}^{kt} = 2$, a round trip is defined, 0 otherwise.

The three-index formulation model is defined by (28)−(44):

$$\min \sum_{i \in \mathcal{V}} \sum_{t \in \mathcal{T} \cup \{0\}} h_i \cdot I_i^t + \sum_{t \in \mathcal{T}} \mu \cdot p^t + \sum_{t \in \mathcal{T}} \gamma \cdot y^t + \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{E}} c_{ij} \cdot x_{ij}^{kt} \tag{28}$$

subject to

$$I_0^t = I_0^{t-1} + p^t - \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}'} q_i^{kt} \qquad t \in \mathcal{T} \tag{29}$$

$$I_i^t = I_i^{t-1} + \sum_{k \in \mathcal{K}} q_i^{kt} - d_i^t \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{30}$$

$$\sum_{k\in\mathcal{K}} q_i^{kt} \leq U_i - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{31}$$

$$q_i^{kt} \leq \min\left\{Q, U_i, \sum_{t'=t}^{\rho} d_i^{t'}\right\} \cdot z_i^{kt} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{32}$$

$$p^t \leq \min\left\{C, \sum_{i\in\mathcal{V}'} \sum_{t'=t}^{\rho} d_i^{t'}\right\} \cdot y^t \qquad t \in \mathcal{T} \tag{33}$$

$$\sum_{i\in\mathcal{V}'} q_i^{kt} \leq Q \cdot z_0^{kt} \qquad k \in \mathcal{K}, t \in \mathcal{T} \tag{34}$$

$$\sum_{k\in\mathcal{K}} z_i^{kt} \leq 1 \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{35}$$

$$\sum_{\substack{j\in\mathcal{V}\\ i<j}} x_{ij}^{kt} + \sum_{\substack{j\in\mathcal{V}\\ i>j}} x_{ji}^{kt} = 2 \cdot z_i^{kt} \qquad i \in \mathcal{V}, k \in \mathcal{K}, t \in \mathcal{T} \tag{36}$$

$$\sum_{i\in\mathcal{S}} \sum_{\substack{j\in\mathcal{S}\\ i<j}} x_{ij}^{kt} \leq \sum_{i\in\mathcal{S}} z_i^{kt} - z_m^{kt} \qquad \mathcal{S} \subseteq \mathcal{V}', |\mathcal{S}| \geq 2, m \in \mathcal{S}, k \in \mathcal{K}, t \in \mathcal{T} \tag{37}$$

$$0 \leq I_i^t \leq U_i \qquad i \in \mathcal{V}, t \in \mathcal{T} \cup \{0\} \tag{38}$$

$$z_i^{kt} \in \{0,1\} \qquad i \in \mathcal{V}, k \in \mathcal{K}, t \in \mathcal{T} \tag{39}$$

$$x_{ij}^{kt} \in \{0,1\} \qquad i,j \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{40}$$

$$x_{0j}^{kt} \in \{0,1,2\} \qquad j \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{41}$$

$$0 \leq q_i^{kt} \leq \min\left\{Q, U_i, \sum_{t'=t}^{\rho} d_i^{t'}\right\} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{42}$$

$$0 \leq p^t \leq \min\left\{C, \sum_{i\in\mathcal{V}'} \sum_{t'=t}^{\rho} d_i^{t'}\right\} \qquad t \in \mathcal{T} \tag{43}$$

$$y^t \in \{0,1\} \qquad t \in \mathcal{T}. \tag{44}$$

The objective function (28) and constraints (29)−(33) are similar to (1) and (2)−(6) of Section 3.1. Constraints (34) ensure that the total quantity delivered does not exceed the vehicle capacity, while constraints (35) avoid split deliveries. Linking and subtour elimination conditions are imposed by (36) and (37), and the variables domain is defined by constraints (38)−(44).

As in Section 3.1, formulation (28)−(44) address the PRP instances of Adulyasak et al. [3] under the ML policy. To consider the OU policy, the following constraints are required:

$$\sum_{k\in\mathcal{K}} q_i^{kt} \geq U_i \cdot \left(\sum_{k\in\mathcal{K}} z_i^{kt}\right) - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T}. \tag{45}$$

In addition, the OU policy (or when $h_0 > h_i$ in the ML policy) also requires some modifications to constraints

(32) and (33), and on the variables domain (42) and (43):

$$q_i^{kt} \leq \min\{Q, U_i\} \cdot z_i^{kt} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{46}$$

$$p^t \leq C \cdot y^t \qquad t \in \mathcal{T} \tag{47}$$

$$0 \leq q_i^{kt} \leq \min\{Q, U_i\} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{48}$$

$$0 \leq p^t \leq C \qquad t \in \mathcal{T}. \tag{49}$$

To handle the IRP instances of [9, 10] under the ML policy, the production setup variables y^t must be removed, the quantity produced must be $p^t = r^t$, and constraints (46)−(48) must be added.

According to the timing of activities considered by Archetti et al. [8] and Boudia et al. [22], constraints (31) should be replaced by:

$$\sum_{k \in \mathcal{K}} q_i^{kt} \leq U_i + d_i^t - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T}. \tag{50}$$

The OU policy is reformulated according to:

$$\sum_{k \in \mathcal{K}} q_i^{kt} \geq \left(U_i + d_i^t\right) \cdot \left(\sum_{k \in \mathcal{K}} z_i^{kt}\right) - I_i^{t-1} \qquad i \in \mathcal{V}', t \in \mathcal{T}. \tag{51}$$

As pointed out in Section 3.1, the PRP instances of Boudia et al. [22] require $y^1 = p^1 = 0$, avoiding production in $t = 1$.

The set of VIs of the three-index formulation is presented in Appendix B.

4. A three-front parallel branch-and-cut algorithm

Our 3FP-B&C runs continuously on three independent fronts in an integrated framework. A controller receives all solutions found by any front and shares them with the two other fronts; the controller also handles all stopping criteria. In this framework, both B&C algorithms are responsible for proving optimality, improving upper bounds, and finding candidate solutions to be handled by the LSM front. When an optimal solution is proved by 2I-B&C (3I-B&C), the controller aborts the 3I-B&C (2I-B&C) and the LSM. In addition, each B&C also updates the controller with new upper bounds and partial solution candidates

obtained from each explored node from the search tree, added to a list $\mathcal{R}$. Each element of $\mathcal{R}$ contains the quantities delivered to customers.

On the other hand, the LSM front operates in two independent phases. In the first one, LSM completes and polishes an element from $\mathcal{R}$ by solving a capacitated vehicle routing problem (CVRP) and a lot-sizing and scheduling problem, deciding when and how much to produce, as well as how much to deliver to each customer. The second phase applies MIP formulations with different neighborhood structures to optimize the best solution found by any of the three fronts.

In what follows, in Section 4.1 we detail both B&C fronts, and in Section 4.2 we describe the LSM front.

4.1. Branch-and-cut fronts

We now present the B&C algorithm used by the two- and three-index formulations from Sections 3.1 and 3.2. We designed a modified version of the traditional B&C algorithm from the literature to support a collaborative integration with the controller. In particular, our algorithm shares the BKS (S_{Best}), a stopping criterion flag ($StopAll$), and the list $\mathcal{R}$.

The algorithm starts by specifying a list of active nodes $\mathcal{L}$ in the B&C tree, a feasible solution S and its objective function value $z(S)$, and a stopping flag $Stop$. In the beginning, the list $\mathcal{L}$ contains only the root node $\mathcal{N}_0$ corresponding to the linear problem LP_0. Then an iterative process runs until an stopping criterion is satisfied. In particular, a B&C front stops when the list of active nodes is empty or when it receives a stopping signal from the controller. As soon as a B&C front stops, the controller issues $StopAll = true$ and stops the other fronts. At each iteration, the B&C front updates the flag $Stop$, the solution S, and its objective function value $z(S)$. If $Stop = true$, then another front has stopped the optimization. Otherwise, the list of active nodes $\mathcal{L}$ is updated by removing all nodes that cannot be better than S_{Best}. This step helps prune some nodes using an external upper bound and decrease the size of $\mathcal{L}$.

Then, a node $N_i \in \mathcal{L}$ is selected and its corresponding LP_i is solved. Next, a pruning step is applied to potentially remove $\mathcal{N}_i$ if: *i*) LP_i is infeasible (prune by infeasibility), *ii*) S_i is worse than the current best solution S (prune by bound), or *iii*) S_i is an integer feasible solution (prune by integrality). During the pruning step, if LP_i is feasible, the algorithm also updates the list $\mathcal{R}$ of partial solutions. Each element added to $\mathcal{R}$ contains all visited customers in S_i and the quantities delivered. After extensive preliminary tests, we identified that solutions S_i which violated integrality conditions perform worse than integer ones. We have then adopted two strategies for adding a solution to $\mathcal{R}$: integer solutions are added to the top of the list to be selected first by the LSM front; fractional solutions are added to the bottom of the list to be polished only if no integer solution remains.

Finally, after the pruning step, the algorithm decides whether to add cuts or apply a branching step. In the first case, valid cuts that are violated in S_i are separated and added to the corresponding LP_i which must be re-optimized. According to the formulation, different families of valid cuts can be added to the relaxed problem (see Sections 4.1.1 and 4.1.2). Adding valid cuts to LP_i helps strengthen the current relaxation. However, sometimes it is important to select a fractional variable v_i of S_i for branching. The branching step creates two new problems LP_i^1 and LP_i^2 imposing $v_i \leq \lfloor \overline{v}_i \rfloor$ and $v_i \geq \lceil \overline{v}_i \rceil$ to LP_i, where $\overline{v}_i$ is the value of v_i. Then, the corresponding new nodes $\mathcal{N}_i^1$ and $\mathcal{N}_i^2$ are added to $\mathcal{L}$ and a new iteration is performed.

4.1.1. Branch-and-cut for the two-index formulation

The two-index formulation does not require specific cuts to be added along the search tree to ensure a feasible solution. However, subtour elimination can be reinforced with the following cuts:

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ i<j}} x_{ij}^t \leq \sum_{i \in \mathcal{S}} z_i^t - z_m^t \qquad \mathcal{S} \subseteq \mathcal{V}', |\mathcal{S}| \geq 2, m \in \mathcal{S}, t \in \mathcal{T}. \tag{52}$$

Cuts (52) are the so-called subtour elimination constraints (SEC) and were also used by Manousakis et al. [34] to solve the IRP. To separate these cuts, we implemented a TS inspired by the procedure developed by Augerat et al. [14]. The algorithm creates two disjoint sets $\mathcal{S} = \{i\}$ and $\mathcal{S}' = \mathcal{V} \backslash \mathcal{S}$ for a customer i whose variable z_i^t is positive in the solution S_i of LP_i. We then apply an expansion phase iteratively moving customers from $\mathcal{S}'$ to S such that the difference between the LHS and RHS of (52) is maximized. A positive value for $f(S) = LHS - RHS$ indicates a violated cut associated with the cut-set $[\mathcal{S}, \mathcal{S}']$. The expansion phase ends when there are no more customers in $\mathcal{S}'$ adjacent to at least one customer in $\mathcal{S}$. Next, we apply an interchange phase to reallocate customers from $\mathcal{S}$ to $\mathcal{S}'$ and vice-versa. If a customer i is removed (added) from (to) $\mathcal{S}$ at any iteration, the reverse movement is defined tabu over a given number of iterations (tabu length list - tll), avoiding repeated exchanges in consecutive iterations.

After preliminary tests (see Section 5), we add a maximum of 25 cuts for each LP_i. All cuts were separated by adopting $tll = 5$, and the TS algorithm is applied for each period $t \in \mathcal{T}$. Also, to obtain a positive trade-off between branching on a fractional variable or adding cuts, we separate cuts only when the B&C front has explored 400 nodes with no lower bound improvement.

4.1.2. Branch-and-cut for the three-index formulation

In contrast to the two-index formulation, the model presented in Section 3.2 can only be generated for small instances. For realistic size problems, the number of SEC (37) grows exponentially with the number of customers and their full enumeration is impracticable. Thus, at the beginning of the B&C front, we create the root node $\mathcal{N}_0$ of the B&C tree containing the model given by (28)−(44) disregarding (37) and all valid inequalities (100)−(122). Then, whenever subtours are found in the current solution S_i of LP_i, violated SECs are dynamically identified and added to the model. To this end, we apply the TS algorithm described in Section 4.1.1. However, after preliminary tests with a set of instances solved by the three-index formulation, we adopted a more aggressive strategy when compared to the two-index formulation, generating up to 25 cuts at each 50 nodes without lower bound improvement. This strategy is only valid when S_i is fractional. Otherwise, when all variables x_{ij}^{kt} assume binary values in S_i, an exhaustive search is applied to identify subtours.

In addition to SECs, we also separate the families of disaggregate parity and vehicle-disaggregate parity inequalities [18]. These families of valid cuts were inspired by the inequalities of Bertazzi et al. [20] for the multi-depot IRP. To formulate these cuts we consider a set $\delta(S)$ of edges (i, j) incident to the vertices $i \in S \subset \mathcal{V}$, where $j \notin S$, obtaining:

$$\sum_{(i,j)\in\delta(S)\setminus F} x_{ij}^{kt} \geq \sum_{(i,j)\in F} x_{ij}^{kt} - |F| + 1 \qquad t \in \mathcal{T}, k \in \mathcal{K}, F \subseteq \delta(S), |F| \text{ odd} \tag{53}$$

$$\sum_{(i,j)\in\delta(S)\setminus F} \sum_{k\in\mathcal{K}} x_{ij}^{kt} \geq \sum_{(i,j)\in F} \sum_{k\in\mathcal{K}} x_{ij}^{kt} - |F| + 1 \qquad t \in \mathcal{T}, F \subseteq \delta(S), |F| \text{ odd.} \tag{54}$$

To separate inequalities (53) and (54), we adopted the heuristic procedure described by Aráoz et al. [13].

4.2. Local search matheuristic front

The LSM front is responsible for obtaining an initial solution via a constructive procedure (CP) and for performing improvements on a solution.

The improvement efforts are executed in two phases. The first one repairs and polishes a solution candidate from the list $\mathcal{R}$ by applying a routing heuristic procedure (RHP) and a lot-sizing and scheduling procedure (LSSP). If the new solution cost $z(S)$ is better than $z(S_{Best})$, then S_{Best} is updated and made available to all other fronts. If $z(S) < (1 + \epsilon) \cdot z(S_{Best})$, then a reallocation and polishing procedure (RPP) is applied.

The second phase intensifies the improvement efforts by recursively applying MIPs, aiming to reorganize

the delivery routes and solve the associated lot-sizing problem whenever a new best solution is found. Here, RPP is continuously applied with varying parameters.

4.2.1. Constructive procedure (CP)

CP generates an initial solution for the problem in which all deliveries are scheduled based on direct deliveries, thus removing all routing variables x from the model while minimizing inventory, lot-sizing, and distribution costs. The CP is formulated as follows:

$$\min \sum_{i \in \mathcal{V}} \sum_{t \in \mathcal{T} \cup \{0\}} h_i \cdot I_i^t + \sum_{t \in \mathcal{T}} \mu \cdot p^t + \sum_{t \in \mathcal{T}} \gamma \cdot y^t + \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}'} 2 \cdot c_{0i} \cdot z_i^{kt} \tag{55}$$

subject to (29)–(35) $\cup$ (38)–(39) $\cup$ (42)–(44) and to:

$$z_0^{kt} \leq \sum_{i \in \mathcal{V}'} z_i^{kt} \qquad k \in \mathcal{K}, t \in \mathcal{T}. \tag{56}$$

Here, constraints (56) ensure that vehicle k leaves the plant only if at least one customer is visited by it in period t. The MIP previously described is applied to obtain a delivery and production plan. To complete the solution, we determine the sequence of visits for each vehicle k and period t by solving a traveling salesman problem using the B&C algorithm of Padberg and Rinaldi [36].

4.2.2. Routing heuristic procedure (RHP)

The routing heuristic procedure (RHP) solves a CVRP for an element of $\mathcal{R}$, which contains the quantities delivered q_i^t for each customer i in period t. We use the hybrid genetic search algorithm (HGS-CVRP) of Vidal et al. [47] and recently improved by Vidal [46]. HGS-CVRP is applied until 500 iterations without improvement in the best solution.

4.2.3. Lot-sizing and scheduling procedure (LSSP)

Since all routes are known, production and inventory decisions can be optimized to complete the solution with the LSSP as follows. Let ψ_i^{kt} be a binary parameter equal to 1 if the vertex $i \in \mathcal{V}$ is visited by vehicle $k \in \mathcal{K}$ in period $t \in \mathcal{T}$ in the current solution, and 0 otherwise. The LSSP is formulated by:

$$\min \sum_{i \in \mathcal{V}} \sum_{t \in \mathcal{T} \cup \{0\}} h_i \cdot I_i^t + \sum_{t \in \mathcal{T}} \mu \cdot p^t + \sum_{t \in \mathcal{T}} \gamma \cdot y^t \tag{57}$$

subject to (29)–(31) ∪ (33) ∪ (38) ∪ (42)–(44) and to:

$$q_i^{kt} \leq \min \left\{ Q, U_i, \sum_{t'=t}^{\rho} d_i^{t'} \right\} \cdot \psi_i^{kt} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{58}$$

$$\sum_{i \in \mathcal{V}'} q_i^{kt} \leq Q \cdot \psi_0^{kt} \qquad k \in \mathcal{K}, t \in \mathcal{T}. \tag{59}$$

Constraints (58) and (59) are similar to the constraints (32) and (34) of Section 3.2.

4.2.4. Reallocation and polishing procedure (RPP)

Different from combining RHP and LSSP to create a feasible solution, the RPP is a MIP-based improvement heuristic that aims to simultaneously reallocate customers on existing routes, as well as polish the lot-sizing and distribution decisions. The RPP slightly improves a solution by removing and/or inserting customers in the routes, scheduling production, and swapping customers among routes. All parameters used in the RPP are shown below.

- ψ_i^{kt}: binary parameter equal to 1 if customer i is visited by vehicle k in period t, 0 otherwise.
- a_i^{kt}: routing reduction cost if customer i is removed from the route of vehicle k in period t. This cost parameter is valid when $\psi_i^{kt} = 1$ and follows the cheapest removal rule.
- b_i^{kt}: routing insertion cost if customer i is added to the route of vehicle k in period t. This cost parameter is valid when $\psi_i^{kt} = 0$ and follows the cheapest insertion rule.

In addition, new binary variables are defined:

- $\delta_i^{kt} = 1$ if customer i is removed from the route of vehicle k in period t, where $\psi_i^{kt} = 1$.
- $\omega_i^{kt} = 1$ if customer i is added to the route of vehicle k in period t, where $\psi_i^{kt} = 0$.

The RPP is formulated by:

$$\min \sum_{i\in\mathcal{V}} \sum_{t\in\mathcal{T}\cup\{0\}} h_i \cdot I_i^t + \sum_{t\in\mathcal{T}} \mu \cdot p^t + \sum_{t\in\mathcal{T}} \gamma \cdot y^t - \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}'} a_i^{kt} \cdot \delta_i^{kt} + \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}'} b_i^{kt} \cdot \omega_i^{kt} \tag{60}$$

subject to (29)–(31) ∪ (33) ∪ (38) ∪ (42)–(44) and to:

$$q_i^{kt} \leq \min\left\{Q, U_i, \sum_{t'=t}^{\rho} d_i^{t'}\right\}\left(\psi_i^{kt} - \delta_i^{kt} + \omega_i^{kt}\right) \qquad i\in\mathcal{V}', k\in\mathcal{K}, t\in\mathcal{T} \tag{61}$$

$$\sum_{i\in\mathcal{V}'} q_i^{kt} \leq Q \qquad k\in\mathcal{K}, t\in\mathcal{T} \tag{62}$$

$$\sum_{k\in\mathcal{K}} \left(\psi_i^{kt} - \delta_i^{kt} + \omega_i^{kt}\right) \leq 1 \qquad i\in\mathcal{V}', t\in\mathcal{T} \tag{63}$$

$$\omega_i^{kt} \leq 1 - \psi_i^{kt} \qquad i\in\mathcal{V}, k\in\mathcal{K}, t\in\mathcal{T} \tag{64}$$

$$\delta_i^{kt} \leq \psi_i^{kt} \qquad i\in\mathcal{V}, k\in\mathcal{K}, t\in\mathcal{T} \tag{65}$$

$$\delta_i^{kt}, \omega_i^{kt} \in \{0,1\} \qquad i\in\mathcal{V}', k\in\mathcal{K}, t\in\mathcal{T}. \tag{66}$$

The objective function (60) minimizes the inventory holding, production decisions, and removal and insertion costs. Constraints (61) link insertion and removal variables with delivered quantities, while (62) guarantee that the vehicle capacity is not exceeded. Constraints (63) avoid split deliveries and constraints (64) prevent adding a customer to a route that already serves it, while (65) ensure that a customer can only be removed from a route visiting it. Finally, (66) define the domain of the variables.

We also define a constant β to control the maximum number of removals and insertions allowed in the solution used to formulate a set of six different families of inequalities $\mathcal{N} = \{1, 2, ..., 6\}$:

$$\sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{V}'} \left(\delta_i^{kt} + \omega_i^{kt}\right) \leq \beta \qquad k\in\mathcal{K} \qquad (\alpha = 1) \tag{67}$$

$$\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \left(\delta_i^{kt} + \omega_i^{kt}\right) \leq \beta \qquad i\in\mathcal{V}' \qquad (\alpha = 2) \tag{68}$$

$$\sum_{i\in\mathcal{V}'} \left(\delta_i^{kt} + \omega_i^{kt}\right) \leq \beta \qquad k\in\mathcal{K}, t\in\mathcal{T} \qquad (\alpha = 3) \tag{69}$$

$$\sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}'} \left(\delta_i^{kt} + \omega_i^{kt}\right) \leq \beta \qquad t\in\mathcal{T} \qquad (\alpha = 4) \tag{70}$$

$$\sum_{t\in\mathcal{T}} \left(\delta_i^{kt} + \omega_i^{kt}\right) \leq \beta \qquad i\in\mathcal{V}', k\in\mathcal{K} \qquad (\alpha = 5) \tag{71}$$

$$\sum_{k\in\mathcal{K}} \left(\delta_i^{kt} + \omega_i^{kt}\right) \leq \beta \qquad i\in\mathcal{V}', t\in\mathcal{T} \qquad (\alpha = 6). \tag{72}$$

Each family of inequalities $\alpha \in \mathcal{N}$ represents a different neighborhood with respect to the current solution. In this sense, all families of inequalities allow us to explore distinct regions in the search space, alternating between intensification and exploration. In particular, inequalities (67) enable at most β changes in the route of each vehicle k, while (68) and (70) do the same for each customer i and period t, respectively. On the other hand, inequalities (69), (71), and (72) are more restrictive, enabling up to β removals or insertions for each vehicle k and period t, customer i and vehicle k, and customer i and period t, respectively.

We solve RPP several times while running the LSM front. At each execution, we choose a family of inequalities $\alpha \in \mathcal{N}$ and a value for the parameter β to setup RPP(α, β). We point out that when RPP($\alpha = 3$, $\beta = 1$) the procedure considers real transportation costs. Otherwise, the solution is approximate. However, all families are useful for exploring different search spaces.

5. Computational results

The computational experiments were performed on a grid of AMD EPYC 7532 2.4 GHz processors, CentOS Linux, with 500GB of RAM memory. The 3FP-B&C was coded in C++ and we used Gurobi 9.5.1 as the MIP solver. Our 3FP-B&C runs for up to 7200s for the IRP instances, and 14400s for the PRP instances. The parameters of our LSM were set after preliminary experiments as follows. RPP is invoked for all values of α starting randomly and β starts with 5 and is decreased until 1, until a new best solution is found, when these values are reset. The value of ϵ was set to 0.05.

Section 5.1 presents the computational results for the IRP. We used the small-size set [9], consisting of 956 feasible instances with three and six periods, and 5 to 50 customers, and also a large-size set [10] with 300 instances with six periods, totalling 1316 instances. The number of vehicles varies from 1 to 6. The 3FP-B&C is the first method to solve the large instances with $|\mathcal{K}| = 6$.

Section 5.2 presents the computational results for the PRP instances proposed by Adulyasak et al. [3], Archetti et al. [8], and Boudia et al. [22]. The first set considers ML and OU policies, 10 to 50 customers, planning horizon of three, six, and nine periods, and two, three, and six vehicles. In the second set we have four different classes with $n \in \{14, 50, 100\}$, with six periods, and unlimited fleet when $n \in \{50, 100\}$. Each class has 120 instances. The last set consists of more difficult instances with 50 (B1) and 100 customers (B2), five and nine vehicles, and 20 periods. Our computational resources could not solve the root node for instances with 200 customers, and the B3 group was not assessed.

All instance sets and detailed solutions are available online at `https://www.leandro-coelho.com/three-front-parallel/`.

5.1. Inventory-routing problem

In this section, we present the computational results of our 3FP-B&C on the IRP. In Table 1 we report a brief summary of exact and heuristic methods used in the comparison. Table 2 presents the number of instances tested by each exact method. The '−' symbol means that a method was not run in these instances. We can observe the 3FP-B&C is the first method to consider all instances.

Table 1: IRP algorithms used in the comparison.

Reference	Type	Algorithm	Hardware	Threads	Solver
Archetti et al. [10]	Heuristic	HAIR	Intel Dual Core 1.86 GHz	Default	CPLEX 10.1
Coelho and Laporte [25]	Exact	B&C [CL]	Xeon 2.66 GHz	6	CPLEX 12.3
Adulyasak et al. [3]	Exact/Matheur.	B&C [Ad]/ALNS	Intel Xeon 2.67 GHz	8	CPLEX 12.3
Desaulniers et al. [29]	Exact	BP&C	Intel Core i7-2600 3.4 GHz	1	CPLEX 12.2
Avella et al. [16]	Exact	B&C [Av1]	Pentium Quad-core 2.6 GHz	1	Xpress 7.3
Archetti et al. [11]	Matheuristic	MORTAR	Intel Xeon W3680 3.33 GHz	8	CPLEX 12.5
Avella et al. [17]	Exact	B&C [Av2]	Intel core i7-2620 2.7 GHz	1	Xpress 7.6
Alvarez et al. [5]	Heuristic	ILS	Intel Core i7-2600 3.4 GHz	1	-
Alvarez et al. [5]	Heuristic	SA	Intel Core i7-2600 3.4 GHz	1	-
Chitsaz et al. [24]	Matheuristic	CCJ-DH	Xeon X5650 2.67 GHz	1	CPLEX 12.6
Guimarães et al. [31]	Exact	I-B&C [G]	Xeon E5-2630 v2 2.60 GHz	6	Gurobi 8.1.0
Alvarez et al. [6]	Heuristic	ILS + MIP	Intel Xeon X5650 2.67 GHz	1	CPLEX 12.8
Diniz et al. [30]	Heuristic	D-H	Intel Core i7-8700 K 3.7 GHz	1	LEMON library
Manousakis et al. [34]	Exact	I-B&C [M]	Core i7-7700 3.60 GHz	8	Gurobi 8.1.0
Archetti et al. [12]	Matheuristic	KS-MIRP	Intel Xeon 3.5 GHz	1	CPLEX 12.10
Vadseth et al. [44]	Matheuristic	V-H	Xeon Gold 6144 3.5 GHz	1	Gurobi 9.0
Vadseth et al. [45]	Matheuristic	VACS-M	Xeon Gold 6144 3.5 GHz	1	Gurobi 9.1
Skålnes et al. [41]	Exact	SOTAF	Intel E5-2670v3 2.3 GHz	12	Gurobi 9.0.2
Skålnes et al. [41]	Exact	CSF	Intel E5-2670v3 2.3 GHz	12	Gurobi 9.0.2
Solyalı and Süral [42]	Matheuristic	MATHIRP	Xeon X5650 2.67 GHz	1	CPLEX 12.7
This paper	Exact	3FP-B&C	AMD EPYC 7532 2.4 GHz	24	Gurobi 9.5.1

Table 2: Number of instances tested by each exact method.

Class	$\|\mathcal{K}\|$	#	B&C [CL]	BP&C	B&C [Av1]	B&C [Av2]	B&C [Ad]	I-B&C [M]	I-B&C [G]	SOTAF	CSF	3FP-B&C
small	1	160	160	-	-	-	-	-	160	-	-	160
	2	160	160	158	40	40	100	160	160	40	40	160
	3	160	160	159	40	40	150	160	160	40	40	160
	4	160	160	160	40	40	50	160	160	40	40	160
	5	158	158	158	40	40	-	158	158	40	40	158
	6	158	-	-	-	-	-	158	-	-	-	158
large	1	60	60	-	-	-	-	40	60	-	-	60
	2	60	40	-	-	-	-	40	60	-	-	60
	3	60	40	-	-	-	-	40	60	-	-	60
	4	60	-	-	-	-	-	40	60	-	-	60
	5	60	-	-	-	-	-	40	60	-	-	60
	6	60	-	-	-	-	-	-	-	-	-	60
Total		**1316**	**938**	**635**	**160**	**160**	**300**	**996**	**1098**	**160**	**160**	**1316**

Table 3 shows the number of optimal solutions found by each method. For each class and number of vehicles, we highlight in bold the method that obtained the largest number of optimal solutions. We can observe that

the I-B&C [G] of Guimarães et al. [31] was the method in the literature that had found the largest number of optimal solutions (598). Furthermore, considering all exact methods in the literature, optimal solutions were known for 733 instances. Our 3FP-B&C proves optimality for 769 instances (58% of all instances or 78% considering only the small class). We found new optimal solutions (column # new OPT) for 47 instances. Therefore, our method found more optimal solutions than all other exact methods combined, dominating all of them.

Table 3: Number of optimal solutions found for the IRP.

Class	$\|\mathcal{K}\|$	#	B&C [CL]	BP&C	B&C [Av1]	B&C [Av2]	B&C [Ad]	I-B&C [M]	I-B&C [G]	SOTAF	CSF	All combined	3FP-B&C	# new OPT
small	1	160	**160**	-	-	-	-	-	**160**	-	-	**160**	**160**	0
	2	160	152	75	14	14	100	139	155	18	18	156	**160**	**4**
	3	160	112	77	0	0	117	96	119	4	3	123	**136**	**13**
	4	160	76	84	0	0	20	80	82	1	1	106	**108**	**6**
	5	158	51	90	0	0	-	78	61	0	1	**103**	**103**	**7**
	6	158	-	-	-	-	-	64	-	-	-	64	**79**	**15**
large	1	60	17	-	-	-	-	13	21	-	-	21	**23**	**2**
	2	60	0	-	-	-	-	0	0	-	-	0	0	0
	3	60	0	-	-	-	-	0	0	-	-	0	0	0
	4	60	-	-	-	-	-	0	0	-	-	0	0	0
	5	60	-	-	-	-	-	0	0	-	-	0	0	0
	6	60	-	-	-	-	-	-	-	-	-	-	0	0
Total		**1316**	568	326	14	14	237	470	598	23	23	733	**769**	**47**

Table 4 details the number of BKS found by each exact method. The 3FP-B&C found the BKS for 94% of the tested instances (1243 of 1316 instances), being the method that obtained the largest number of BKS in the literature, even more than heuristic ones. Our method found the BKS for 99% of the small class instances. On the large class, it found the BKS for 82% of the instances. The method with the second largest number of BKS was the I-B&C [G] [31], which found the BKS in 638 instances (58% of the tested instances). The 3FP-B&C found new BKS for 423 instances (column # new BKS), that is, for 32% of the instances. Likewise, the number of BKS of the 3FP-B&C is twice the number of BKS considering all heuristic methods from Table 1.

The 3FP-B&C is also the method with the largest number of best lower bounds, as shown in Table 5. Our method found the best lower bounds for 1169 instances (89% of the instances). We have new lower bounds for 441 instances (column # new LB), 68% of them for the large class.

Table 6 summarizes the improvements presented by 3FP-B&C as shown in the previous tables. Because it is the method with the largest number of BKS and lower bounds, we also have the smallest gaps between the upper and lower bounds. We can see that we proved optimality for all instances of the small class with 1 or 2 vehicles. For the other instances of this class, we have gaps smaller than 1%. For instances of the large class, we have an average gap smaller than 1% for a single vehicle, and slightly larger as the number

Table 4: Number of best-known solutions for the IRP.

Class	$\|\mathcal{K}\|$	#	B&C [CL]	BP&C	B&C [Av1]	B&C [Av2]	B&C [Ad]	I-B&C [M]	I-B&C [G]	SOTAF	CSF	All exacts	All heuristics†	3FP-B&C	# new BKS
small	1	160	**160**	-	-	-	-	-	**160**	-	-	**160**	150	**160**	0
	2	160	155	137	16	15	91	146	158	27	22	158	133	**160**	**2**
	3	160	117	105	2	0	120	123	124	7	9	134	118	**159**	**21**
	4	160	79	90	0	0	17	109	90	4	6	125	97	**160**	**35**
	5	158	56	96	0	0	-	109	74	8	4	118	92	**155**	**36**
	6	158	-	-	-	-	-	102	-	-	-	102	0	**155**	**59**
large	1	60	19	-	-	-	-	15	32	-	-	32	2	**49**	**26**
	2	60	0	-	-	-	-	1	0	-	-	1	17	**43**	**42**
	3	60	0	-	-	-	-	2	0	-	-	2	12	**46**	**46**
	4	60	-	-	-	-	-	0	0	-	-	0	13	**47**	**47**
	5	60	-	-	-	-	-	0	0	-	-	0	11	**49**	**49**
	6	60	-	-	-	-	-	-	-	-	-	-	-	**60**	**60**
Total		**1316**	586	428	18	15	228	607	638	46	41	832	645	**1243**	**423**

† Considering HAIR, ALNS, MORTAR, ILS, SA, CCJ-DH, ILS+MIP, D-H, KS-MIRP, V-H, VACS-M, MATHIRP from Table 1.

Table 5: Number of best lower bounds for the IRP.

Class	$\|\mathcal{K}\|$	#	B&C [CL]	BP&C	B&C [Av1]	B&C [Av2]	B&C [Ad]	I-B&C [M]	I-B&C [G]	SOTAF	CSF	All exacts	3FP-B&C	# new LB
small	1	160	**160**	-	-	-	-	-	**160**	-	-	**160**	**160**	0
	2	160	152	75	14	10	100	139	152	10	9	156	**160**	**4**
	3	160	112	98	3	0	117	96	114	1	1	146	**138**	**14**
	4	160	76	**122**	2	0	20	80	81	1	1	145	117	**15**
	5	158	51	**131**	1	0	-	78	61	0	1	145	109	**13**
	6	158	-	-	-	-	-	64	-	-	-	64	**158**	**94**
large	1	60	26	-	-	-	-	14	26	-	-	38	**43**	**22**
	2	60	0	-	-	-	-	2	0	-	-	2	**58**	**58**
	3	60	0	-	-	-	-	3	0	-	-	3	**58**	**57**
	4	60	-	-	-	-	-	6	0	-	-	6	**55**	**54**
	5	60	-	-	-	-	-	10	0	-	-	10	**53**	**50**
	6	60	-	-	-	-	-	-	-	-	-	-	**60**	**60**
Total		**1316**	**577**	**426**	**20**	**10**	**237**	**492**	**594**	**12**	**12**	**875**	**1169**	**441**

of vehicles increases. Still, the average gaps were not more than 4.2%.

Table 6: Average percentage gaps for the IRP.

Class	$\|\mathcal{K}\|$	#	B&C [CL]	BP&C	B&C [Av1]	B&C [Av2]	B&C [Ad]	I-B&C [M]	I-B&C [G]	SOTAF	CSF	3FP-B&C
small	1	160	**0.0**	-	-	-	-	-	**0.0**	-	-	**0.0**
	2	160	0.1	0.9	2.3	2.2	**0.0**	0.2	0.1	0.6	1.1	**0.0**
	3	160	2.0	1.4	9.2	4.9	1.2	0.8	1.6	1.8	1.8	**0.2**
	4	160	6.6	3.4	7.4	6.2	5.1	1.2	4.4	1.7	3.6	**0.5**
	5	158	11.4	5.7	9.2	7.6	-	1.0	7.5	1.7	3.2	**0.6**
	6	158	-	-	-	-	-	1.4	-	-	-	**0.9**
large	1	60	9.3	-	-	-	-	0.9	1.4	-	-	**0.8**
	2	60	28.5	-	-	-	-	2.6	8.3	-	-	**2.0**
	3	60	46.4	-	-	-	-	4.7	18.2	-	-	**3.1**
	4	60	-	-	-	-	-	5.8	26.8	-	-	**3.8**
	5	60	-	-	-	-	-	6.3	36.3	-	-	**4.0**
	6	60	-	-	-	-	-	-	-	-	-	**4.2**

Finally, Table 7 presents a general analysis of the 3FP-B&C compared to all exact and heuristic methods of Table 1. Columns Exact, Heuristic, and All present the average values of the objective function of the best solutions obtained in each group of instances considering exact algorithms, heuristics, and all combined. The Δ% columns show the percentage improvement of 3FP-B&C concerning these methods. Negative values indicate improvements. Our method improved the average objective function values for all sets of instances (except in the small class and $|\mathcal{K}|$ equal to 1 and 2, for which all optimal solutions were known). Instances of the large class with $|\mathcal{K}| = 6$ were not compared as our method is the first one to solve them. This table demonstrates once again that our method dominates the IRP literature and is the new state-of-the-art algorithm to solve this problem.

Table 7: Average solution improvement for the IRP.

Class	$\|\mathcal{K}\|$	#	Best Exact	Best Heuristic	Best All	3FP-B&C	Δ% Best Exact	Δ% Best Heuristic	Δ% Best All
small	1	160	7319.0	7319.7	7319.0	7319.0	**0.00**	**-0.01**	**0.00**
	2	160	7875.1	7877.9	7875.1	7874.9	**0.00**	**-0.04**	**0.00**
	3	160	8592.8	8594.5	8589.6	8585.6	**-0.08**	**-0.10**	**-0.05**
	4	160	9334.5	9345.6	9330.7	9323.3	**-0.12**	**-0.24**	**-0.08**
	5	158	10115.9	10131.1	10113.4	10104.6	**-0.11**	**-0.26**	**-0.09**
	6	158	10932.8	-	10932.8	10913.4	**-0.18**	**-**	**-0.18**
large	1	60	40482.5	40487.7	40389.0	40336.4	**-0.36**	**-0.37**	**-0.13**
	2	60	42584.0	41163.6	41114.2	41074.7	**-3.54**	**-0.22**	**-0.10**
	3	60	45212.2	42348.8	42333.7	42257.1	**-6.54**	**-0.22**	**-0.18**
	4	60	49008.9	43760.5	43752.9	43644.7	**-10.95**	**-0.26**	**-0.25**
	5	60	52477.6	45267.4	45267.4	45134.1	**-13.99**	**-0.29**	**-0.29**
	6	60	-	-	-	46695.1	**-**	**-**	**-**

In Appendix C we present a detailed comparison of the performance profiles of several exact algorithms versus our 3FP-B&C, with adjusted runtimes to account for different hardware.

5.2. Production-routing problem

In this section, we present the computational results of our 3FP-B&C applied to the PRP considering three sets of instances from the literature: 336 instances of Adulyasak et al. [3], 1440 instances of Archetti et al. [8], and 60 instances of Boudia et al. [22]. In each set, our method was compared with the best methods in the literature.

5.2.1. Instances of Adulyasak et al. [3]

This section presents a comparison of our 3FP-B&C with the algorithms that used the set of instances proposed by Adulyasak et al. [3]. We consider the B&C and ALNS [3], the LS-B&C [40], and the EXM [48]. Table 8 presents the computational environment of each method.

Table 8: Papers compared for the PRP instances of Adulyasak et al. [3].

Reference	Type	Algorithm	Hardware	Threads	Solver
Adulyasak et al. [3]	Exact/Math.	B&C [Ad]/ALNS	Intel Xeon 2.67 GHz	8/1	CPLEX 12.3
Schenekemberg et al. [40]	Exact	LS-B&C	Intel Xeon 2.60 GHz	6	Gurobi 8.1.0
Zhang et al. [48]	Exact	EXM	Intel Xeon E5-2623 3.0 GHz	1	CPLEX 12.6.3
This paper	Exact	3FP-B&C	AMD EPYC 7532 2.4 GHz	24	Gurobi 9.5.1

Table 9 presents the number of instances for which the exact methods proved the optimal solutions. We can observe that the performance of both B&C of the literature is similar, while the 3FP-B&C was able to prove a greater number of optimal solutions under the ML policy (88% of the instances) and the OU policy (77% of the instances). The EXM algorithm performs similarly to 3FP-B&C under OU. The 3FP-B&C proved the optimal solution for 19 new instances (5.7%).

Table 9: Number of optimal solutions found for the PRP instances of Adulyasak et al. [3].

Policy	$\|\mathcal{K}\|$	#	B&C [Ad]	LS-B&C	EXM	3FP-B&C	# new OPT
ML	2	48	47	**48**	-	**48**	0
	3	84	64	67	-	**76**	6
	4	36	14	16	-	**24**	7
OU	2	48	**48**	**48**	45	**48**	0
	3	84	53	53	**65**	64	1
	4	36	5	7	17	**18**	5
Total		**336**	231	239	127	**278**	**19**

Regarding the number of BKS, we can see in Table 10 that 3FP-B&C found the BKS for all instances of this set except one (OU policy and $|\mathcal{K}| = 4$) and presented an average improvement of 0.024%. The other methods in the literature found the BKS in only 70% of instances (B&C [Ad]), 7.7% (ALNS), and 75.6% (LS-B&C). The EXM found the BKS for 84% of the instances with OU policy. The 3FP-B&C improved the BKS for 74 out of 336 instances (22%).

Table 10: Number of best known solutions for the PRP instances of Adulyasak et al. [3].

Policy	$\|\mathcal{K}\|$	#	B&C [Ad]	ALNS	LS-B&C	EXM	3FP-B&C	# new BKS	Δ% UB
ML	2	48	47	7	47	-	**48**	0	**-0.000**
	3	84	68	2	76	-	**84**	**6**	**-0.005**
	4	36	14	0	19	-	**36**	**15**	**-0.051**
OU	2	48	**48**	**9**	**48**	46	**48**	0	**-0.000**
	3	84	54	8	58	69	**84**	**14**	**-0.028**
	4	36	5	0	6	26	**35**	**10**	**-0.058**
Total		**336**	236	26	254	141	**335**	**74**	**Avg. = -0.024**

The 3FP-B&C is also the method providing the largest number of best lower bounds (Table 11). Consequently, the 3FP-B&C has the best gaps between the lower and upper bounds (Table 12). Our method finds the best lower bound in 99.4% of the ML instances and in 86.9% of the OU instances. We improved the lower bound for 52 instances. Regarding the average gap, the 3FP-B&C has a gap of only 0.09% while the B&C [Ad] has a gap of 0.82% and the LS-B&C has a gap of 0.81%. The EXM has a gap of 0.20% in the instances under the OU policy.

Table 11: Number of best lower bounds for the PRP instances of Adulyasak et al. [3].

Policy	$\|\mathcal{K}\|$	#	B&C [Ad]	LS-B&C	EXM	3FP-B&C	# new LB
ML	2	48	47	**48**	-	**48**	0
	3	84	65	68	-	**83**	12
	4	36	14	17	-	**36**	18
OU	2	48	**48**	**48**	45	**48**	0
	3	84	53	52	70	**76**	13
	4	36	5	7	**27**	22	9
Total		**336**	232	240	142	**313**	**52**

In Appendix D we present the performance profile of our method against the literature, showing the excellent performance of our algorithm.

5.2.2. Instances of Archetti et al. [8]

This section presents the 3FP-B&C performance using the 1440 PRP instances of Archetti et al. [8]. Table 13 presents the algorithms that used this instance set and their computational environment. The comparison

Table 12: Average gaps for the PRP instances of Adulyasak et al. [3].

Policy	$\|\mathcal{K}\|$	#	B&C [Ad]	LS-B&C	EXM	3FP-B&C
ML	2	48	0.02	**0.00**	-	**0.00**
	3	84	0.40	0.21	-	**0.05**
	4	36	1.61	0.96	-	**0.21**
OU	2	48	**0.00**	**0.00**	0.03	**0.00**
	3	84	0.82	1.08	0.20	**0.10**
	4	36	3.18	3.60	0.41	**0.26**
Avg.			0.82	0.81	0.20	**0.09**

was performed considering the nine methods with the best results.

Table 13: Papers compared for the PRP instances of Archetti et al. [8].

Reference	Type	Algorithm	Hardware	Threads	Solver
Adulyasak et al. [4]	Matheuristic	ALNS-500/ALNS-1000	Duo 2.10 GHz	Default	CPLEX 12.2
Absi et al. [1]	Matheuristic	MS/DMS	Intel Xeon 2.67 GHz	Default	CPLEX 12.1
Solyalı and Süral [43]	Heuristic	5P	PC 2.40 GHz	1	CPLEX 12.5
Russell [38]	Matheuristic	SP-VRP	Intel i7 3930k 3.5 GHz	Default	CPLEX 12.6
Qiu et al. [37]	Heuristic	VNS	Intel Core 2 Duo P8600 2.40 GHz	Default	CPLEX 12.6
Chitsaz et al. [24]	Matheuristic	CCJ-DH	Xeon X5650 2.67 GHz	1	CPLEX 12.6
Avci and Yildiz [15]	Matheuristic	MA	Intel Core i5-760X 4.0 GHz	Default	CPLEX 12.6
Li et al. [33]	Heuristic	TLH	Intel Core i7 2.5 GHz	Default	CPLEX 12.6
Vadseth et al. [45]	Matheuristic	VACS	Xeon Gold 6144 3.5 GHz	1	Gurobi 9.1
Manousakis et al. [35]	Matheuristic	HISM	Intel Core i7-7700 3.60 GHz	1	Gurobi 9.0.2
This paper	Exact	3FP-B&C	AMD EPYC 7532 2.4 GHz	24	Gurobi 9.5.1

Table 14 presents a general analysis comparing the number of BKS found by each method. We can see that the number of BKS found by 3FP-B&C is much larger than any other methods, and in fact larger than all other methods combined. Our algorithm obtained the BKS in 1105 out of 1440 instances (76.74%). Furthermore, 3FP-B&C was able to find new BKS for 584 instances (40.56%). Even considering the BKS found by all methods combined, our 3FP-B&C has a larger number of BKS (1105 against 845). The method in the literature with the largest number of BKS was HISM [35], with only 516 instances, less than half of ours.

Table 15 presents a detailed analysis of the 3FP-B&C results. We observe that the average gap between the lower and upper bounds was only 0.61%, and optimal solutions were obtained for all instances with 14 customers. Furthermore, the 3FP-B&C proved the optimality for 26 instances with 50 customers. This is the first exact method to prove optimality for instances with 50 customers. When comparing against the BKS of the literature (column Best All), we notice that the 3FP-B&C improved the average upper bound in six subsets of instances. In the other three subsets with 100 customers, our method is close to the best upper bounds of the literature, even though the 3FP-B&C is an exact method.

Table 14: General solution comparison for the PRP of Archetti et al. [8].

n	Class	MS	DMS	5P	CCJ-DH	MA	TLH	VNS	HISM	VACS	All† methods	3FP-B&C	# new BKS
14	1	80	68	73	19	106	45	31	102	0	119	**120**	0
	2	81	68	73	19	99	46	34	105	0	120	**120**	0
	3	51	44	61	5	82	36	21	89	0	112	**120**	0
	4	89	76	75	28	102	48	26	110	0	118	**120**	0
50	1	0	0	0	1	1	0	0	15	21	36	**93**	**84**
	2	0	0	0	0	0	1	0	22	0	23	**106**	**97**
	3	0	0	0	0	0	0	0	17	24	40	**87**	**80**
	4	0	0	0	1	1	0	0	22	11	36	**95**	**84**
100	1	0	0	1	2	1	0	0	1	57	63	**61**	**57**
	2	0	0	0	0	0	1	0	8	24	33	**87**	8**87**
	3	0	0	0	0	0	1	0	16	**78**	96	24	**24**
	4	0	0	0	0	0	0	0	9	40	49	**72**	**71**
Total		**301**	**256**	**283**	**75**	**392**	**178**	**112**	**516**	**255**	**845**	**1105**	**584**

† Both ALNS-500 and ALNS-1000 found only two BKS and for the same instances ($n = 15$ Class 1 and $n = 15$ Class 4).

Table 15: Average UBs and LBs for the PRP of Archetti et al. [8].

n	Class	3FP-B&C						Avg. UB	Δ%UB	Min Δ% UB	Max Δ% UB
		Avg. UB	Avg. LB	Avg. gap %	Min gap %	Max gap %	# OPT	Best all			
14	1	51555.8	51555.8	0.00	0.00	0.00	120	51555.8	0.000	0.00	0.00
	2	370153.9	370153.9	0.00	0.00	0.00	120	370153.9	0.000	0.00	0.00
	3	93243.5	93243.5	0.00	0.00	0.00	120	93244.6	**-0.001**	-0.18	0.00
	4	202099.8	202099.8	0.00	0.00	0.00	120	202099.9	0.000	0.00	0.00
50	1	164182.6	163415.9	0.50	0.01	1.76	**1**	164197.5	**-0.009**	-0.14	0.12
	2	1275971.0	1275471.9	0.10	0.01	0.30	**16**	1276056.7	**-0.007**	-0.05	0.03
	3	215411.8	211963.7	1.70	0.45	4.32	0	215555.5	**-0.067**	-0.99	0.51
	4	691131.0	690603.7	0.30	0.01	1.70	**9**	691165.6	**-0.005**	-0.45	0.15
100	1	300384.9	297920.4	0.90	0.21	2.73	0	300242.5	0.047	-0.33	1.03
	2	2364036.5	2362201.7	0.10	0.02	0.34	0	2364120.9	**-0.004**	-0.06	0.04
	3	397008.8	384810.5	3.30	1.03	6.52	0	394920.2	0.529	-0.31	2.28
	4	1276027.1	1274341.2	0.50	0.02	2.67	0	1275977.8	0.004	-0.14	1.34
Avg.		**616767.2**	**614815.1**	**0.61**	-	-	-	**616607.6**	**0.041**		

5.2.3. Instances of Boudia et al. [22]

This section presents the computational results for the 60 instances proposed by Boudia et al. [22] with 50 and 100 customers. Table 16 presents the algorithms tested on these instances. Schenekemberg et al. [40] are the only authors that report results of an exact method for this instance set. This way, the lower bounds of 3FP-B&C will be compared only against this method. Due to the difficulty of this set, heuristics have the BKS for these instances.

Table 16: Papers compared for the PRP instances of Boudia et al. [22].

Reference	Type	Algorithm	Hardware	Threads	Solver
Boudia and Prins [21]	Heuristic	MA	Duo 2.10 GHz	Default	CPLEX 12.2
Adulyasak et al. [4]	Matheuristic	ALNS-500/ALNS-1000	Duo 2.10 GHz	Default	CPLEX 12.2
Absi et al. [1]	Matheuristic	IM-VRP/IM-MTSP	Intel Xeon 2.67 GHz	Default	CPLEX 12.1
Solyalı and Süral [43]	Heuristic	5P	PC 2.40 GHz	1	CPLEX 12.5
Qiu et al. [37]	Heuristic	VNS	Intel Core 2 Duo P8600 2.40 GHz	Default	CPLEX 12.6
Chitsaz et al. [24]	Matheuristic	CCJ-DH	Xeon X5650 2.67 GHz	1	CPLEX 12.6
Li et al. [33]	Heuristic	TLH	Intel Core i7 2.5 GHz	Default	CPLEX 12.6
Schenekemberg et al. [40]	Exact	LS-B&C	Intel Xeon 2.60 GHz	6	Gurobi 8.1.0
Vadseth et al. [45]	Matheuristic	VACS	Xeon Gold 6144 3.5 GHz	1	Gurobi 9.1
Manousakis et al. [35]	Matheuristic	HISM	Intel Core i7-7700 3.60 GHz	1	Gurobi 9.0.2
This paper	Exact	3FP-B&C	AMD EPYC 7532 2.4 GHz	24	Gurobi 9.5.1

Table 17 presents a comparison of the lower and upper bounds obtained by the 3FP-B&C and LS-B&C, in addition to a comparison with the BKS considering all methods of Table 16. We observe that the 3FP-B&C obtained significant improvements in terms of lower and upper bounds. The average gap was reduced from 25.1% for LS-B&C to 8.4% for 3FP-B&C. We can also notice that the upper bound of the 3FP-B&C had an average percentage deviation (column Δ%BKS) of 0.8% with respect to the BKS of the literature. In the worst case, we had a deviation of 1.7% (instance `50Clients21`). On instances with 100 customers, our 3FP-B&C again improves the gaps, the lower, and the upper bounds of the LS-B&C. However, in these instances, the average gap of the 3FP-B&C was 31.1%. Despite this, the mean percentage deviation for the BKS from the literature was 6.3%. In the worst case, the upper bound of the 3FP-B&C had a deviation of 9.6% (instance `100Clients9`).

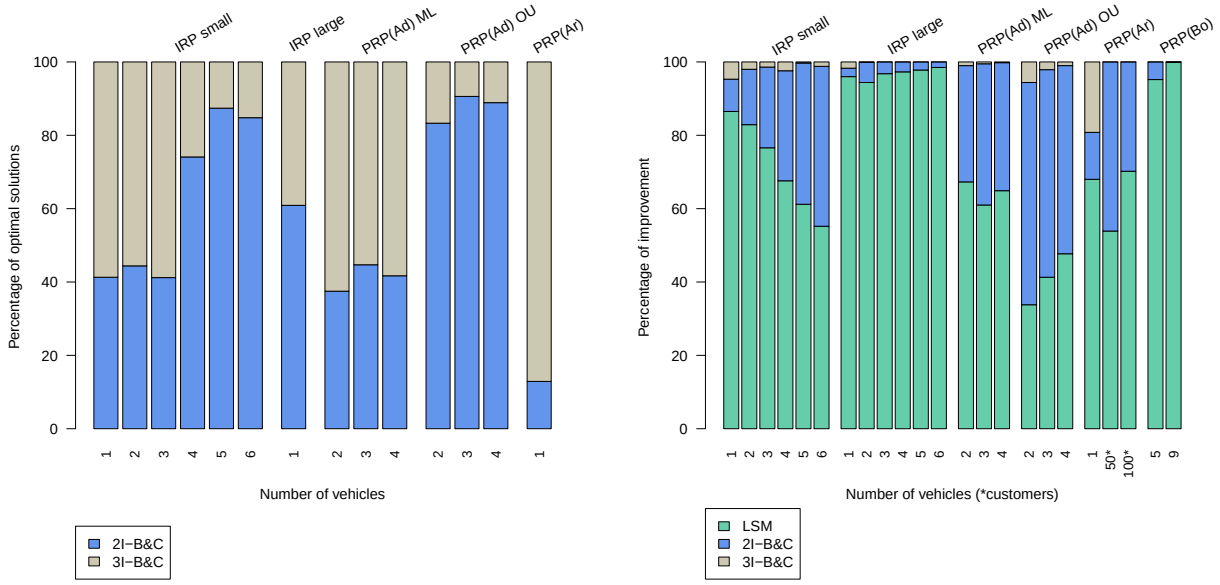
Table 17: Results for the PRP of Boudia et al. [22] (B1 and B2).

Class	#	LS-B&C			3FP-B&C			Δ%LB	Δ%UB	BKS	Δ%BKS	Gap BKS
		LB	UB	Gap	LB	UB	Gap					
B1	30	262632.1	350801.8	25.1	313857.8	342657.9	8.4	19.6	-2.3	339823.3	0.8	8.3
B2	30	335385.8	705030.4	52.4	458973.6	666703.0	31.1	36.9	-5.4	626913.8	6.3	36.6

5.3. Analysis of 3FP-B&C components

In this section, we analyze the contribution of each front to the lower and upper bound improvements. First, we evaluate the fraction of optimal solutions found by each B&C front of our algorithm, considering which one found the optimal lower bound. Then, we present the percentage of improvement over the initial solution obtained by the 2I-B&C, the 3I-B&C, and the LSM fronts. The bar plots in Figure 1 demonstrate the value of the three fronts to solve the IRP and the PRP.

Figure 1a compares the optimal solutions proved by 2I-B&C and 3I-B&C for the IRP and PRP instances. Here, each bar represents all optimal solutions found by our algorithm. In addition, the bars are grouped by set of instances where at least one optimum has been proven, where 'IRP small' indicates the small-size instances of [9], 'IRP large' reports the large-size instances of [10], 'PRP(Ad) ML' and 'PRP(Ad) OR' refer to the ML and OU policies for the instances of Adulyasak et al. [3], and 'PRP(Ar)' represents the instances of Archetti et al. [8]. We observe that for the instances with up to three vehicles, 3I-B&C proved optimality for 60% of them under an ML policy. On the other hand, 2I-B&C proved optimality for 80% of the small IRP instances with 4 or more vehicles and for up to 90% of the PRP instances under the OU policy. These data demonstrate that there is no dominance between the two B&C fronts.



(a) Percentage of optimal solutions. **(b)** Percentage of improvement.

Figure 1: Analysis of 3FP-B&C components in terms of percentage of optimal solutions proved and improvement of the initial solution.

Figure 1b shows the fraction of the gains over the initial solution obtained by each front of the 3FP-B&C.

We show in each bar the total improvement over the initial solution until obtaining the final solution. The first two groups of six bars in Figure 1b show the results for the small and large IRP instances, indicating that the LSM front was responsible for a significant part of the improvement. Particularly, in the large set, LSM found 94% of the improvement over the initial solution, and most of the remaining improvement was found by the 2I-B&C. For the PRP instances, the first six bars refer to ML and OU policy from [3], the next three bars refer to Archetti et al. [8], and the last two bars refers to Boudia et al. [22]. We observe that under the ML policy and for the instances of [8], the LSM front is responsible for 60% of the improvement obtained, while for the instances of [22] the improvement obtained by LSM was up to 99.9% for instances with 100 customers (nine vehicles). Therefore, we can observe that the matheuristic front is also valuable for the convergence of the 3FP-B&C.

6. Conclusions

This paper introduces a three-front parallel branch-and-cut (3FP-B&C) algorithm to solve production and inventory routing problems. Our method combines three independent fronts consisting of two B&Cs derived from two- and three-index formulations and a local search matheuristic (LSM). These three fronts run in parallel and share all improving solutions found. The LSM first repairs and polishes solutions and then reorganizes the delivery routes and optimizes the inventory and production decisions according to new routes by recursively applying a MIP procedure. Computational experiments on several benchmark instances show the efficiency of our method and the advantage of combining the three fronts in an integrated approach.

For the IRP, our 3FP-B&C has found 1243 BKS out of 1316 small and large instances of [9, 10], with 434 new BKS. Our algorithm proved 769 optimal solutions against 733 obtained by all other exact algorithms from the literature combined, being 47 new ones. Finally, our method reported 1169 best lower bounds, with 432 new ones.

On the PRP, our method obtained 278 optimal solutions for the 336 instances of Adulyasak et al. [3], being 19 new ones, besides finding 335 BKS (74 new ones) and 313 best lower bounds (52 new ones). For the PRP instances of Archetti et al. [8], our algorithm found 1105 BKS out of 1440 instances, with 584 new ones, in addition to providing the first lower bounds for the subset of instances with an unlimited fleet. Finally, the results for the very large instances of Boudia et al. [22] offered much tighter bounds for instances with 50 and 100 customers.

A comparative analysis of the contribution of each front to the performance of the algorithm shows that 3I-B&C provides better lower bounds than 2I-B&C for small to medium size IRP instances. Regarding solution improvement, our experiments have shown that LSM dominates the other fronts, but as the number

of vehicles increases, 2I-B&C becomes competitive.

Appendices

A. Valid inequalities for the two-index formulation

We now describe the valid inequalities considered in the 2I-B&C used to tighten the formulation presented in Section 3.1. These VIs were adapted from the two-index formulation of Manousakis et al. [34] which were derived from the single-vehicle IRP of Archetti et al. [9] and the multi-vehicle IRP of Coelho and Laporte [25] and Coelho and Laporte [26].

$$x_{0i}^t \leq z_i^t \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{73}$$

$$x_{ij}^t \leq z_i^t \qquad i < j \in \mathcal{V}', t \in \mathcal{T} \tag{74}$$

$$x_{ij}^t \leq z_j^t \qquad i < j \in \mathcal{V}', t \in \mathcal{T} \tag{75}$$

$$\sum_{t'=1}^{t} z_i^{t'} \geq \left\lceil \frac{\sum_{t'=1}^{t} d_i^{t'} - I_i^0}{\min\{Q, U_i\}} \right\rceil \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{76}$$

$$\sum_{t=t_1}^{t_2} z_i^t \geq \left\lceil \frac{\sum_{t=t_1}^{t_2} d_i^t - U_i}{\min\{Q, U_i\}} \right\rceil \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T} \tag{77}$$

$$\sum_{t=t_1}^{t_2} z_i^t \geq \frac{\sum_{t=t_1}^{t_2} d_i^t - I_i^{t_1-1}}{\min\{Q, U_i\}} \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T} \tag{78}$$

$$\sum_{t=t_1}^{t_2} z_i^t \geq \frac{\sum_{t=t_1}^{t_2} d_i^t - I_i^{t_1-1}}{\sum_{t=t_1}^{t_2} d_i^t} \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T}. \tag{79}$$

Inequalities (73) guarantee that if a customer is visited, then a vehicle departs from the plant. Inequalities (74) and (75) ensure that edge (i, j) is traveled only if i and j are served in t. Inequalities (76) establish the minimum number of visits to avoid a stock-out for customer i on the interval $[1, t]$, while inequalities (77)−(79) do the same for the interval $[t_1, t_2]$. Manousakis et al. [34] also introduced new VIs for the IRP, which are also valid for the PRP, as follows:

$$f_{ij}^t \geq d_j^t \cdot x_{ij}^t - I_j^{t-1} \qquad i < j \in \mathcal{V}, t \in \mathcal{T} \tag{80}$$

$$f_{ji}^t \geq d_i^t \cdot x_{ij}^t - I_i^{t-1} \qquad i < j \in \mathcal{V}, t \in \mathcal{T} \tag{81}$$

$$\sum_{\substack{i \in \mathcal{V} \\ i \neq j}} f_{ij}^t \geq 2 \cdot q_j^t \qquad j \in \mathcal{V}', t \in \mathcal{T} \tag{82}$$

$$\sum_{t \in \mathcal{T}} q_i^t = \sum_{t \in \mathcal{T}} d_i^t - I_i^0 \qquad i \in \mathcal{V}'. \tag{83}$$

Inequalities (80) and (81) ensure that the flow heading to customer i is at least the difference between the demand in t and the inventory in $t-1$. The RHS of (82) limits the sum of direct and inverse flows in j to be greater than or equal to twice the delivered quantity. Finally, inequalities (83) are valid only when $h_i \geq h_0$ under the ML policy, and calculate the total quantity delivered to customer i as the net demand throughout the planning horizon.

Manousakis et al. [35] also present new inequalities for the production setup variables, as follows:

$$y^t \cdot C \geq \sum_{i \in \mathcal{V}'} \left(d_i^t - I_i^{t-1}\right) - I_0^{t-1} \qquad t \in \mathcal{T} \tag{84}$$

$$\sum_{t'=1}^{t} y^{t'} \geq \left\lceil \frac{\sum_{i \in \mathcal{V}'} \sum_{t'=1}^{t} d_i^{t'} - I_i^0 - I_0^0}{C} \right\rceil \qquad t \in \mathcal{T}, t > 1 \tag{85}$$

$$\sum_{t'=t_1}^{t_2} y^{t'} \geq \left\lceil \frac{\sum_{i \in \mathcal{V}'} \sum_{t'=t_1}^{t_2} d_i^{t'} - U_i - U_0}{C} \right\rceil \qquad t_1 \leq t_2 \in \mathcal{T} \tag{86}$$

$$\sum_{t'=t_1}^{t_2} y^{t'} \geq \frac{\sum_{i \in \mathcal{V}'} \left(\sum_{t'=t_1}^{t_2} d_i^{t'}\right) - I_i^{t_1-1} - I_0^{t_1-1}}{C} \qquad t_1 \leq t_2 \in \mathcal{T}. \tag{87}$$

Inequalities (84) impose a production setup in period t when the sum of inventories is not sufficient to satisfy the demand of the customers, while inequalities (85)−(87) compute the minimum number of setups required during interval $[t_1, t_2]$.

Under the ML policy, Archetti et al. [8] present inequalities (88) to prevent the quantity produced in period t being greater than the aggregated demand of all customers over the planning horizon.

$$p^t \leq \left(\sum_{i \in \mathcal{V}'} \sum_{t'=t}^{\rho} d_i^{t'}\right) \cdot y^t \qquad t \in \mathcal{T}. \tag{88}$$

Adulyasak et al. [3] also introduce a set of valid inequalities for the PRP. The authors denote $t' = argmin_{1 \leq t \leq \rho}\{\sum_{i \in \mathcal{V}'} \max\{0, \sum$ $I_i^0\} - I_0^0 > 0\}$ as the earliest period that a production batch should take place, and $t'' = \min_{i \in \mathcal{V}'} t_i''$ as the earliest period that a customer must be replenished to avoid a stock-out, where $t_i'' = argmin_{1 \leq t \leq \rho}\{\sum_{t_1=1}^{t} d_i^{t_1} - I_i^0 > 0\}$. Let s be the net demand in t'', given by $s = \sum_{i \in \mathcal{V}'} \max\{0, \sum_{t_1=1}^{t''} d_i^{t_1} - I_i^0\}$. Stock-outs can be avoided by imposing the following inequalities:

$$\sum_{t=1}^{t'} y^t \geq 1 \tag{89}$$

$$\sum_{t=1}^{t''} z_0^{kt} \geq \left\lceil \frac{s}{Q} \right\rceil. \tag{90}$$

The authors also introduce (91) to strengthen the customer replenishment:

$$I_i^{t-s-1} \geq \left(\sum_{j=0}^{s} d_i^{t-j} \right) \cdot \left(1 - \sum_{j=0}^{s} z_i^{(t-j)} \right) \qquad i \in \mathcal{V}', t \in \mathcal{T}, s = 0, ..., t-1. \tag{91}$$

In order to take into account the timing of activities considered by Archetti et al. [8] and Boudia et al. [22], inequalities (76), (77), and (78) must be replaced by:

$$\sum_{t'=1}^{t} z_i^{t'} \geq \left\lceil \frac{\sum_{t'=1}^{t} d_i^{t'} - I_i^0}{\min\left\{Q, U_i + \max_{t \in [1,t]} d_i^t\right\}} \right\rceil \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{92}$$

$$\sum_{t=t_1}^{t_2} z_i^t \geq \left\lceil \frac{\sum_{t=t_1}^{t_2} d_i^t - U_i}{\min\left\{Q, U_i + \max_{t \in [t_1,t_2]} d_i^t\right\}} \right\rceil \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T} \tag{93}$$

$$\sum_{t=t_1}^{t_2} z_i^t \geq \frac{\sum_{t=t_1}^{t_2} d_i^t - I_i^{t_1-1}}{\min\left\{Q, U_i + \max_{t \in [t_1,t_2]} d_i^t\right\}} \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T}. \tag{94}$$

Since an optimal solution for these problems can be associated with a FIFO inventory usage, Desaulniers et al. [29] introduce a suitable notation as follows. Let $I_i^{0,s} = max\left\{0, I_i^0 - \sum_{t=1}^{s} d_i^t\right\}$ be the residual initial inventory for customer i at the end of period $s \in \mathcal{T}$. The residual demand is the fraction of the demand not satisfied by the initial inventory and is defined as follows:

$$\bar{d}_i^s = \begin{cases} \max\left\{0, d_i^1 - I_l^0\right\} & \text{if } s = 1 \\ \max\left\{0, d_i^s - I_i^{0,s-1}\right\} & \text{otherwise} \end{cases} \quad \forall s \in \mathcal{T}. \tag{95}$$

Set P_{it}^+ represents all periods in which a sub-delivery for customer i in period t can be consumed in the same period, or be kept in inventory to satisfy future demands:

$$P_{it}^+ = \left\{ t | \bar{d}_i^t > 0 \right\} \cup \left\{ s > t | \bar{d}_i^s > 0 \text{ and } \sum_{t'=t}^{s-1} d_i^{t'} < U_i \right\} \cup \left\{ \rho + 1 | \sum_{t'=t}^{\rho} d_i^{t'} < U_i \right\}. \tag{96}$$

Finally, $P_{is}^- = \left\{ t \in \mathcal{T} | s \in P_{it}^+ \right\}$ represents all periods for which a quantity delivered can be used to satisfy the demand of customer i. Based on that, Desaulniers et al. [29] introduce the following inequalities:

$$\sum_{t \in P_{is}^-} z_i^t \geq 1 \qquad i \in \mathcal{V}', s \in \mathcal{T}, \text{ with } P_{is}^- \neq \emptyset. \tag{97}$$

Using the remaining quantity to tighten I_i^t and q_i^t, Lefever et al. [32] propose the following inequalities:

$$I_i^t \geq I_i^{0,t} \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{98}$$

$$q_i^t \leq U_i - I_i^{0,t} \qquad i \in \mathcal{V}', t \in \mathcal{T}. \tag{99}$$

B. Valid inequalities for the three-index formulation

Except for the flow constraints (80)–(82) which are valid only for the two-index formulation, all VIs presented in Appendix A can be extended to the three-index formulation by just adding an index k for each vehicle:

$$x_{0i}^{kt} \leq 2 \cdot z_i^{kt} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{100}$$

$$x_{ij}^{kt} \leq z_i^{kt} \qquad i < j \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{101}$$

$$x_{ij}^{kt} \leq z_j^{kt} \qquad i < j \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{102}$$

$$\sum_{k \in \mathcal{K}} \sum_{t'=1}^{t} z_i^{kt'} \geq \left\lceil \frac{\sum_{t'=1}^{t} d_i^{t'} - I_i^0}{\min\{Q, U_i\}} \right\rceil \qquad i \in \mathcal{V}', t \in \mathcal{T} \tag{103}$$

$$\sum_{k \in \mathcal{K}} \sum_{t=t_1}^{t_2} z_i^{kt} \geq \left\lceil \frac{\sum_{t=t_1}^{t_2} d_i^{t} - U_i}{\min\{Q, U_i\}} \right\rceil \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T} \tag{104}$$

$$\sum_{k \in \mathcal{K}} \sum_{t=t_1}^{t_2} z_i^{kt} \geq \frac{\sum_{t=t_1}^{t_2} d_i^{t} - I_i^{t_1-1}}{\min\{Q, U_i\}} \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T} \tag{105}$$

$$\sum_{k \in \mathcal{K}} \sum_{t=t_1}^{t_2} z_i^{kt} \geq \frac{\sum_{t=t_1}^{t_2} d_i^{t} - I_i^{t_1-1}}{\sum_{t=t_1}^{t_2} d_i^{t}} \qquad i \in \mathcal{V}', t_1 \leq t_2 \in \mathcal{T} \tag{106}$$

$$\sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} q_i^{kt} = \sum_{t \in \mathcal{T}} d_i^t - I_i^0 \qquad i \in \mathcal{V}' \tag{107}$$

$$y^t \cdot C \geq \sum_{i \in \mathcal{V}'} \left(d_i^t - I_i^{t-1}\right) - I_0^{t-1} \qquad t \in \mathcal{T} \tag{108}$$

$$\sum_{t'=1}^{t} y^{t'} \geq \left\lceil \frac{\sum_{i \in \mathcal{V}'} \sum_{t'=1}^{t} d_i^{t'} - I_i^0 - I_0^0}{C} \right\rceil \qquad t \in \mathcal{T}, t > 1 \tag{109}$$

$$\sum_{t'=t_1}^{t_2} y^{t'} \geq \left\lceil \frac{\sum_{i \in \mathcal{V}'} \sum_{t'=t_1}^{t_2} d_i^{t'} - U_i - U_0}{C} \right\rceil \qquad t_1 \leq t_2 \in \mathcal{T} \tag{110}$$

$$\sum_{t'=t_1}^{t_2} y^{t'} \geq \frac{\sum_{i \in \mathcal{V}'} \left(\sum_{t'=t_1}^{t_2} d_i^{t'}\right) - I_i^{t_1-1} - I_0^{t_1-1}}{C} \qquad t_1 \leq t_2 \in \mathcal{T} \tag{111}$$

$$p^t \leq \left(\sum_{i \in \mathcal{V}'} \sum_{t'=t}^{\rho} d_i^{t'} \right) \cdot y^t \qquad t \in \mathcal{T} \tag{112}$$

$$\sum_{t=1}^{t'} y^t \geq 1 \tag{113}$$

$$\sum_{k \in \mathcal{K}} \sum_{t=1}^{t''} z_0^{kt} \geq \left\lceil \frac{s}{Q} \right\rceil \tag{114}$$

$$I_i^{t-s-1} \geq \left(\sum_{j=0}^{s} d_i^{t-j} \right) \cdot \left(1 - \sum_{k \in \mathcal{K}} \sum_{j=0}^{s} z_i^{k(t-j)} \right) \qquad i \in \mathcal{V}', t \in \mathcal{T}, s = 0, ..., t-1 \tag{115}$$

$$\sum_{k \in \mathcal{K}} \sum_{t \in P_{is}^-} z_i^{kt} \geq 1 \qquad i \in \mathcal{V}', s \in \mathcal{T}, \text{ with } P_{is}^- \neq \emptyset. \tag{116}$$

$$q_i^{kt} \leq U_i - I_i^{0,t} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T}. \tag{117}$$

The modifications pointed out in Appendix A to consider the OU policy and the different types of instances are also valid for all VIs. Furthermore, some symmetry breaking inequalities exploring the index k of each vehicle can be formulated. The following inequalities were presented by Archetti et al. [9] for the single-vehicle IRP, and Coelho and Laporte [25] for the multi-vehicle IRP, and are also valid for the PRP.

$$z_i^{kt} \leq z_0^{kt} \qquad i \in \mathcal{V}', k \in \mathcal{K}, t \in \mathcal{T} \tag{118}$$

$$z_0^{kt} \leq z_0^{(k-1)t} \qquad k \in \mathcal{K} \backslash \{1\}, t \in \mathcal{T} \tag{119}$$

$$z_i^{kt} \leq \sum_{\substack{j \in \mathcal{V}' \\ j<i}} z_j^{(k-1)t} \qquad i \in \mathcal{V}', k \in \mathcal{K} \backslash \{1\}, t \in \mathcal{T}. \tag{120}$$

Inequalities (118) enable vehicle k to visit customer i only if this vehicle departs from the plant in t. Inequalities (119) ensure that vehicle k can leave the plant only if vehicle $k-1$ is used in the same period, while inequalities (120) impose that if a customer i is visited by vehicle k, then at least one other customer with a smaller index is visited by vehicle $k-1$. Darvish et al. [28] also presented some other symmetry breaking inequalities that are valid for the PRP, given by:

$$z_i^{kt} \leq \sum_{j=1}^{i-1} z_j^{lt} \qquad k \in \mathcal{K} \backslash \{1\}, l \in \{1, 2, ..., k-1\}, i \in \{k, k+1, ..., n\}, t \in \mathcal{T} \tag{121}$$

$$(k-1) \cdot z_i^{kt} \leq \sum_{j=1}^{i-1} \sum_{l=1}^{k-1} z_j^{lt} \qquad i \in \mathcal{V}' \backslash \{1\}, k \in \mathcal{K} \backslash \{1\}, t \in \mathcal{T}. \tag{122}$$

Inequalities (121) guarantee that if customer i is visited by vehicle k, every vehicle with an index lower than

k must visit a customer with an index lower than i. Additionally, inequalities (122) ensure that customers with smaller indices are also assigned to vehicles with smaller indices.

C. Performance profile for the IRP instances

Figure 2 shows a $\log_2$ scaled view of the performance profiles of the B&C [CL], BP&C, I-B&C [M], I-B&C [G], and 3FP-B&C. We used a performance profile with a convergence test to evaluate the level of accuracy of the different methods. Computation times were adjusted to account for different hardware configurations using `https://www.cpubenchmark.net/singleThread.html`. We can observe that 3FP-B&C presents the best results in terms of computational time and accuracy, i.e, on these IRP instances the 3FP-B&C clearly dominates all other methods. For the level of accuracy (defined as $gap < 1\%$), the 3FP-B&C found the target solution for 66% of the instances, while I-B&C [G] found it for 48%, B&C [CL] for 45%, I-B&C [M] for 44%, and BP&C for 34% of the instances. We also observe in the graph that 3FP-B&C was the fastest solver for 44% of the instances, while BP&C was the fastest solver for 12%, B&C [CL] and I-B&C [G] for 5% of the instances, and I-B&C [M] for fewer than 1% of the instances. Other performance differences can be observed between the approaches. For example, consider the performance profiles at a performance ratio of $\tau = 2$ (x-axis = 1 as $log_2(\tau) = 1$): the 3FP-B&C found the target solution for 50% of instances within a factor of two from the best performance, while BP&C found the target for 15%, and B&C [CL] and I-B&C [M] for 10% of the instances. This is a significant difference, highlighting once again the excellent performance of our 3FP-B&C.

D. Performance profile for the PRP instances of Adulyasak et al. [3]

Figure 3 presents the performance profile of the runtime for instances with the ML policy, while Figure 4 presents the same for the OU policy. Again, the level of accuracy is $gap < 1\%$ and the computational times were once again adjusted. We observe that in both cases the 3FP-B&C curves dominate all other methods. Under ML, the 3FP-B&C is the fastest solver in 52% of the instances and found the target solution for 96% of the instances. The LS-B&C and B&C [Ad] found the target solution for 89% and 85%, respectively. For the OU policy, our method is the fastest for 45% of the instances and the EXM is the fastest solver for 41% of the instances. However, the 3FP-B&C found the target solution for 99% of the instances, while EXM found it for 94%. Both methods are better than B&C [Ad] and LS-B&C.

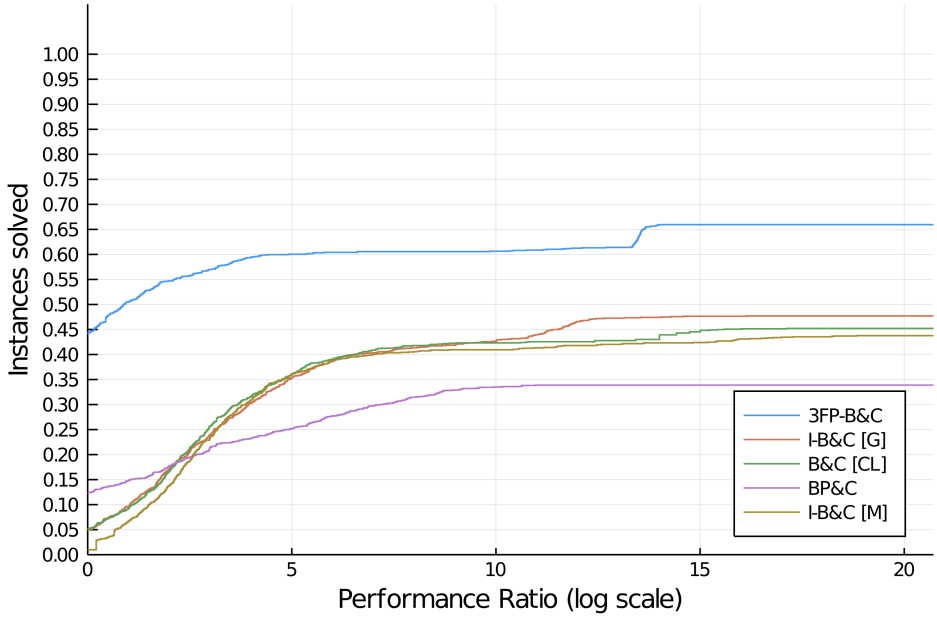


Figure 2: Performance profile of runtime for B&C [CL] , BP&C, I-B&C [M], I-B&C [G], and 3FP-B&C algorithms.

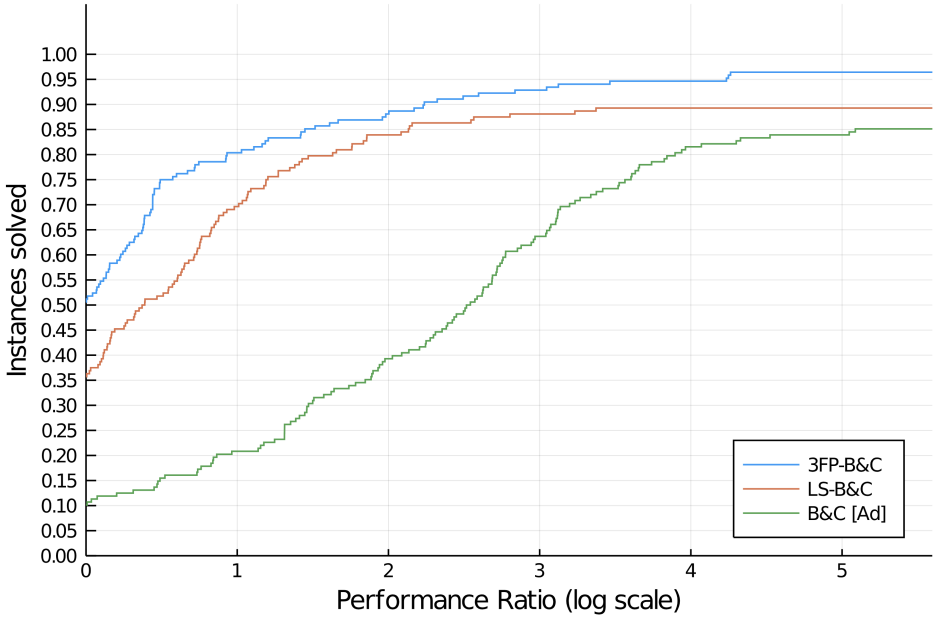


Figure 3: Performance profile of the runtime for B&C [Ad], LS-B&C, and 3FP-B&C algorithms for ML instances.

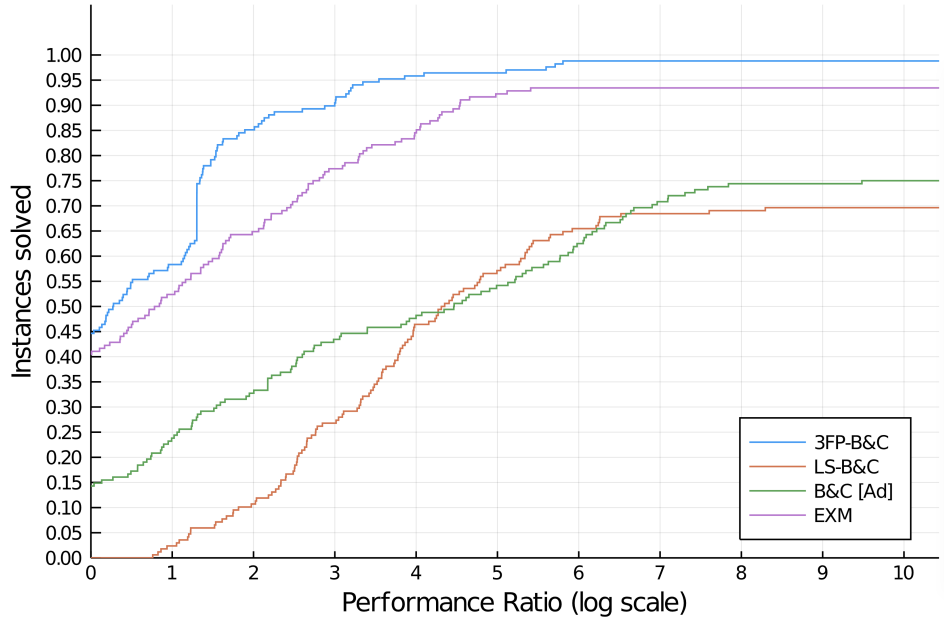


Figure 4: Performance profile of the runtime for B&C [Ad], LS-B&C, EXM, and 3FP-B&C algorithms for OU instances.

References

[1] N. Absi, C. Archetti, S. Dauzère-Pérès, and D. Feillet. A two-phase iterative heuristic approach for the production routing problem. *Transportation Science*, 49(4):784–795, 2015.

[2] Y. Adulyasak and R. Cordeau, J.-F. Jans. The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, 55:141–152, 2015.

[3] Y. Adulyasak, J.-F. Cordeau, and R. Jans. Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, 26(1):103–120, 2014.

[4] Y. Adulyasak, J.-F. Cordeau, and R. Jans. Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science*, 48(1):20–45, 2014.

[5] A. Alvarez, P. Munari, and R. Morabito. Iterated local search and simulated annealing algorithms for the inventory routing problem. *International Transactions in Operational Research*, 25(6):1785–1809, 2018.

[6] A. Alvarez, J.-F. Cordeau, R. Jans, P. Munari, and R. Morabito. Formulations, branch-and-cut and a hybrid heuristic algorithm for an inventory routing problem with perishable products. *European Journal of Operational Research*, 283(2):511–529, 2020.

[7] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Løketangen. Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9):1515–1536, 2010.

[8] A. Archetti, L. Bertazzi, G. Paletta, and M. G. Speranza. Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research*, 38(12):1731–1746, 2011.

[9] C. Archetti, L. Bertazzi, G. Laporte, and M. G. Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41(3):382–391, 2007.

[10] C. Archetti, L. Bertazzi, A. Hertz, and M. G. Speranza. A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116, 2012.

[11] C. Archetti, N. Boland, and M. G. Speranza. A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, 29(3):377–387, 2017.

[12] C. Archetti, G. Guastaroba, D. L. Huerta-Muñoz, and M. G. Speranza. A kernel search heuristic for the multivehicle inventory routing problem. *International Transactions in Operational Research*, 28(6): 2984–3013, 2021.

[13] J. Aráoz, E. Fernández, and O. Meza. Solving the prize-collecting rural postman problem. *European Journal of Operational Research*, 196(3):886–896, 2009.

[14] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. Separating capacity constraints in the CVRP using tabu search. *European Journal of Operational Research*, 106(2):546–557, 1998.

[15] M. Avci and S. T. Yildiz. A matheuristic solution approach for the production routing problem with visit spacing policy. *European Journal of Operational Research*, 279(2):572–588, 2019.

[16] P. Avella, M. Boccia, and L. A. Wolsey. Single-item reformulations for a vendor managed inventory routing problem: Computational experience with benchmark instance. *Networks*, 65(2):129–138, 2015.

[17] P. Avella, M. Boccia, and L. A. Wolsey. Single-period cutting planes for inventory routing problems. *Transportation Science*, 52(3):497–508, 2018.

[18] F. Barahona and M. Grötschel. On the cycle polytope of a binary matroid. *Journal of Combinatorial Theory - Series B*, 40(1):40–62, 1986.

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

[20] L. Bertazzi, L. C. Coelho, A. De Maio, and D. Laganà. A matheuristic algorithm for the multi-depot inventory routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 122: 524–544, 2019.

[21] M. Boudia and C. Prins. A memetic algorithm with dynamic population management for an integrated production–distribution problem. *European Journal of Operational Research*, 195(3):703–715, 2009.

[22] M. Boudia, M. A. O. Louly, and C. Prins. A reactive GRASP and path relinking for a combined production–distribution problem. *Computers & Operations Research*, 34(11):3402–3419, 2007.

[23] P. Chandra and M. L. Fisher. Coordination of production and distribution planning. *European Journal of Operational Research*, 72(3):503–517, 1994.

[24] M. Chitsaz, J.-F. Cordeau, and R. Jans. A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing*, 31(1):134–152, 2019.

[25] L. C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558–565, 2013.

[26] L. C. Coelho and G. Laporte. Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155:391–397, 2014.

[27] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2014.

[28] M. Darvish, L. C. Coelho, and R. Jans. Comparison of symmetry breaking and input ordering techniques for routing problems. Technical Report CIRRELT-2020-22, Montréal, 2020.

[29] G. Desaulniers, J. G. Rakke, and L. C. Coelho. A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 50(3):1060–1076, 2015.

[30] P. Diniz, R. Martinelli, and M. Poggi. An efficient matheuristic for the inventory routing problem. In M. Baïou, B. Gendron, O. Günlük, and A. R. Mahjoub, editors, *Combinatorial Optimization. ISCO 2020. Lecture Notes in Computer Science*, pages 273–285. Springer International Publishing, 2020.

[31] T. A. Guimarães, C. M. Schenekemberg, L. C. Coelho, C. T. Scarpin, and J. E. Pécora-Jr. Mechanisms for feasibility and improvement for inventory-routing problems. Technical Report CIRRELT-2020-12, Montréal, 2020.

[32] W. Lefever, E.-H. Aghezzaf, K. Hadj-Hamou, and B. Penz. Analysis of an improved branch-and-cut formulation for the inventory-routing problem with transshipment. *Computers & Operations Research*, 98:137–148, 2018.

[33] Y. Li, F. Chu, C. Chu, and Z. Zhu. An efficient three-level heuristic for the large-scaled multi-product production routing problem with outsourcing. *European Journal of Operational Research*, 272(3):914–927, 2019.

[34] E. Manousakis, P. Repoussis, E. Zachariadis, and C. Tarantilis. Improved branch-and-cut for the inventory routing problem based on a two-commodity flow formulation. *European Journal of Operational Research*, 290(3):870–885, 2021.

[35] E. G. Manousakis, G. A. Kasapidis, C. T. Kiranoudis, and E. E. Zachariadis. An infeasible space exploring matheuristic for the production routing problem. *European Journal of Operational Research*, 298(2):478–495, 2022.

[36] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[37] Y. Qiu, L. Wang, X. Xu, X. Fang, and P. M. Pardalos. A variable neighborhood search heuristic algorithm for production routing problems. *Applied Soft Computing*, 66:311–318, 2018.

[38] R. A. Russell. Mathematical programming heuristics for the production routing problem. *International Journal of Production Economics*, 193:40–49, 2017.

[39] C. M. Schenekemberg, C. T. Scarpin, J. E. Pécora-Jr, T. A. Guimarães, and L. C. Coelho. The two-echelon inventory-routing problem with fleet management. *Computers & Operations Research*, 121: 104944, 2020.

[40] C. M. Schenekemberg, C. T. Scarpin, J.-E. Pécora-Jr., T. A. Guimarães, and L. C. Coelho. The two-echelon production-routing problem. *European Journal of Operational Research*, 288(2):436–449, 2021.

[41] J. Skålnes, H. Andersson, G. Desaulniers, and M. Stålhane. An improved formulation for the inventory routing problem with time-varying demands. *European Journal of Operational Research*, 302(3):1189–1201, 2022.

[42] O. Solyalı and H. Süral. An effective matheuristic for the multivehicle inventory routing problem. *Transportation Science*, 56(4):1044–1057, 2022.

[43] O. Solyalı and H. Süral. A multi-phase heuristic for the production routing problem. *Computers & Operations Research*, 87:114–124, 2017.

[44] S. T. Vadseth, H. Andersson, and M. Stålhane. An iterative matheuristic for the inventory routing problem. *Computers & Operations Research*, 131:105262, 2021.

[45] S. T. Vadseth, H. Andersson, M. Stålhane, and M. Chitsaz. A multi-start route improving matheuristic for the production routing problem. Technical report, Trondheim, 2021.

[46] T. Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:1–11, 2022.

[47] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60:611–624, 2012.

[48] Z. Zhang, Z. Luo, R. Baldacci, and A. Lim. A benders decomposition approach for the multivehicle production routing problem with order-up-to-level policy. *Transportation Science*, 55(1):160–178, 2021.