The Consistent Production Routing Problem

Aldair Alvarez, Jean-François Cordeau, Raf Jans

HEC Montréal and GERAD, Montréal, Canada

Abstract

This paper introduces the consistent production routing problem in a setting with multiple plants and products. The problem consists in finding minimum-cost production-routing plans that also meet specific consistency requirements. In our context, consistency is defined as the degree to which some specified features of the solution remain invariant over time. We consider four forms of consistency, namely: driver, source, product and plant consistency. For each of these consistency requirements, there is a target maximum value defining the decision-maker's tolerance to deviations from a perfectly consistent solution. These targets are enforced as soft constraints whose violations need to be minimized when optimizing the integrated production and routing plan. We present a mathematical formulation for the problem and an exact branch-and-cut algorithm, enhanced with valid inequalities and specific branching priorities. We also propose a heuristic solution method based on iterated local search and several mathematical programming components. Experiments on a large benchmark set of newly introduced instances show that the enhancements substantially improve the performance of the exact algorithm and that the heuristic method performs robustly for production routing problems with different consistency requirements as well as for standard versions of the problem. We also analyze the cost-consistency trade-off of the solutions, confirming that it is possible to impose consistency without excessively increasing the cost. The results also reveal the impact of the first time period when optimizing and measuring the consistency features we study.

Keywords: production routing, consistency, multiple plants, branch-and-cut, iterated local search.

1 Introduction

Production routing problems (PRPs) are planning problems that integrate several activities across the supply chain, such as production, storage, distribution and routing. PRPs have received increased attention in the past two decades due to the substantial costs and performance advantages they bring. However, in addition to these benefits, companies might be interested in finding solutions that meet additional standards and offer reliable service levels to their customers. For example, it could be desirable to have PRP solutions in which some features remain relatively invariant over time. This property, often referred to as *consistency*, is especially relevant when dealing with the planning of several costly resources.

The concept of consistency in logistics and transportation optimization has drawn significant interest in recent years, mainly because of its relevance in some practical settings in several industries. On the one hand, consistency at the customer level, e.g., reducing the number of different drivers that visit each customer over a certain period, can increase their satisfaction due to familiarity and customization of the service, which can, in turn, positively impact the revenues of the company. From a server point of view, this form of consistency can also improve the efficiency of the transportation operations since drivers become familiar with a stable set of customers, their locations and needs. On the other hand, consistency at the production level, e.g., reducing the variety of different products made over a defined planning horizon, can increase the productivity of the plants and potentially improve the quality of the processes.

In the literature, such aspects related to consistency have been addressed in several logistics settings, particularly in the vehicle routing problem (VRP) context. Groër et al. (2009) introduced the concept of consistency in the VRP by defining the so-called consistent VRP (ConVRP). This problem consists of a multi-day VRP in which each customer has to be served by the same driver (driver consistency) and at approximately the same time (time consistency) each day they require service. An upper bound on the maximum difference between the service start times at each customer on different days is imposed. The authors showed through computational experiments on synthetic and real-life instances that, on average, finding solutions meeting all the consistency standards may require a significant increase in total travel time. In some problem instances, however, only a slight increase in the total travel time was observed. Smilowitz et al. (2013) addressed a periodic VRP which, in addition to routing costs, also aims at minimizing the number of different drivers assigned to each customer and the number of times each driver visits distinct regions. Their work differs from the ConVRP by modeling the consistency requirements as objectives rather than as constraints. The authors assessed the value of taking these forms of consistency into account as objectives. They observed that depending on the importance given to these objectives, the consistency metrics of the solutions can be improved without sacrificing too much on other operational metrics. Next, motivated by the fact that the ConVRP may be too restrictive in some applications, Kovacs et al. (2015) proposed the generalized ConVRP. This variant extends the ConVRP by permitting each customer to be served by a set of drivers (instead of a single one) and by allowing violations to the time consistency (which they penalize in the objective function). The authors analyzed the trade-off between travel cost and the studied consistency features, observing that the considered generalizations can result in significant cost savings, especially when the weight of the time consistency violation is relatively low. Motivated by these studies, many VRPs with consistency requirements have later been described and analyzed in the literature. Some examples include potential applications in health care logistics with time consistency (Tellez et al., 2021) and driver consistency (Braekers and Kovacs, 2016), delivery of pharmaceutical products with driver and time consistency (Campelo et al., 2019), and food distribution with driver consistency (Lespay and Suchan, 2021). For more applications, we refer the interested reader to the review presented by Kovacs et al. (2014).

Coelho et al. (2012) and Diabat et al. (2021) presented studies considering consistency in the context of the optimization of integrated supply chain activities, specifically for the inventory routing problem (IRP). Coelho et al. (2012) addressed different requirements in the IRP, such as consistency in the delivery quantities and driver assignments, minimum and maximum intervals between two consecutive visits to each customer, and vehicle filling rates. The authors evaluated the compromise between these forms of consistency and the solution cost. Diabat et al. (2021) studied an IRP that includes two forms of consistency by separating customers into mutually exclusive clusters which are fixed throughout the horizon. In their problem setting, the distribution to each cluster is performed by different vehicles, allowing to reach solutions with driver consistency as well as route consistency (in which both the clusters and the route within each cluster are fixed). The authors showed that the added cost of imposing these forms of consistency can be relatively small.

To the best of our knowledge, the study of consistency requirements in IRPs is limited to the two aforementioned papers, while consistency has not been studied for PRPs. As such, in this paper we propose and study the consistent PRP in a setting with multiple production plants and multiple products. We refer to this problem as to the ConMPRP. Notice that the multiple plant aspect has been addressed in several other settings, such as the lot-sizing problem (Sambasivan and Yahya, 2005; Nascimento and Toledo, 2008), the joint production-inventory-location problem (Shahabi et al., 2018), network design (Cardona-Valdés et al., 2014) as well as in the PRP with a single product (Li et al., 2020). PRPs with multiple plants and multiple products were studied by Marchetti et al. (2014) and Zhang et al. (2017) in the context of the industrial gas supply chain. In this paper, we model the ConMPRP extending the standard multi-plant multi-product PRP (MPRP) by considering four forms of consistency, as follows.

- *Driver consistency*: in this setting, it is desired that a limited number of different drivers visit each customer over the planning horizon. Solutions with this property can lead to enhanced service quality and efficiency since drivers visit the same customer locations more often (Smilowitz et al., 2013).
- *Source consistency*: this form of consistency favors reducing the number of different plants from which a specific customer receives its orders over the entire horizon. The idea is to reduce the variability in the quality of the products dispatched to each customer by providing solutions in which each customer receives products from a limited set of plants. This variability might appear due to differences in processes and technologies at different production plants, even if they are owned by the same company and apply the same production standards for the individual products. From the manufacturer's point of view, this form of consistency can help to simplify the tracing of lots in case of specific production problems. Besides, at the customer level, reducing the number of sources for the individual products can potentially help to reduce the inventory management complexity.
- *Product consistency*: at each production plant, it is preferred to produce a limited number of different products throughout the planning horizon. The idea is to take advantage of limited process flexibility, i.e., reducing the set of products that each plant produces, to increase their productivity and quality. Since total process flexibility (i.e., equipping every plant to manufacture every product) can be excessively costly (Jordan and Graves, 1995), considering the concept of limited flexibility becomes relevant in this context. Whereas the value of limited process flexibility has been analyzed mostly in a stochastic context (Jordan and Graves, 1995; Graves and Tomlin, 2003; Mak and Shen, 2009), Fiorotto et al. (2018) indicated that limited flexibility also has value in the context of deterministic production planning.
- *Plant consistency*: in this configuration of the problem, it is preferred that each product is manufactured at a limited number of plants over the planning horizon, e.g., to achieve economies of scale.

Solutions with this feature might also benefit from enhanced product quality variability. Moreover, from an item point of view, it might not be necessary to design the system with full redundancy (i.e., every product can be manufactured at every plant) to minimize disruption risks under the presence of uncertainties (Schmitt, 2011).

In the ConMPRP, the different consistency requirements are addressed by defining a target maximum value for each of the features. These requirements are optimized simultaneously with the production and routing plan. We impose these targets as soft constraints given that enforcing them as hard constraints may be too restrictive and, in practice, a certain degree of violation might be acceptable. The ConMPRP then favors finding solutions considering the consistency requirements by penalizing the violations of the corresponding maximum targets in the objective function. This type of approach is in line with those used in the works of Smilowitz et al. (2013) and Kovacs et al. (2015).

The contributions of this paper are fourfold. First, we introduce the ConMPRP, which is defined formally in Section 2. Second, in Section 3, we propose a mathematical formulation for the ConMPRP, which is solved using a branch-and-cut algorithm containing some valid inequalities and enhanced branching priorities. Third, we present a heuristic solution method based on iterated local search (ILS) and several mathematical programming components. This heuristic method, described in Section 4, performs robustly for the ConMPRP, the MPRP and the standard PRP. Finally, we perform a series of computational experiments, on the basis of new and existing benchmark instances, to evaluate the trade-off between cost and consistency of the solutions in the PRP context. In particular, we analyze the change in the cost and solution structure when changing the relative importance of each of the consistency features. These analyses are presented in Section 5, followed by the main conclusions of the paper, in Section 6.

2 Problem Definition

We now describe the MPRP and then introduce the ConMPRP with the different forms of consistency that we consider in this study.

The MPRP considers a network with multiple production facilities (plants) and multiple customers that demand several products over a multi-period discrete planning horizon. We define this problem on a graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \mathcal{M} \cup \mathcal{C}$ is the set of all the facilities of the network, $\mathcal{M}$ is the set of plants and $\mathcal{C}$ is the set of customers. Moreover, $\mathcal{E} = \{(i,j)\colon i \in \mathcal{M}, j \in \mathcal{C}\} \cup \{(i,j)\colon i,j \in \mathcal{C}, i < j\}$ is the edge set that links the facilities and represents the distribution network. We denote by $\mathcal{T}$ the set of periods of the planning horizon and by $\mathcal{P}$ the set of products. Furthermore, each plant i has its own fleet of homogeneous vehicles, represented by the set $\mathcal{K}_i$, with $\mathcal{K} = \bigcup_{i\in\mathcal{M}} \mathcal{K}_i$ representing the joint fleet of all the plants.

Each customer i demands d_{ip}^t units of product p in period t and, at the beginning of the planning horizon, each facility i has I_{ip}^0 units of product p in inventory. In each period, every plant i faces a limited production capacity C_i, and the inventory level at each facility i cannot exceed L_i. Also, each vehicle k has a transportation capacity Q. The costs involved in the system are as follows: f_{ip} is the fixed production setup cost of product p at plant i; e_{ip} represents the unit production cost of product p at plant i; h_{ip} is the unit inventory holding cost of product p at facility i, and; c_{ij} is the travel cost between facilities i and j. We assume that the travel costs satisfy the triangle inequality.

The MPRP consists of determining a minimum-cost production and routing plan defining, for each period, the setup and production quantities of each product at each plant; the delivery quantities to each customer from each plant; and the vehicle routes required to deliver those quantities. This plan must satisfy all the production, storage and transportation capacities, ensuring that no out-of-stock occurs at any facility. Following the practice in the literature, the timing of the events in the MPRP is as follows: production at the plants, deliveries to the customers, and demand consumption.

The ConMPRP builds upon the MPRP by including some features of the solutions to define consistency metrics. In this work, we study four forms of consistency, as follows: *driver consistency*, in which it is desired that a limited number of different drivers visits each customer over the planning horizon; *source consistency*, in which it is favored that there is a limited number of different plants dispatching to each specific customer over the entire horizon; *product consistency*, in which it is preferred producing a limited number of different products at each plant over the horizon; *plant consistency*, in which it is desired that a limited number of different plants manufacture each product over the planning horizon. For each of these features, there is a target (or ideal) maximum value, which needs to be taken into account in the optimization of the production and routing plan. For this purpose, consider the following additional notation. Let V_i be the maximum number of different drivers that should visit customer $i \in \mathcal{C}$ over the planning horizon. This value defines the decision-maker's tolerance in terms of driver consistency requirements for each individual customer. Analogously, let E_i, O_i and M_p be the target maximum number of different plants that should deliver to customer $i \in \mathcal{C}$, products that plant $i \in \mathcal{M}$ should produce, and plants at which each product $p \in \mathcal{P}$ should be manufactured, respectively, over the entire planning horizon.

In the ConMPRP, we impose these targets as soft constraints, given that it might be too restrictive to enforce them as hard constraints. Besides, in practice, their violation may be acceptable to some extent. As such, the violations above the given maximum targets are penalized in the objective function. Let o^{driver}, o^{source}, o^{product} and o^{plant}, be the unit cost of exceeding V_i, E_i, O_i and M_p, respectively. These costs penalize the violation of the targets and define the importance of each consistency feature with respect to (w.r.t.) the other cost components of the objective function. Using penalties to favor the consistency requirements of certain solution features provides flexibility to the system by giving the decision-maker the freedom to choose which consistency features to optimize and with what significance, according to specific situations. Another advantage of this approach is that it allows solving the problem using methods that do not require specific operators to handle or to improve the parts of the solutions concerning these requirements.

Note that there are some relationships between these forms of consistency, although they have different natures and relate to distinct parts of the solutions. For example, when we try to limit the number of different drivers that visit a given customer, we might be implicitly reducing the number of distinct plants that serve as a product source for the customer. Analogously, narrowing down the set of different plants that dispatch to a customer will likely reduce the number of different drivers that visit it. However, since each plant has its own vehicle fleet, even if only a single plant serves as the source for a customer, the solutions might not have a one-to-one customer-driver assignment. In the ConMPRP, it is assumed that vehicle and driver are coupled together and, therefore, they are considered a single entity and these terms will be used interchangeably. From a plant-product assignment point of view, notice that we might have perfect plant consistency, for example, when only a single plant makes all the products throughout the horizon. In this scenario, every product is always manufactured at the same plant, but the product consistency of the plant might exceed the given targets (as it makes all the products). On the other hand, even if every plant produces only a single product, we might still have a given product being manufactured at different plants. In our computational experiments, we empirically analyze these and other relations between the forms of consistency that we address in this study.

3 Mathematical Formulation

This section presents, first, a base mathematical model for the problem followed by some details of the separation routine used in the branch-and-cut algorithm and the additional components that we use to enhance its performance.

3.1 Base Formulation

In this section we propose a mathematical formulation for the ConMPRP. This model involves the following integer decision variables:

- x_{ij}^{kt}: integer variable indicating the number of times vehicle k traverses edge (i, j) in period t;
- y_{i}^{kt}: binary variable equal to one if vehicle k visits facility i in period t, and zero otherwise;
- z_{ip}^{t}: binary variable equal to one if product p is produced at plant i in period t, and zero otherwise;
- λ_{i}^{k}: binary variable equal to one if driver k visits customer i, and zero otherwise;
- α_{ij}: binary variable equal to one if plant j dispatches products to customer i, and zero otherwise;
- θ_{ip}: binary variable equal to one if plant i produces product p, and zero otherwise.

Additionally, consider the following continuous variables:

- I_{ip}^{t}: inventory level of product p at facility i at the end of period t;
- P_{ip}^{t}: quantity of product p produced in plant i in period t;
- q_{ip}^{kt}: quantity of product p delivered to customer i by vehicle k in period t;
- Λ_i: violation level of the driver consistency target for customer i;
- Ω_i: violation level of the source consistency target for customer i;
- Θ_i: violation level of the product consistency target for plant i;
- Φ_p: violation level of the plant consistency target for product p.

Based on these variables and the notation introduced in Section 2, the formulation is as follows:

$$\min \quad \sum_{t\in\mathcal{T}}\left(\sum_{i\in\mathcal{M}}\sum_{p\in\mathcal{P}}\left(f_{ip}z_{ip}^t+e_{ip}P_{ip}^t\right)+\sum_{i\in\mathcal{N}}\sum_{p\in\mathcal{P}}h_{ip}I_{ip}^t+\sum_{(i,j)\in\mathcal{E}}\sum_{k\in\mathcal{K}}c_{ij}x_{ij}^{kt}\right) + \sum_{i\in\mathcal{C}}o^{\text{driver}}\Lambda_i+\sum_{i\in\mathcal{C}}o^{\text{source}}\Omega_i+\sum_{i\in\mathcal{M}}o^{\text{product}}\Theta_i+\sum_{p\in\mathcal{P}}o^{\text{plant}}\Phi_p \tag{1}$$

$$\text{s.t.} \quad I_{ip}^t = I_{ip}^{t-1}+P_{ip}^t-\sum_{k\in\mathcal{K}_i}\sum_{j\in\mathcal{C}}q_{jp}^{kt} \qquad i\in\mathcal{M},\, p\in\mathcal{P},\, t\in\mathcal{T}, \tag{2}$$

$$I_{ip}^t = I_{ip}^{t-1}+\sum_{k\in\mathcal{K}}q_{ip}^{kt}-d_{ip}^t \qquad i\in\mathcal{C},\, p\in\mathcal{P},\, t\in\mathcal{T}, \tag{3}$$

$$\sum_{p\in\mathcal{P}}I_{ip}^t\le L_i \qquad i\in\mathcal{M},\, t\in\mathcal{T}, \tag{4}$$

$$\sum_{p\in\mathcal{P}}I_{ip}^t\le L_i-\sum_{p\in\mathcal{P}}d_{ip}^t \qquad i\in\mathcal{C},\, t\in\mathcal{T}, \tag{5}$$

$$P_{ip}^t\le\min\left\{C_i,\sum_{j\in\mathcal{C}}\sum_{\tau=t}^{|\mathcal{T}|}d_{jp}^{\tau}\right\}z_{ip}^t \qquad i\in\mathcal{M},\, p\in\mathcal{P},\, t\in\mathcal{T}, \tag{6}$$

$$\sum_{p\in\mathcal{P}}P_{ip}^t\le C_i \qquad i\in\mathcal{M},\, t\in\mathcal{T}, \tag{7}$$

$$\sum_{p\in\mathcal{P}}q_{ip}^{kt}\le\min\{Q,L_i\}y_i^{kt} \qquad i\in\mathcal{C},\, k\in\mathcal{K},\, t\in\mathcal{T}, \tag{8}$$

$$\sum_{j\in\mathcal{C}}\sum_{p\in\mathcal{P}}q_{jp}^{kt}\le Qy_i^{kt} \qquad i\in\mathcal{M},\, k\in\mathcal{K}_i,\, t\in\mathcal{T}, \tag{9}$$

$$\sum_{\substack{j\in\mathcal{N}:\\(j,i)\in\mathcal{E}}}x_{ji}^{kt}+\sum_{\substack{j\in\mathcal{C}:\\(i,j)\in\mathcal{E}}}x_{ij}^{kt}=2y_i^{kt} \qquad i\in\mathcal{C},\, k\in\mathcal{K},\, t\in\mathcal{T}, \tag{10}$$

$$\sum_{\substack{j\in\mathcal{N}:\\(i,j)\in\mathcal{E}}}x_{ij}^{kt}=2y_i^{kt} \qquad i\in\mathcal{M},\, k\in\mathcal{K}_i,\, t\in\mathcal{T}, \tag{11}$$

$$\sum_{k\in\mathcal{K}}y_i^{kt}\le 1 \qquad i\in\mathcal{C},\, t\in\mathcal{T}, \tag{12}$$

$$\sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}:\\j>i}}x_{ij}^{kt}\le\sum_{i\in\mathcal{B}}y_i^{kt}-y_\ell^{kt} \qquad \forall\mathcal{B}\subseteq\mathcal{C}\colon|\mathcal{B}|\ge 2,\, k\in\mathcal{K},\, t\in\mathcal{T},\, \ell\in\mathcal{B}, \tag{13}$$

$$\sum_{t\in\mathcal{T}}y_i^{kt}\le|\mathcal{T}|\lambda_i^k \qquad i\in\mathcal{C},\, k\in\mathcal{K}, \tag{14}$$

$$\Lambda_i\ge\sum_{k\in\mathcal{K}}\lambda_i^k-V_i \qquad i\in\mathcal{C}, \tag{15}$$

$$\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_j}y_i^{kt}\le|\mathcal{T}|\alpha_{ij} \qquad i\in\mathcal{C},\, j\in\mathcal{M}, \tag{16}$$

$$\Omega_i\ge\sum_{j\in\mathcal{M}}\alpha_{ij}-E_i \qquad i\in\mathcal{C}, \tag{17}$$

$$\sum_{t \in \mathcal{T}} z_{ip}^t \leq |\mathcal{T}| \theta_{ip} \qquad i \in \mathcal{M},\, p \in \mathcal{P}, \tag{18}$$

$$\Theta_i \geq \sum_{p \in \mathcal{P}} \theta_{ip} - O_i \qquad i \in \mathcal{M}, \tag{19}$$

$$\Phi_p \geq \sum_{i \in \mathcal{M}} \theta_{ip} - M_p \qquad p \in \mathcal{P}, \tag{20}$$

$$I_{ip}^t \geq 0 \qquad i \in \mathcal{N},\, p \in \mathcal{P},\, t \in \mathcal{T}, \tag{21}$$

$$P_{ip}^t \geq 0 \qquad i \in \mathcal{M},\, p \in \mathcal{P},\, t \in \mathcal{T}, \tag{22}$$

$$q_{ip}^{kt} \geq 0 \qquad i \in \mathcal{C},\, p \in \mathcal{P},\, k \in \mathcal{K},\, t \in \mathcal{T}, \tag{23}$$

$$\Lambda_i \geq 0 \qquad i \in \mathcal{C}, \tag{24}$$

$$\Omega_i \geq 0 \qquad i \in \mathcal{C}, \tag{25}$$

$$\Theta_i \geq 0 \qquad i \in \mathcal{M}, \tag{26}$$

$$\Phi_p \geq 0 \qquad p \in \mathcal{P}, \tag{27}$$

$$z_{ip}^t \in \{0,1\} \qquad i \in \mathcal{M},\, p \in \mathcal{P},\, t \in \mathcal{T}, \tag{28}$$

$$y_i^{kt} \in \{0,1\} \qquad i \in \mathcal{C},\, k \in \mathcal{K},\, t \in \mathcal{T}, \tag{29}$$

$$y_i^{kt} \in \{0,1\} \qquad i \in \mathcal{M},\, k \in \mathcal{K}_i,\, t \in \mathcal{T}, \tag{30}$$

$$x_{ij}^{kt} \in \{0,1\} \qquad (i,j) \in \mathcal{E}\colon i \notin \mathcal{M},\, k \in \mathcal{K},\, t \in \mathcal{T}, \tag{31}$$

$$x_{ij}^{kt} \in \{0,1,2\} \qquad (i,j) \in \mathcal{E}\colon i \in \mathcal{M},\, k \in \mathcal{K},\, t \in \mathcal{T}, \tag{32}$$

$$\lambda_i^k \in \{0,1\} \qquad i \in \mathcal{C},\, k \in \mathcal{K}, \tag{33}$$

$$\alpha_{ij} \in \{0,1\} \qquad i \in \mathcal{C},\, j \in \mathcal{M}, \tag{34}$$

$$\theta_{ip} \in \{0,1\} \qquad i \in \mathcal{M},\, p \in \mathcal{P}. \tag{35}$$

The objective function (1) consists of minimizing the total cost, which is the sum of setup, production, inventory, and transportation costs, plus the total penalty for violating the maximum targets of the consistency features. Constraints (2) and (3) define the inventory balance at the plants and customers, respectively. Constraints (4) and (5) impose the storage capacity at the plants and customer locations, respectively. Note that the storage capacity of the customers is imposed before demand consumption, as is the usual practice in the standard IRP and PRP literature (Coelho et al., 2012; Adulyasak et al., 2014a). Constraints (6) allow positive production quantities only if a setup is carried out, and constraints (7) impose the maximum production capacity at the plants. Likewise, constraints (8) allow positive delivery quantities only if a visit occurs, while (9) impose the transportation capacity of the vehicles. Constraints (10) and (11) require the number of edges incident to a node to be 2 if a customer is visited (for $i \in \mathcal{C}$) or if the vehicle leaves its plant (for $i \in \mathcal{M}$), respectively. Constraints (12) allow at most one visit to every customer in each time period and constraints (13) are subtour elimination constraints (SECs). The violation of the driver consistency target is computed with constraints (14) and (15), where (14) activate the variable λ if the corresponding customer is visited at least once by the given vehicle over the horizon while (15) capture the violation level, if any, w.r.t. the target V_i. Analogously, constraints (16) and (17) compute the source consistency violation w.r.t. the target E_i. Constraints (18) activate the variable θ when the production of the given product occurs at the corresponding plant at least once over the entire horizon. Then, constraints (19) and (20) compute the violation level of the product and plant consistency, respectively. Note that we only consider a violation as the level by which the given consistency feature exceeds the target values (V_i, E_i, O_i, and M_p). Finally, constraints (21)-(35) define the domain of the decision variables.

Given these definitions and assumptions, it is possible to show that, for any ConMPRP instance that has an optimal solution, the optimal solutions have the following property:

Property 1. *The number of vehicles that visit each customer is always larger than or equal to the number of plants that dispatch to the customer.*

To show that this property holds, notice that for a given customer $i \in \mathcal{C}$ and plant $j \in \mathcal{M}$, the term $\sum_{k\in\mathcal{K}_j} \lambda_i^k$ defines the number of vehicles of plant j that visit customer i over the planning horizon. Next, given that α_{ij} specifies whether or not plant j dispatches some products to customer i (i.e., one of its vehicles visits the customer), the relation $\alpha_{ij} \leq \sum_{i\in\mathcal{K}_j} \lambda_i^k$ holds for every customer $i \in \mathcal{C}$ and plant $j \in \mathcal{M}$. Hence, summing over all the plants on both sides, the relation $\sum_{j\in\mathcal{M}} \alpha_{ij} \leq \sum_{j\in\mathcal{M}} \sum_{k\in\mathcal{K}_j} \lambda_i^k$ also holds for every customer $i \in \mathcal{C}$. Notice that $\sum_{j\in\mathcal{M}} \alpha_{ij}$ is the number of plants that dispatch products to customer i, while $\sum_{j\in\mathcal{M}} \sum_{k\in\mathcal{K}_j} \lambda_i^k$ is the total number of vehicles that visit it.

3.2 Branch-and-Cut Algorithm

We solve formulation (1)-(35) using a branch-and-cut algorithm. For this purpose, the SECs (13) are initially dropped from the model and dynamically added when violated. To separate the SECs, we employ the minimum $s-t$ cut algorithm of the Concorde TSP solver (Applegate et al., 2018) over the support graph of the solutions found in the tree. This type of separation algorithm has been successfully applied in the literature (e.g., Adulyasak et al., 2014a; Alvarez et al., 2021; Diabat et al., 2021) and was originally proposed by Padberg and Rinaldi (1990). Whenever we find a violated SEC, we add it for $\ell = \arg\max_{i\in\mathcal{B}}\{\bar{y}_i^{kt}\}$, for every vehicle $k \in \mathcal{K}$ and time period $t \in \mathcal{T}$ in which it is violated (where $\bar{y}$ represents the value of the variable in the current solution). This separation procedure is applied at the root node and every time an integer solution is found during the search process.

3.3 Valid Inequalities and Enhanced Branching Priorities

To strengthen the formulation and enhance the performance of the branch-and-cut method, we can add several valid inequalities that have been used for PRPs and IRPs (Archetti et al., 2007, 2011; Adulyasak et al., 2014a).

$$y_i^{kt} \leq y_{j^k}^{kt} \qquad i \in \mathcal{C},\ k \in \mathcal{K},\ t \in \mathcal{T}, \tag{Ineq1}$$

$$y_i^{k^2 t} \leq y_i^{k^1 t} \qquad i \in \mathcal{M},\ k^1, k^2 \in \mathcal{K}_i\colon k^1 < k^2,\ t \in \mathcal{T}, \tag{Ineq2}$$

$$\sum_{j\in\mathcal{C}} 2^{(|\mathcal{C}|-j)} y_j^{k^2 t} \leq \sum_{j\in\mathcal{C}} 2^{(|\mathcal{C}|-j)} y_j^{k^1 t} \qquad i \in \mathcal{M},\ k^1, k^2 \in \mathcal{K}_i\colon k^1 < k^2,\ t \in \mathcal{T}. \tag{Ineq3}$$

Constraints (Ineq1) allow a vehicle to visit a customer only if it leaves its plant, where $j^k \in \mathcal{M}$ is the plant of vehicle k. (Ineq2) and (Ineq3) are symmetry breaking constraints, where (Ineq2) allow using a vehicle only if another vehicle of the same plant and with a lower index is utilized too. Notice that they apply only for vehicles of the same plant. On the other hand, (Ineq3) are lexicographic ordering constraints (see, e.g., Jans (2009) and Adulyasak et al. (2014a)), which assign a unique number to each possible subset of customers and order the routes of each plant according to the number associated with the customers they visit.

We can also use the instance information to add valid inequalities a priori. In this context, constraints (Ineq4) impose that the number of routes over the whole horizon is at least a lower bound computed with the customer demands and the capacity of the vehicles:

$$\sum_{i\in\mathcal{M}}\sum_{k\in\mathcal{K}_i}\sum_{t\in\mathcal{T}} y_i^{kt} \geq \left\lceil \frac{\sum_{i\in\mathcal{C}}\sum_{p\in\mathcal{P}} \max\{0, \sum_{t\in\mathcal{T}} d_{ip}^t - I_{ip}^0\}}{Q} \right\rceil . \tag{Ineq4}$$

Additionally, to further enhance the performance of the branch-and-cut method, we also explored the potential of defining specific branching priorities in the algorithm (instead of letting the solver decide). As such, we define a priori the variable selection order for the solver. The idea is to branch first on variables that can constrain the problem to higher degrees, potentially increasing the lower bounds in many branch-and-bound nodes. In our branch-and-cut method we first branch on the z variables, next on the y_i^{kt} variables for $i \in \mathcal{M}$, then on y_i^{kt} for $i \in \mathcal{C}$, and finally on the x variables. When the consistency requirements are considered, the associated variables (λ, α and θ) are given the highest priority. For each group of variables we let the solver select the specific variable to branch on.

Branching on the setup variables (z) before branching on the visit variables (y) allows focusing on determining in which plant and period each product must be produced, instead of focusing on the customer visits. Since, typically, there are fewer plants than customers, such priority can be very effective as the setup decisions affect the service to all the customers and appear directly in the objective function. Analogously, when we branch on y_i^{kt} first for the plants and then for the customer nodes, we focus on defining in which periods each vehicle must be used (which affects the service to a set of customers) rather than on individual customer visit decisions. Finally, the x variables are given the lowest priority since they have a relatively weak impact on the bounds.

Although the idea of defining specific branching priorities is not new in the branch-and-bound literature, it has often been neglected in the literature concerning branch-and-cut-based methods for IRPs and PRPs. As will be shown with the results of the computational experiments (Section 5.2.1), this strategy resulted in a substantially enhanced performance in our branch-and-cut algorithm, allowing to find a significantly larger number of optimal and feasible solutions within the time limit and also speeding up the algorithm.

4 Iterated Local Search for the ConMPRP

In this section, we describe the ILS-based heuristic method that we propose to solve the problem. ILS is a metaheuristic algorithm based on the idea of iteratively escaping from local optima by perturbing the current solution and then reoptimizing with a local search heuristic (Lourenço et al., 2019). In our implementation, the idea is to handle the various decisions of the problem using different components, embedding the framework within a multi-start approach. In this context, we use a construction heuristic based on the solution of a production-distribution (PD) formulation, a local search heuristic that explores several routing neighborhoods, and a perturbation procedure that can change multiple attributes of the solution simultaneously. Also, we use an improvement routine based on a mixed-integer programming (MIP) formulation, an intensification operator that employs a state-of-the-art metaheuristic, and a linear programming (LP) model to set the values of the continuous variables of the problem.

An overview of the method, which consists of two main phases, *construction* and *improvement*, is presented in Algorithm 1. In the *construction phase* (steps 3-8) we set up a solution pool S with different initial solutions. The purpose of the pool is to provide the method with diverse initial solutions, allowing it to explore different regions of the search space from the beginning. To generate those solutions, we use a construction heuristic based on a PD model. This model generates a partial solution, consisting of a production and distribution plan, which is afterwards completed by generating vehicle routes for each plant and time period. To generate different solutions with the PD model every time we solve it, we include combinatorial cuts that enforce changes w.r.t. all the previously generated solutions. Also, in this phase, we try to improve each of the solutions (step 5) by applying a solution improvement (SI) routine based on a MIP formulation that models a customer assignment problem. This phase can generate up to n^{sol} solutions if every call to the PD model and the routing method produces a feasible solution.

In the *improvement phase* (steps 9-27) the method iteratively calls the perturbation procedure (step 14) and the local search heuristic (step 15) over the solutions of the pool. Next, we apply the SI routine to the resulting solution (s^*) when it is better than the solution s from which it was generated (step 17). Finally, the best feasible solution is updated in case of improvement (steps 18-19). Notice that, over the search process, the size of the pool is constant while its solutions can change over the search process (step 17). A detailed description of all the components of the method is presented in the next sections.

4.1 Construction Heuristic

To generate the initial solutions for our method, we apply a construction heuristic that solves a PD formulation followed by a vehicle routing procedure. The PD formulation generates a production and distribution plan, which specifies the production and delivery quantities over the planning horizon. Afterward, the routing procedure solves a capacitated vehicle routing problem (CVRP) using a state-of-the-art metaheuristic.

The PD model derives from the original formulation, dropping the routing constraints and adjusting the delivery and visit variables to indicate the plant from which the delivery is made rather than the vehicle performing it. To formulate this problem, consider the following additional notation. Let q_{ipj}^{t} be a continuous variable indicating the quantity of product p delivered to customer i from plant j in period t, and y_{ij}^{t} be a binary variable equal to one if and only if a vehicle of plant j serves customer i in period t.

Algorithm 1: Iterated Local Search

```
1  begin
2      $s^{\text{best}} = \emptyset$ and $f(s^{\text{best}}) = \infty$
3      for $i = 1, \ldots, n^{sol}$ do
4          Generate initial solution $s$
5          $s = \text{SolutionImprovementMIP}(s)$
6          if $f(s) < f(s^{best})$ then $s^{\text{best}} = s$
7          Add $s$ to the solution pool $S$
8      end
9      if $S \neq \emptyset$ then
10         $iterations = 0$
11         $iterations\ without\ improvement = 0$
12         while $runtime < max\ runtime$ and $iterations < max\ iterations$ and
            $iterations\ without\ improvement < max\ iterations\ without\ improvement$ do
13             forall $s \in S$ do
14                 $s^* = \text{Perturbation}(s)$
15                 $s^* = \text{LocalSearch}(s^*)$
16                 if $f(s^*) < f(s)$ then
17                     $s = \text{SolutionImprovementMIP}(s^*)$
18                     if $f(s) < f(s^{best})$ then
19                         $s^{\text{best}} = s$
20                         $iterations\ without\ improvement = 0$
21                     end
22                 end
23                 $iterations = iterations + 1$
24                 $iterations\ without\ improvement = iterations\ without\ improvement + 1$
25             end
26         end
27     end
28 end
```

Also, let σ_{ij} be a cost estimate of servicing customer i from plant j, which we compute as

$$\sigma_{ij} = \min \left\{ 2c_{ij}, \min_{\substack{m,n \in \mathcal{N}: m \neq n, \\ m \neq i, n \neq i}} (c_{mi} + c_{in}) \right\}, \tag{36}$$

i.e., as the minimum between the round-trip cost from the given plant and the travel cost from and to its two nearest nodes. The mathematical formulation for the PD subproblem is then as follows:

$$\min \quad \sum_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{M}} \sum_{p \in \mathcal{P}} (f_{ip} z_{ip}^t + e_{ip} P_{ip}^t) + \sum_{i \in \mathcal{N}} \sum_{p \in \mathcal{P}} h_{ip} I_{ip}^t \right.$$
$$\left. + \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{M}} \sigma_{ij} y_{ij}^t \right) + \sum_{i \in \mathcal{C}} o^{\text{source}} \Omega_i + \sum_{i \in \mathcal{M}} o^{\text{product}} \Theta_i + \sum_{p \in \mathcal{P}} o^{\text{plant}} \Phi_p \tag{37}$$

$$\text{s.t.} \quad (4)-(7),(17)-(22),(25)-(28),(34)-(35),$$

$$I_{ip}^t = I_{ip}^{t-1} + P_{ip}^t - \sum_{j \in \mathcal{C}} q_{jpi}^t \qquad i \in \mathcal{M},\, p \in \mathcal{P},\, t \in \mathcal{T}, \tag{38}$$

$$I_{ip}^t = I_{ip}^{t-1} + \sum_{j \in \mathcal{M}} q_{ipj}^t - d_{ip}^t \qquad i \in \mathcal{C},\, p \in \mathcal{P},\, t \in \mathcal{T}, \tag{39}$$

$$\sum_{p \in \mathcal{P}} q_{ipj}^t \leq \min\{\pi^1 Q, L_i\} y_{ij}^t \qquad i \in \mathcal{C},\, j \in \mathcal{M},\, t \in \mathcal{T}, \tag{40}$$

$$\sum_{j \in \mathcal{C}} \sum_{p \in \mathcal{P}} q_{jpi}^t \leq \pi^2 |\mathcal{K}_i| Q \qquad i \in \mathcal{M},\, t \in \mathcal{T}, \tag{41}$$

$$\sum_{j \in \mathcal{M}} y_{ij}^t \leq 1 \qquad i \in \mathcal{C},\, t \in \mathcal{T}, \tag{42}$$

$$\sum_{t \in \mathcal{T}} y_{ij}^t \leq |\mathcal{T}| \alpha_{ij} \qquad i \in \mathcal{C},\, j \in \mathcal{M}, \tag{43}$$

$$q_{ipj}^t \geq 0 \qquad i \in \mathcal{C},\, j \in \mathcal{M},\, p \in \mathcal{P},\, t \in \mathcal{T}, \tag{44}$$

$$y_{ij}^t \in \{0,1\} \qquad i \in \mathcal{C},\, j \in \mathcal{M},\, t \in \mathcal{T}. \tag{45}$$

The objective function (37), which is similar to (1), minimizes the total cost, but using the cost estimate of the travel cost σ_{ij} instead of the actual travel cost c_{ij}. Note that no term for penalizing the violation of driver consistency appears in the objective function since the formulation does not contain any variable with a vehicle index. Constraints (38) and (39) define the inventory balance at the plants and customers, respectively. Constraints (40) allow positive delivery quantities from a plant only if the customer is assigned to the plant, while (41) impose the available transportation capacity according to the fleet of each plant. The parameters $\pi^1 \in (0,1]$ and $\pi^2 \in (0,1]$ are introduced to reduce the individual and total delivery quantities, respectively. They are used to prevent solutions with very large delivery quantities, favoring the feasibility of the solution in the subsequent routing step. Constraints (42) enforce that each customer can be assigned to at most one plant per time period. Constraints (43) activate the variable α if the corresponding customer is served at least once by the given plant over the horizon. Finally, constraints (44)-(45) define the domain of the new decision variables.

Once a solution has been generated using the PD model, we can attempt to obtain a feasible solution for the ConMPRP by finding vehicle routes for the defined delivery quantities. As such, based on the solution $\bar{q}$ and $\bar{y}$ of the PD model, the resulting problem reduces to a CVRP for each plant and period. To solve this problem, we use the Hybrid Genetic Search (HGS) with advanced diversity control of Vidal et al. (2012) and Vidal (2020). This method is available as open-source code and stands as a leading metaheuristic with regards to solution quality, speed, and conceptual simplicity.

Since we use the PD model to construct solutions for the pool S, we then need to make sure we generate different PD plans every time we solve it. For this purpose, we include combinatorial cuts that enforce changes in the setups and plant-customer assignments w.r.t. all the previously generated solutions. Let s be the index of the last solution found in the construction phase, and let $\bar{z}$ and $\bar{y}$ indicate the solution values of z and y in the solution s, respectively. We then add the following inequalities to the PD model:

$$\sum_{i \in \mathcal{M}} \sum_{p \in \mathcal{P}} \sum_{\substack{t \in \mathcal{T}: \\ \bar{z}_{ip}^{t(s)}=1}} (1 - z_{ip}^t) + \sum_{i \in \mathcal{M}} \sum_{p \in \mathcal{P}} \sum_{\substack{t \in \mathcal{T}: \\ \bar{z}_{ip}^{t(s)}=0}} z_{ip}^t \geq 1, \tag{46}$$

$$\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{M}} \sum_{\substack{t \in \mathcal{T}: \\ \bar{y}_{ij}^{t(s)}=1}} (1 - y_{ij}^t) + \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{M}} \sum_{\substack{t \in \mathcal{T}: \\ \bar{y}_{ij}^{t(s)}=0}} y_{ij}^t \geq 1. \tag{47}$$

The inequality (46) forces at least one setup variable to be different w.r.t. solution s, while (47) does the same for the assignment variables. These inequalities are accumulated over the construction phase and we add them to provide the heuristic with diverse initial solutions, allowing it to explore different regions of the search space from the beginning.

To avoid using an excessive amount of time in the construction phase of the method, we stop the solver used to optimize the PD model when either: a solution within 1% of optimality is found; the number of nodes explored during the branch-and-bound process exceeds 5,000; or a maximum CPU time of 15 seconds is reached. Similar to Adulyasak et al. (2014b), this configuration does not significantly affect the performance of the complete method. The values of π^1 and π^2 in the PD model were set to 0.6 and 0.9, respectively. The value of π^1 is reduced to 0.5 if the routing phase fails to find a feasible solution for the problem. This combination allowed us to find feasible solutions for all the problem instances considered in our computational experiments. Every time we call the HGS algorithm for a given plant i and period t, we impose a maximum CPU time of 0.5 seconds if only one vehicle is required (i.e., if $\sum_{j \in \mathcal{C}} \sum_{p \in \mathcal{P}} q_{jpi}^t < Q$), otherwise we allow up to 1 second. The maximum number of initial solutions in the pool n^{sol} is set to 5 to avoid excessive computing times in the construction phase. Also, it is worth mentioning that this construction method considers explicitly the source, product, and plant consistency since their constraints are maintained in the PD model. On the other hand, driver consistency is not explicitly considered at this point. In our implementation, we explored the usage of a MIP formulation modeling a route reassignment problem that optimizes driver consistency, similar to those presented by Smilowitz et al. (2013) and Diabat et al. (2021). However, this additional step did not improve the algorithm performance and, therefore, we did not consider it in the final version of the heuristic.

4.2 Solution Improvement MIP

This section describes the SI routine that we use as a component within our heuristic. It consists of a mathematical formulation that models a ConMPRP in which we decide on the insertion and removal of customers into the routes of a solution given as input. The SI formulation performs, therefore, a local search over the input solution, helping to identify the periods in which we should visit each customer. This feature is particularly useful when the travel costs are relatively high compared to the rest of the costs involved in the problem. This type of component was first introduced by Archetti et al. (2012) and later on used in several other solution methods for IRPs and PRPs.

Similar to the PD model, the SI formulation is derived from the original formulation using additional variables to indicate the insertion and removal operations. Consider the following notation. Let γ_i^{kt} be a binary variable equal to 1 if and only if customer i is inserted into the route of vehicle k in period t, and Γ_i^{kt} be the cost associated with that insertion. We set this cost as the cheapest insertion in the route. Also, let δ_i^{kt} be a binary variable equal to 1 if and only if customer i is removed from the route of vehicle k in period t, and Δ_i^{kt} be the associated removal cost. This cost is computed as $c_{hi} + c_{ij} - c_{hj}$, where h and j are the predecessor and successor of the customer in the route, respectively. Finally, for the input solution, let $\bar{y}_i^{kt}$ be the value of y_i^{kt} and $\bar{x}^{kt}$ be a binary parameter indicating whether or not vehicle k was used in period t. The SI formulation can be stated as follows:

$$\begin{aligned}\min \quad & \sum_{t\in\mathcal{T}}\Bigg(\sum_{i\in\mathcal{M}}\sum_{p\in\mathcal{P}}\Big(f_{ip}z_{ip}^t + e_{ip}P_{ip}^t\Big) + \sum_{i\in\mathcal{N}}\sum_{p\in\mathcal{P}} h_{ip}I_{ip}^t + \sum_{i\in\mathcal{C}}\sum_{k\in\mathcal{K}}\Gamma_i^{kt}\gamma_i^{kt} - \sum_{i\in\mathcal{C}}\sum_{k\in\mathcal{K}}\Delta_i^{kt}\delta_i^{kt}\Bigg) \\ & + \sum_{i\in\mathcal{C}} o^{\text{driver}}\Lambda_i + \sum_{i\in\mathcal{C}} o^{\text{source}}\Omega_i + \sum_{i\in\mathcal{M}} o^{\text{product}}\Theta_i + \sum_{p\in\mathcal{P}} o^{\text{plant}}\Phi_p \qquad (48)\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \quad & (2)-(7),(15),(17)-(28),(33)-(35), && \\
& \sum_{p\in\mathcal{P}} q_{ip}^{kt} \le \min\{Q, L_i\}(\bar{y}_i^{kt} - \delta_i^{kt} + \gamma_i^{kt}) && i\in\mathcal{C},\, k\in\mathcal{K},\, t\in\mathcal{T}, \qquad (49)\\
& \sum_{i\in\mathcal{C}}\sum_{p\in\mathcal{P}} q_{ip}^{kt} \le Q\bar{x}^{kt} && k\in\mathcal{K},\, t\in\mathcal{T}, \qquad (50)\\
& \sum_{k\in\mathcal{K}}(\bar{y}_i^{kt} - \delta_i^{kt} + \gamma_i^{kt}) \le 1 && i\in\mathcal{C},\, t\in\mathcal{T}, \qquad (51)\\
& \delta_i^{kt} \le \bar{y}_i^{kt} && i\in\mathcal{C},\, k\in\mathcal{K},\, t\in\mathcal{T}, \qquad (52)\\
& \gamma_i^{kt} \le 1 - \bar{y}_i^{kt} && i\in\mathcal{C},\, k\in\mathcal{K},\, t\in\mathcal{T}, \qquad (53)\\
& \sum_{i\in\mathcal{C}}(\delta_i^{kt} + \gamma_i^{kt}) \le \beta\bar{x}^{kt} && k\in\mathcal{K},\, t\in\mathcal{T}, \qquad (54)\\
& \sum_{t\in\mathcal{T}}(\bar{y}_i^{kt} - \delta_i^{kt} + \gamma_i^{kt}) \le |\mathcal{T}|\lambda_i^k && i\in\mathcal{C},\, k\in\mathcal{K}, \qquad (55)\\
& \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_j}(\bar{y}_i^{kt} - \delta_i^{kt} + \gamma_i^{kt}) \le |\mathcal{T}|\alpha_{ij} && i\in\mathcal{C},\, j\in\mathcal{M}, \qquad (56)\\
& \gamma_i^{kt}, \delta_i^{kt} \in \{0,1\} && i\in\mathcal{C},\, k\in\mathcal{K},\, t\in\mathcal{T}. \qquad (57)
\end{aligned}$$

The objective function (48) consists of minimizing the total cost, considering now the insertion and removal costs. Note that the violation of the driver consistency is included in this objective function since the formulation considers the explicit assignment of customers to vehicles (drivers). Constraints (49) allow positive delivery quantities only if either: a visit occurs already and the customer is not removed from the route ($\bar{y}_i^{kt} = 1$ and $\delta_i^{kt} = 0$) or the customer is inserted in the route of the vehicle ($\bar{y}_i^{kt} = 0$ and $\gamma_i^{kt} = 1$). The transportation capacity of the vehicles is ensured by constraints (50) while constraints (51) allow at most one visit to every customer in each time period. Constraints (52) permit removing a customer from a route only if it is visited by the route. Analogously, constraints (53) enforce that if a customer is already visited by a route then it cannot be reinserted into the same route. Constraints (54) forbid the insertion and removal of customers into and from routes of vehicles that are not used in the given time period. It is worth mentioning that if more than one customer is inserted or removed from a route, then Δ and Γ provide only an approximation of the actual insertion and removal costs, respectively. For this reason, the right-hand side of (54) contains the parameter β which limits the number of changes that can be performed to every single route of the solution. Finally, constraints (55) and (56) activate the λ and α variables, respectively, depending on the insertion and removal decisions, whose domain is defined in constraints (57).

To stop the optimization solver in the SI routine, we used the same criteria that we set for the PD model, except for the maximum CPU time which we set to 20 seconds. The value of β was set to 1, i.e., we allow at most one change per route in the solution.

4.3 Local Search Heuristic

In the improvement phase of our method, every time the local search step is invoked (line 15 of Algorithm 1), we use a local search heuristic that focuses on exploring the neighborhood of the routing part of the solutions. Specifically, we make use of a variable neighborhood descent heuristic (Mladenović and Hansen, 1997) with a random ordering of the operators. This heuristic applies a set of local search operators to improve the incumbent solution progressively. The heuristic is always initialized with a set of predefined local search operators. In each iteration of the heuristic, while the set is not empty, an operator is randomly selected and applied to the incumbent solution. If the operator fails to improve the solution, the heuristic removes it from the set. Conversely, if the operator improves the solution, the set is reestablished to its initial form, containing all the operators. The local search heuristic stops when none of the operators can improve the solution further, i.e., when the set of operators becomes empty.

In our implementation of the local search heuristic, we use the following classical vehicle routing operators: 2-opt; Or-opt-k, $k \in \{1, 2, 3\}$; Shift(k), $k \in \{1, 2, 3\}$; and Swap(k_1, k_2), $k_1, k_2 \in \{1, 2\}, k_1 \geq k_2$. The first two operate over a single route at a time. In 2-opt, two nonadjacent arcs of the route are deleted and another two are added such that a new route is generated. Or-opt-k removes k adjacent customers and inserts them in another position of the same route. On the other hand, Shift(k) and Swap(k_1, k_2) are operators that change two routes at a time. In our implementation, we apply these operators over the routes of vehicles belonging to the same plant for every time period. Shift(k) tries to transfer k adjacent customers from their route to the routes of other vehicles (departing from the same plant in that period) while Swap(k_1, k_2) tries to exchange k_1 with k_2 adjacent customers from two different routes. In all these operators, we explore the search space using the first improvement strategy and do not allow infeasible solutions in the search process.

4.4 Perturbation Procedure

Following the concept of using separate components to handle different types of decisions in our method, we developed a perturbation procedure that operates mostly over the production setups and visits of the solutions. The idea is to simultaneously change various solution attributes given the interrelation of the many decisions of the problem. For this purpose, our perturbation operates similarly to the local search heuristic of Section 4.3. Specifically, given an initial set of operators, the procedure randomly chooses one of them in each iteration and applies it to the solution. If the operator fails to change the solution (e.g., when all the possible moves of the operator are either infeasible or do not alter the solution structure), the procedure removes it from the set, and the perturbation continues with the remaining operators. On the other hand, if the operator manages to change the solution, the procedure restores the set to its initial configuration. The procedure stops when the set is empty, i.e., when all the operators fail at changing the solution or when the number of successfully applied operators reaches φ. This parameter defines the strength of the perturbation by limiting the number of changes that the perturbation procedure carries out on the input solution. In our experiments, the value of φ was set to $1 + \lceil 0.03|\mathcal{C}| \rceil$, where $|\mathcal{C}|$ is the number of customers of the problem instance.

Our perturbation procedure utilizes several operators that alter different parts of the solution and to different degrees. In total, we apply 12 operators, divided into four categories, whose details are described next. In the description of the operators, $\bar{z}$ and $\bar{y}$ represent the value of the corresponding variables in the input solution.

Removal operators: these operators are responsible for deleting parts of the input solution. In particular, we developed operators that remove active production setups, visits to the customers, or entire vehicle routes. The specific operators are presented next.

- Setup removal: the operator deactivates one of the currently active production setup operations by randomly selecting a tuple (i, p, t) such that $\bar{z}^t_{ip} = 1$ and then setting $z^t_{ip} = 0$;
- Visit removal: this operator removes a visit to one of the customers of the solution by choosing a random customer-period combination (i, t) such that customer i is visited in period t (i.e., $\sum_{k\in\mathcal{K}} \bar{y}^{kt}_i = 1$) and removing the customer from its corresponding route;
- Route removal: the operator randomly chooses and deletes a complete route from the solution. This operator can, therefore, remove multiple visits simultaneously.

Insertion operators: the operators of this category activate new parts of the solution, instead of removing parts of it, every time they are called. In total, we explored three different insertion operators.

- Setup insertion: this operator turns on an inactive production setup by randomly choosing a tuple (i, p, t) such that $\bar{z}^t_{ip} = 0$ and then setting the corresponding setup variable z^t_{ip} to 1;
- Visit insertion: the operator inserts a visit to a customer in one of the periods in which it is not visited, choosing the customer and the period at random. If there are routes in the chosen period, the operator inserts the visit into the cheapest insertion position of a randomly selected route. Otherwise, if there are no routes in the period, a round-trip route is inserted from a random plant and with a randomly selected vehicle (from the fleet of the plant);
- Route insertion: the operator chooses a random route from the solution and inserts a copy of the route in a different period, selecting the new time period, plant and vehicle at random. The operator

removes from the newly inserted route the customers that are already visited by other routes of the period.

Transfer operators: the third category of operators transfers active operations between parts of the solution. These operators share features found in the removal and insertion categories since they delete active operations and activate them in a different part of the solution. We developed three transfer operators, which are described next.

- Setup transfer: this operator selects a random plant-product combination (i, p) and performs two changes: deactivate one of the currently active production setup operations and turn on an inactive production setup, choosing the two periods at random. This is therefore a setup transfer between different periods for a given plant and product;
- Visit transfer: the operator selects a customer at random and transfers one of the visits from its current period to another one, choosing both periods at random. Similar to the visit insertion operator, if the destination period has vehicle routes, the operator inserts the visit into the cheapest insertion position of a randomly selected route. Otherwise, if there are no routes in the period, a round-trip route is inserted from a random plant and with a randomly selected vehicle out of the plant fleet;
- Route transfer: the operator selects a random route from the solution and transfers it to a different time period. The new period, plant and vehicle of the route are chosen at random. The operator deletes from the new route the customers that are already visited by the other routes of the period.

Change operators: these operators either activate or deactivate parts of the solution, depending on the status of the part chosen. They can, therefore, behave as insertion or removal operators. The description of the operators of this category is as follows.

- Setup change: the operator chooses a random tuple (i, p, t) and changes the associated setup status, i.e., z_{ip}^{t} is set to 0 if $\bar{z}_{ip}^{t} = 1$, otherwise it is set to 1;
- Visit change: this operator selects a random customer-period combination (i, t) and inserts a visit to customer i if it is not visited in period t; otherwise, if the customer is visited in that period, then the visit is removed;
- Route change: this operator works by changing the plant and vehicle of a route chosen randomly. The period and the customers of the route are maintained unaltered.

Observe that the changes performed by the operators may be infeasible or result in inactive actions in the solution. Infeasibilities may appear due to, for example, stockouts at a plant because of a deactivated setup or at customer locations due to the elimination of visits. On the other hand, inactive operations refer to idle actions in the solutions. For example, a new setup that has a zero production level or a new visit with no associated delivery quantity. We deal with these issues depending on the category of the operators. The removal operators reject infeasible moves until a successful change is applied or until all the possible removals are tested (and rejected). The insertion operators (whose operations are always feasible) discard inactive moves until one active change is achieved (i.e., an actually used visit or production setup) or until the operator tests all the possible insertions. Finally, in the transfer and change operators, infeasible and inactive moves are rejected until the operator performs a successful transfer/change or until all the possible operations are tested. We verify the feasibility and activeness of the moves by solving an LP formulation over the resulting solution. This LP determines the optimal solution (if any) of the continuous variables for the given input ($\bar{y}$, $\bar{z}$) and is presented in the next section.

4.5 LP to Determine the Continuous Quantities

This section presents the LP we use to determine, given the values of the setup and visit variables ($\bar{z}$ and $\bar{y}$, respectively), the optimal values of the continuous variables (production and delivery quantities as well as inventory levels) that minimize the total cost. This formulation can also be used to determine the feasibility of the given setup and visit input solution. The formulation, which is derived from the original model (1)-(35), is as follows:

$$\min \quad \sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}}\left(\sum_{i\in\mathcal{M}} e_{ip}P_{ip}^{t} + \sum_{i\in\mathcal{N}} h_{ip}I_{ip}^{t}\right) \tag{58}$$

$$\text{s.t.} \quad (2)-(5),(7),(21)-(23),$$

$$P_{ip}^{t} \leq \min\left\{C_i, \sum_{j\in\mathcal{C}}\sum_{\tau=t}^{|\mathcal{T}|} d_{jp}^{\tau}\right\}\bar{z}_{ip}^{t} \qquad i\in\mathcal{M},\ p\in\mathcal{P},\ t\in\mathcal{T}, \tag{59}$$

$$\sum_{p\in\mathcal{P}} q_{ip}^{kt} \leq \min\{Q, L_i\}\bar{y}_i^{kt} \qquad i\in\mathcal{C},\ k\in\mathcal{K},\ t\in\mathcal{T}, \tag{60}$$

$$\sum_{j\in\mathcal{C}}\sum_{p\in\mathcal{P}} q_{jp}^{kt} \leq Q\bar{y}_i^{kt} \qquad i\in\mathcal{M},\ k\in\mathcal{K}_i,\ t\in\mathcal{T}. \tag{61}$$

In this LP, the objective function (58) minimizes the total production and inventory cost while constraints (59)-(61) link the production and delivery variables with the input values $\bar{z}$ and $\bar{y}$ as their counterparts in the original model. This LP can be solved straightforwardly with an optimization software. Also, we do not need to build it from scratch every time we are going to solve it. We rather build it only once and the right-hand sides of the constraints are updated when necessary.

The solution found after solving this LP, if solved to optimality, will always have a cost that is better than or equal to the original cost of the input solution. We can, however, try to further improve the final solution by deleting unnecessary setups and visits, as follows. Given a solution ($\bar{P}$, $\bar{q}$) of the LP, we verify if there is any $\bar{P}_{ip}^{t} = 0$ such that $\bar{z}_{ip}^{t} = 1$ for a given $i\in\mathcal{M}$, $p\in\mathcal{P}$, $t\in\mathcal{T}$. In such a case, the setup variable is set to 0 and the total setup cost is updated. Analogously, if there is any $\sum_{p\in\mathcal{P}}\bar{q}_{ip}^{kt} = 0$ such that $\bar{y}_i^{kt} = 1$ for a given $i\in\mathcal{C}$, $k\in\mathcal{K}$, $t\in\mathcal{T}$, the customer is removed from its route in the solution and the routing cost is updated accordingly. The consistency penalties are also updated every time we solve the LP if the values of $\bar{z}$ and $\bar{y}$ change.

4.6 Additional Details

This section describes some additional details of the method and the criteria that we used to stop the method. In this context, we used the HGS metaheuristic as an intensification phase in our method. Every time the incumbent solution is updated (steps 6 and 19 of Algorithm 1), we call the HGS routine to try to further improve the routes of the solution. This routine is called for every plant-period combination, solving the corresponding CVRP and replacing the routes if a better solution is found. However, this routine is not applied when driver consistency is being optimized since the new routes might change considerably, possibly worsening the customer-driver assignments of the solution. The stopping criteria of the heuristic were set to 500 seconds, 8,000 iterations and 4,000 iterations without improvement.

5 Computational Experiments

This section describes the experiments carried out with the different methods and the results of these experiments. We divide the results into two parts. The first one (Section 5.2) shows and assesses the performance of the solution methods. The second part (Section 5.3) focuses on analyzing the impact of taking the consistency requirements into account and the trade-offs arising from their inclusion.

The branch-and-cut algorithm and the heuristic algorithm were coded in C++, using CPLEX 20.1 as the MIP and LP solver. All the parameters of the solver were kept at their default values, except for the relative MIP optimality gap tolerance which was set to 10^{-8}. For the branch-and-cut algorithm, we set a time limit of two hours. All computations were executed on the Cedar cluster of the Compute Canada cluster, which is equipped with 2.1 GHz Intel E5-2683 v4 Broadwell processors. We used a single thread for all the experiments and a RAM limit of 8GB.

5.1 Problem Instances

For our experiments, we generated a test set based on benchmark instances from the PRP literature. We did so because, as far as we know, there is no benchmark test set for the MPRP. Specifically, we used the benchmark set proposed by Adulyasak et al. (2014a) and the set A3 of the benchmark set of Archetti et al. (2011). The first benchmark set contains 168 instances with up to 50 customers, nine periods and four vehicles. The set A3 of Archetti et al. (2011) consists of 480 problem instances with 100 customers and six time periods. All these instances are further divided into four classes, exhibiting different characteristics w.r.t. the relative importance of the cost components. The first class represents the base case in which the costs are balanced while the second class represents a case with higher production and setup costs. The third class depicts a case with higher transportation costs and the fourth class presents the case with no inventory costs at the customer facilities.

We used these instances as a basis to generate our MPRP instances as follows. For each of the 168 instances of Adulyasak et al. (2014a), we generated an instance by setting the number of plants $|\mathcal{M}|$ to 2. We also generated instances with 4 plants from the instances with 4 vehicles. For each of these cases, we generated four instances with 2, 3, 4, and 5 products ($|\mathcal{P}|$). Then, for each of the specified combinations we created instances by generating the missing parameters (e.g., coordinates, costs and demands). We used the same base distribution as in the original instances, but with two different levels of variability: low (0.2) and high (0.5), w.r.t. the mean value to generate the data for the different products and plants. This led to a subtotal of 1,632 instances. Besides, we generated additional instances by taking those from the original set with a maximum number of vehicles equal to three (48 instances, with up to 25 customers) and generated a new one with four vehicles, four plants, the four values for the number of products and the two levels of variability. This process results in 384 additional instances. Finally, we generated large-sized instances using a similar procedure for the set A3 of Archetti et al. (2011). Using the same combinations for the number of plants, products and data variability, we took a random instance of each of the four classes and generated a new one with three and four vehicles. Additional instances were then generated by taking only the first 75 customers, instead of the original 100. This procedure led to a subtotal of 192 large-sized instances, and a total of 2,208 problem instances for the MPRP.

5.2 Computational Performance

In this section, we show the results of the experiments carried out to assess the impact of the enhancements of the branch-and-cut algorithm (Section 5.2.1) and the heuristic method (Section 5.2.2). The experiments in this section do not include any of the consistency requirements.

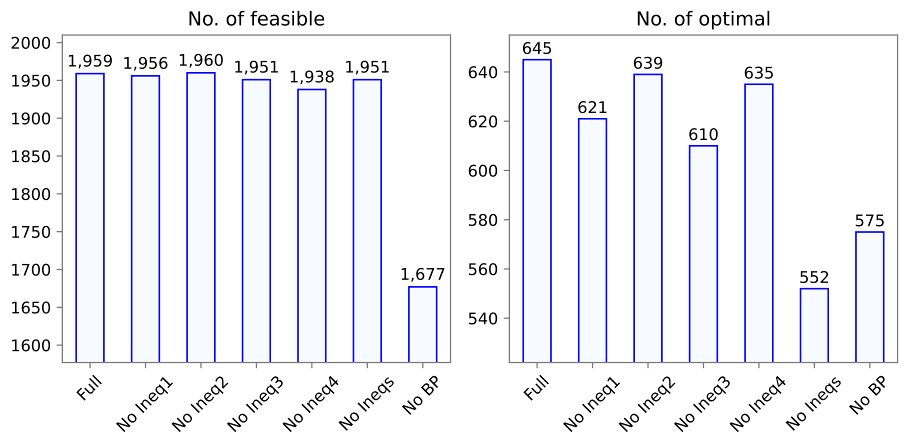
5.2.1 Effect of the Valid Inequalities and Branching Priorities

To show the impact of the valid inequalities and branching priorities presented in Section 3.3, in this section we show a comparison of the performance of the branch-and-cut algorithm for different configurations. In this context, we compare the method using all the components (valid inequalities plus branching priorities) simultaneously (Full) with configurations obtained by removing the valid inequalities from the method individually (No Ineq1, No Ineq2, No Ineq3, No Ineq4) or altogether (No Ineqs). We also include the case in which the branching priorities are not considered while keeping all inequalities (No BP). In summary, we compare the full version of the algorithm with six other configurations for different performance measures such as the number of feasible and optimal solutions, as well as the CPU time of the algorithm.

Figure 1 shows the number of instances (out of 2,208) for which the algorithm could find a feasible or an optimal solution, with each configuration, within the two-hour time limit. The figure shows that disregarding the valid inequalities generally leads to a substantial reduction in the number of instances solved to optimality by the method. In particular, when they are all removed simultaneously (No Ineqs), the number goes from 645 to 552 instances (nearly 15% fewer optimal solutions). The figure also shows the benefits of the predefined branching priorities w.r.t. the case in which we let the solver decide (No BP). In this case, a total of 70 fewer instances are solved to optimality within the time limit, and the algorithm finds a feasible solution for 282 fewer instances within the time limit. Notice that disregarding Ineq2 leads to one instance more for which the algorithm can find a feasible solution. However, this configuration also leads to six fewer instances solved to optimality. This is not desirable for us since our analysis and managerial insights of Section 5.3 are based on the results of optimal solutions only.

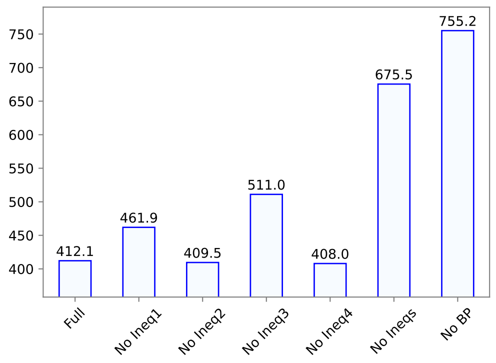
Figure 2 shows, considering only those instances solved to optimality for all the configurations, the average CPU time (in seconds) of the branch-and-cut algorithm. The full version of the method uses

Figure 1: Number of feasible and optimal solutions for all the configurations



only 61% and 55% of the time required by the settings that do not use the valid inequalities and the branching priorities, respectively. These analyses allow us, therefore, to observe how the valid inequalities and branching priorities enhanced the performance of the branch-and-cut algorithm with regard to different metrics.

Figure 2: CPU time over all the instances with optimal solutions for all the configurations



5.2.2 Performance of the Heuristic

To test the performance of the heuristic, in a first set of experiments, we compared its results w.r.t. the results of the full version of the branch-and-cut algorithm. The comparison is shown in Table 1, whose results are grouped by the number of plants and products of the instances. The table headings indicate, for each group, the number of instances (#), the number of feasible (#F) and optimal (#O) solutions found by the branch-and-cut algorithm, as well as the average relative optimality gap (Opt gap) and the CPU time (CPU time), in seconds. For the heuristic, we display the number of instances for which the heuristic found a feasible (#F) solution and a better (#B) solution w.r.t. those of the branch-and-cut algorithm. Finally, we show the average relative difference (Rel diff) of the objective value of the heuristic solutions w.r.t. the branch-and-cut solutions and the CPU time (CPU time) of the heuristic method, in seconds. The relative differences are computed as $100 \times (z^h - z^b)/z^b$, where z^h is the objective value of the solution of the heuristic and z^b is the objective value of the branch-and-cut solution. The relative differences are computed only over those instances for which the branch-and-cut algorithm found a feasible solution within the time limit (#F). Notice also that a negative relative difference value indicates that the solution of the heuristic is better than the solution of the exact method. We consider that the heuristic finds a better solution when $z^h < z^b$ and also when the branch-and-cut algorithm fails to find a feasible one. The last row (Total) displays the average values over all the instances for the gaps and CPU time values and the sum of the column for the counter columns (#, #F, #O and #B).

In the table, is it possible to notice the advantages of the heuristic w.r.t. the branch-and-cut method. The first one is that the heuristic finds feasible solutions for all the instances while the exact method fails to do so. Moreover, nearly half of these solutions are better than the solutions of the branch-and-cut algorithm. For all the groups, the heuristic finds solutions that are on average better than those of the branch-and-cut method (negative values of 'Rel diff'). On average, the heuristic finds solutions that are 0.82% better than the ones of the branch-and-cut algorithm while only spending a fraction of the CPU time, about 8%. Besides, this analysis also allows us to verify that the branch-and-cut performance worsens when the number of products increases, especially for the instances with four plants, while the heuristic performance is consistent over all the groups for the different numbers of plants.

Table 1: Comparison between the heuristic method and the branch-and-cut algorithm

			Branch-and-cut				Heuristic			
No. of plants	No. of products	#	#F	#O	Opt gap (%)	CPU time	#F	#B	Rel diff (%)	CPU time
2	2	368	338	153	4.43	4,361.74	368	139	-0.61	297.59
	3	368	335	143	4.86	4,542.36	368	152	-0.54	346.73
	4	368	334	125	5.16	4,832.85	368	163	-0.52	375.70
	5	368	331	118	5.17	4,990.46	368	170	-0.51	397.67
4	2	184	165	40	9.76	5,889.49	184	98	-1.22	413.83
	3	184	156	30	11.91	6,167.43	184	111	-1.50	450.25
	4	184	148	20	13.18	6,376.43	184	126	-1.42	470.60
	5	184	152	16	14.57	6,577.27	184	127	-1.57	474.78
Total		2,208	1,959	645	7.25	5,175.76	2,208	1,086	-0.82	387.07

We also measured the heuristic performance only for the instances solved to optimality by the branch-and-cut method. Table 2 shows the results of this analysis. The idea is to verify if the heuristic can find solutions close to the optimal ones. The table headings indicate, for each group, the number of instances for which the exact method found and proved the optimality within the time limit (#O) and the average CPU time (CPU time), in seconds. For the heuristic, we present the optimality gap of its solutions (Opt gap) and the CPU time (CPU time), also in seconds. We compute the optimality gaps as $100 \times (z^h - z^b)/z^h$. The results show that our heuristic can find near-optimal solutions over all the groups, with average optimality gaps under 1% for all but one group, requiring only about 18% of the time spent by the exact method. This result further highlights the benefits of the heuristic algorithm for the MPRP.

To further test the performance of the heuristic, we also used it to solve the standard PRP. We compared our results with the state-of-the-art heuristic methods from the literature. Specifically, we

Table 2: Comparison between the heuristic method and the branch-and-cut algorithm over the instances solved to optimality

		Branch-and-cut		Heuristic	
No. of plants	No. of products	#O	CPU time	Opt gap (%)	CPU time
2	2	153	929.83	0.61	132.19
	3	143	973.87	0.51	169.53
	4	125	874.92	0.60	190.10
	5	118	1,002.00	0.61	225.01
4	2	40	1,794.06	1.27	214.64
	3	30	1,830.54	0.83	283.92
	4	20	1,105.39	0.66	294.83
	5	16	1,283.84	0.74	300.09
Total		645	1,051.87	0.64	190.05

compare with the following methods: the optimization-based ALNS of Adulyasak et al. (2014b) (ALNS), the multi-start iterative methods of Absi et al. (2015) (IM-MS and IM-VRP), the multi-phase heuristic of Solyalı and Süral (2017) (MP), the VNS heuristic of Qiu et al. (2018) (VNS), the three-level heuristic of Li et al. (2019) (TLH), the decomposition matheuristic of Chitsaz et al. (2019) (DM), the matheuristic algorithm of Avci and Yildiz (2019) (MA) and the infeasible space exploration matheuristic of Manousakis et al. (2021) (IMA). For this experiment, we used the benchmark set of Archetti et al. (2011), which has three subsets (A1, A2 and A3), containing instances with 14, 50 and 100 customers, respectively. Each subset has 480 problem instances and can be further divided into four classes presenting different characteristics. For this experiment, the stopping criteria of the heuristic were extended to 600 seconds, 10,000 iterations or 5,000 iterations without improvement.

Table 3 presents, for each instance set, class and method, the relative differences of the methods w.r.t. the best solution found by all of them. The results show that our heuristic is highly competitive in terms of solution quality compared to these methods, in particular considering that many of these methods were specifically designed to tackle the standard PRP. The heuristic finds solutions with relative differences of less than or equal to 1% on average for all but two groups (class 3, set A2 and A3). A total of 27 better solutions were found by our heuristic, in comparison with all the other methods simultaneously. The average CPU times for set A1, A2 and A3 is 34, 584 and 600 seconds, respectively, indicating that our heuristic can also solve the standard PRP within reasonable times.

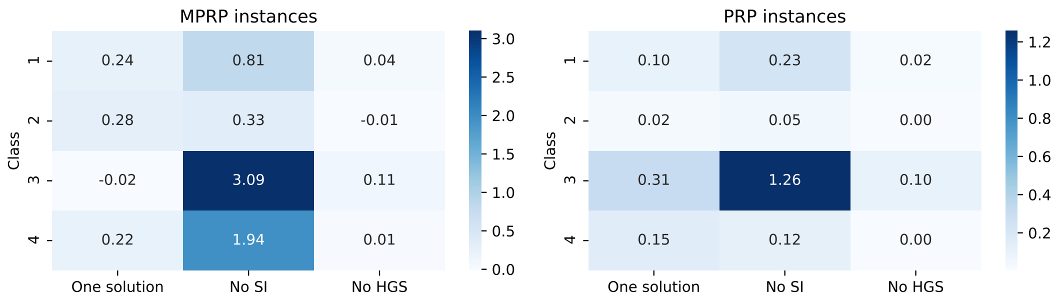
Finally, we evaluated the contribution of some of the enhancements of the heuristic to its overall performance. For this purpose, we compared three different configurations of the heuristic (where some of the enhancements were not used) w.r.t its full version (as described in Section 4). The three configurations that we analyzed consisted of a version using only a single initial solution in the pool ('One solution'), one not using the SI routine ('No SI'), and one not using the HGS as an intensification phase ('no HGS'). Figure 3 shows the average relative differences of the solutions found with these configurations w.r.t. the full version, separated by instance set (MPRP and standard PRP) and for the different instance classes. This analysis shows that the SI phase contributes significantly to the performance of the heuristic since the version of the method without it ('No SI') finds solutions that can be up to 3.09% worse (for the MPRP test set, class 3). Also, the usage of multiple initial solutions has a significant impact on the PRP

Table 3: Comparison between the different heuristic methods for the instances of Archetti et al. (2011)

			Adulyasak	Absi	Solyali	Qiu	Li	Chitsaz	Avci	Manousakis	This paper
Set	Class	#	ALNS	IM	MP	VNS	TLH	DM	MA	IMA	ILS-ACJ
A1	1	120	1.70	0.09	0.03	0.28	0.15	0.24	0.01	0.05	0.15
	2	120	0.37	0.02	0.01	0.05	0.03	0.04	0.01	0.01	0.03
	3	120	8.42	0.57	0.18	1.52	0.75	1.49	0.04	0.38	1.00
	4	120	0.93	0.02	0.03	0.56	0.08	0.11	0.01	0.00	0.06
	Total	480	2.85	0.18	0.06	0.60	0.25	0.47	0.02	0.11	0.31
A2	1	120	1.35	0.38	0.27	0.26	0.21	0.16	0.18	0.02	0.32
	2	120	0.22	0.10	0.06	0.08	0.05	0.06	0.06	0.00	0.05
	3	120	4.39	1.83	0.92	0.87	0.82	0.63	0.78	0.06	1.40
	4	120	0.35	0.26	0.22	0.14	0.19	0.13	0.17	0.07	0.16
	Total	480	1.58	0.64	0.37	0.34	0.32	0.25	0.30	0.04	0.48
A3	1	120	1.05	0.27	0.12	0.23	0.08	0.23	0.07	0.19	0.50
	2	120	0.16	0.06	0.04	0.08	0.03	0.04	0.05	0.00	0.09
	3	120	3.97	1.54	0.46	0.62	0.47	1.38	0.55	0.37	1.89
	4	120	0.35	0.27	0.08	0.10	0.06	0.08	0.11	0.14	0.27
	Total	480	1.38	0.54	0.17	0.26	0.16	0.43	0.19	0.18	0.69

instances, in particular for class 3. Besides, although it seems that the HGS intensification phase does not contribute significantly to the heuristic performance (in which case we could ignore it in the method), this component is particularly important to refine the routing part of the solutions and does not lead to longer CPU times.

Figure 3: Relative differences of different configurations of the heuristic w.r.t. the base case



5.3 Analysis of the Consistency Features

To analyze the consistency features of the solutions, we carried out a series of experiments to observe the effect of using different values for the corresponding violation penalties. The idea is to verify the change in the cost and solution structure when varying the unit cost of exceeding the targets of the consistency features, i.e., when altering the relative importance of the consistency requirements. For this purpose, we need to define the values of the penalties and metrics to measure the corresponding violations (if any). First, we set the values of the unit penalty costs as:

$$o^{\text{driver}} = o^{\text{source}} = \rho \times \max_{i \in \mathcal{M}, j \in \mathcal{C}} \{2c_{ij}\}, \tag{62}$$

$$o^{\text{product}} = o^{\text{plant}} = \omega \times \max_{i \in \mathcal{M}, p \in \mathcal{P}} \{f_{ip}\}. \tag{63}$$

In Eq. (62), we set the driver and source consistency unit penalties as the maximum round-trip cost of the instance, multiplied by a weight term ρ. Similarly, in Eq. (63), we set the product and plant consistency unit penalties as the instance maximum setup cost, also multiplied by a weight ω. Notice that the weights ρ and ω define the relative importance of each consistency feature w.r.t. the other cost components and allow us to perform sensitivity analyses with regards to different values for these weights.

In addition to defining the values used to penalize the violations, it is necessary to define metrics for the consistency features of the solutions w.r.t. their respective targets. As such, we have used and extended the metric proposed by Diabat et al. (2021) to measure driver consistency for the IRP. To introduce these metrics, consider the following notation. For a given solution, let $\bar{\lambda}_i^k$ be the value of the variables λ_i^k (whether or not vehicle k visits customer i over the planning horizon). The driver consistency metric κ^{driver} introduced by Diabat et al. (2021) is as follows:

$$\kappa^{\text{driver}} = \frac{\displaystyle\sum_{i \in \mathcal{C}} \sum_{k \in \mathcal{K}} \bar{\lambda}_i^k - |\mathcal{C}|}{|\mathcal{C}|} \times 100. \tag{64}$$

The value of κ^{driver}, which is larger than or equal to 0, measures the percentage deviation from the ideal customer-driver assignment, which is one driver per customer. If every customer i is assigned to a single driver, then $\sum_{k \in \mathcal{K}} \bar{\lambda}_i^k$ equals 1, and then the numerator would be 0, leading to a κ^{driver} value equal to 0 too. This case indicates a perfect customer-driver assignment with regards to a target value of one driver per customer.

We extend this metric to a general case in which the target value can be different than 1. We also propose analogous metrics for all the different consistency features that we study. For driver consistency, with a target value of at most V_i drivers for customer $i \in \mathcal{C}$, the metric is as follows:

$$\kappa^{\text{driver}} = \frac{\displaystyle\sum_{i \in \mathcal{C}} \max \left\{ \sum_{k \in \mathcal{K}} \bar{\lambda}_i^k - V_i, 0 \right\}}{|\mathcal{C}|} \times 100. \tag{65}$$

Notice that this metric considers, for each customer i, a deviation from the target only when the number of different drivers assigned to i exceeds V_i, i.e., when $\sum_{k \in \mathcal{K}} \bar{\lambda}_i^k$ is larger than V_i. When we have $V_i = 1$, $\forall i \in \mathcal{C}$, then (65) reduces to (64), given that in any instance with no redundant customers the value of $\sum_{k \in \mathcal{K}} \bar{\lambda}_i^k$ must be at least one.

Based on this metric, we present analogous ones for the remaining consistency features. Consider the following additional notation. For a given solution, let $\bar{\alpha}_{ij}$ and $\bar{\theta}_{ip}$ be the values of the variables α_{ij} and θ_{ip}, respectively. The metrics for source, product and plant consistency are as follows:

$$\kappa^{\text{source}} = \frac{\displaystyle\sum_{i \in \mathcal{C}} \max \left\{ \sum_{j \in \mathcal{M}} \bar{\alpha}_{ij} - E_i, 0 \right\}}{|\mathcal{C}|} \times 100, \tag{66}$$

$$\kappa^{\text{product}} = \frac{\displaystyle\sum_{i\in\mathcal{M}} \max\left\{\sum_{p\in\mathcal{P}} \bar{\theta}_{ip} - O_i, 0\right\}}{|\mathcal{M}|} \times 100, \tag{67}$$

$$\kappa^{\text{plant}} = \frac{\displaystyle\sum_{p\in\mathcal{P}} \max\left\{\sum_{i\in\mathcal{M}} \bar{\theta}_{ip} - M_p, 0\right\}}{|\mathcal{P}|} \times 100. \tag{68}$$

The value of κ, for each case, measures the average percentage deviation from the corresponding targets (E_i, O_i, and M_p for source, product and plant consistency, respectively). Their values, which are always larger than or equal to zero, indicate how far the solution is from not violating the predefined targets.

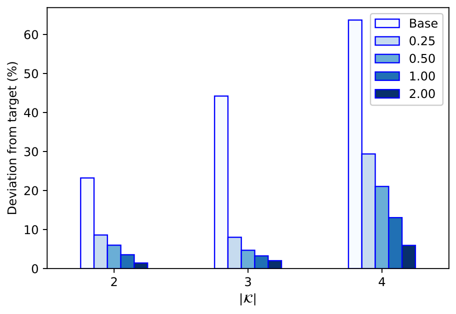
Using these metrics, we perform sensitivity analyses with regards to different values for the weights of each individual consistency feature. We solved all the instances for different values of the weights using the branch-and-cut algorithm. For the sake of comparability, we only analyze the results for instances solved to optimality within the time limit for all the weight values (for each consistency requirement). Also, in this part we did not include the valid inequalities in the model to avoid any conflict between them and the consistency features of the solutions.

5.3.1 Driver Consistency

We initially investigate driver consistency and its impact on the solutions. For this purpose, we analyze the value of κ^{driver} and the solution structure change for different values of the weight ρ. As such, we solved each instance, setting the value of ρ to 0.25, 0.50, 1.00, and 2.00, in addition to the baseline case in which no penalty is included. The target value V_i was set to 1 for every customer $i \in \mathcal{C}$. In this experiment, we considered the results of 416 instances, corresponding to the ones for which the branch-and-cut algorithm could find (and prove) the optimal solution within the time limit for all the configurations. The results are shown in Figures 4 and 5. We first analyze, in Figure 4, the average value of the driver consistency metric (over all the considered instances) for the different weights. We show the results separated by the number of vehicles ($|\mathcal{K}|$) of the instances. In the figure, we can observe that when no penalty is included (Base) the solutions have relatively large deviations from the target values. In particular, an average deviation of about 60% w.r.t. the target is observed for the instances with four vehicles. On the other hand, as expected, when the weight of the violation penalty increases, the deviations tend to decrease, getting close to the target value of one driver per customer.

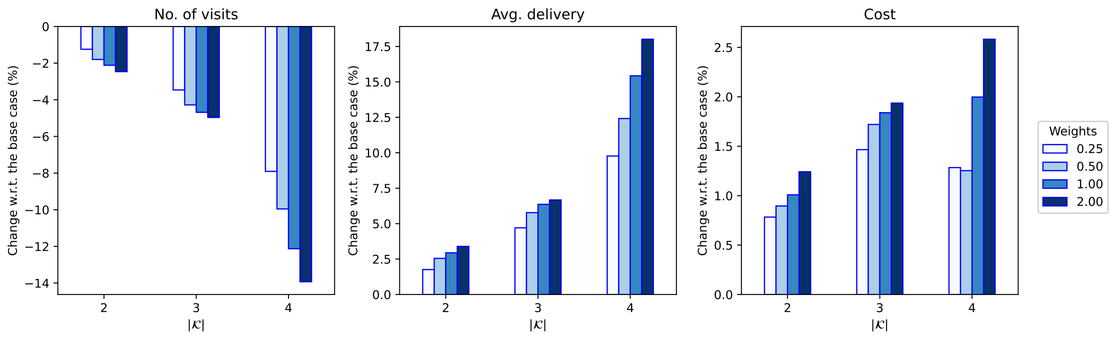
Figure 5 shows the changes in the number of visits, average delivery size, and cost (without violation penalties) of the solutions, when changing the weight value of the penalty term. We computed the changes, for each value of ρ, w.r.t. the baseline case ($\rho = 0$). We chose these features since they presented the clearest trends among several other solution traits such as the number of setups, inventory levels, lot sizes and different capacity usages. The figure shows that to reduce the relative deviation from the target, the solutions have fewer visits with larger delivery sizes. This change implies reducing the number

Figure 4: Value of κ^{driver} for the different weights



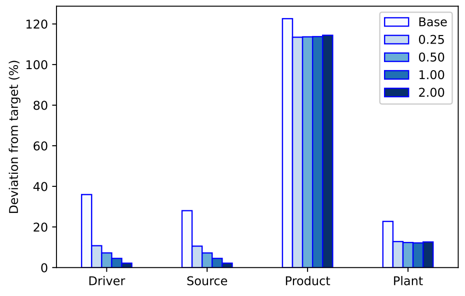
of customers per route, which is necessary to have more dedicated customer-driver assignments. These changes are remarkably higher for the instances with more vehicles, reducing the number of visits up to 13% and increasing the average delivery quantity up to 17%, for the instances with four vehicles. Finally, as a result of these changes, the cost of the solutions increases, although in a relatively low proportion (up to 2.5% on average for the instances with four vehicles and $\rho = 2$). This conclusion is in line with the one found by Diabat et al. (2021) for the IRP and indicates that, if improving driver consistency is desirable for decision-makers, in the PRP context the cost-consistency trade-off can be achieved without compromising excessively on the cost component.

Figure 5: Change in the solutions w.r.t. the base case for driver consistency



To further investigate the impact of driver consistency on other parts of the solutions, we analyzed the change of the other consistency metrics (for $E_i = 1\ \forall i \in \mathcal{C}$, $O_i = 1\ \ \forall i \in \mathcal{M}$, and $M_p = 1\ \ \forall p \in \mathcal{P}$) when changing the weight of the driver consistency violation penalty. In this experiment, we do not optimize for the other consistency features but rather just compute their corresponding metrics for the solutions when optimizing for driver consistency. Figure 6 displays the values of the different metrics, showing the trend of the source consistency metric, that follows that of the driver consistency and at a similar change rate. Since in our problem every driver is assigned to a single plant over the entire horizon, the driver and source consistency metrics improve at similar rates. On the other hand, the product and plant consistency metrics show an initial decrease when a low penalty is included (for driver consistency) and a relatively stable behavior for the rest of the weights. These results reveal that improving driver consistency has little-to-no effect on the solution features that affect plant and product consistency.

Figure 6: Consistency metrics when optimizing for driver consistency



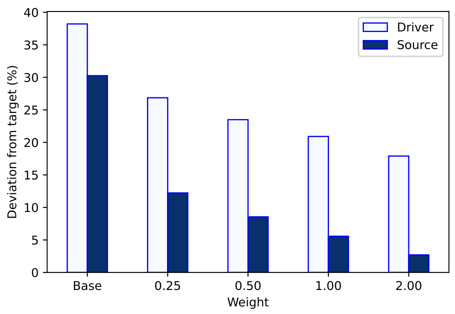
5.3.2 Source Consistency

To analyze the impact of imposing source consistency, we also explored different values for ρ (0.25, 0.50, 1.00, 2.00) and checked the change in the cost and several solution features when optimizing for source consistency with a target E_i set to 1 for every customer $i \in \mathcal{C}$. The results of 434 instances were considered in this analysis. Given the close relation between driver and source consistency in the ConMPRP, we expected to observe results similar to those shown in Section 5.3.1 and this was confirmed by our experiments. As such, we observed comparable impacts of the penalty weight on the solution structure (number of visits and average delivery quantity) and on the solution costs (slightly above 2% on average for the instances with four vehicles and $\rho = 2$). However, the results reveal that source consistency does not reduce driver consistency at the same rate that the latter reduces the former. In this context, Figure 7 shows the average value of the driver and source consistency metric (over all the considered instances) for different weights, when optimizing just for source consistency. Note that, for both metrics, the average deviation from the target (one driver and one source per customer) decreases when we increase the penalty weight. For these instances, the average source consistency metric gets very close to zero (ideal customer-plant assignment for every customer) for the highest value tested (3% deviation for $\rho = 2$). On the other hand, the driver consistency metric does not decrease at the same rate, remaining relatively high for the same case. This observation contrasts with the results of Section 5.3.1, showing that it is possible to have a customer-plant assignment close to the target and still be relatively far from the desired customer-driver relation. These observations are a result of *Property 1*, which indicates that the number of vehicles that visit each customer is always larger than or equal to the number of plants that dispatch to the customer.

5.3.3 Product Consistency

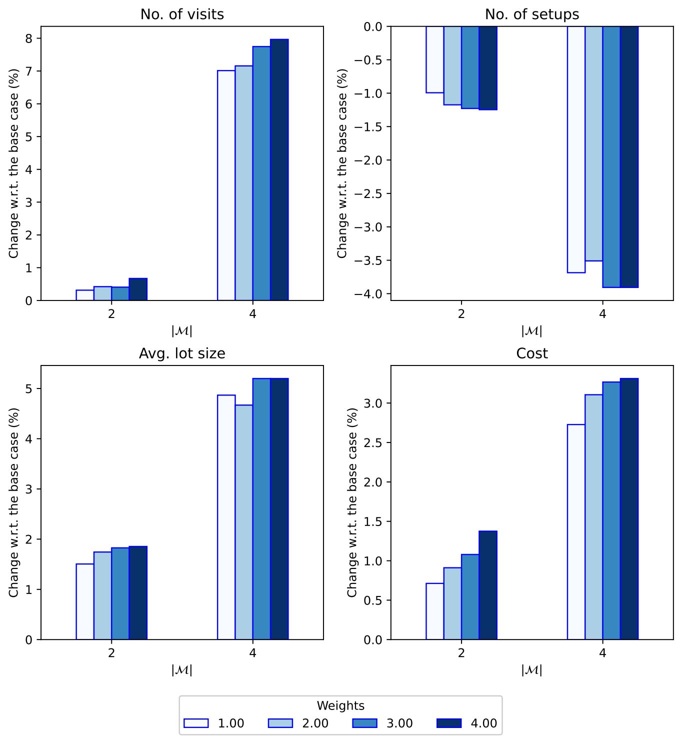
To evaluate the effect of imposing product consistency, in the form of reducing the number of different products made at each plant, we performed experiments analogous to those carried out in the two previous sections. For this purpose, we analyzed the value of κ^{product} and the change in the solution structure for values of the weight ω in $\{1.0, 2.0, 3.0, 4.0\}$ and a target O_i equal to 1 for every plant $i \in \mathcal{M}$. A total of 487 instances (for all configurations) were solved to optimality within the time limit by the branch-and-cut algorithm. Figure 8 shows the changes in the number of visits, setups, average lot size, and cost

Figure 7: Value of κ^{driver} and κ^{source} when optimizing for source consistency



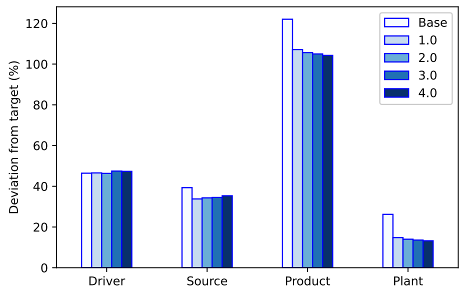
(without violation penalties) of the solutions for the different values of ω. We computed the changes w.r.t. the baseline case ($\omega = 0$) and display the results separated by the number of plants ($|\mathcal{M}|$). The figure shows that the weight increase results, in general, in fewer setups of larger batches. The change is particularly significant for the instances with more plants (batches up to 5% larger than the baseline case for the cases with four plants). This observation may be due to the fact that the production of the individual products is now more concentrated at fewer plants, which produce larger batches of fewer different products over the horizon. This change requires more visits to the customers by vehicles from different plants. As expected, the cost of the solutions increases due to these changes (up to 3% on average for the instances with four plants and $\omega = 4$).

Figure 8: Change in the solutions w.r.t. the base case for product consistency



We also analyzed the consistency metrics change for different weights for the violation penalty of the product consistency. Again, in this experiment, we optimize for product consistency and compute the metrics for the remaining features (for a maximum target of 1 for each of the consistency features). The results are displayed in Figure 9, which shows that it is very difficult to improve the product consistency of the solutions under the current configuration. As such, the values of the percentage deviations from the targets remain close to 100% for product consistency, even for relatively high penalty costs ($\omega = 4.0$). It is reasonable to attribute this observation to the fact that it might be impossible to have a one-to-one plant-product assignment. The first limitation in this sense appears when the number of products is larger than the number of plants. Another limitation is that some customers might need more than one product in the first period (when their initial inventory for those products does not cover the first period demand), requiring deliveries from more than one plant in a single period (if each product is produced at a single plant), which is not allowed in our problem setting. These factors reveal once again the difficulties in performing the changes required to significantly improve the solution attributes that affect product consistency. Regarding the other consistency metrics, note that they remain relatively stable for increasing weights, showing initial alterations when a relatively low penalty is included and a stable behavior afterwards.

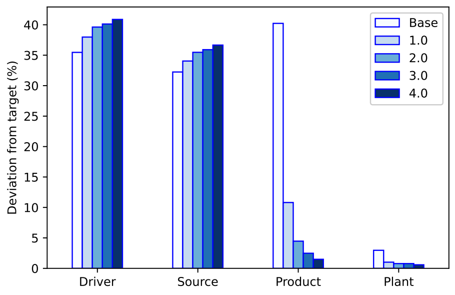
Figure 9: Consistency metrics when optimizing for product consistency



To verify the effect of the first time period and the initial inventories on the consistency features, we carried out an additional analysis excluding the first period when computing the corresponding violations and metrics. This experiment requires modifying the equations of the model that account for the violations so that they do not include the first period. The same applies to the equations used to compute the consistency metrics κ. Note that these changes require running the branch-and-cut algorithm for the modified version of the model so that the violations are optimized disregarding the first period. In this case, the branch-and-cut algorithm could prove the optimality of the solutions within the time limit with all the configurations for a total of 504 instances. The results of this experiment are displayed in Figure 10, which shows all consistency metrics for the different values of ω (optimizing for product consistency only). The figure shows that the increasing penalty weight significantly improves the product consistency when we ignore the first period for computation purposes. This result contrasts highly with the previous analysis, revealing the substantial impact of the first period when measuring this consistency. These results also indicate that low initial inventories at the customer locations can negatively impact the consistency features of the solutions. Low initial inventories might appear, for example, when the minimization of the total cost leads to low inventory levels at the end of the planning horizon. This phenomenon is known as the end-of-horizon effect (Archetti et al., 2017; Ben Ahmed et al., 2021).

On the other hand, the analysis of this scenario also shows that product consistency improves plant consistency. This behavior might appear since when we favor product consistency, each plant focuses on reducing the variety of products made over the horizon (without the first period), implying that each product will likely be produced in fewer plants over the horizon as well. Besides, we observe relatively small increases in the deviations from the target values for the driver and source consistency. This change likely results from the additional routes needed to satisfy the customer demands, which are now typically served from a more diverse set of plants.

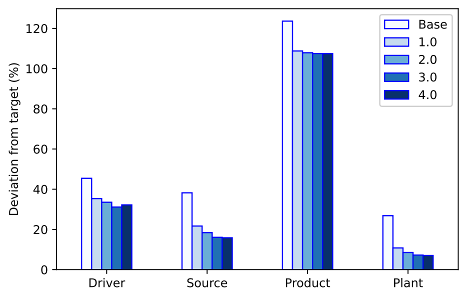
Figure 10: Consistency metrics when optimizing for product consistency ignoring the first period



5.3.4 Plant Consistency

Finally, we evaluated the impact of plant consistency on the solutions. It is worth remembering that this form of consistency favors reducing the number of plants at which each product is made. As in the previous experiments, we analyzed the consistency metrics change for different weights for the violation penalty of the plant consistency and a target M_p equal to 1 for every product $p \in \mathcal{P}$. This experiment includes the results of 507 instances that the branch-and-cut algorithm solved to optimality with all configurations. The results are shown in Figure 11, in which it is possible to observe how the plant consistency metric improves when the penalty increases. The average value of the deviation from the target (one plant per product) decreases sharply (going from 26% to 10%) when we set $\omega = 1.0$, and then reduces slowly reaching an average value of about 7% when $\omega = 4.0$. Although these deviations are relatively low, their reduction implies that the production of each product is likely being focused on fewer plants rather than distributed evenly over all the plants. This aggregation results in improved source and driver consistency of the solutions as fewer plants produce the different products and deliver them to the customers. Furthermore, this experiment confirms the difficulty in obtaining solutions with a relatively low deviation of the product consistency metric for the reasons discussed in the previous section. Finally, our analysis also shows that in order to reduce the violation of the plant consistency measure, the average cost of the solutions increases around 2.5%, which shows that the cost-consistency trade-off can be also achieved with relatively low cost compromise in the plant consistency context.

Figure 11: Consistency metrics when optimizing for plant consistency



5.4 Results with the Heuristic for the Consistency Features

Finally, we also evaluate the heuristic performance for the case with the different consistency features. We compared the consistency metrics of the solutions found by the heuristic with the metrics of the optimal solutions (found by the branch-and-cut algorithm). These instances correspond to the same used for the analyses of Section 5.3. The results are displayed in Figure 12, which shows the average value of the percentage deviations (κ), separated by each of the consistency features. The figure shows that, with regards to the consistency metrics, the heuristic solutions follow the same behavior as the optimal ones. Increasing the penalties for the consistency violations results, in general, in decreasing deviations w.r.t. the target set for each case. The rate of the deviation decrease for the heuristic solutions is very close to the one of the optimal solutions. Besides, for every feature and value of the weights, the average value of the consistency metrics found by the heuristic method presents only relatively small differences to the optimal value. Moreover, the average optimality gap of the solutions of the heuristic is 0.94%, 0.58%, 0.68% and 0.54%, for driver, source, product and plant consistency, respectively. To find these solutions, the heuristic took on average less than 150 seconds, for every consistency feature. These results highlight that the heuristic is also suitable to solve the ConMPRP and, therefore, could be used to support decision-making in this context.

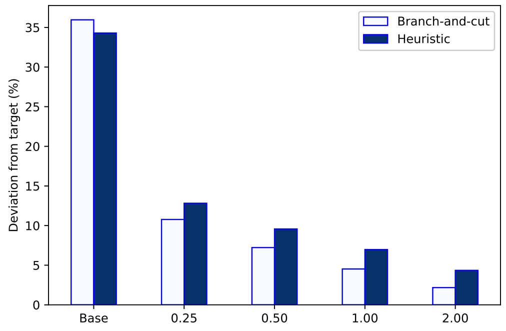
6 Conclusions

In this paper, we introduced the consistent PRP (ConMPRP) as an extension of the MPRP in which there exist some consistency requirements. We considered four forms of consistency, namely: driver, source, product and plant consistency. In the ConMPRP, the different consistency requirements were addressed by setting a target maximum value for each of them. These targets are imposed through soft constraints, whose violations are minimized when optimizing the production and routing plan. This approach brings flexibility when considering the consistency requirements and solving the corresponding problems.

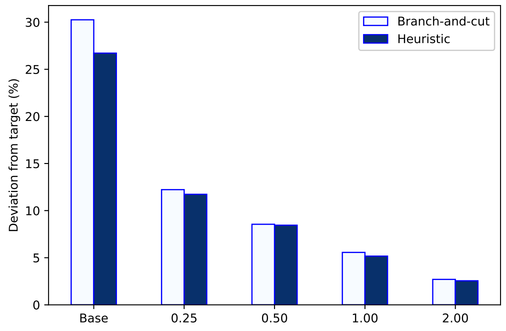
We proposed a mathematical formulation for the ConMPRP and presented a branch-and-cut algorithm, which was enhanced with valid inequalities and specific branching priorities. We also presented a heuristic solution method based on iterated local search and several mathematical programming components. Experiments on a large benchmark set of newly introduced instances showed that the additional enhancements improve the performance of the exact algorithm and that the heuristic method performs

Figure 12: Comparison of the consistency metrics of the heuristic and branch-and-cut solutions

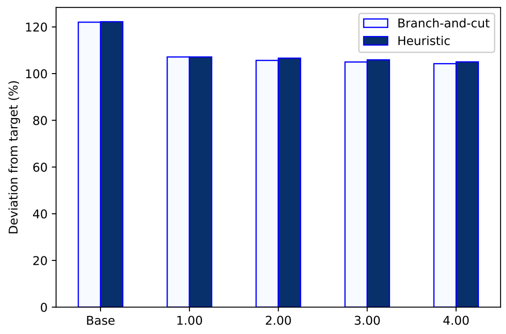
(a) Driver consistency



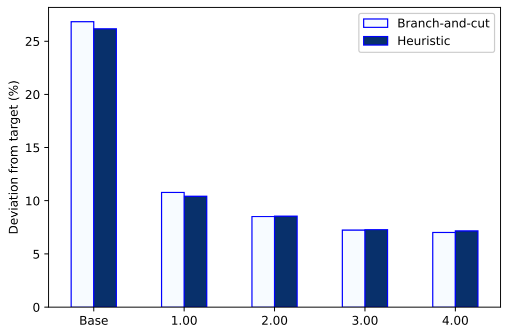
(b) Source consistency



(c) Product consistency



(d) Plant consistency



robustly for the ConMPRP, the MPRP and the standard PRP. We also analyzed the trade-off between cost and consistency of the solutions in the PRP context. In particular, we verified that it is possible to achieve such trade-offs without compromising excessively on the cost component. The results also revealed the impact of the first period when optimizing and measuring the consistency requirements we studied. In the future, the introduced problem could be extended to settings allowing additional decisions like transshipments between customers and transfers between plants. Also, studying the impact of such decisions on the consistency features becomes a clear path for research.

Acknowledgments

This research has been funded by GERAD, and the Chair in Supply Chain Operations Planning and the Chair in Logistics and Transportation at HEC Montréal.

References

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

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

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

Alvarez, A., Cordeau, J.-F., Jans, R., Munari, P., and Morabito, R. (2021). Inventory routing under stochastic supply and demand. *Omega*, 102:102304.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2018). Concorde TSP solver. `http://www.math.uwaterloo.ca/tsp/concorde.html`. Accessed: 2018-07-20.

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

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

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

Archetti, C., Desaulniers, G., and Speranza, M. G. (2017). Minimizing the logistic ratio in the inventory routing problem. *EURO Journal on Transportation and Logistics*, 6(4):289–306.

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

Ben Ahmed, M., Okoronkwo, O. L., Okoronkwo, E. C., and Hvattum, L. M. (2021). Long-term effects of short planning horizons for inventory routing problems. *International Transactions in Operational Research*, pages 1–36.

Braekers, K. and Kovacs, A. A. (2016). A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological*, 94:355–377.

Campelo, P., Neves-Moreira, F., Amorim, P., and Almada-Lobo, B. (2019). Consistent vehicle routing problem with service level agreements: A case study in the pharmaceutical distribution sector. *European Journal of Operational Research*, 273(1):131–145.

Cardona-Valdés, Y., Álvarez, A., and Pacheco, J. (2014). Metaheuristic procedure for a bi-objective supply chain design problem with uncertainty. *Transportation Research Part B: Methodological*, 60:66–84.

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

Coelho, L., Cordeau, J.-F., and Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270–287.

Diabat, A., Archetti, C., and Najy, W. (2021). The fixed-partition policy inventory routing problem. *Transportation Science*, 55(2):353–370.

Fiorotto, D. J., Jans, R., and de Araujo, S. A. (2018). Process flexibility and the chaining principle in lot sizing problems. *International Journal of Production Economics*, 204:244–263.

Graves, S. C. and Tomlin, B. T. (2003). Process flexibility in supply chains. *Management Science*, 49(7):907–919.

Groër, C., Golden, B., and Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643.

Jans, R. (2009). Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints. *INFORMS Journal on Computing*, 21(1):123–136.

Jordan, W. C. and Graves, S. C. (1995). Principles on the benefits of manufacturing process flexibility. *Management Science*, 41(4):577–594.

Kovacs, A. A., Golden, B. L., Hartl, R. F., and Parragh, S. N. (2014). Vehicle routing problems in which consistency considerations are important: A survey. *Networks*, 64(3):192–213.

Kovacs, A. A., Golden, B. L., Hartl, R. F., and Parragh, S. N. (2015). The generalized consistent vehicle routing problem. *Transportation Science*, 49(4):796–816.

Lespay, H. and Suchan, K. (2021). A case study of consistent vehicle routing problem with time windows. *International Transactions in Operational Research*, 28(3):1135–1163.

Li, Y., Chu, F., Chu, C., and Zhu, Z. (2019). 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.

Li, Y., Chu, F., Côté, J.-F., Coelho, L. C., and Chu, C. (2020). The multi-plant perishable food production routing with packaging consideration. *International Journal of Production Economics*, 221:107472.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2019). Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 129–168. Springer.

Mak, H.-Y. and Shen, Z.-J. M. (2009). Stochastic programming approach to process flexibility design. *Flexible Services and Manufacturing Journal*, 21(3):75–91.

Manousakis, E. G., Kasapidis, G. A., Kiranoudis, C. T., and Zachariadis, E. E. (2021). An infeasible space exploring matheuristic for the production routing problem. *European Journal of Operational Research*.

Marchetti, P. A., Gupta, V., Grossmann, I. E., Cook, L., Valton, P.-M., Singh, T., Li, T., and André, J. (2014). Simultaneous production and distribution of industrial gas supply-chains. *Computers & Chemical Engineering*, 69:39–58.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100.

Nascimento, M. C. V. and Toledo, F. M. B. (2008). A hybrid approach for the capacitated lot sizing problem with setup carryover. *International Journal of Production Research*, 50(6):1582–1597.

Padberg, M. and Rinaldi, G. (1990). Facet identification for the symmetric traveling salesman polytope. *Mathematical programming*, 47(1):219–257.

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

Sambasivan, M. and Yahya, S. (2005). A lagrangean-based heuristic for multi-plant, multi-item, multi-period capacitated lot-sizing problems with inter-plant transfers. *Computers & Operations Research*, 32(3):537–555.

Schmitt, A. J. (2011). Strategies for customer service level protection under multi-echelon supply chain disruption risk. *Transportation Research Part B: Methodological*, 45(8):1266–1283.

Shahabi, M., Tafreshian, A., Unnikrishnan, A., and Boyles, S. D. (2018). Joint production-inventory-location problem with multi-variate normal demand. *Transportation Research Part B: Methodological*, 110:60–78.

Smilowitz, K., Nowak, M., and Jiang, T. (2013). Workforce management in periodic delivery operations. *Transportation Science*, 47(2):214–230.

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

Tellez, O., Vercraene, S., Lehuédé, F., Péton, O., and Monteiro, T. (2021). The time-consistent dial-a-ride problem. *Networks*.

Vidal, T. (2020). Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. Technical report. PUC-Rio. Available in ArXiV: https://arxiv.org/abs/2012.10384.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

Zhang, Q., Sundaramoorthy, A., Grossmann, I. E., and Pinto, J. M. (2017). Multiscale production routing in multicommodity supply chains with complex production facilities. *Computers & Operations Research*, 79:207–222.