An Infeasible Space Exploring Matheuristic for the Production Routing Problem

Eleftherios G. Manousakis[a,*], Grigoris A. Kasapidis[a], Chris T. Kiranoudis[b], Emmanouil E. Zachariadis[a]

[a] *Department of Management Science and Technology, School of Business, Athens University of Economics and Business, Athens, Greece*

[b] *Department of Process Analysis and Plant Design, School of Chemical Engineering, National Technical University of Athens, Athens, Greece*

Abstract

This paper proposes a novel matheuristic algorithm for solving the Production Routing Problem (PRP). The PRP is a hard-to-solve combinatorial optimization problem with numerous practical applications in the field of freight transportation, logistics and supply chain management. A manufacturer is responsible for determining production decisions, as well as the timing and the quantity of replenishment services offered to a set of geographically dispersed customers over a multi-period time horizon. The problem calls for jointly optimizing the production, inventory, distribution and routing decisions. The paper provides a novel two-commodity flow formulation and proposes a two-phase infeasible space matheuristic algorithm for solving the examined problem. The first phase deals with a relaxation of the problem to construct production-distribution plans. In the second phase, these are completed with routing information and optimized via a local search framework which oscillates between the feasible and the infeasible solution space. The framework is equipped with mixed integer programming components for restoring infeasibility and for diversifying the conducted search. Computational experiments demonstrate that the infeasibility space exploration significantly contributes to the quality of the final solutions. The results obtained by the proposed matheuristic out-match the results of state-of-the-art approaches. More specifically, 594 and 55 new best solutions out of 1440 and 90 instances of two well-established benchmark data sets of small-medium and large instances are reported, respectively.

Keywords: transportation, integrated routing problem, production routing, matheuristic, infeasible space search

1. Introduction

A supply chain of a product consists of several activities such as the production, inventory control, distribution and routing that can take place in geographically different areas. All these activities are interdependent and their coordination is required to ensure good system performance. According to recent research, the total logistics industry costs for 2018 are calculated to the staggering amount of \$1,700 billion and \$9,044 billion for EU and worldwide, respectively (Mazareanu 2020). Despite the evident importance, the common practices for decision-making rely on the fragmentation of the above processes in discrete stages that are optimized hierarchically, usually from production to consumption, without feedback of every stage to the other ones. Practically, this means that the system is divided into

*Corresponding author

Email addresses: lmanousakis@aueb.gr (Eleftherios G. Manousakis), gkasapidis@aueb.gr (Grigoris A. Kasapidis), kyr@chemeng.ntua.gr (Chris T. Kiranoudis), ezach@aueb.gr (Emmanouil E. Zachariadis)

isolated sub-problems and the decision-making process becomes myopic, inelastic and often ineffective. However, regardless of the significantly lower quality of this decision-making approach, it remains the common practice as the integrated decision making often leads to extremely complex logistics management problems that are difficult to tackle (Archetti and Speranza 2016).

During the last few years, computational advances have enabled researchers to develop more and more case-specific and realistic integrated models, as well as optimization methodologies to tackle these arising intractable problems. The new models exploit the benefits of integrated optimization as opposed to the "traditional" hierarchical optimization processes which are usually adopted for tackling realistic industrial problems. Regarding the well-known Inventory Routing Problem (IRP) that jointly coordinates distribution, timing, quantities and routing decisions, Archetti and Speranza (2016) claim that inventory and routing cost savings achieved by the Vendor Managed Inventory (VMI) policy in comparison to the Retailer Managed Inventory (RMI) policy may range up to approximately 10% for well-known benchmark data sets. In the context of the Production Routing Problem which generalizes the IRP by incorporating the production decisions, Absi et al. (2016) compare an integrated approach with the performance of two sequential ones: results show that the dominance of the integrated approach depends on the balance between production and distribution costs and on the balance between production setup and inventory costs in production facility. By incorporating location decisions into a production routing variant, Darvish and Coelho (2018) show that all tested sequential decision-making approaches are dominated by an integrated method.

One integrated problem arising in numerous supply chains, is the well-studied Production Routing Problem (PRP). The PRP is an NP-hard combinatorial optimization problem aimed at jointly optimizing the production, inventory, distribution and routing decisions over a period of time. It incorporates the cost-saving VMI policy according to which a retailer is responsible for replenishing the inventory of geographically dispersed customers ensuring that no stock-outs occur at any time. The suppliers have complete knowledge of the inventory level of customers throughout the planning horizon, and therefore can decide on the timing, quantities and transportation plans for customer inventory replenishment. Thus, VMI creates a win-win situation, as on the one hand, customers save resources from inventory control and order placement and on the other hand, suppliers are able to better coordinate production, inventory and distribution processes (Archetti and Speranza 2016). Additionally, PRP incorporates the lot-sizing problem in terms of deciding the timing and the quantities of the production. Therefore, it combines two well-known problems (lot-sizing - LSP, vehicle routing - VRP). It can also be seen as a generalization of the lot-sizing problem with direct shipment and of the IRP (Adulyasak et al. 2015b).

The Production Routing Problem has been introduced in 1993 (Chandra 1993) and it has been pointed out that an integrated production, inventory and routing planning can reduce the total operating cost by 3%–20% (Chandra and Fisher 1994). It has received increasing research attention, with most works published during the last few years. The combined nature of the problem makes it directly applicable to various industry fields. In practice, companies such as Kellogg (Brown et al. 2001) and Frito-Lay (Çetinkaya et al. 2009) have recorded significant savings by jointly optimizing the decision levels considered by PRP. The broad applicability of PRP has driven researchers to introduce numerous variants to cope with a realistic spectrum of business settings. For instance, Adulyasak et al. (2015a) consider a stochastic PRP variant with demand uncertainty, whereas Qiu et al. (2019) consider the PRP for perishable products and experiment with different selling policies to minimize value losses. In a similar manner, Dayarian and Desaulniers (2019) model and solve the problem of a catering company that delivers meals with short life-span, including features such as multi-trips and time-windows. Motivated by the petrochemical industry, Schenekemberg et al. (2021) study the Two-Echelon PRP with pickups and inventory control of ethanol from the suppliers, production and inventory of pure and commercial gasoline, and deliveries of commercial gasoline to the final customers. The interested reader is referred to the reviews of Díaz-Madroñero et al. (2015) and Adulyasak et al. (2015a) for surveys on optimization models for integrated production, inventory and routing decisions.

Due to the high complexity of PRP, the majority of early approaches are based on decomposing the master problem to subproblems that are usually tackled sequentially via metaheuristics such as Greedy Randomized Adaptive Search Procedure (GRASP) with path relinking (Boudia et al. 2007), two-stage local search algorithms (Boudia et al. 2008), memetic algorithms with population management (Boudia and Prins 2009), tabu seach schemes with path relinking and both short and long-term memories (Armentano et al. 2011), etc. The exact approaches are scarce and capable of tackling relatively small instances. For example, for the single vehicle case, Archetti et al. (2011) apply an exact approach solving six period instances, whereas Qiu et al. (2018b) propose a Branch-and-Cut (BnC) with new valid inequalities solving instances with only three periods and up to 100 customers. Adulyasak et al. (2014b) compares the performance of vehicle-indexed and non-vehicle indexed formulations for several cases via single core and parallel computing BnC. Recently, the same relatively small instances are solved for perishable inventory by a BnC strengthened with lot-sizing and lifted Miller–Tucker–Zemlin subtour elimination constraints (Qiu et al. 2019).

Hybrid algorithms that combine metaheuristics and mathematical programming methods, referred to as matheuristics, have promising performance at solving complex routing problems (Archetti and Speranza 2014). The majority of recent PRP works focus on the combination of metaheuristic algorithms with exact Mixed Integer Programming (MIP). Adulyasak et al. (2014a) introduces an adaptive large neighborhood search heuristic with production, inventory, and shipment quantities determined by solving a minimum cost network flow subproblem. Similarly, Absi et al. (2015) and Miranda et al. (2018) propose two-phase iterative schemes with production-distribution and routing subproblems solved by Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) solvers within a local search framework. Decomposition techniques that lead to the formulation of MIP subproblems have been also used within the context of genetic algorithms, in order to generate good quality individuals from the population (Senoussi et al. 2018).

The basic PRP version, that serves as a starting point for numerous extensions, has been introduced by Adulyasak et al. (2015a). For this standard PRP variant, the best performing methods are mainly multi-phase matheuristics. Russell (2017) uses mathematical programming for a relaxed PRP version, to determine an initial solution which is completed by a metaheuristic tabu search scheme based on the concept of seed routes. Solyalı and Süral (2017) propose a five-phase heuristic, with overlapping subproblems formulated as MIPs and solved via exact algorithms. In a similar manner, other multi-phase approaches decompose the problem into the setup, distribution and routing decision levels which can be faced either by exact algorithms (Chitsaz et al. 2019), or by fix-and-optimize strategies (Li et al. 2019). A more straightforward scheme is the multi-start scheme of Avci and Yildiz (2019) in which an MIP-based local search procedure is iteratively applied. On the contrary, hybrid algorithms combining metaheuristics components have been also introduced (skewed general variable neighborhood search and guided variable neighborhood descent) (Qiu et al. 2018a). Most recently, Schenekemberg et al. (2021) introduced a parallelized hybrid of local search with BnC.

The concept of exploiting information from infeasible solutions has produced promising results in the VRP literature. One of the first such papers, proposed the concept of intermediate infeasible solutions in terms of capacity and length violations within a Tabu Search (TS) scheme (Gendreau et al. 1994). Since then, this approach has been employed to deal with hard-to-solve extensions of the classic VRP, such as the VRP with stochastic demands and customers (Gendreau et al. 1996) and the periodic and Multi-Depot VRP with time windows (Cordeau et al. 2001). A guide for designing heuristic and metaheuristic algorithms suggests that the blend of feasible and infeasible solutions reduces the chances of getting trapped in local optima during the search (Cordeau et al. 2002). More recently, Felipe et al. (2011) used intermediate infeasible solutions to diversify the search process. The concept of infeasible solution search is coupled with procedures that restore feasibility by using infeasibility measures to control the "distance" from the feasible space.

The contribution of the paper lies on the benefits enabled by integrated logistics models, the high

complexity of PRP and the effectiveness of infeasible space search. In specific, the present paper presents a novel matheuristic algorithm that oscillates between feasible and infeasible space of the basic PRP model. A novel PRP two-commodity flow formulation is proposed along with adapted and new valid inequalities. The proposed matheuristic scheme is mainly based on a hybrid local search framework equipped with MIP and LP procedures to tackle various subproblems. The production-distribution decisions are taken by solving a PRP relaxation which considers approximated routing costs for customer visits. Afterwards, a heuristic GRASP algorithm completes the partial solution with routing information. The complete solution is then iteratively improved by a local search framework. The main novelty of the proposed local search framework is the exploration of both the feasible and the infeasible regions of the solution space. Specifically, solutions that violate the capacity constraint are allowed and then repaired by employing an MIP procedure capable of inserting, removing and relocating customers. Extensive computational experiments have been conducted on 1530 well-known PRP instances to analyze the proposed algorithm performance and to access its effectiveness. The obtained results outperform the results of the state-of-the-art PRP methods. Several new best solutions for two well-known benchmark data sets have been generated.

The remainder of the paper is organized as follows. Section 2 provides the two-commodity flow formulation along with valid inequalities. The proposed matheuristic is described in detail in Section 3 which also provides the motivation of the algorithmic design and describes the course of the algorithm development. Next, Section 4 reports extensive computational experiments for tuning and analyzing the infeasibility space search impact, as well as to compare the proposed algorithm against state-of-the-art PRP approaches. Finally, Section 5 concludes the paper and offers some promising research directions.

2. Two-commodity flow formulation

The generic PRP variant examined in this paper describes the situation in which a manufacturer of a product is responsible for production and replenishment of customers inventories over a given time horizon, ensuring that no stock-outs occur. The decision-maker is responsible for deciding: i) the time periods at which the production takes place, ii) the product quantities that are produced, iii) the timing for replenishing each customer inventory, iv) the associated replenishment quantity and v) the routing of all customer services, with respect to the minimization of the total cost of the system.

For the sake of completeness of the paper, we provide a new formulation for the PRP which extends the well-performing formulation of Manousakis et al. (2021) for the similar IRP. The basic model and valid inequalities are given for a complete PRP definition, and they are also employed for modeling individual subproblems tackled in the context of the proposed algorithm.

Let an undirected graph $G = (V, E)$, where $V = \{0, 1, \ldots, n, n+1\}$ is the set of nodes and $E = \{(i, j) : i, j \in V,\ i < j\}$ is the edge set. Node 0 represents the production facility and node $n+1$ represents the production facility clone, where the inverse flow originates from. The set $V_c = \{1, 2, \ldots, n\}$ is used to denote the customers. The problem spans over a set of time periods $T = \{1, 2, \ldots, |T|\}$. Each node $i \in V$ starts with initial inventory I_i^0 at period 0 and incurs a unit holding cost h_i for every period $t \in T$. Customer $i \in V_c$ faces a per period $t \in T$ demand d_i^t and has a limited maximum inventory capacity L_i. At the start of period $t \in T$, the production facility (depot) may produce any non-negative quantity p^t up to the per period production capacity limit C, whereas the end inventory I_0^t cannot exceed the production facility storage capacity L_0. The production facility faces a setup cost s^t for each period that any non-zero quantity is produced, whereas a unit production cost of u^t also applies for each unit of production. The produced quantity may be directly used to satisfy customer demand at the same period t. Each edge $(i, j) \in E$ is associated with a non-negative travel cost c_{ij} that represents the cost for traversing this edge. The supplier delivers any non-negative quantity to each customer $i \in V_c$ at $t \in T$, such that no stock-outs occur. For every period, a homogeneous vehicle fleet of vehicles $K = \{1, 2, \ldots, |K|\}$ each of capacity Q is available at the depot.

We assume a symmetric transportation cost matrix, i.e., $c_{ij} = c_{ji}$. We use binary undirected routing variables $x_{ij}^t = 1$ for $i, j \in V$, $i < j$ iff any vehicle $k \in K$ traverses edge (i, j) in any direction at period $t \in T$. Binary variables z_i^t take a value of 1 iff i is visited by any vehicle at period $t \in T$. Similarly, binary variables y^t take a value of 1 iff production takes place at the production facility during period t. Non-negative continuous flow variables f_{ij}^t and f_{ji}^t represent the load and the residual capacity of the vehicle travelling from i to j at time $t \in T$, respectively. Let p^t be the quantity produced at period t, whereas q_i^t denotes non-negative product quantity delivered to customer $i \in V_c$ at time $t \in T$. Finally, continuous variables I_i^t for each node $i \in V$ represent the inventory at the end of period $t \in T$. The objective of the PRP with the ML inventory policy is to minimize the overall fixed and variable production, transportation and inventory holding costs for both the supplier and the customers. Below, the two-commodity flow formulation for the PRP with the maximum level (ML) inventory policy is provided, henceforward named *TCF*:

$$\underset{x,y,z,f,p,q,I}{\text{minimize}}\ g = \sum_{t \in T} \left(u^t p^t + s^t y^t + \sum_{i \in V} h_i I_i^t + \sum_{i \in V} \sum_{j \in V,\, i<j} c_{ij} x_{ij}^t \right) \tag{1}$$

subject to

$$\sum_{j \in V, i>j} x_{ji}^t + \sum_{j \in V, i<j} x_{ij}^t = 2z_i^t \qquad i \in V_c \quad t \in T \tag{2}$$

$$\sum_{j \in V_c} x_{0j}^t \leq |K| \qquad t \in T \tag{3}$$

$$\sum_{j \in V_c} x_{0j}^t = \sum_{i \in V_c} x_{i(n+1)}^t \qquad t \in T \tag{4}$$

$$f_{ij}^t + f_{ji}^t = Q x_{ij}^t \qquad i, j \in V,\ i < j \quad t \in T \tag{5}$$

$$\sum_{j \in V, i \neq j} f_{ij}^t = Q z_i^t - q_i^t \qquad i \in V_c \quad t \in T \tag{6}$$

$$\sum_{j \in V_c} f_{0j}^t = \sum_{i \in V_c} q_i^t \qquad t \in T \tag{7}$$

$$\sum_{i \in V_c} f_{i(n+1)}^t = 0 \qquad t \in T \tag{8}$$

$$p^t \leq min\left\{ C, \sum_{i \in V_c} \sum_{t'=t}^{|T|} d_i^{t'} \right\} y^t \quad t \in T \tag{9}$$

$$p^t \leq \sum_{i \in V_c} \left(\sum_{t'=t}^{|T|} d_i^{t'} - I_i^{t-1} \right) - I_0^{t-1} \quad t \in T \tag{10}$$

$$I_0^t = I_0^{t-1} + p^t - \sum_{i \in V_c} q_i^t \qquad t \in T \tag{11}$$

$$I_i^t = I_i^{t-1} + q_i^t - d_i^t \qquad i \in V_c \quad t \in T \tag{12}$$

$$q_i^t \leq L_i + d_i^t - I_i^{t-1} \qquad i \in V_c \quad t \in T \tag{13}$$

$$x_{ij}^t \in \{0, 1\} \qquad i, j \in V,\ i < j \quad t \in T \tag{14}$$

$$y^t \in \{0, 1\} \quad t \in T \tag{15}$$

$$z_i^t \in \{0,1\} \qquad i \in V_c \quad t \in T \tag{16}$$

$$0 \leq f_{ij}^t \leq Q \qquad i,j \in V,\ i \neq j \quad t \in T \tag{17}$$

$$0 \leq q_i^t \leq min\left\{ L_i + d_i^t, Q, \sum_{t'=t}^{|T|} d_i^{t'} \right\} \qquad i \in V_c \quad t \in T \tag{18}$$

$$0 \leq p^t \leq min\left\{ C, \sum_{i \in V_c} \sum_{t'=t}^{|T|} d_i^{t'} \right\} \quad t \in T \tag{19}$$

$$0 \leq I_i^t \leq L_i \qquad i \in V \quad t \in T \tag{20}$$

The objective function g represents the sum of the depot setup and unit production costs, the transportation costs and the inventory holding costs of both the depot and the customers over the planning horizon. Note that the holding costs of the initial inventory at period $t = 0$ are also taken into account, whereas no inventory holding costs are considered for the artificial depot node. Constraints (2) are the degree constraints for the customers, while constraints (3) and (4) equate the number of vehicles leaving and returning to the depot and ensure that this number does not exceed $|K|$. Constraints (5) ensure that the total flow (normal and inverse of an edge) is equal to the vehicle capacity if and only if the edge is traversed by a vehicle. Constraints (6) imply the flow as they force the total sum of the flows originating from a node to be reduced by the delivery quantity absorbed by the node. The total product quantity starting from the depot is given by (7). Accordingly, constraints (8) ensure that no products arrive at the depot when routes are terminated, thus guaranteeing that the total product quantity leaving the depot is equal to the quantity delivered to the customers. Constraints (9) ensure that the produced quantity may be positive only when the production facility is open. In this case, the produced quantity cannot exceed the total production capacity or the sum of the remaining demand of all customers. More specifically, according to constraints (10) the produced quantity and the remaining depot inventory cannot exceed the remaining demand after subtracting the existing customer stocks as in any other case the solution would be sub-optimal. Constraints (11) and (12) represent the inventory flow preservation over the periods of the planning horizon for the depot and the customers, whereas constraints (13) implement the ML policy. Finally, constraints (14)–(20) enforce integrality, as well as lower and upper bounds on the decision variables. The flow variables y_{ij}^t are defined by (17) for each edge and direction as the existence of flows imposes direction to the model. Note that period $t = 0$ contributes to the overall holding costs; however, no transportation takes place at this period. It should be underlined that the *TCF* formulation, as a flow model, directly implies the sub-tour elimination from the flow related constraints (5)–(8). Therefore, there is no need to introduce $2^{|V|}$ additional constraints to implement the classic DFJ sub-tour elimination constraints (i.e., one for each subset of V) or new variables for the alternative MTZ sub-tour elimination constraints. In our formulation sub-tours are prevented by imposing the continuous flow of the load and the residual capacity. Hence, it can be solved as-is via any off-the-shelf Mixed Integer Linear Programming (MILP) solver. For completeness, adapted and new valid inequalities are provided in Appendix A.

Regarding the maximum delivery quantity of customer i at period t, we distinguish between two cases that have appeared in the PRP literature. The common practice, is to allow the delivered quantity q_i^t to exceed the maximum capacity L_i making sure that the excessive quantity is consumed during this period t, so that the inventory I_i^t at the end of the period does not exceed maximum capacity L_i. However, other research works, adopt a stricter assumption forcing the delivery quantity $q_i^t \leq L_i, \forall i \in V_c, t \in T$ (Adulyasak et al. 2014b, 2015b, Qiu et al. 2018a). In the present work and to enable comparisons, we consider the former case. However, the proposed model can be modified to

capture the latter case by replacing constraints (13) and (18) with (21) and (22):

$$q_i^t \leq L_i - I_i^{t-1} \qquad i \in V_c \quad t \in T \tag{21}$$

$$0 \leq q_i^t \leq min\left\{L_i, Q, \sum_{t'=t}^{l} d_i^{t'}\right\} \qquad i \in V_c \quad t \in T \tag{22}$$

Furthermore, in order to facilitate the methodology description, additional notation is introduced for all the algorithmic procedures that refer to a specific solution S. We define $\tilde{y}^t$, $\tilde{z}_i^t$, $\tilde{x}_{ij}^t$, $\tilde{p}^t$ $\tilde{q}_i^t$, $\tilde{I}_i^t$ and $\tilde{f}_{ij}^t$ to be the currently assigned values of y^t, z_i^t, x_{ij}^t, p^t q_i^t, I_i^t and f_{ij}^t in the solution S. Every solution consists of a set of routes R and each route $r \in R$ is assigned to a specific vehicle. Let function $\zeta : (V_c, T) \rightarrow R$, return the route r in which customer $i \in V_c$ is assigned for period $t \in T$. Additionally, let $V_c^r \subseteq V_c$ be the subset of customers that are included in route $r \in R$.

3. Methodology

This section presents the proposed matheuristic approach for the generic version of PRP. It starts with a high-level outline of the proposed algorithmic scheme. Next, a relaxation for obtaining a partial solution, as well as a randomized construction heuristic for generating routes and completing the partial solution are introduced. Then, the core of the approach, which is a local search algorithm integrated with exact optimization components, is described. Finally, the section is concluded with the motivation of design of the proposed approach.

We propose a two-phase matheuristic approach for solving the PRP. Inspired by the use of mathematical programming components and infeasible space search, the algorithm is named Hybrid Infeasible Space Matheuristic (*HISM*). In Phase I, a partial initial solution is generated by solving a relaxation of the *TCF* master problem, called *PD-Rel*. This generates a partial PRP solution, and more specifically a production-distribution schedule which ignores the routing decision level (Section 3.1). In Phase II, the partial solution is firstly completed with routing information and then iteratively improved. To complete the partial solution, a greedy randomized adaptive search procedure (*GRASP*) probabilistically constructs routes according to the predetermined production-distribution plans (Section 3.2). Afterwards, the complete solution is optimized by a matheuristic local search algorithm (Section 3.3) equipped with exact components (Section 3.4). This algorithm oscillates between infeasible and feasible space by allowing vehicle capacity infeasibilities and applying an infeasibility restoration MIP procedure, respectively. It is referred as Feasible-Infeasible Local Search (*FILS*).

The outline of the overall *HISM* framework is presented in Algorithm 1.

Phase I is an initialization phase. More specifically, the *PD-Rel* relaxation is solved, so that an initial solution which consists of the delivery schedule and quantities for all customers, is generated (line 2). The second phase is repeated for ϵ^r restarts (line 3). The routing decisions are incorporated in the partial solution by means of the GRASP routing heuristic (line 4) which constructs the routes and assigns customers to vehicles with respect to the determined delivery quantities. The complete initial solution S_0 is then improved by *FILS* to obtain a solution S_b which denotes the best solution generated through restart b (line 5). Obviously, the final solution returned by the *HISM* is the best solution identified over all restarts. Note that, all restarts share the same production-distribution schedule and quantities, however each restart is differentiated according to the probabilistically generated route set. The individual components of *HISM* are described in the following sections.

3.1. Production-distribution relaxation

The construction of an initial PRP solution is a challenging task. Common construction heuristics cannot straightforwardly decide the continuous production and delivery quantities which obviously are interconnected with the routing decision level. A simple heuristic design is difficult to ensure feasibility

Algorithm 1 *Overall Scheme of HISM*

Phase I
$S_0 \leftarrow \emptyset$, $S^* \leftarrow \emptyset$
y, z, p, q, $I \leftarrow$ *PD-Rel()*
Phase II
for $b \leftarrow 1$ to ϵ^r **do**
  $S_0 \leftarrow$ *GRASP(z, q, I)*
  $S_b \leftarrow$ *FILS(S_0)*
  if $Z(S_b) < Z(S^*)$ **then**
    $S^* \leftarrow S_b$
  end if
end for
return S^*

due to the three interconnected decision levels. To overcome this challenge, our proposed heuristic solves a production-distribution relaxation of the proposed *TCF* formulation, named *PD-Rel*. The aim is to generate a partial solution with approximated routing costs. Note that the production setup decisions are crucial for the quality of the final solution, due to the fact that they usually make up for the largest part of the total objective. This means that it is highly unlikely to obtain a good quality solution from a partial solution with poor production schedule characteristics, and vice versa.

The ineffectiveness of modifying the production setup decisions during an iterative improvement procedure is also stated in Adulyasak et al. (2014a). Indeed, preliminary experiments have indicated that modifying the production setup in the context of the hybrid local search causes excessive diversification due to the drastic objective value changes. Therefore, in the proposed scheme, the production setup information defined by *PD-Rel* is maintained throughout the optimization process, whereas the production quantities, the delivery quantities, and distribution plans may be modified.

We adapt the production-distribution relaxation of Adulyasak et al. (2014a) to our two-commodity flow formulation. *PD-Rel* ignores the original routing cost matrix and considers aggregated vehicle capacity over each period. Therefore, the complexity of the problem is drastically reduced, and thus, a high-quality solution of the relaxed problem is usually obtained within limited computational time. Similar to Adulyasak et al. (2014a) for each node $i \in V$, an approximation of the routing (transportation) cost σ_i is calculated as shown in (23)

$$\sigma_i = \min \left\{ 2c_{0i}, \min_{j,b \in V, j \neq b} \{c_{ij} + c_{ib}\} \right\} \tag{23}$$

The approximation of the cost of visiting a node i, is the minimum between: i) the sum of the distances of the two closest nodes and ii) twice the distance between i and the depot (direct shipping). This is the most optimistic approximation of the visit cost of each node.

The *PD-Rel* formulation is presented below:

$$\underset{y,z,p,q,I}{\text{minimize}}\, \hat{g} = \sum_{t \in T} \left(u^t p^t + s^t y^t + \sum_{i \in V} h_i I_i^t + \sum_{i \in V_c} \sigma_i z_i^t \right) \tag{24}$$

subject to

$$\sum_{i \in V_c} q_i^t \leq |K| Q \qquad t \in T \tag{25}$$

Constraints (9)–(13), (15), (16), (18)–(20).

The objective function (24) is similar to the objective function of the original problem, except for the fact that the last sum involves the approximated and not the actual transportation costs. Constraints (25) ensure that the total delivered quantity for each period t does not exceed the aggregated capacity of all available vehicles. Note that split deliveries are forbidden by the original *TCF* model, thus the aggregated capacity constraint may generate delivery schedules which are *TCF* infeasible. Therefore, it is possible that for a period t, although the aggregated capacity constraint is satisfied for all vehicles together, the delivered quantities cannot be split in a way that the capacity constraint of each individual vehicle is respected. To ensure feasibility, the RHS of inequality (25) is commonly multiplied by a factor between zero and one. The factor is iteratively reduced until a feasible *TCF* production and distribution schedule is constructed (Absi et al. 2015, Chitsaz et al. 2019). However, this is quite restrictive and sacrifices the quality of the solution for the sake of feasibility. Since our algorithm handles infeasible solutions, we have not adopted such a restricting approach.

3.2. Routing construction heuristic

A PRP partial solution obtained via *PD-Rel* incorporates the production-distribution plans, but lacks actual routing sequences. Routing plans need to be integrated in this partial solution to form a complete *TCF* solution. To do so, a GRASP construction heuristic inspired by Feo and Resende (1995) is applied. The goal is to construct the routes of every period, to implement the predetermined customer visits (z), while respecting all other decision variables also determined by the *PD-Rel*. The GRASP algorithm is executed for each period separately: one independent VRP is solved for each period $t \in T$.

Customer visits are iteratively pushed in the solution. More specifically, a customer visit represented by $z_i^t = 1$ is assigned to period t. Then, all available insertion positions (both in terms of the various routes of period t, as well as the various positions within each of these routes) are evaluated and stored in a candidate list RCL. The cost of each insertion is set to promote solution feasibility: Let c_i be the actual routing cost increase of a candidate insertion. If the insertion is infeasible due to violated vehicle capacity constraint (the *TCF* model contrary to the *PD-Rel* relaxation forbids split deliveries), this actual cost is augmented by a very large positive value M. This is to favor feasible insertions by guaranteeing that the cost of every feasible insertion is better than the cost of any infeasible one. RCL is then sorted in increasing order of the associated insertion costs. The insertion applied is randomly selected among the top RCL_N candidates of the RCL, with all candidates sharing the same selection probability.

3.3. Feasible-infeasible local search

The *FILS* method is the core optimization component of the overall framework. It is responsible for improving the partial solution generated by the *PD-Rel* relaxation and completed by the *GRASP* heuristic. As its name suggests, *FILS* is able to handle both the feasible and infeasible solutions of the *TCF* problem. It is an iterative local search algorithm which explores six different types of neighborhoods which deal with the timing, as well as the routing decisions (Section 3.3.1). In addition, through neighborhood search exploration, it employs an efficient procedure to uniquely define customer delivery quantities when the associated replenishment schedules are modified. A central characteristic of this procedure is that the delivery schedule of a single customer is modified by each tentative local search move. Thus, the delivery quantities for the affected customer are determined by assuming all other customer delivery quantities fixed (Section 3.3.2). To counterbalance this assumption which greedily generates delivery quantities from the perspective of the affected customer, the *FILS* scheme occasionally uses the *QOPT* Linear Programming (LP) component for jointly re-optimizing all delivery and production quantities (Section 3.4.1). Finally, as it will be seen in the computational results section, the most powerful feature of *FILS* is that it dynamically switches between both feasible and infeasible

solutions. In this way, feasible solution regions that could not be examined under computational time restrictions may be explored. Obviously, the optimal solution of a convex polytope is an extreme point/vertex. Therefore, the infeasible space may be used to drive the search out of these extreme points. By using the *FR* MIP component (Section 3.4.2), the infeasible solution is restored and the search is diversified towards a distant feasible region of the solution polytope.

FILS employs three inventory operators and three routing operators, the application of which is described in the following. The boolean variable o^{inv} indicates if the inventory operators are enabled during the neighborhood exploration. In the beginning, only routing operators are used in order to speed up the initial solution improvements (unless the starting solution is infeasible). The inventory operators are switched on whenever the search is moved to the infeasible space and they remain enabled until a new best feasible solution is identified. It should be noted that routing operators affect only the routing aspect of the solution and are much faster than the inventory ones. Therefore, the proposed scheme has a twofold role: it accelerates the search, and promotes the identification of high quality routing plans for given delivery schedule and quantity configurations (since repetitive inventory modifications prevent the algorithm from identifying optimal routing plans).

The oscillation between feasible and infeasible search phases is controlled by two counters. The enter counter i^{in} designates the iteration number at which the search is allowed to enter the infeasible space. This transition is enabled by increasing vehicle capacity to $Q^e = \gamma Q$ (henceforward referred to as effective vehicle capacity). Parameter γ is randomly valued from $[1.02, 1.04, 1.06]$ with 50%, 30% and 20% probability, respectively, each time the infeasible phase begins. The exit counter i^{out} designates the iteration number at which the search is tunneled back to the feasible space by applying *FR* to solution S^{rep} which represents the infeasible phase solution to be repaired. The i^{in} and i^{out} counters are updated as follows, where i denotes the current local search iteration:

$$i^{in} = \begin{cases} i^{out} + \theta^{in}|V| + \mathcal{U}(1,|V|) & \text{if infeasible search is enabled} \\ i + \theta^{in}|V| + \mathcal{U}(1,|V|) & \text{otherwise} \end{cases} \tag{26}$$

$$i^{out} = i^{in} + \theta^{out}|V| + \mathcal{U}(1,|V|) \tag{27}$$

The term $\mathcal{U}(1,|V|)$ denotes the discrete uniform distribution between 1 and $|V|$ and it is used to randomly extend both feasible and infeasible phases, and thus promote diversification. Note that in the computational results, we provide experiments on the optimal value of the θ^{in} and θ^{out} parameters which have been seen to strongly impact the algorithm performance.

The *FR* MIP is solved for a solution S^{rep} obtained during the infeasible space search. During the infeasible space search, we record the best solutions in terms of the total objective, in terms of the routing objective, as well as the incumbent solution. The solution to be repaired S^{rep}, is selected probabilistically among the aforementioned solutions with selection probability of 70%, 15% and 15%, respectively. These probabilities were seen to lead to an effective intensification-diversification balance. The second exact component, the *QOPT* LP is executed every ϵ^{QOPT} local search iterations if the incumbent is feasible.

For instances involving an unlimited vehicle fleet, the algorithm has a dynamic control over the number of vehicles, since it makes sure that exactly one additional empty vehicle is always available. *FILS* terminates after ϵ^t feasible-infeasible space transitions with no solution improvement or after a CPU hour. The outline is presented in Algorithm 2.

The algorithm is initialized by setting the incumbent S' and the best S^* solutions to S_0 and $\emptyset$, respectively. Furthermore, the inventory operators are disabled, unless S_0 is infeasible, in which case the o^{inv} switch is set to true. Additionally, the number of feasible-infeasible space transitions without improvement ϕ and the iteration counters are initialized (line 1). While the number of non-improving feasible-infeasible transitions does not exceed ϵ^t or the time limit is not violated, the following repetitive procedure is applied: At each iteration, the best solution is updated if improved (line 4). In this case,

Algorithm 2 FILS

Input: S_0
1: $S' \leftarrow S_0$, $S^* \leftarrow \emptyset$, $o^{inv} \leftarrow false$, $\phi \leftarrow 0$, $i \leftarrow 0$
2: **while** $\phi < \epsilon^t$ **and** $noTimeViolation()$ **do**
3: **if** $isFeasible(S')$ **and** $Z(S') < Z(S^*)$ **then**
4: $S^* \leftarrow S'$
5: $updateCounter(i^{in}, i^{out})$, $o^{inv} \leftarrow false$, $\phi \leftarrow 0$, $Q^e = Q$
6: **continue**
7: **end if**
8: $S' \leftarrow neighborhoodExploration(S', o^{inv}, Q^e)$
9: **if** $i = i^{in}$ **then**
10: $Q^e \leftarrow \gamma Q$
11: $updateCounter(i^{in})$, $o^{inv} \leftarrow true$
12: **else if** $i = i^{out}$ **then**
13: $S' \leftarrow FR(S^{rep})$
14: $updateCounter(i^{out})$, $\phi \leftarrow \phi + 1$, $Q^e = Q$
15: **else if** i **mod** $\epsilon^{QOPT} = 0$ **and** $isFeasible(S')$ **then**
16: $S' \leftarrow QOPT(S')$
17: **end if**
18: $i \leftarrow i + 1$
19: **end while**
20: **return** S^*

the enter and exit counters are also updated, and the inventory operators are disabled operators polish the solution routing aspects during the next iterations. Of course, the ϕ and the effective vehicle capacity Q^e are reset (line 5) and the algorithm skips to the next iteration (line 6). The incumbent solution is set to the best solution of the neighborhoods explored (line 8). Next, the algorithm checks if the counters dictate to transition from feasible to infeasible space or vice versa (lines 9 and 12). If the infeasibility phase starts, the effective capacity is determined (line 10), the enter counter value is set to designate the beginning of the next infeasible phase, and the inventory operators are enabled (line 11). On the other hand, if the number of the iteration matches the value of i^{out}, the infeasibility phase is terminated (line 12): the FR method repairs S^{rep}, to create the new incumbent solution S' (line 13). Afterwards, the exit, as well as the feasible-infeasible space transitions counters are updated and the effective vehicle capacity is reset to the actual capacity (line 14). Every ϵ^{QOPT} iterations and if the solution is feasible, the $QOPT$ method jointly optimizes the production and delivery quantities (line 16). At the end of each iteration, the counter i is updated (line 18). Finally, upon termination, the best solution obtained S^* is returned (line 20).

3.3.1. Neighborhood exploration

The proposed local search framework employs six operators and operated according to the best admissible scheme (i.e., neighborhoods are explored and the search is moved to the highest quality admissible neighbor identified). Note that the effective vehicle capacity is used to decide on move admissibility.

The operators deal either with the timing aspect of the delivery plans (inventory operators) or the routing decisions of each period t (routing operators). The inventory operators are: the *Period Deletion*, the *Period Insertion* and the *Period Relocation*. Obviously, when a delivery service is introduced in a customer schedule, a customer visit is necessary. To identify this additional service position, all routing positions are examined and the one which is admissible and minimizes the additional routing cost is selected. The aforementioned operators affect the timing of replenishment services, and thereafter, the delivery quantities, as well as the inventory levels. Regarding the routing operators, the

well-known intra- and inter-route operators *Customer Relocation*, *Customer Swap* and *2-Opt* are used. The neighborhood structures of the aforementioned operators are summarized below:

- *Period Deletion*: includes all solutions derived by removing a visit of any customer i from any period t, such that $\tilde{z}_i^t = 1$ $(O(|V_c||T|))$.
- *Period Insertion*: includes all solutions derived by inserting a new visit for any customer i at any period t, such that $\tilde{z}_i^t = 0$ $(O(|V_c||T||K|))$.
- *Period Relocation*: includes all solutions derived by relocating an existing visit of customer i from period t_1 such that $\tilde{z}_i^{t_1} = 1$, to period t_2 such that $\tilde{z}_i^{t_2} = 0$ $(O(|V_c||T||K|))$.
- *Customer Relocation*: includes all solutions derived by changing the service position of any customer i at any period t, to every other available service position of the same period t $(O(|V_c|^2|T||K|))$.
- *Customer Swap*: includes all solutions derived by exchanging the service positions of customers i and j visited at any period t $(O(|V_c|^2|T||K|))$.
- *2-Opt*: includes all solutions derived by applying the 2-opt operator to any edge pair traversed at any period t $(O(|V_c|^2|T||K|))$.

Note that a move is considered admissible, if it respects the problem constraints and respects the following tabu policy (Zachariadis et al. 2015): When a *Period Deletion*, or a *Period Relocation* move is applied to a solution S, so that the service of customer i is removed from period t, we set a threshold value $\rho_i^t = Z(S)$. The insertion of any customer i into period t leading to solution S' is only allowed when $\rho_i^t > Z(S')$. Similarly, when a move eliminates a set of edges $F \subseteq E$ at period t from a solution S, we set a threshold value $\rho_{ij}^t = Z(S), \forall (i,j) \in F$. Local search moves introducing any edge $(i,j) \in F$ at period t leading to solution S', are only allowed when $\rho_{ij}^t > Z(S')$. It is worth to highlight that the thresholds described above can be used to diversify the local search process, and for this reason, they are periodically set to infinity each time a new best solution is found, a number of iterations ϵ^ρ is completed, or the search transitions from infeasible to feasible space.

Since all problem aspects are intertwined, it is not possible to optimize against one aspect of the problem, without modifying the others. However, to employ local search on the delivery schedules of customers, a systematic way for determining customer delivery quantities is necessary. To do so, when inserting (or removing) any customer i into (or from) period t, the customer delivery quantities for all time periods are decided, so that the sum of the depot and customer i inventory cost is minimized, under the assumption that all other decisions are fixed. The procedure that determines delivery quantities, when the replenishment schedule of a customer is modified is given in section 3.3.2. Note that changes in the routing component of the objective function can be assessed in $O(1)$, since they involve the replacement of a constant number of edges. On the contrary, the complexity of the proposed delivery quantity determination (necessary for inventory cost change evaluation) is $O(|T|)$. The neighborhood exploration returns to the *FILS* the best tabu feasible solution after exploring all defined operators.

3.3.2. Customer inventory calculation

A challenging aspect of LS approaches for problems with continuous decisions variables, such as the PRP, is their quantization. During the development of local search framework for the PRP, the management and the calculation of the customer delivered quantities (q) is not a straightforward task. The PRP literature contains some ways to overcome this obstacle. One of the most common, is to solve an exact subproblem to identify the delivered quantities (e.g., Minimum Cost Flow problem (Adulyasak et al. 2014a)). This approach requires solving an exact subproblem for evaluating each solution. Therefore, it is difficult to be integrated within neighborhood exploration, since it would lead

to excessive computational times. Another approach is to adopt the well-known OU replenishment policy and therefore the common assumption that the quantity delivered to a customer at the time of visit, should be equal to the quantity required to reach the maximum stock level. This approach, despite being computationally efficient, is practically unrealistic and significantly deteriorates the quality of the final solutions (Archetti et al. 2011). This is particularly the case for instances that involve high maximum customer capacity, and thus just a few customers may fully occupy the available capacity of a vehicle, leading to a very inefficient routing and possibly causing shortages to other customers. In addition, since the unit holding cost of customers and the depot can significantly differ, the OU policy may lead to increased total inventory costs. To avoid the disadvantages of the aforementioned approaches, we propose an efficient computational method for calculating the delivered quantities to a customer, assuming all other replenishment variables (delivery schedule of the customer involved, delivery schedule and quantities for all other customers) are fixed. This method is used when one of the operators modifies the visit schedule (z_i) of a customer. We distinguish two separate cases according to the relationship of customer i and depot unit holding costs (h_i, h_0):

- *Case 1.* $h_i > h_0$: Inventory accumulations with early deliveries of product to the customer lead to increased total holding costs. Therefore, it is preferable to preserve this stock in the depot location. In other words, customer i is preferably served as late as possible to minimize the inventory costs incurred.

- *Case 2.* $h_i \leq h_0$: Inventory accumulations with early deliveries of product to the customer lead to decreased total holding costs. Therefore, it is preferable to preserve this stock in the customer location. In other words, customer i is preferably served as early as possible to minimize the inventory costs incurred.

On this basis, we propose two delivery quantities calculation algorithms for Case 1 and Case 2 customers, respectively. When the neighborhood exploration is performed, given the incumbent solution S and the visit schedule of customer i ($\tilde{z}_i$), one of the aforementioned algorithms is applied to calculate the delivery quantities that minimize the total inventory costs. Note that, each one of the two methods uniquely defines a set of delivered quantities for the affected customer. For the exceptional case of instances, where $h_i = 0 \quad \forall i \in V_c$, we randomly classify each customer to one of the two cases with equal probability. This is to balance the delivery volumes through the planning horizon. Both methods can be executed with a single pass of the periods of the planning horizon ($O(|T|)$), thus they can be efficiently executed during the neighborhood exploration stage of the *FILS* algorithm.

To facilitate presentation, we introduce some auxiliary variables for any given solution: Auxiliary variable q_i^- runs through the planning horizon. For any $t \in T$, q_i^- represents the deficit quantity for customer i equal to the current stock-out. In other words, q_i^- is the minimum delivery quantity that must be delivered to the customer i at period $t \in T$ to avoid stock-outs. On the contrary, the variable q_i^+ denotes the current customer surplus quantity (current inventory) or in other words the product quantity that can be taken out from customer i without stock-out. Let $\hat{I}_0^t$ be a equal to the depot inventory slack of period $t \in T$. It should be highlighted, that all customer currently decided quantities $\tilde{q}_i^t \forall t \in T$ are reallocated to the depot upon the beginning of the described procedure. Similarly, let $\hat{Q}_r^t$ be the remaining capacity (slack) of vehicle travelling route r at period t.

For the Case 1 customers, the goal is to deliver quantities that minimize the customer holding cost, since it is preferable to preserve stock in depot rather than the customer location. To do so, for each period the minimum of five different quantities must be identified:

1. Depot slack $\hat{I}_0^t$
2. Vehicle slack $\hat{Q}_r^t$
3. Maximum customer capacity L_i
4. Minimum quantity to avoid stock-out q_i^-

5. Total remaining demand for the remaining planning horizon $\sum_{t \in T} d_i^t - I_i^0 - \sum_{\tau=t+1}^{|T|} q_i^\tau$

Note that, for customers of Case 1, deliveries should be made as late as possible. Thus, the method traverses the planning horizon backwards. At any iteration, if the delivery quantity is negative, the evaluated visit schedule is considered infeasible. Algorithm 3 presents the replenishment quantity calculation for Case 1 customers.

Algorithm 3 Customer q_i calculation (Case 1)

Input: S, i
1: $q_i^- \leftarrow 0$
2: **for** $t \leftarrow 1$ to $|T|$ **do**
3: $\quad \hat{I}_0^t \leftarrow \tilde{I}_0^t + \sum_{\tau=1}^{t} \tilde{q}_i^\tau$
4: **end for**
5: **for** $t \leftarrow |T|$ to 1 **do**
6: $\quad q_i^t \leftarrow 0$
7: $\quad q_i^- \leftarrow q_i^- + d_i^t$
8: $\quad$ **if** $\tilde{z}_i^t = 1$ **then**
9: $\quad\quad r \leftarrow \zeta(i,t)$
10: $\quad\quad \hat{Q}_r^t \leftarrow Q^e - \sum_{j \in V_c^r \setminus \{i\}} \tilde{q}_j^t$
11: $\quad\quad q_i^t \leftarrow \min(\hat{I}_0^t, \hat{Q}_r^t, L_i + d_i^t, q_i^-, \sum_{t \in T} d_i^t - I_i^0 - \sum_{\tau=t+1}^{|T|} q_i^\tau)$
12: $\quad\quad q_i^- \leftarrow q_i^- - q_i^t$
13: $\quad\quad \hat{I}_0^\tau \leftarrow \hat{I}_0^\tau - q_i^t \quad \forall \tau \in [t, |T|]$
14: $\quad$ **end if**
15: **end for**
16: **return** q_i

The procedure starts with the initialization of the customer deficit quantity (line 1) and the calculation of the depot inventory slack (lines 2-4). The depot slack considers the inventory for period t and all the quantities from $\tau = 1$ until t horizon that have been assigned to the customer. Therefore, we temporarily assign all quantities of customer i back to the depot. Lines 5-14 represent the loop that starts at period $t = |T|$ and ends at period $t = 1$. Line 6 initializes the delivery quantity to zero for the case that no customer delivery takes places at the specific period. Line 7 increases the temporary deficit quantity by customer demand at t (product consumption). If customer is replenished at t, the route r serving the customer is retrieved (line 9). On line 10, the vehicle capacity slack $\hat{Q}_r^t$ is calculated as by considering the delivery quantity for all other customers (the $\tilde{q}_i^t$ has been assigned back to depot). The new quantity for period t is set to the minimum of the vehicle capacity slack, the depot inventory surplus, the maximum allowed customer delivery quantity, the deficit stock-out quantity and the total remaining demand (line 11). After obtaining, the new delivery quantity the deficit quantity and the depot inventory surplus are updated accordingly (lines 12-13). Finally, the new delivery quantities for customer i are returned.

For the Case 2 customers, the goal is to front-load the delivery schedule. This is to deliver large product quantities to the customers as early as possible, to minimize the inventory holding cost incurred by storing products at the depot. To do so, for each period the minimum of four different quantities must be identified:

1. Minimum depot slack $\hat{I}_0^{min}$ for the remaining periods
2. Vehicle slack $\hat{Q}_r^t$
3. Current remaining customer capacity $L_i - q_i^+$
4. Total remaining demand for the remaining planning horizon $\sum_{t \in T} d_i^t - I_i^0 - \sum_{\tau=1}^{t-1} q_i^\tau$

Since it is preferable to preserve stock in depot rather than the customer location, deliveries should be made as early as possible and therefore, the planning horizon is traversed forward. At any iteration, if the delivery quantity is negative, the evaluated visit schedule is considered infeasible. Algorithm 4 presents the delivery schedule calculation for Case 2 customers.

Algorithm 4 Customer q_i calculation (Case 2)

Input: S, i
1: $q_i^+ \leftarrow 0$
2: **for** $t \leftarrow 1$ to $|T|$ **do**
3: $\quad \hat{I}_0^t \leftarrow \tilde{I}_0^t + \sum_{\tau=1}^{t} \tilde{q}_i^\tau$
4: **end for**
5: **for** $t \leftarrow 1$ to $|T|$ **do**
6: $\quad q_i^t \leftarrow 0$
7: $\quad q_i^+ \leftarrow q_i^+ - d_i^t$
8: $\quad$ **if** $\tilde{z}_i^t = 1$ **then**
9: $\quad\quad r \leftarrow \zeta(i,t)$
10: $\quad\quad \hat{Q}_r^t \leftarrow Q^e - \sum_{j \in V_c^r \setminus \{i\}} \tilde{q}_j^t$
11: $\quad\quad \hat{I}_0^{min} \leftarrow \min\left(\hat{I}_0^\tau \quad \forall \tau \in [t+1, |T|]\right)$
12: $\quad\quad q_i^t \leftarrow \min(\hat{I}_0^{min}, \hat{Q}_r^t, L_i - q_i^+, \sum_{t \in T} d_i^t - I_i^0 + \sum_{\tau=1}^{t-1} q_i^\tau)$
13: $\quad\quad q_i^+ \leftarrow q_i^+ + q_i^t$
14: $\quad\quad \hat{I}_0^\tau \leftarrow \hat{I}_0^\tau - q_i^t \quad \forall \tau \in [t, |T|]$
15: $\quad$ **end if**
16: **end for**
17: **return** q_i

The algorithm is similar to the one described in Algorithm 3 with a few differences. Firstly, at each period the consumption is subtracted from the current surplus (inventory) (line 7) and the current surplus is increased by the decided quantity (line 13). Additionally, we use a look-ahead minimum inventory calculation for the depot. In particular, we calculate the minimum depot inventory surplus for all the remaining periods $\hat{I}_0^{min}$ (line 11). This is a upper bound for the customer delivery, as any quantity exceeding this bound would cause a depot inventory stock-out during one or more of the following periods (but not during the current period). Finally, the calculation of the total remaining demand until the end of the time horizon considers periods 1 to $t-1$, due to the fact that the planning horizon is traversed from start to end.

3.4. Mixed integer linear programming components

The *FILS* incorporates two exact components: i) the *QOPT* which simultaneously optimizes the continuous variables of the delivery, production and inventory quantities, and ii) the *FR* which restores the feasibility of a capacity infeasible solution. Both exact components are presented below.

3.4.1. Quantity optimization method QOPT

The quantity optimization method *QOPT* is responsible for optimizing the production and delivery quantities in pursuit of total inventory costs minimization, given the $\tilde{y}^t$, $\tilde{z}_i^t$, $\tilde{x}_{ij}$ and $\tilde{f}_{ij}^t$ values of an incumbent solution S. *QOPT* is responsible for counterbalancing the assumption of fixed decision when a customer delivery quantities are calculated during neighborhood exploration (see Section 3.3.2). To do so, it jointly optimizes all delivery and production quantities. The quantity optimization

sub-problem tackled by $QOPT$ is as follows:

$$\underset{p,q,I}{\text{minimize}}\, g_1 = \sum_{t\in T}\left(u^t p^t + \sum_{i\in V} h_i I_i^t\right) \tag{28}$$

subject to

$$\sum_{i\in V_c^r} q_i^t \leq Q \qquad r \in R \quad t \in T \tag{29}$$

(10)–(13) and the variable bounds (9), (18), (20).

The objective function (28) is made up of the production and inventory terms of the original TCF objective function. Note that, routing and setups decisions are not modified by the $QOPT$ model. Constraints (29) ensure that the vehicle capacity is not violated. Finally, since y variables are unaffected by the model, the RHS of (9) is a constant stronger variable bound and thus, replaces (19).

3.4.2. Feasibility repair method FR

Similarly, to the $QOPT$, the FR, optimizes the production and delivery quantities of a given solution S. However, FR can also insert, remove and relocate customer visits. In addition, as earlier stated $QOPT$ is applied only to feasible solutions, whereas FR is applied to both feasible, as well as infeasible solutions. Therefore, the underlying model faced by FR extends the $QOPT$ sub-problem presented above. To ensure a fast performance, all customer removal and insertions costs are approximated. The number of customer visits that can be added, removed or relocated is bounded, to ensure that: i) the repaired solution is similar to the original one, and ii) the approximation impact on the solution quality is kept to manageable levels. To eliminate capacity infeasibilities, the sub-problem faced by FR penalizes vehicle capacity violations. Note that, FR not only restores the feasibility of a solution in a near optimal way, but also diversifies the solution by applying several routing modifications, contrary to the $FILS$ that can only apply minor routing changes at each iteration. The use of approximation costs instead of the original ones, further enhances the model ability to diversify the search by performing a leap in the solution space. Every repaired solution is further optimized by solving the TSP associated with each route of each period to balance the approximation trade-off.

Let ξ denote the per unit penalty for excess vehicle load. The continuous variable e_r^t denotes the excess load of route r at period t over the actual capacity Q and up to the effective capacity Q^e. Obviously, if any $e_r^t > 0$, route r at period t is infeasible. Also, let constant Δ_i^t denote the minimal insertion cost of customer i in any of the routes of period t. Note that, this is defined even for customers already visited in period t, since the model allows them to be relocated to other routes, or even within the same route. Given the triangular inequalities, the cheapest insertion for any customer cannot be in an empty route. For cases, where the number of vehicles is not limited, this fact does not allow the algorithm to exploit routing alternatives, which involve additional vehicles. In addition, this leads to routes serving many customers which are difficult to be repaired. For these cases, if FR cannot restore feasibility, it is executed again with the following modification: the "cheapest" insertion of a customer is set to an empty route with a 25% probability. Let function $\zeta' : (V_c, T) \rightarrow R$ return the route of cheapest insertion of customer i in period t. Similarly, constants Λ_i^t are the savings of removing customer i from period t and are defined only when $\tilde{z}_i^t = 1$.

Additionally, let us introduce binary variables δ_i^t and λ_i^t equal to 1 iff customer i is added or removed from period t, respectively. Variables λ_i^t can be only defined for customers such that $\tilde{z}_i^t = 1$. To avoid non-linear constraints, we introduce delivery quantity variables q'^t_i representing the delivery quantity of relocated customers (moved to another service position of the same period). Therefore, if a customer i is relocated (by being removed and re-inserted), q'^t_i is used instead of q_i^t for the delivery quantity. Finally, parameter a bounds the number of insertions and deletions per route. The value

of a depends on the relationship of the objective of the solution to be repaired $Z(S^{rep})$ and the best objective $Z(S^*)$ obtained so far: if $\frac{|Z(S^{rep})-Z(S^*)|}{Z(S^*)} < 0.01$, then $a = a^-$ and otherwise $a = a^+$. This is to control the structural changes caused by the MIP to the solution, depending on the distance of the infeasible solution objective from the best known objective. For conciseness, we denote the constant upper bound of the delivery quantity to customer i at period t, $\overline{q}_i^t$, as shown in equation (30):

$$\overline{q}_i^t = min\left\{ L_i + d_i^t, Q, \sum_{t'=t}^{|T|} (d_i^{t'}) \right\} \qquad i \in V_c \quad t \in T \tag{30}$$

Below, the *FR* MIP is presented:

$$\underset{\delta,\lambda,p,q,q',I}{\text{minimize}}\, g_2 = \sum_{t\in T}\left(u^t p^t + \sum_{i\in V} h_i I_i^t + \sum_{r\in R} \xi e_r^t + \sum_{i\in V_c} \Delta_i^t \delta_i^t + \sum_{i\in V_c:\tilde{z}_i^t=1} \Lambda_i^t \lambda_i^t \right) \tag{31}$$

subject to

$$\sum_{i\in V_c^r} q_i^t + \sum_{i\in V_c:\zeta'(i,t)=r,\tilde{z}_i^t=0} q_i^t + \sum_{i\in V_c:\zeta'(i,t)=r,\tilde{z}_i^t=1} {q'}_i^t \le Q^e \qquad r \in R \quad t \in T \tag{32}$$

$$e_r^t \ge \sum_{i\in V_c^r} q_i^t + \sum_{i\in V_c:\zeta'(i,t)=r,\tilde{z}_i^t=0} q_i^t + \sum_{i\in V_c:\zeta'(i,t)=r,\tilde{z}_i^t=1} {q'}_i^t - Q \qquad r \in R \quad t \in T \tag{33}$$

$$\delta_i^t \le \lambda_i^t \qquad i \in V_c \quad t \in T : \tilde{z}_i^t = 1 \tag{34}$$

$$q_i^t \le (1-\lambda_i^t)\overline{q}_i^t \qquad i \in V_c \quad t \in T : \tilde{z}_i^t = 1 \tag{35}$$

$${q'}_i^t \le \delta_i^t \overline{q}_i^t \qquad i \in V_c \quad t \in T : \tilde{z}_i^t = 1 \tag{36}$$

$${q'}_i^t \le \lambda_i^t \overline{q}_i^t \qquad i \in V_c \quad t \in T : \tilde{z}_i^t = 1 \tag{37}$$

$$q_i^t \le \delta_i^t \overline{q}_i^t \qquad i \in V_c \quad t \in T : \tilde{z}_i^t = 0 \tag{38}$$

$$\sum_{i\in V_c:\zeta'(i,t)=r} \delta_i^t \le a \qquad r \in R \quad t \in T \tag{39}$$

$$\sum_{i\in V_c^r} \lambda_i^t \le a \qquad r \in R \quad t \in T \tag{40}$$

$$\delta_i^t \in \{0,1\} \qquad i \in V_c \quad t \in T \tag{41}$$

$$\lambda_i^t \in \{0,1\} \qquad i \in V_c \quad t \in T : \tilde{z}_i^t = 1 \tag{42}$$

$$0 \le e_r^t \le Q^e - Q \qquad r \in R \quad t \in T \tag{43}$$

$$0 \le {q'}_i^t \le \overline{q}_i^t \qquad i \in V_c \quad t \in T : \tilde{z}_i^t = 1 \tag{44}$$

(10)–(13) and variable bounds (9), (18), (20).

Objective function (31) minimizes the total unit production and holding cost, the excessive capacity penalty, as well as the costs for inserting and removing customers. Constraints (32) ensure that the augmented total capacity Q^e (allowing infeasibility) of any vehicle will not be exceeded by the sum of the existing customers delivery quantities (first term), the customers inserted delivery quantities (second term), as well as to customers relocated to the associated route (third term). In a manner similar, constraints (33) sets the augmented vehicle capacity slack variable for excess load. Next,

constraints (34) guarantee solution consistency by allowing the addition of an existing customer i in period t iff this customer is removed from its current delivery position (relocation). For existing customers (i.e., $\tilde{z}_i^t = 1$), constraints (35) ensure that $q_i^t = 0$ if customer i is removed. Note that, if customer i is relocated to new delivery position of the same period t, $q_i^t = 0$ and the delivery quantity associated with the new delivery position is represented by a non-zero $q'^t_i = 0$. Constraints (36) and (37) ensure that the relocated delivery quantity q'^t_i may be positive only if customer i is both inserted and removed (i.e., relocated). For any customer i inserted in any period t (i.e., $\tilde{z}_i^t = 0$), constraint (38) sets the associated delivery quantity to q_i^t. Inequalities (39) and (40) limit the number of insertions and deletions per route period to a. Additionally, (41)–(44) provide the bounds of the new variables introduced. Note that, the removal, as well as the relocated delivery quantity variables, are defined iff customers are not already routed at the associated period. Finally, since y variables are unaffected by the model, the RHS of (9) is a constant stronger variable bound and thus, replaces (19). It should be noted that, for the interested reader, a substantially more compact and comprehensible formulation of the aforementioned model is provided in Appendix B. This model does not make use of the q' variables and thus is non-linear. Preliminary experiments with the non-linear model demonstrated significantly inferior performance compared to the linear one.

3.5. Motivation of the proposed algorithm structure

This section summarizes the course of the proposed algorithm development and provides the motivation behind the main algorithmic components. Our central goal was to build an algorithm capable of generating high quality PRP solutions within acceptable computational times, even for the largest scale instances considered in the literature. Initially, the local search framework. which has proven effective for vehicle routing problems, was selected as the backbone of the proposed algorithm. To avoid cycling the well-tested promise mechanism of Zachariadis et al. (2015) was integrated into the local search. Preliminary experiments with operators that modified the production setup decisions (y variables) through the algorithm execution resulted in a repetitive drastic modification of the objective function, so that the local search becomes excessively diversified. This is mainly because for most of the test instances the production setup costs make up for the largest share of the overall objective, as also discussed by Adulyasak et al. (2014a). To overcome this issue, a two phase approach was decided: in the first phase a production-distribution relaxation is solved, whereas in the second phase, all decisions are optimized given the production setup schedule determined in the first phase. Experiments demonstrated that a high quality production plans lead to high quality final solutions, thus there is no need to iterate through different production plans.

Regarding the second phase which constitutes the core of our optimization algorithm, one of the basic challenges faced was how to handle the continuous production and delivery quantities (p and q variables, respectively) through the local search. Initially, we designed the $QOPT$ method to jointly re-optimize the production and delivery quantities, given the routing and visit decisions of the incumbent solution. However, the computational overhead of repetitively calling the $QOPT$ method when the local search move evaluation takes place led to excessive computational times. As a result, we focused on heuristic mechanisms. On this basis, the algorithms for determining the customer delivery quantities (Algorithms 3 and 4) were developed. Their goal is to optimize the delivery quantities of a customer, given the corresponding visit schedule and assuming that all other decisions cannot be modified. These algorithms were efficient to be integrated within the neighborhood explorations. To add some flexibility on the delivery quantities, the $QOPT$ method is periodically used to override the decisions made by the aforementioned algorithms. Experiments with the above described components demonstrated satisfactory performance, but still worse than the majority of the best known PRP solution scores. By analyzing the objective curves of the solutions visited through the local search, we realized that some extra diversification should be introduced to drive the search towards unexplored regions. To do so, we experimented with allowing violations of the vehicle capacities. We observed

that the increase of vehicle capacities let additional customers to be served by already used routes and this resulted in significant routing and inventory savings. Thus, an appropriate feasibility restoration mechanism would be adequate to introduce drastically modified and at the same time promising solution features. Our first attempt with a heuristic feasibility restoration method was rather ineffective for the following two reasons: i) feasibility cannot be straightforwardly guaranteed and ii) even when it is achieved, the solution quality might be excessively deteriorated. Thus, we decided to proceed with the development of a mathematical programming component (*FR* method) for effectively restoring feasibility. Our final goal was to decide which of the infeasible solutions visited through the infeasible phase had to be restored by the *FR* method. The algorithm demonstrated the best performance when the solution was randomly selected among i) the lowest total cost solution observed through the infeasibility phase, ii) the lowest routing cost solution observed through the infeasibility phase and iii) the incumbent infeasible solution. This can be attributed to the fact that diverse solution features are built and introduced to the incumbent solution.

4. Computational results

This section summarizes computational experiments and results obtained by the proposed matheuristic on well-known PRP benchmark data sets. Initially, the benchmark data sets are introduced. In addition, state-of-the-art methodologies proposed for the PRP variant are briefly described. Then, tuning experiments on key algorithm parameters are reported. To motivate the proposed algorithm design, detailed experiments on the infeasible space contribution are provided. Finally, we draw analytic comparisons of the proposed algorithm results against state-of-the-art PRP methodologies.

The proposed algorithm (*HISM*) was coded in C#. All exact components (i.e., *PD-Rel*, *QOPT*, *FR* and *TSP*) were tackled by Gurobi 9.0.2. The overall algorithm (both local search and exact components) were executed as a single thread to enable comparisons. The computational experiments were carried out on an Intel Core i7-7700 3.60 GHz CPU with 16 gigabyte RAM x64 Windows 10 machine.

4.1. Benchmark data sets

Tuning experiments and performance comparisons were carried out on the two most widely used benchmark data sets: i) a data set of small-medium PRP instances, named *A* (Archetti et al. 2011) and ii) a data set of large scale PRP instances data set, named *B* (Boudia et al. 2007). Data set *A* consists of *A1*, *A2* and *A3* subsets involving six time periods and 14, 50, and 100 customers, respectively. The per period demand is constant and the customer starting inventory is non zero. Plant production and inventory capacity are set to infinity, whereas no plant starting inventory is considered. Each of the three subsets includes 480 problems: five instances for each of 96 instance types ($96 \times 5 = 480$). Subset *A1* considers one vehicle, whereas *A2* and *A3* assume unlimited vehicles. The 96 instance types are further categorized into four main classes: Class I (1-24) corresponds to the instances with the basic configuration of production, inventory and transportation costs. Class II (25-48) and Class III (49-72) involve high production costs (i.e., base unit production costs are multiplied with 10) and transportation variable costs (i.e., node coordinates multiplied by 5), respectively. Finally, Class IV (73-94) considers no inventory holding cost for customers.

The second benchmark data set *B* is significantly more challenging: it consists of three subsets *B1*, *B2* and *B3*, each one with 30 large or very large instances of 50, 100, and 200 customers and 20 time periods. Unlike data set *A*, the maximum production capacity per period, plant inventory capacity, and maximum number of vehicles are restricted. In addition, contrary to data set *A*, where $d^t = d$ for $\forall t \in T$, the customer demand is time-varying and no inventory holding cost is considered for customers. According to Boudia et al. (2007), production in period t becomes available not earlier that period $t + 1$, but no inventory costs are incurred for period t. Thus, two clarifications need to be

made: firstly, the demand at period $t = 1$ has to be satisfied from the initial inventory at the plant. We adopt the common literature practice which sets the initial inventory at the plant equal to the total customer demand in the first period. As earlier stated no inventory holding cost is incurred for the initial inventory between periods 0 and 1. The second clarification is that according to our model, the quantity produced for any period t is available directly, whereas according to Boudia et al. (2007) the quantity is available at $t + 1$ but without incurred holding costs. To represent this business setting, we adopt the workaround of Adulyasak et al. (2014a), which sets $y^1 = 0$, enforcing $p^1 = 0$, i.e., the production made available in period 1 is equal to zero and thus the total customer demand is satisfied by the initial inventory at the plant. In this way, the quantity produced at any period $t > 1$ for our model, would have been produced at period $t - 1$ in the case of Boudia et al. (2007) without extra holding costs incurred. Therefore, the exact same operational scenario of Boudia et al. (2007) is solved and thus comparisons on the instances of data set B are enabled.

Overall, the characteristics of data sets A and B which comprise of 1530 PRP instances are demonstrated in Table 1.

Table 1: Overview of the Benchmark Instances (Adulyasak et al. 2014a)

Benchmark instance set	Archetti et al. (2011)			Boudia et al. (2007)		
	A1	A2	A3	B1	B2	B3
No. of instances	480	480	480	30	30	30
No. of periods	6	6	6	20	30	30
No. of customers	14	50	100	50	100	200
No. of trucks	1	∞	∞	5	9	13
Demand	C	C	C	V	V	V
Production capacity	∞	∞	∞	C	C	C
Plant inventory capacity	∞	∞	∞	C	C	C
Customer inventory capacity	C	C	C	C	C	C
Initial inventory at plant	0	0	0	V	V	V
Initial inventory at customers	V	V	V	0	0	0
Vehicle capacity	C	C	C	C	C	C

V: Varying, C: Constant, ∞: Unlimited

4.2. State-of-the-art approaches

The results of the proposed method are compared against the results of state-of-the-art PRP approaches which are summarized in Table 2.

Table 2: Benchmark algorithms for A and B data sets

Reference	Approach	Threads	Solver	Data sets
Adulyasak et al. (2014a)	MIP-based ALNS	Default	CPLEX 12.2	*A*, *B*
Absi et al. (2015)	Iterative MIP heuristic	Default	CPLEX 12.1	*A2*, *A3*, *B*
Solyalı and Süral (2017)	MIP-based heuristic	12	CPLEX 12.5	*A*, *B*
Russell (2017)	MIP-based heuristic	Default	CPLEX 12.6	*A2**, *A3**, *B*
Qiu et al. (2018a)	VNS/MIP	1	CPLEX 12.6	*A*, *B*
Chitsaz et al. (2019)	Decomposition Matheuristic	1	CPLEX 12.6	*A*, *B*
Li et al. (2019)	MIP-based heuristic	Default	CPLEX 12.6	*A*, *B*
Avci and Yildiz (2019)	MIP-based Local Search	Default	CPLEX 12.6	*A*
Schenekemberg et al. (2021)	ALNS-BnC hybrid	6	Gurobi 8.1	*B1*, *B2*
This paper	MIP-based Local Search	1	Gurobi 9.0.2	*A*, *B*

*: the author solves only the first 96 problems

Due to the complexity of PRP, the most effective approaches are hybrid combinations of heuristic and exact components. Adulyasak et al. (2014a) (ACJ) propose an ALNS scheme with exact optimization components, whereas Absi et al. (2015) (AADF) develop a two-stage approach that solves the lot-sizing

and routing subproblems iteratively, feeding information of the former stage to the latter, and vice versa. Similarly, Solyalı and Süral (2017) (SS) propose a one-shot five stage heuristic integrated with exact components that tackles a sequence of overlapping problems. Russell (2017) (RR) uses the concept of predetermined routes and seed routes and solves a set-covering problem. The lot-sizing and routing subproblems are tackled with exact components and Tabu Search, respectively. Qiu et al. (2018a) (QWXFP) employ a skewed general variable neighborhood search and guided variable neighborhood descent, whereas Chitsaz et al. (2019) (CCJ) propose a three-phase decomposition matheuristic that relies on the iterative solution of different subproblems. Li et al. (2019) (LCCZ) suggest a mathematical programming based heuristic made up of a two-phase iterative method, a repairing strategy, and a fix-and-optimize procedure to find near-optimal solutions. Avci and Yildiz (2019) (AY) propose an iterative MIP-based local search procedure. Most recently, Schenekemberg et al. (2021) (SSPGC) present both an ALNS algorithms enhanced by MIPs, as a well as a parallel hybrid ALNS and BnC algorithm.

4.3. Setting of general parameters

Preliminary runs on PRP data sets, led to the following decisions regarding general *HISM* parameters. The RCL length of the GRASP heuristic is set to $RCL_N = \lceil \frac{min(|V|,50)}{5} \rceil$, whereas $QOPT$ is used every $\epsilon^{QOPT} = 200$ iterations. The promise reinitialization is set to $\epsilon^{\rho} = |V|$ throughout the experiments, whereas each one of the $\epsilon^{r} = 100$ restarts stops after $\epsilon^{t} = 20$ consecutive feasible-infeasible space transitions without solution improvement. For the $QOPT$ and TSP, the default solver settings are used with the exception of lazy constraints addition used for the TSP. On the opposite, for FR, we impose a maximum time of 40 seconds, MIPFocus is set to one to quickly generate feasible solutions and the per unit penalty is set to $\xi = 10,000$, to promote the infeasibility restoration. The time limit for the *PD-Rel* relaxation is set to three minutes for data set A, whereas the time limit is set to three, five and five minutes for *B1*, *B2* and *B3*, respectively. Also, note that in order to ensure acceptable computational times, for the very large scale *B3* instances, the local search algorithm examines only two randomly selected neighborhoods instead of all six for every solution.

4.4. Entering and exiting the infeasible space

Some of the most crucial parameters of the proposed *HISM* algorithm are the θ^{in} and θ^{out}, which jointly determine the number of iterations before entering and before exiting the infeasible space. To tune these parameters, we selected a subset of problems from the more challenging and realistic data set B. Two problems for each customer size (i.e., 50, 100, 200) are selected according to the following criterion: maximization of the gap between the best solutions reported (see Table 2). This can be interpreted as a measure of the complexity of an instance. The new data set $B' \subset B$ contains the following six problems: instances 9 and 30 of *B1*, instances 20 and 27 of *B2* and instances 9 and 30 *B3*. Table 3 summarizes the results of the experiment for θ^{in} (rows) and θ^{out} (columns) ranging within $[1-5]$ and $[1-4]$, respectively. Each entry represents the average percentage gap for all six instances. The gap of each problem is the relative difference of the objective achieved by the corresponding parameter pair and the best objective achieved over all tested pairs. Note that for the sake of more meaningful comparisons, the considered objective ignores the production setup cost which is identical for all parameter pairs.

Table 3: Tuning of θ^{in} and θ^{out} parameters

$\theta^{in} \backslash \theta^{out}$	1	2	3	4	Avg.
1	0.360	0.294	0.148	0.199	0.250
2	**0.068**	0.269	0.319	0.239	**0.224**
3	0.350	0.345	0.350	0.350	0.349
4	0.537	0.539	0.532	0.539	0.537
5	0.650	0.650	0.650	0.650	0.650
Avg.	**0.393**	0.419	0.400	0.395	

Low θ^{in} values of up to 3 seem to be consistently better. Therefore, frequent applications of the infeasible local search phase improves the algorithmic performance. The value $\theta^{in} = 2$ yields the best results on average. In terms of the θ^{out} parameter, the iterations spent in the infeasible space search seem to have a negligible impact on the quality of the final solution. This is expected, as the solution to be repaired is either the best infeasible (in terms of total or routing objective) found during the infeasible space search, or the incumbent, and therefore it is not heavily affected by the number of iterations in infeasible space search. The synergy of the two parameters is shown for pairs 1-3 and 1-4: high exit counter values seem to compensate for the low number of iterations spent in the feasible space. The best performing value pair is $\theta^{in} = 2$ and $\theta^{out} = 1$ and is used for all algorithm runs.

4.5. Effect of allowed per modifications on infeasible tunneling

A very important algorithm feature is the flexibility provided to the *FR* MIP, to restore the feasibility of a solution. High flexibility allows more structural differences between the infeasible and the restored feasible solution. Parameter a determines the maximum number of per route insertions and deletions, thus it defines the flexibility provided to *FR* method. As mentioned above, when the objective of the selected infeasible solution is less than 1% worse than the current restart best feasible solution objective identified so far, a is set to a^-. Otherwise, $a = a^+$. To decide on the a^- and a^+ values, tuning experiments were conducted on data set B', with both parameters ranging within $[1, 5]$ (Table 4). As in the previous tuning experiment, each entry represents the is the average gap over all six instances. For each instance, the gap is equal to the relative difference between the objective achieved by the corresponding parameter value pair and the best objective achieved by all tested pairs.

Table 4: Tuning a^- and a^+ parameters

$a^- \backslash a^+$	1	2	3	4	5	Avg.
1	0.603	0.681	0.441	0.693	0.707	0.625
2	**0.383**	0.593	0.422	0.404	0.441	**0.448**
3	0.687	0.639	0.600	0.467	0.593	0.597
4	0.540	0.603	0.462	0.415	0.389	0.482
5	0.519	0.419	0.472	0.402	0.568	0.476
Avg.	0.546	0.587	0.480	**0.476**	0.540	

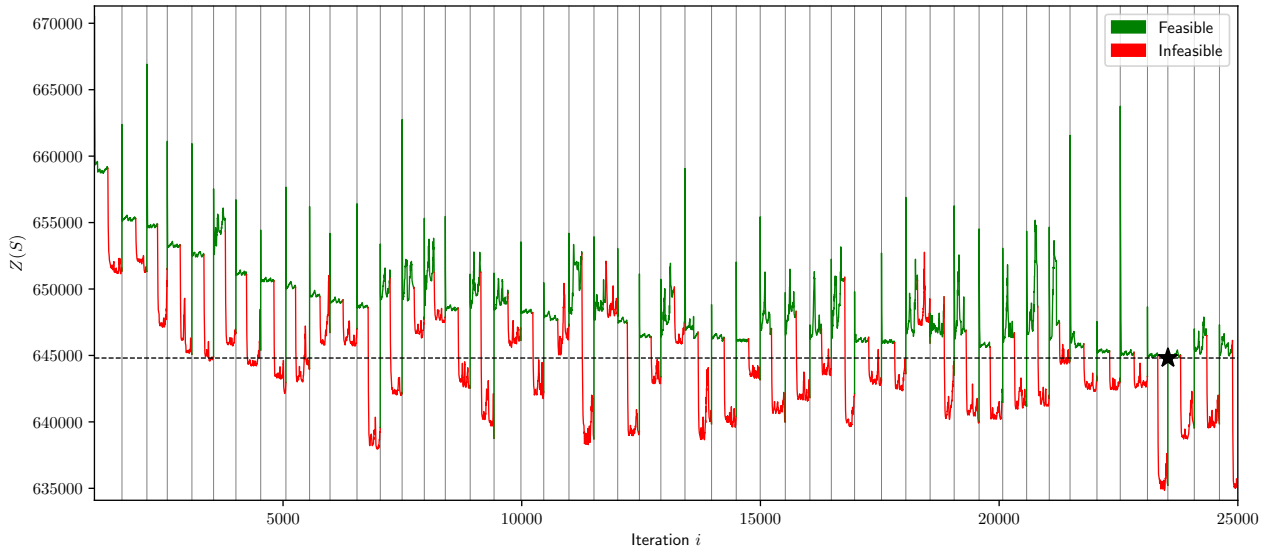
The results highlight the importance of the parameters. When the infeasible solution objective is close to the optimal known, good results are obtained by $a^- = 2$, whereas higher values of 4 and 5 yield similar results, on average. On the contrary, $a^- = 1$ yields the worst results. Conclusively, it seems that high flexibility for restoring solution feasibility is required to obtain high quality solutions. However, given that the solution may be structurally similar to the best solution already identified through the search, a relative conservative value of 2 seems to be the best option to avoid excessive diversification caused by feasibility restoration. The tuning experiments demonstrate lower sensitivity to the a^+ parameter. All tested values yielded comparable average results, with the values of 4 and 3 appearing to be the best options. Again, a balanced flexibility is the best option. However, this balanced flexibility is higher than before due to the fact that the objective of the solution to be repaired

is far from the best identified objective, probably because of a completely different solution structure. Specifically, the optimal values of the parameters tested are $a^- = 2$ and $a^+ = 1$. This pair yields average results similar to the ones provided by pair $(4, 5)$. However, the value pair $a^- = 2$ and $a^+ = 1$ lead to lower *FR* computational times, thus it was selected for the standard parameter setting of the *HISM* algorithm. It should be pointed out that extended experiments with the $(4, 5)$ value pair failed to improve the results yielded by the optimal value pair.

4.6. Contribution of the infeasible space search

This section investigates the role of the infeasible space search, which is the cornerstone of *FILS*. Initially, the relationship between feasible and infeasible space transitions and the objective trajectory is illustrated. Specifically, Figure 1 presents the trajectory of the objective for a restart of one of the most challenging instances solved (instance 20 of subset *B2*). To improve presentation, the first iterations which lead to significant improvements are omitted. The green trajectory line corresponds to feasible solutions, whereas the red line to infeasible ones. The vertical black lines represent *FR* executions aimed at restoring solution feasibility. The black horizontal line and the star mark correspond to the best solution identified.

Figure 1: Feasible and infeasible space transitions (Instance 20 of subset *B2*)



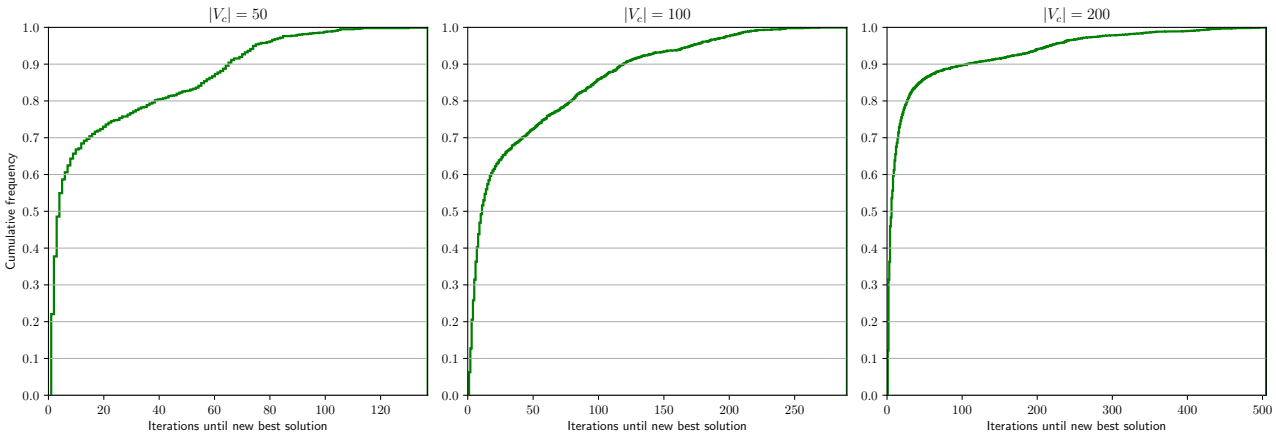
The figure demonstrates the importance of the infeasible space search. The feasible solution objective trajectory (green line) represents the solutions generated by the proposed local search scheme which is equipped with a powerful diversification component (Zachariadis et al. 2015). Despite the diversification by this scheme, plateaus are observed especially for late iterations when the search is trapped around locally optimal solutions. Apparently, by allowing infeasible solutions, a stronger diversification effect is observed which helps the search identify promising solution regions.

Each time the infeasible phase is initialized, a steep objective reduction is observed. After the descent, the search is stable around locally optimal infeasible solutions. A significant objective increase is then observed when the *FR* method is applied to restore feasibility (these steps are represented by the black vertical lines in the figure). This is due to the drastic structural modifications caused by the approximated insertion and removal costs. Starting from a structurally diversified feasible solution, the local search method is capable of quickly improving the incumbent solution. Therefore, the search is driven towards unexplored and promising solution regions and thus the identification of

new best solutions is promoted. From a different perspective, the green plateaus form an objective decreasing trajectory in the early stage of the search (up to approx. 7000 iterations), to be followed by a trajectory with high variability in the later stages. Interestingly, a few iterations before the best solution is identified, a very high objective increase is recorded which probably represents a jump to new feasible region. After this, the infeasible space objective drops significantly lower and then when the feasibility is restored, the best solution is reached.

To demonstrate the effectiveness of the diversification role of the FR method, Figure 2 is provided. In specific, Figure 2 illustrates the distribution of the number of iterations required to obtain a new best solution after the call to the *FR* method. Note that attempts starting from a repaired solution that failed to identify a new best one before reapplying the *FR* method are not taken into account. We use the B' data set and results are aggregated over instances with the same customer number. Hence, each plot contains the total frequencies observed for two instances over 50 restarts.

Figure 2: Cumulative frequency distribution of iterations until new best after feasibility restoration



For $|V_c| = 50$, over 90% of the times that *FR* is used, less than 80 iterations are required to find a new best solution. Similarly, for $V_c = 100$, over 90% of the new best solution are reached within 140 iterations after the feasibility restoration, whereas the cumulative distribution is even more steep for $V_c = 200$. Especially, for these very large instances, above 90% of the new best solutions require less than 150 iterations (i.e., $\frac{3}{4}$ the size of the problem). Note that uncommonly observed high values of iterations required to identify new best solutions are mainly observed at late algorithm stages: this is because at these late stages high-quality local optima have been already identified which are difficult to be improved. Concluding, the contribution of the *FR* MIP is essential for providing solutions with diverse and desirable features that can be quickly transformed to new best solutions.

Appendix C provides additional informative results for the feasible and infeasible space transitions of the algorithm for both data sets *A* and *B* (e.g., feasible-infeasible iterations ratio, time repairing solutions, etc.).

4.7. Comparison of HISM and non-infeasible space algorithm configurations

To provide additional insight on the contribution of allowing infeasible solutions, we perform the following experiment on the most challenging instances of data set *B'*: the infeasibility tunneling is removed from three different algorithmic configurations. This is to assess the possible interplay between the infeasibility procedure and the various algorithm components.

- *HISM-PLS*: plain local search where no capacity infeasibility is allowed. All mathematical programming components, namely *FR* method, *QOPT* method and *TSP* are disabled. This

configuration contains the basic local search framework with the six operators and the promise mechanism, as well as the quantity calculation algorithms.

- *HISM-NCIR*: no capacity infeasibility and no repairs are allowed. The capacity of all vehicles remains Q throughout the execution. Additionally, the *FR* MIP is disabled.
- *HISM-NCI*: no capacity infeasibility is allowed. Note that the *FR* MIP is still used to insert and delete customers to minimize the solution objective, even if the solution is feasible.

Table 5 presents the obtained results of the three tested configurations against the fully featured *HISM* algorithm. Columns PLS (%), NCIR (%) and NCI (%) correspond to the per cent gap of each configuration solution score w.r.t the objective obtained by the *HISM* algorithm. Positive values indicate a higher solution objective compared to the reference *HISM* objective. To provide a more comprehensive comparison, the production costs which are identical for all configurations are ignored. The termination condition for the *HISM-PLS* and *HISM-NCIR* is the completion of 10,000 non-improving iterations, whereas, for *HISM-NCI* the termination condition remains the same with *HISM*. After preliminary runs, the limit of 10,000 iterations was specified which roughly corresponds to the total number of non-improving iterations allowed by *HISM*.

Table 5: Comparison of *HISM* algorithmic variants

Set	Inst.	PLS (%)	NCIR (%)	NCI (%)
B1	9	5.70	1.36	0.10
B1	30	6.35	1.32	0.40
B2	20	5.76	0.88	0.07
B2	27	5.96	2.36	0.19
B3	9	8.93	5.54	1.25
B3	30	9.78	3.44	0.89
Avg.		7.08	2.48	0.48

The *HISM-PLS* configuration is strictly dominated by all others: high gaps of up to 9.78% for the 200-customer problem 30. The incorporation of the *QOPT* and *TSP* components (*HISM-NCIR*) yields considerably better results, indicating that they significantly contribute to the final solution quality. The best results among the three tested configurations are obtained via *HISM-NCI*. However, they are consistently worse than the scores obtained by the complete *HISM* algorithm. In particular, the average gaps is equal to 0.48%.

4.8. Comparison with state-of-the-art algorithms

This section compares the *HISM* results against those of state-of-the-art PRP algorithms. Note that, for the *B1* subset, Schenekemberg et al. (2021) reported results under the assumption of unlimited customer capacity. To enable comparisons, after coming in contact with the authors we were provided with new corrected results that take into account the maximum customer capacity. Additionally, the results of Russell (2017) for all instances of the *B3* subset improve all previously published results by very large gaps. This is unjustifiably inconsistent with the behaviour of the Russell (2017) algorithm for every other data set which is comparable to most algorithms. After personal communication with the author of the aforementioned article, we were unable to confirm the reported objectives: the provided solution objectives did not match the reported ones. Considering the above, it is likely that different assumptions, or objective calculations have been used for instance subset *B3*. Therefore, the Russell (2017) results for subset *B3* are not used for comparisons.

Firstly, we present the number of best solutions reported by each one of the compared algorithms for data set *A* (Table 6), as well as data set *B* (Table 7). The # Inst. column represents the number

of instances involved in each row. The columns names correspond to the algorithm initials that are reported in section 4.2.

Table 6: Number of best known solutions for data set A

Set	Class	# Inst.	ACJ	AADF	SS	RR	QWXFP	CCJ	LCCZ	AY	Our
A1	I	120	1	-	74	-	31	11	46	**107**	102
	II	120	0	-	72	-	35	10	46	99	**104**
	III	120	0	-	67	-	21	4	38	89	**97**
	IV	120	1	-	76	-	27	20	49	104	**112**
A2	I	120	0	0	6	1	1	12	5	15	**91**
	II	120	0	0	0	0	0	1	2	1	**116**
	III	120	0	0	6	1	1	11	2	6	**96**
	IV	120	0	2	4	3	9	3	2	6	**92**
A3	I	120	0	11	13	5	1	20	**26**	**26**	18
	II	120	0	27	0	1	0	5	3	1	**84**
	III	120	0	5	23	7	2	7	15	30	**31**
	IV	120	0	18	8	7	7	8	12	4	**56**
Total		1440	2	63	349	25	135	112	246	488	**999**

Table 7: Number of new best known solutions for data set B

Set	# Inst.	ACJ	AADF	SS	RR	QWXFP	CCJ	LCCZ	SSPGC	Our
B1	30	0	0	0	6	0	0	0	0	**24**
B2	30	0	0	3	8	0	0	1	0	**18**
B3	30	0	2	1	-	0	0	**14**	-	13
Total	90	0	2	4	14	0	0	15	0	**55**

For both data sets, the proposed algorithm generates the highest number of best known solutions (BKS): for data set A 999 BKS are obtained compared to 488 BKS reported by the second most effective algorithm, whereas for data set B 55 BKS are obtained compared to 15 BKS reported by the second most effective algorithm. For data set A, our approach achieves comparable results with the best performing algorithms for the smaller *A1* instances. For the larger instances of *A2* and *A3*, our algorithm significantly improves the results obtained by all compared methods. Specifically, for the high inventory cost Class II, the superiority of *HISM* is more evident. Note that as mentioned earlier, RR provides results for only the first instance of each instance type. Similar conclusions can be drawn for the B data set. Our approach again obtains the highest number of best known solutions: HISM obtains 24 out of the 30, and 18 out of the 30 best known solutions for *B1* and *B2* instances, respectively. RR achieves the second highest number of best solution for these sets. For the largest *B3* set LCCZ reports 14, followed by our approach with 13. It seems that the performance of ACJ, QWXFP, CCJ and SSPGC does not scale efficiently for this data set.

Except for the number of best solutions, objective comparisons are reported for data set A (Table 8) and for data set B (Table 9). Each column provides the per cent optimality gap of the respective algorithm compared to the best known solution. The gap is calculated as $\frac{Z(S^*_{alg})-Z(S^*)}{Z(S^*)}$, where $Z(S^*_{alg})$ is solution objective by an algorithm alg and $Z(S^*)$ is the best known solution.

Table 8: Average gap for best known solutions for data set A

Set	Class	%ACJ	%AADF	%SS	%RR	%QWXFP	%CCJ	%LCCZ	%AY	%Our
A1	I	1.70	-	0.03	-	0.28	0.47	0.15	**0.01**	0.05
	II	0.37	-	0.01	-	0.05	0.08	0.03	**0.01**	0.01
	III	8.42	-	0.18	-	1.52	2.20	0.75	**0.04**	0.38
	IV	0.93	-	0.03	-	0.56	0.23	0.08	0.01	**0.00**
A2	I	1.34	0.34	0.26	0.31	0.26	0.30	0.20	0.18	**0.01**
	II	0.22	0.08	0.06	0.05	0.08	0.08	0.05	0.06	**0.00**
	III	4.42	1.59	0.95	0.76	0.90	1.19	0.85	0.81	**0.09**
	IV	0.35	0.21	0.21	0.14	0.13	0.15	0.18	0.17	**0.06**
A3	I	1.07	0.24	0.15	0.13	0.25	0.34	0.10	**0.10**	0.22
	II	0.33	**0.04**	0.21	0.04	0.25	0.22	0.20	0.21	0.17
	III	4.03	1.36	0.51	**0.29**	0.67	1.96	0.52	0.60	0.42
	IV	0.44	0.18	0.16	**0.08**	0.19	0.17	0.15	0.20	0.23
Average		1.97	0.50	0.23	0.23	0.43	0.62	0.27	0.20	**0.14**

Table 9: Average gap for best known solutions for data set B

Set	%ACJ	%AADF	%SS	%RR	%QWXFP	%CCJ	%LCCZ	%SSPGC	%Our
B1	2.11	2.24	1.09	0.33	1.78	1.91	1.35	3.26	**0.03**
B2	1.69	1.43	0.80	0.58	1.35	1.83	0.64	10.51	**0.15**
B3	2.70	1.30	3.34	-	1.27	1.07	0.41	-	**0.20**
Average	2.17	1.66	1.74	0.45	1.46	1.60	0.80	6.89	**0.13**

For data set A, almost all algorithms report high quality solutions with gaps that are up to 0.7% above the BKS scores. The A set mainly involves instances for which the production costs make up the the largest portion of the total costs. Consequently, the solutions may be structurally different in terms of the inventory and routing decisions, however the total objective is comparable as it mainly depends on the production cost. ACJ demonstrates larger gaps due to the Class III results where the routing costs contribute more strongly to the overall objective. Our methodology obtains the lowest average gaps, and outperforms compared methods for all classes of the *A2* subset. The SS method which mainly relies on exact components appears to be more powerful for the smaller instances of *A1* subset compared to metaheuristic-based algorithms, such as CCJ and LCCZ. The results reported for the B set, demonstrate the superiority of the proposed algorithm. The average optimality gap obtained by *HISM* is limited to 0.13%. On average the AADF, QWXFP, CCJ and SS methods have comparable performance. The exact-based SSPGC and SS algorithms appear to scale poorly with the problem size (the average gap for SS is equal to 3.34% for the larger instances of *B3*, whereas SSPGC could only be applied to instances of up to 100 customers for which a large average gap of 10.51% is reported). On the contrary, LCCZ provides consistently high quality results for all three subsets.

Except for the average objective gaps, we also provide box-and-whisker plots which graphically depict the individual gaps obtained for every instance of each of the subsets considered: Figure 3 and Figure 4 consist the box-and-whisker plots for the data sets A and B, respectively. Both involve three plots, one plot for each instance subset. The whiskers extend to a maximum of 3.5 times the interquartile range.

For data set A, the ACJ gaps of Class III as mentioned earlier are large and hence the boxes for all other algorithms are seem narrow. Especially for *A1* our algorithm, SS and AY demonstrate significant consistency, whereas all algorithms except for ACJ are comparable. Similarly, for subset *A2*, our approach produces new BKS for the majority of instances, thus the interquartile range is close to zero, whereas the rest of the algorithms except ACJ demonstrate similar performance. For the larger instances subset *A3*, the relevant plot illustrates greater variability of the gap dispersion of the algorithms. Methods RR, SS, LCCZ and AY appear to be the most consistent. Again, the more

Figure 3: Optimality gap box-and-whisker plot for data set *A*

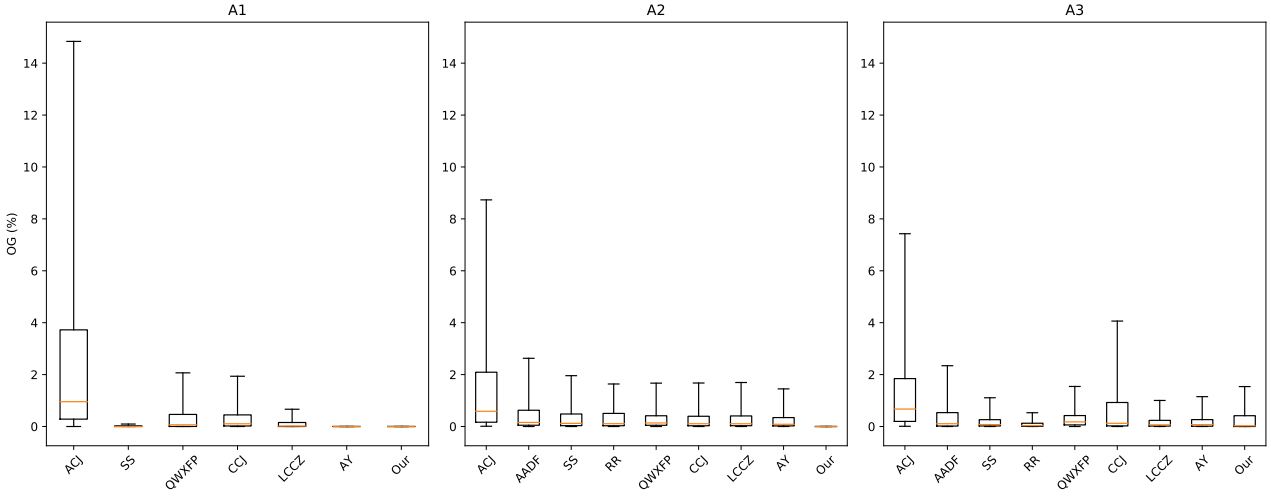
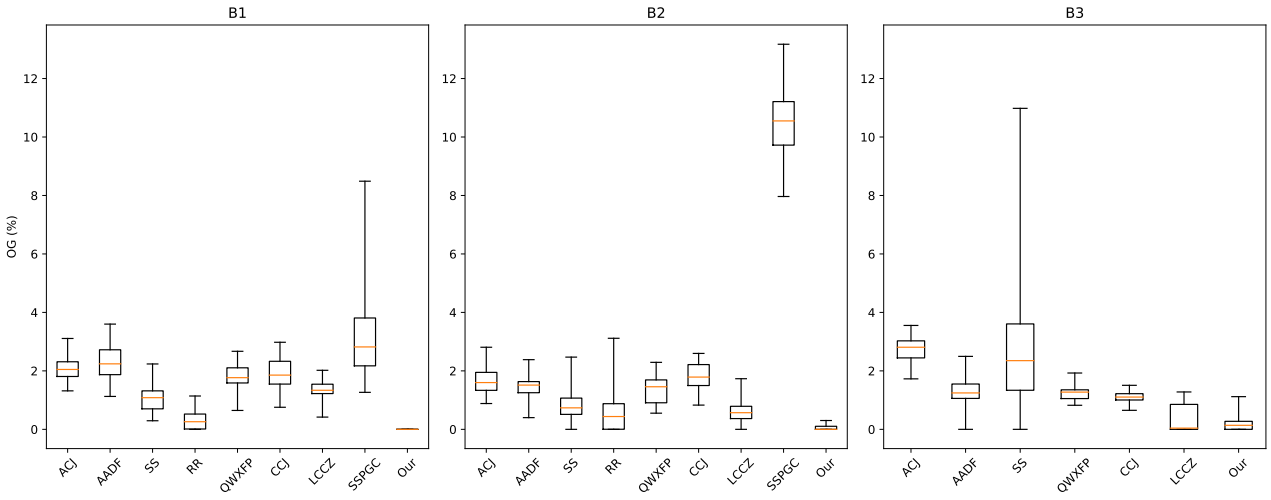


Figure 4: Optimality gap box-and-whisker plot for data set *B*



balanced and challenging data set *B* provides a stronger insight on the performance of the compared algorithms. For subset *B1*, our algorithm produces several new BKS, and therefore the gaps are consistently close to zero. Algorithms SS and RR exhibit an effective and stable performance, whereas other algorithms appear consistent but with higher medians. For subset *B2*, SSPGC demonstrates significantly worse results than all other approaches. Our proposed methodology seems the most effective and consistent, whereas RR and LCCZ also perform well but with larger variance of their gaps. Our approach and LCCZ produced comparable results for subset *B3*, with our algorithm demonstrating a more stable behaviour. Notably, heuristics QWXFP and CCJ yield large gaps with low variance. Finally, SS which heavily relies on exact components produces gaps with the largest variance.

4.9. Summary results

Our algorithm matches or improves the BKS scores for the majority of the PRP instances examined. Tables 10 and 11 summarize the results of *HISM* for the two benchmark data sets. Column #B reports the number of matched or new BKS, whereas #NB is the number of new BKS identified. BKSG

represents the gap between our scores and previously published BKS scores. Thus, negative values indicate improvement of the BKS. Finally, the Time column presents total running time. respectively.

Table 10: *HISM* results for data set *A*

Set	Class	# Inst	#B	#NB	BKSG%	Time
A1	I	120	102	4	0.05	0.86
	II	120	104	4	0.01	0.86
	III	120	97	9	0.38	1.05
	IV	120	112	1	0.00	0.98
A2	I	120	91	86	-0.10	15.13
	II	120	116	116	-0.03	15.63
	III	120	96	95	-0.34	18.32
	IV	120	92	91	-0.02	14.09
A3	I	120	18	18	0.21	101.74
	II	120	84	83	0.15	110.73
	III	120	31	31	0.29	110.13
	IV	120	56	56	0.21	117.60
Total		1440	999	594	-	-

Table 11: *HISM* results for data set *B*

Set	# Inst	#B	#NB	BKSG%	Time
B1	30	24	24	-0.29	317.58
B2	30	18	18	-0.17	821.94
B3	30	13	13	-0.07	2021.11
Total	90	55	55	-	-

For data set *A*, we manage to report 594 new best solution out of the 999 best known solution found. The optimality gaps are on average low, and specifically they are negative for all classes of *A2*. On average about half the iteration are in the infeasible space and the times remaining manageable as the largest *A3* instances required under two minutes on average. For data set *B* all 55 reported best solution are new best solutions and consequently all average optimality gaps are negative.

Individual solution scores and computational times for each of the 1530 PRP instances are provided in the supplementary material of the paper.

5. Conclusions

This paper proposes a novel matheuristic algorithm named *HISM* for the challenging Production Routing Problem. Initially, the problem is formulated as a two-commodity flow model, which is further strengthened by valid inequalities. The proposed solution methodology consists of two phases, a production-distribution relaxation and a local search matheuristic oscillating between feasible and infeasible search space. Initial production-distribution plans are obtained by a master problem relaxation. The associated partial solution is completed in terms of routing information by a GRASP method. The constructed complete is then iteratively improved by the proposed *FILS* matheuristic algorithm which uses a blend of six inventory and routing operators. Vehicle capacity infeasibility is allowed through the search process. To handle the continuous delivery quantity decision variables, a computationally effective procedure is proposed. The obtained delivery plans and production decisions are then jointly optimized via an LP model. The proposed matheuristic incorporates the *FR* MIP capable of inserting, removing and relocating a bounded number of customers for each route to restore solution feasibility.

The main feature of *HISM* is the transition between feasible and infeasible solutions during the search process. Allowing infeasible solutions effectively diversifies the search towards new regions of the solution space. In addition, the employed feasibility restoration procedure *FR* results in significant solution structure modifications. This strong diversification effect is controlled by the proposed local search method which can quickly intensify around locally optimal solutions. Conducted experiments showed that the tunneling through infeasible space plays an important role for a fast, diversified and effective exploration of the solution space. The proposed matheuristic was compared against the most recent and state-of-the-art methods of the PRP literature on 1530 well known benchmark instances. Our obtained results outperform the known results by matching or generating new solutions for the majority of the instances examined. More specifically, for 1440 small-medium and 90 large problem instances, our algorithm matched or improved the best known solutions for 999 and 55 test cases, respectively. More precisely, 594 and 55 of these are new best solutions.

Regarding future research directions, it would be interesting to examine how can the setup production schedule be modified within a local search framework. Another interesting algorithmic extension is the introduction of novel neighborhood structures specifically aimed at maximizing the benefits of infeasible space search. Finally, apart from vehicle capacity infeasibility, other types of infeasibility could be considered.

Acknowledgements

This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014-2020" in the context of the project "Quantitative Decision Support Tools for the Combined Production, Inventory, Distribution and Routing Problem" (MIS 5049910).

The authors would like to thank Dr. Cleder Marcos Schenekemberg for providing updated results for Schenekemberg et al. (2021) and two anonymous reviewers for providing comments that improved the manuscript quality.

References

Absi, N., Archetti, C., Dauzère-Pérès, S., Feillet, D., Speranza, M.G., 2016. Comparing sequential and integrated approaches for the production routing problem. European Journal of Operational Research 269, 633–646. doi:10.1016/j.ejor.2018.01.052.

Absi, N., Archetti, C., Dauzère-Pérès, S., Feillet, D., 2015. A two-phase iterative heuristic approach for the production routing problem. Transportation Science 49, 784–795. doi:10.1287/trsc.2014.0523.

Adulyasak, Y., Cordeau, J.F., Jans, R., 2014a. Optimization-based adaptive large neighborhood search for the production routing problem. Transportation Science 48, 20–45. doi:10.1287/trsc.1120.0443.

Adulyasak, Y., Cordeau, J.F., Jans, R., 2015a. Benders decomposition for production routing under demand uncertainty. Operations Research 63, 851–867. doi:10.1287/opre.2015.1401.

Adulyasak, Y., Cordeau, J.F., Jans, R., 2015b. The production routing problem: A review of formulations and solution algorithms. Computers and Operations Research 55, 141–152. doi:10.1016/j.cor.2014.01.011.

Adulyasak, Y., Cordeau, J.F.F., Jans, R., 2014b. Formulations and Branch-and-Cut Algorithms for Multivehicle Production and Inventory Routing Problems. INFORMS Journal on Computing 26, 103–120. doi:10.1287/ijoc.2013.0550.

Archetti, C., Bertazzi, L., Paletta, G., Speranza, M.G., 2011. Analysis of the maximum level policy in a production-distribution system. Computers and Operations Research 38, 1731–1746. doi:10.1016/j.cor.2011.03.002.

Archetti, C., Speranza, M.G., 2014. A survey on matheuristics for routing problems. EURO Journal on Computational Optimization 2, 223–246. doi:10.1007/s13675-014-0030-7.

Archetti, C., Speranza, M.G., 2016. The inventory routing problem: The value of integration. International Transactions in Operational Research 23. doi:10.1111/itor.12226.

Armentano, V.A., Shiguemoto, A.L., Løkketangen, A., 2011. Tabu search with path relinking for an integrated production-distribution problem. Computers and Operations Research 38, 1199–1209. doi:10.1016/j.cor.2010.10.026.

Avci, M., Yildiz, S.T., 2019. A matheuristic solution approach for the production routing problem with visit spacing policy. European Journal of Operational Research 279, 572–588. doi:10.1016/j.ejor.2019.05.021.

Boudia, M., Louly, M.A.O., Prins, C., 2007. A reactive GRASP and path relinking for a combined production – distribution problem. Computers & Operations Research 34, 3402–3419. doi:10.1016/j.cor.2006.02.005.

Boudia, M., Louly, M.A.O., Prins, C., 2008. Fast heuristics for a combined production planning and vehicle routing problem. Production Planning and Control 7287. doi:10.1080/09537280801893356.

Boudia, M., Prins, C., 2009. A memetic algorithm with dynamic population management for an integrated production-distribution problem. European Journal of Operational Research 195, 703–715. doi:10.1016/j.ejor.2007.07.034.

Brown, G., Keegan, J., Vigus, B., Wood, K., 2001. The Kellogg Company Optimizes Production, Inventory, and Distribution. Interfaces 31, 1–15. doi:10.1287/inte.31.7.1.9646.

Çetinkaya, S., Üster, H., Easwaran, G., Keskin, B.B., 2009. An integrated outbound logistics model for Frito-Lay: Coordinating aggregate-level production and distribution decisions. Interfaces 39, 460–475. doi:10.1287/inte.1090.0450.

Chandra, P., 1993. A Dynamic Distribution Model with Warehouse and Customer Replenishment Requirements. Journal of the Operational Research Society 44, 681–692.

Chandra, P., Fisher, M.L., 1994. Coordination of Production and Distribution Planning. European Journal of Operational Research 72, 503–517. doi:10.1534/genetics.112.141861.

Chitsaz, M., Cordeau, J.F., Jans, R., 2019. A unified decomposition matheuristic for assembly, production, and inventory routing. INFORMS Journal on Computing 31, 134–152. doi:10.1287/ijoc.2018.0817.

Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., Semet, F., 2002. A guide to vehicle routing heuristics. Journal of the Operational Research Society 53, 512–522. doi:10.1057/palgrave/jors/2601319.

Cordeau, J.F., Laporte, G., Mercier, A., 2001. A unified tabu search algorithm for vehicle routing problems with time windows. Journal of the Operational Research Society 59, 663–673. doi:10.1057/palgrave.jors.2602371.

Darvish, M., Coelho, L.C., 2018. Sequential versus integrated optimization: Production, location, inventory control, and distribution. European Journal of Operational Research 268. doi:10.1016/j.ejor.2018.01.028.

Dayarian, I., Desaulniers, G., 2019. A branch-price-and-cut algorithm for a production-routing problem with short-life-span products. Transportation Science 53, 829–849. doi:10.1287/trsc.2018.0854.

Díaz-Madroñero, M., Peidro, D., Mula, J., 2015. A review of tactical optimization models for integrated production and transport routing planning decisions. Computers and Industrial Engineering 88, 518–535. doi:10.1016/j.cie.2015.06.010.

Felipe, A., Teresa Ortuño, M., Tirado, G., 2011. Using intermediate infeasible solutions to approach vehicle routing problems with precedence and loading constraints. European Journal of Operational Research 211, 66–75. doi:10.1016/j.ejor.2010.11.011.

Feo, T.A., Resende, M.G., 1995. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization 6, 109–133. doi:10.1007/BF01096763.

Gendreau, M., Hertz, A., Laporte, G., 1994. Tabu search heuristic for the vehicle routing problem. Management Science 40, 1276–1290. doi:10.1287/mnsc.40.10.1276.

Gendreau, M., Laporte, G., Seguin, R., 1996. A Tabu Search Heuristic for the Vehicle Routing Problem with Stochastic Demands and Customers. Operations Research 44, 469–477.

Li, Y., Chu, F., Chu, C., 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, 914–927. doi:10.1016/j.ejor.2018.07.018.

Manousakis, E., Repoussis, P., Zachariadis, E., Tarantilis, C., 2021. Improved branch-and-cut for the Inventory Routing Problem based on a two-commodity flow formulation. European Journal of Operational Research 290, 870–885. doi:10.1016/j.ejor.2020.08.047.

Mazareanu, E., 2020. Logistics industry- global costs by region 2018. Technical Report. URL: https://www.statista.com/statistics/943492/global-logistics-industry-costs-by-region/.

Miranda, P.L., Cordeau, J.F., Ferreira, D., Jans, R., Morabito, R., 2018. A decomposition heuristic for a rich production routing problem. Computers and Operations Research 98, 211–230. doi:10.1016/j.cor.2018.05.004.

Qiu, Y., Qiao, J., Pardalos, P.M., 2019. Optimal production, replenishment, delivery, routing and inventory management policies for products with perishable inventory. Omega (United Kingdom) 82, 193–204. doi:10.1016/j.omega.2018.01.006.

Qiu, Y., Wang, L., Xu, X., Fang, X., Pardalos, P.M., 2018a. A variable neighborhood search heuristic algorithm for production routing problems. Applied Soft Computing Journal 66, 311–318. doi:10.1016/j.asoc.2018.02.032.

Qiu, Y., Wang, L., Xu, X., Fang, X., Pardalos, P.M., 2018b. Formulations and branch-and-cut algorithms for multi-product multi-vehicle production routing problems with startup cost. Expert Systems with Applications 98, 1–10. doi:10.1016/j.eswa.2018.01.006.

Russell, R.A., 2017. Mathematical programming heuristics for the production routing problem. International Journal of Production Economics 193, 40–49. doi:10.1016/j.ijpe.2017.06.033.

Schenekemberg, C.M., Scarpin, C.T., Pécora, J.E., Guimarães, T.A., Coelho, L.C., 2021. The two-echelon production-routing problem. European Journal of Operational Research 288, 436–449. doi:10.1016/j.ejor.2020.05.054.

Senoussi, A., Dauzère-Pérès, S., Brahimi, N., Penz, B., Kinza Mouss, N., 2018. Heuristics Based on Genetic Algorithms for the Capacitated Multi Vehicle Production Distribution Problem. Computers and Operations Research 96, 108–119. doi:10.1016/j.cor.2018.04.010.

Solyalı, O., Süral, H., 2017. A multi-phase heuristic for the production routing problem. Computers and Operations Research 87, 114–124. doi:10.1016/j.cor.2017.06.007.

Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2015. The load-dependent vehicle routing problem and its pick-up and delivery extension. Transportation Research Part B: Methodological 71, 158–181. doi:10.1016/j.trb.2014.11.004.