A Decomposition Heuristic for a Rich Production Routing Problem

Pedro L. Miranda[a], Jean-François Cordeau[b], Deisemera Ferreira[c], Raf Jans[b,*], Reinaldo Morabito[a]

[a]*Department of Production Engineering, Federal University of São Carlos*
Rodovia Washington Luís - Km 235, CEP: 13565-905, São Carlos-SP, Brazil
[b]*HEC Montréal and CIRRELT*
3000 chemin de la Côte-Sainte-Catherine, Montréal H3T 2A7, Canada
[c]*Department of Physics, Chemistry and Mathematics, Federal University of São Carlos*
Rodovia João Leme dos Santos - Km 110, CEP: 18052-780, Sorocaba-SP, Brazil

Abstract

We propose a decomposition heuristic to solve a rich production routing problem arising in the context of a make-to-order company. The problem is motivated by the operations of a Brazilian furniture manufacturer and considers several important features, such as multiple products, sequence-dependent setup times, a heterogeneous fleet of vehicles, routes extending over one or more periods, multiple time windows and customer deadlines, among others. An integrated mathematical model is presented and is used as a basis to develop the heuristic, which solves the problem by decomposing it in two parts that are solved iteratively. The first subproblem focuses on the production planning and customer assignment decisions, and uses an approximation for the routing costs and travel times. The second subproblem makes the routing decisions, which are further improved by a local search algorithm. The solution of the second subproblem is then used to update the approximation of the routing costs and travel times in the first subproblem. We use a large set of random instances to benchmark our heuristic against a general-purpose solver. Numerical results show that our method provides,

*Corresponding author
Email addresses: pmiranda@dep.ufscar.br (Pedro L. Miranda), jean-francois.cordeau@hec.ca (Jean-François Cordeau), deise@ufscar.br (Deisemera Ferreira), raf.jans@hec.ca (Raf Jans), morabito@ufscar.br (Reinaldo Morabito)

in shorter computing times, solutions of similar quality to those obtained by the solver for instances with up to 15 customers. For larger instances, with 20 to 50 customers, the heuristic clearly outperforms the solver, which in most cases cannot find any solution after 24 hours of computing time.

Keywords: Production routing problem, integrated production and distribution planning, decomposition heuristic, lot-sizing

1. Introduction

The coordination of production and distribution activities has drawn the attention of many researchers since the seminal work of Chandra and Fisher [17]. Production activities comprise decisions about lot sizing and scheduling to determine what to produce, when to produce, how much to produce, and how much to keep in stock, aiming to minimize the production, setup and inventory costs incurred in the production process [27, 29, 33]. Distribution activities focus on shipment dates and quantities and on vehicle routing decisions to determine the optimal set of routes to be performed by a fleet of vehicles in order to serve a given set of customers at minimum cost [21, 25, 36].

In a typical supply chain, these activities of production and distribution are often planned and optimized sequentially [5]. The common practice is to first determine the lot sizes and the inventory levels over the planning horizon, and then use these decisions as input to decide how products should be delivered to customers. Since the distribution decisions are limited by the preceding production decisions, the benefits of a more global and integrated planning are lost. However, today's companies should use their resources more efficiently in order to increase customer service levels and reduce lead times and total costs [18, 22]. In this sense, integrating different supply chain decisions, such as production and distribution, may help to increase efficiency and save costs across the company.

In the literature, the problem concerned with the integration of the aforementioned decisions is known as the production routing problem (PRP) [5, 34], the integrated production, inventory, distribution and routing problem (PIDRP) [10, 11, 30] or the integrated production and distribution problem (IPDP) [8, 14, 17, 24], and it aims to jointly minimize production, inventory, setup and routing costs in the system.

The economic impact of integrating production and routing decisions at the operational level was first studied by Chandra and Fisher [17]. The authors found that integrating these decisions may lead to savings ranging from 3 % to 20 % in comparison with the traditional sequential approach, in which routing decisions are made after the production plan has been determined. Subsequentially, a number of papers have proposed both mathematical formulations and solution procedures for different variants of the problem. Most of the research on the PRP has focused on a problem with a single production facility that produces one product and owns a limited fleet of homogeneous vehicles. Solution methods for this problem include branch-and-cut algorithms [2, 34], Benders-based branch-and-cut [4], branch-and-price-based heuristics [9, 11], mathematical programming-based heuristics [1, 7, 12, 19], greedy randomized adaptive search procedure (GRASP) [13], genetic algorithm with population management (GAPM) [14], reactive tabu search (RTS) [10] and adaptive large neighborhood search (ALNS) [3].

A notable exception to the standard problem described above is the work presented by Lei et al. [30] in the context of chemical industries. The problem consists of a manufacturer owning multiple production facilities that produce the same product and a fleet of heterogeneous vehicles that may perform multiple trips per period. The authors formulated the problem as a mixed integer program (MIP) and proposed a two-phase decomposition heuristic to solve it.

The problem with multiple products has received less attention. Because of its complexity, solution strategies have been limited to heuristic and meta-heuristic procedures, such as decomposition heuristics [17], Lagrangian heuristics [24], mathematical programming-based heuristics [15] and tabu search-based algorithms [8, 35]. The work of Amorim et al. [6] does not propose a specific solution method for the problem, but includes a comparison of two different formulations with sequencing decisions in the context of perishable goods.

More recently, Miranda et al. [31] formulated a MIP and proposed several relax-and-fix (RF) heuristics to solve a particular PRP in small furniture companies. The authors considered a scenario in which the manufacturer has one production line and only one vehicle that may perform multiple trips over the planning horizon. The formulation includes some features rarely considered in the related literature, but commonly found in real-world applications, such as producing and stocking multiple parts (instead of final products), distribution routes extending over one or more periods, multiple

time windows and deadlines on customer deliveries.

Miranda et al. [32] extended their previous work in furniture companies by considering a more general situation with sequence-dependent setup times on the production line and a limited fleet of heterogeneous vehicles for distribution. Two different formulations were proposed and evaluated on a large set of random instances by using a general-purpose solver. Computational results showed that integrating production and distribution decisions can lead to significant savings in the system, in comparison with a sequential procedure that mimics the common practice in furniture industries.

In this paper we study a rich PRP arising in the context of make-to-order companies and motivated by the operations of a Brazilian furniture manufacturer. The problem includes producing multiple items needed to assemble different final products, sequence-dependent setup times, a heterogeneous fleet of vehicles, routes that may span for one or more periods, multiple time windows and customer deadlines, among others. This problem has been previously studied by Miranda et al. [32], who used the CPLEX solver to evaluate two different MIP models. Computational experiments showed that instances with at most 15 customers can be solved within reasonable computing times and, therefore, efficient solution methods are necessary to solve larger instances. We address this issue here, by proposing a decomposition heuristic to solve the problem.

Our heuristic is inspired by the work of Absi et al. [1] and decomposes the problem into two subproblems, which are then solved iteratively until a given stopping criterion is met. The first subproblem makes all the production decisions as well as the assignment of customers to routes without considering routing decisions explicitly. At this point, the routing costs and travel times are only taken into account in an approximate way.

The second subproblem aims to optimize the distribution routes based on the previous results of the first subproblem. Since timing constraints on the routing side are not taken into account in the first subproblem, it is possible that the second one is infeasible. If this is the case, inequalities inspired by local branching are added to the first subproblem to guide it to a different solution. Whenever feasible routes are obtained after solving the second subproblem, a simple local search algorithm is applied to further reduce the routing costs. The incumbent solution is then used to update the current approximation of both routing costs and travel times before starting a new iteration. This procedure iterates until the incumbent solution is not improved in a given number of iterations, which triggers a restart mechanism

that resets the whole procedure using new approximations of the routing costs.

Our approach is benchmarked against a state-of-the-art optimization software on a set of randomly generated instances. Numerical results show that the heuristic provides competitive solutions for the smaller instances, and is able to find good feasible solutions for the larger ones, which are beyond the current capabilities of the solver.

The rest of the paper is organized as follows. Section 2 presents the problem description. A mathematical formulation, which is used as basis to develop our heuristic, is presented in Section 3. The decomposition heuristic is then described in detail in Section 4. Section 5 is devoted to computational experiments and, finally, concluding remarks and future research directions are highlighted in Section 6.

2. Problem description

The production routing problem considered consists of a make-to-order company owning a production facility that produces a set $\mathcal{C}$ of items, indexed by $a \in \mathcal{C}$, required to meet the demand for a set $\mathcal{P}$ of final products, indexed by $p \in \mathcal{P}$. Assembling the items into final products is only done at the customers, and hence the production planning only relates to the items. This is, for example, the case in the furniture industry where final products are designed to share interchangeable items and sub-assemblies, and therefore there is no inventory or production of final products, but only of the items used to assemble them. To assemble one unit of product p it is necessary to produce η_{ap} units of item a. We assume that $\eta_{ap} = 0$ if $a \notin \mathcal{F}_p$, where $\mathcal{F}_p \subseteq \mathcal{C}$ represents the subset of items required for assembling product p. The unit inventory holding cost of item a is h_a, and the initial and minimum inventory levels of item a are denoted by I_{a0} and $I_a^{\min}$, respectively.

The set of planning periods is denoted by $\mathcal{T}$. The line capacity measured in time units is K_t, $t \in \mathcal{T}$, and the time required to produce one unit of item a is ρ_a. Items produced in period t are only available for shipping in period $t+1$ and so shipments in period t must come from the inventory available at the end of period $t-1$. Multiple items can be produced in each period and switching production from item a to item b requires ς_{ab} time units. This time represents a loss of capacity due to cleaning, adjustment, calibration, change of tooling, etc., performed on the line before starting the production of a new item. Similarly, the cost of a changeover from item a to item b is

denoted by $\hat{c}_{ab}$, which means that both setup times and costs are sequence-dependent. We also assume that setups can be carried-over between periods and, therefore, the last product of a given period t may be produced without an additional setup time or cost at the start of the next period $t+1$.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ define a complete directed graph, where $\mathcal{N} = \{0, 1, \ldots, n+1\}$ is the set of nodes and $\mathcal{A} = \{(i,j) : i, j \in \mathcal{N}, i \neq j\}$ is the set of arcs. The depot is denoted by both 0 and $n+1$ and $\bar{\mathcal{C}} = \mathcal{N} \backslash \{0, n+1\}$ denotes the set of customers. The travel time (cost) to go from node i to node j is denoted by τ_{ij} (c_{ij}), $i, j \in \mathcal{N}$. For each customer i, $i \in \bar{\mathcal{C}}$, let d_{pi} be the demand of product p at customer i and $[\delta_{it}, \bar{\delta}_{it}]$ denote the time window of customer i in period t. Similarly, let $[\delta_{it}, \bar{\delta}_{it}]$ for $i \in \{0, n+1\}$ denote the time window of nodes 0 and $n+1$ in period t, respectively. If customer i is visited in period t, then the service at node i must start anytime between the initial time instant δ_{it} and the final time instant $\bar{\delta}_{it}$. If the vehicle arrives at the node before δ_{it}, then it has to wait until δ_{it} to start the service. Also, if the vehicle arrives after $\bar{\delta}_{it}$, then it has to wait until $\delta_{i,t+1}$ to start the service. Each customer i places exactly one order for several products and must be visited only once (i.e., split deliveries are not allowed), before a preset deadline Δ_i, which denotes the latest instant of time to serve customer i. Time windows on node 0 represent the time intervals within which loading operations and dispatching of vehicles are allowed. Outside these time windows no loading or dispatching operation is possible, for example because there is no personnel at the facilities. On the other hand, time windows on node $n+1$ denote the intervals at which the depot is available for the return of the vehicles. In practice, these latter time windows span the whole day, which means that the vehicles may get back to the depot at any time.

Deliveries are performed by a set $\mathcal{V}$ of heterogeneous vehicles. The capacity, in weight units, of vehicle $v \in \mathcal{V}$ is θ_v and each unit of product p on board of vehicle v consumes φ_p units of capacity. Each vehicle v can perform several routes over the planning horizon, and each route $r = 1, \ldots, R$ must depart from node 0 and arrive at node $n+1$. R denotes an upper bound on the number of routes performed over the planning horizon (for example, $R = n$). Before starting a new route, vehicles must be reloaded at the depot. We assume that the loading time is proportional to the service time of the customers served by the route, and therefore it is variable. The service time of customer i, s_i, is completely determined by its demand, since it only depends on the time required to unload the products. The time to load/unload per unit of weight is denoted by λ. We do not impose a limit on the length

of the route, so that a route can depart from node 0 in period $t \in \mathcal{T}$ and arrive at node $n+1$ at period $t' \in \mathcal{T}, t' \geq t$. Therefore, routes can extend over several periods in the planning horizon and it is also possible to perform several short routes within a single period.

The problem consists in determining how much to produce of each item, the sequence in which the items should be produced, the vehicle routes and the time at which each customer should be served over a finite multi-period planning horizon. The objective is to minimize the setup, inventory and routing costs. Note that we do not consider inventory holding costs at the customers, since we do not assume that the manufacturer manages the customers' inventory.

3. Mathematical formulation

This section presents the main notation and a mathematical formulation that integrates both production and routing decisions in a single framework. We use the following notation.

Sets

$\mathcal{P}$: Set of products, indexed by p;
$\mathcal{C}$: Set of components or items, indexed by a and b;
$\mathcal{F}_p$: Set of items needed to make product p, $\mathcal{F}_p \subseteq \mathcal{C}$;
$\mathcal{T}$: Set of time periods, indexed by t;
$\bar{\mathcal{C}}$: Set of customer nodes, indexed by i and j, with $\bar{\mathcal{C}} = \{1, \ldots, n\}$;
$\mathcal{N}$: Set of all nodes, with $\mathcal{N} = \bar{\mathcal{C}} \cup \{0, n+1\}$, where 0 and $n+1$ represent the depot;
$\mathcal{V}$: Set of vehicles, indexed by v.

Parameters

h_a : Unit inventory holding cost of item a;
c_{ij} : Transportation cost from node i to node j;
$\hat{c}_{ab}$: Setup cost from item a to item b;
I_{a0} : Initial inventory of item a;
$I_a^{\min}$: Minimum inventory (or safety stock) of item a;
ρ_a : Unit processing time of item a;

ς_{ab} : Setup time from item a to item b;

η_{ap} : Number of items a needed to produce one unit of product p;

d_{pi} : Demand of product p at customer i;

K_t : Production capacity (in units of time) in period t;

τ_{ij} : Traveling time from node i to node j;

θ_v : Capacity of vehicle v;

δ_{it} : Beginning of time window of node i in period t;

$\bar{\delta}_{it}$: End of time window of node i in period t;

Δ_i : Deadline of node i (i.e., latest time instant to serve customer i);

φ_p : Unit weight of product p;

λ : Time to load/unload per unit of weight;

R : Maximum number of routes over the planning horizon;

s_i : Service time of customer i, with $s_i = \lambda \sum_{p \in \mathcal{P}} \varphi_p d_{pi}$;

M_{0j} : A sufficiently large number, $M_{0j} = \bar{\delta}_{0|\mathcal{T}|} + \sum_{i \in \bar{\mathcal{C}}} s_i + \tau_{0j}$;

M_{ij} : A sufficiently large number, $M_{ij} = \Delta_i + s_i + \tau_{ij}$.

Decision variables

x_{at} : Production quantity of item a in period t;

I_{at} : Inventory of item a at the end of period t;

y_{at} : Equal to 1 if the line is set up for item a at the beginning of period t, 0 otherwise;

z_{abt} : Equal to 1 if there is a changeover from item a to item b in period t, 0 otherwise;

π_{at} : Auxiliary variable for sequencing item a in period t;

w_{ijr} : Equal to 1 if arc (i, j) is traveled by route r, 0 otherwise;

Q_{prt} : Quantity of product p shipped on route r in period t;

ϕ_{irt} : Equal to 1 if node i is visited by route r in period t, 0 otherwise;

μ_{ir} : Starting time at which node i is served by route r;

α_{rv} : Equal to 1 if route r is performed by vehicle v, 0 otherwise.

The full integrated model can be written as follows:

$$\min \sum_{a\in\mathcal{C}} \sum_{t\in\mathcal{T}} h_a I_{at} + \sum_{a\in\mathcal{C}} \sum_{\substack{b\in\mathcal{C}\\ b\neq a}} \sum_{t\in\mathcal{T}} \hat{c}_{ab} z_{abt} + \sum_{r=1}^{R} \sum_{i=0}^{n} \sum_{j=1}^{n+1} c_{ij} w_{ijr} \tag{1}$$

subject to

$$I_{a,t-1} + x_{at} = \sum_{r=1}^{R} \sum_{p\in\mathcal{P}} \eta_{ap} Q_{prt} + I_{at}, \qquad a \in \mathcal{C}, t \in \mathcal{T} \tag{2}$$

$$I_{a,t-1} \geq \sum_{r=1}^{R} \sum_{p\in\mathcal{P}} \eta_{ap} Q_{prt}, \qquad a \in \mathcal{C}, t \in \mathcal{T} \tag{3}$$

$$I_{at} \geq I_a^{\min}, \qquad a \in \mathcal{C}, t \in \mathcal{T} \tag{4}$$

$$\sum_{a\in\mathcal{C}} \rho_a x_{at} + \sum_{a\in\mathcal{C}} \sum_{\substack{b\in\mathcal{C}\\ b\neq a}} \varsigma_{ab} z_{abt} \leq K_t, \qquad t \in \mathcal{T} \tag{5}$$

$$x_{at} \leq \bar{M}_{at} \left(y_{at} + \sum_{\substack{b\in\mathcal{C}\\ b\neq a}} z_{bat} \right), \qquad a \in \mathcal{C}, t \in \mathcal{T} \tag{6}$$

$$y_{at} + \sum_{\substack{b\in\mathcal{C}\\ b\neq a}} z_{bat} = y_{a,t+1} + \sum_{\substack{b\in\mathcal{C}\\ b\neq a}} z_{abt}, \qquad a \in \mathcal{C}, t \in \mathcal{T} \tag{7}$$

$$\sum_{a\in\mathcal{C}} y_{at} = 1, \qquad t \in \mathcal{T} \tag{8}$$

$$\pi_{at} \geq \pi_{bt} + 1 - |\mathcal{C}| \left(1 - z_{bat}\right), \qquad a \in \mathcal{C}, b \in \mathcal{C}, t \in \mathcal{T} \tag{9}$$

$$\sum_{j\in\bar{\mathcal{C}}\cup\{n+1\}} w_{0jr} = 1, \qquad r = 1, \ldots, R \tag{10}$$

$$\sum_{i\in\bar{\mathcal{C}}\cup\{0\}} w_{i(n+1)r} = 1, \qquad r = 1, \ldots, R \tag{11}$$

$$\sum_{\substack{i\in\bar{\mathcal{C}}\cup\{0\}\\ i\neq j}} \sum_{r=1}^{R} w_{ijr} = 1, \qquad j \in \bar{\mathcal{C}} \tag{12}$$

$$\sum_{\substack{j\in\bar{\mathcal{C}}\cup\{n+1\}\\ j\neq i}} w_{ijr} = \sum_{\substack{j\in\bar{\mathcal{C}}\cup\{0\}\\ j\neq i}} w_{jir}, \qquad i \in \bar{\mathcal{C}}, r = 1, \ldots, R \tag{13}$$

$$\sum_{i\in\bar{\mathcal{C}}} w_{0ir} \geq \sum_{i\in\bar{\mathcal{C}}} w_{0i(r+1)}, \qquad r = 1,\ldots,R-1 \tag{14}$$

$$Q_{prt} \leq \min\left\{\left\lfloor \frac{\max\theta_v}{\varphi_p}\right\rfloor, \sum_{i\in\bar{\mathcal{C}}} d_{pi}\right\}\phi_{0rt}, \qquad p\in\mathcal{P}, r=1,\ldots,R, t\in\mathcal{T} \tag{15}$$

$$\sum_{t\in\mathcal{T}} Q_{prt} = \sum_{i\in\bar{\mathcal{C}}} d_{pi}\Bigg(\sum_{\substack{j\in\bar{\mathcal{C}}\cup\{n+1\}\\ j\neq i}} w_{ijr}\Bigg), \qquad p\in\mathcal{P}, r=1,\ldots,R \tag{16}$$

$$\sum_{t\in\mathcal{T}} \phi_{irt} = \sum_{\substack{j\in\bar{\mathcal{C}}\cup\{n+1\}\\ j\neq i}} w_{ijr}, \qquad i\in\bar{\mathcal{C}}, r=1,\ldots,R \tag{17}$$

$$\sum_{t\in\mathcal{T}} \phi_{0rt} = 1 - w_{0(n+1)r}, \qquad r=1,\ldots,R \tag{18}$$

$$\sum_{t\in\mathcal{T}} \phi_{0rt} = \sum_{t\in\mathcal{T}} \phi_{(n+1)rt}, \qquad r=1,\ldots,R \tag{19}$$

$$\sum_{t\in\mathcal{T}} \delta_{it}\phi_{irt} \leq \mu_{ir} \leq \sum_{t\in\mathcal{T}} \bar{\delta}_{it}\phi_{irt}, \qquad i\in\mathcal{N}, r=1,\ldots,R \tag{20}$$

$$\mu_{jr} \geq \mu_{0r} + \lambda\Bigg(\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} \varphi_p Q_{prt}\Bigg) + \tau_{0j} - M_{0j}\left(1 - w_{0jr}\right), \; j\in\bar{\mathcal{C}}, r=1,\ldots,R \tag{21}$$

$$\mu_{jr} \geq \mu_{ir} + s_i + \tau_{ij} - M_{ij}\left(1 - w_{ijr}\right), \quad \begin{array}{l} i\in\bar{\mathcal{C}}, j\in\bar{\mathcal{C}}\cup\{n+1\},\\ r=1,\ldots,R : i\neq j\end{array} \tag{22}$$

$$\mu_{ir} \leq \Delta_i \sum_{\substack{j\in\bar{\mathcal{C}}\cup\{n+1\}\\ j\neq i}} w_{ijr}, \qquad i\in\bar{\mathcal{C}}, r=1,\ldots,R \tag{23}$$

$$\mu_{0s} \geq \mu_{(n+1)r} - \bar{\delta}_{(n+1)|\mathcal{T}|}\Big(2 - \alpha_{rv} - \alpha_{sv}\Big), \quad v\in\mathcal{V}, r,s=1,\ldots,R : r<s \tag{24}$$

$$\sum_{p\in\mathcal{P}} \varphi_p\left(\sum_{t\in\mathcal{T}} Q_{prt}\right) \leq \sum_{v\in\mathcal{V}} \theta_v\alpha_{rv}, \qquad r=1,\ldots,R \tag{25}$$

$$\sum_{v\in\mathcal{V}} \alpha_{rv} = \sum_{t\in\mathcal{T}} \phi_{0rt}, \qquad r=1,\ldots,R \tag{26}$$

$$x_{at}, I_{at}, \pi_{at} \geq 0, \qquad a \in \mathcal{C}, t \in \mathcal{T} \tag{27}$$
$$\mu_{ir} \geq 0, \qquad i \in \mathcal{N}, r = 1, \ldots, R \tag{28}$$
$$Q_{prt} \geq 0, \qquad p \in \mathcal{P}, r = 1, \ldots, R, t \in \mathcal{T} \tag{29}$$
$$y_{at}, z_{abt} \in \{0, 1\}, \qquad a \in \mathcal{C}, b \in \mathcal{C}, t \in \mathcal{T} \tag{30}$$
$$\phi_{irt} \in \{0, 1\}, \qquad i \in \mathcal{N}, r = 1, \ldots, R, t \in \mathcal{T} \tag{31}$$
$$w_{ijr} \in \{0, 1\}, \qquad i, j \in \mathcal{N}, r = 1, \ldots, R \tag{32}$$
$$\alpha_{rv} \in \{0, 1\}, \qquad r = 1, \ldots, R, v \in \mathcal{V}. \tag{33}$$

The objective function (1) minimizes the sequence-dependent setup, inventory and routing costs. Inequalities (2) are the inventory balance constraints. Constraints (3) ensure that the quantity of item a shipped in period t (i.e., $\sum_{r=1}^{R} \sum_{p \in \mathcal{P}} \eta_{ap} Q_{prt}$) is not greater than the inventory at the end of period $t-1$. Constraints (4) impose a minimum inventory level at the end of each period and (5) represent the production capacity constraints.

Constraints (6) guarantee that there is production of item a in period t only if the line is set up for item a at the beginning of period t, or if there is a changeover from some item b to item a in period t. Note that these constraints also give an upper bound on the production quantity of item a in period t, $\bar{M}_{at} = \min \left\{ \lfloor \frac{K_t}{\rho_a} \rfloor, \sum_{i \in \bar{\mathcal{C}}} \sum_{p \in \mathcal{P}} \eta_{ap} d_{pi} \right\}$.

Equalities (7) relate the setup variables at the start of a period, y_{at}, with changeover variables, z_{abt}, in each period of the planning horizon. They are flow constraints and establish that if the line is set up for item a at the beginning of period t, then we have either a changeover from item a to some different item b in period t or the line is still set up for item a at the beginning of period $t+1$. Similarly, if there is a changeover from some item b to item a in period t, then we have either a changeover from item a to some different item b in period t or the line is still set up for item a at the beginning of period $t+1$. If the left-hand side is equal to zero, meaning that there is no production of item a in period t, then there cannot be a changeover from item a to any other item b in period t, nor can the line be set up for item a at the beginning of period $t+1$.

Equations (8) establish that the line is set up for exactly one item a at the beginning of each period t, even if the period is idle (i.e., there is no production in that period).

In constraints (9), π_{at} indicates the ordinal position of item a in period t. The larger the value of π_{at}, the later item a will be scheduled in period

t. Whenever a changeover from item b to item a takes place (i.e., $z_{bat} = 1$), the expression $|\mathcal{C}|\,(1 - z_{bat})$ is equal to zero, and (9) becomes $\pi_{at} \geq \pi_{bt} + 1$, which means that item a is scheduled after item b in period t. Therefore, constraints (9) avoid sub-tours and determine a unique sequence of items in each period.

Remember that the depot is represented by nodes 0 and $n+1$, so that a route starts by departing from node 0 and finishes by arriving at node $n+1$, as stated by constraints (10) and (11), respectively. We assume that a route traveling directly from node 0 to node $n+1$ is an empty route.

Constraints (12) establish that each customer must be visited exactly once over the planning horizon (i.e., split deliveries are not allowed), while flow conservation is guaranteed by the set of constraints (13).

Inequalities (14) are symmetry breaking constraints. They indicate that route $r+1$ can be used only if route r is used. Note that some routes may be empty, so these constraints force those routes to be the last ones.

Constraints (15) set the quantity of product p shipped on route r in period t (Q_{prt}) to zero if route r does not start in period t ($\phi_{0rt} = 0$) and equalities (16) determine the total quantity of product p shipped on route r. Since any non-empty route must satisfy $\sum_{t\in\mathcal{T}} \phi_{0rt} = 1$, and considering (15), it follows that at most one variable Q_{prt} on the left-hand side summation of (16) may be positive.

Constraints (17) guarantee that route r must visit customer i in just one period t, only if customer i belongs to route r. Equalities (18) and (19) ensure that each non-empty route starts and finishes in just one period (not necessarily the same), respectively. Note that $w_{0(n+1)r} = 1$ implies that route r was not used, so $\phi_{0rt} = \phi_{(n+1)rt} = 0$ for all $t \in \mathcal{T}$.

Inequalities (20) model time windows. If node i is not visited by route r (i.e., $\sum_{t\in\mathcal{T}} \phi_{irt} = 0$), then $\mu_{ir} = 0$. Conversely, if node i is visited by route r, then $\phi_{irt} = 1$ for some $t \in \mathcal{T}$ and the constraint becomes $\delta_{it} \leq \mu_{ir} \leq \bar{\delta}_{it}$.

Constraints (21)-(22) impose lower bounds on the time when node j is served by route r. Let j be the first customer on route r, then (21) establishes that the time when customer j starts being served is at least the start time of the route plus the loading time and the travel time between the depot and the customer location. Similarly, for the remaining nodes, $j \in \{\bar{\mathcal{C}}, n+1\}$, constraints (22) ensure that the time when the route starts serving customer j is at least the time when the previous customer i is served plus the service time at i and the travel time between i and j.

Deadline requirements are imposed by constraints (23), and a proper

order among routes assigned to the same vehicle is guaranteed by constraints (24). The term $\bar{\delta}_{(n+1)|\mathcal{T}|}$, the end of the time window of node $n+1$ in the last period of the planning horizon, denotes the latest time any vehicle may get back to the depot. In practice, $\bar{\delta}_{(n+1)|\mathcal{T}|}$ matches the end of the planning horizon, according to our assumption that the depot is always open for the return of vehicles.

Constraints (25) and (26) avoid the vehicle capacity being exceeded and ensure that each non-empty route is assigned to only one vehicle, respectively.

4. Solution method - A decomposition heuristic

In order to solve the integrated problem (1)-(33), we decompose it into two smaller subproblems which are solved iteratively until a given stopping criterion is met. Similar ideas have recently been used by Absi et al. [1] and by Chitsaz et al. [19].

In the first model, named LSDS (*lot-scheduling with direct shipments*), we decide when to produce, how much to produce and in what order to produce the items. Besides these decisions, the model also allocates customers to routes (without considering routing decisions explicitly) and routes to vehicles (taking into account the vehicle capacity). Finally, it decides the periods in which each route should leave from and return to the depot based on an estimation of the duration of the routes.

In this model, the routing costs and times are considered in an approximate way. A distribution cost c_{ir} is accounted if customer i is visited by route r, and a fixed cost c_r is considered if route r is used over the planning horizon. Similarly, to estimate the duration of route r, a distribution time $\bar{\tau}_{ir}$ is considered if customer i is served by route r. The objective of the LSDS model is to minimize the sum of production costs (changeover and inventory holding costs) and distribution costs.

After solving the LSDS model, the values of the production variables, the customers to be served by each route and the vehicle that should perform each route are known. The remaining decisions are the sequence and the time of the visits, which are made by solving a multi-trip vehicle routing problem with time windows (MTVRPTW model).

As we know beforehand the allocation of routes to vehicles, the MTVRPTW model can be decomposed into $|\mathcal{V}|$ smaller and independent subproblems, one per vehicle. It is not necessary to use all the vehicles in a feasible solution, so that the number of subproblems might be less than $|\mathcal{V}|$.

Solving the MTVRPTW subproblems may be infeasible because model LSDS does not consider the actual travel time between customer locations nor the customers time windows. When this situation arises, for each infeasible subproblem we iteratively add inequalities based on local branching that force the LSDS model to produce a different solution. The idea of using local branching constraints as a diversification mechanism has been successfully explored by Adulyasak et al. [3] and Chitsaz et al. [19] to solve a single item PRP.

Whenever a feasible solution is found (i.e., when all the MTVRPTW are feasible), a local search is applied aiming to reduce the routing cost. We have noticed that good routes, given the current assignment of customers, are found after solving the MTVRPTW subproblems. Therefore, we try to improve the current routes by applying local search operators that change the assignment of customers to routes, such as swap and reallocate operators.

The current solution is then used to update the approximations for the distribution costs and times for the next iteration of the model LSDS. Then, an inequality based on local branching is added to force the algorithm to produce a different solution. This scheme is repeated until the solution is not improved for a given number of iterations. Whenever this happens, a restart mechanism is applied and the whole procedure is repeated again. The method stops when the restart mechanism has been applied a given number of times. The main steps of the decomposition heuristic are shown in Algorithm 1 and the details of the different steps are given in the subsequent sections.

4.1. Initialization Phase

Before starting the procedure we define new parameters required by the LSDS model and set initial values for each of them (Algorithm 1, line 3), as follows:

c_r : Estimate of the fixed cost of using route r;

c_{ir} : Estimate of the cost of serving customer i in route r;

$\bar{\tau}_{ir}$: Estimate of the time required to serve customer i in route r.

Initially, we set c_{ir} as the minimum value between the cost of a round-trip from the depot and the cost of visiting the two nearest nodes to customer i, i.e.,

Algorithm 1 Decomposition Heuristic

set $sol \leftarrow \emptyset$ and $best\_sol \leftarrow \emptyset$;
set $restart \leftarrow 0$ and $no\_improvement \leftarrow 0$;
Initialize c_{ir}, c_r and $\bar{\tau}_{ir}$ for each $i \in \bar{\mathcal{C}}, r = 1, \ldots, R$;
repeat
  repeat
    Solve LSDS subproblem;
    for all $v \in \mathcal{V}$ **do**
      Solve a MTVRPTW subproblem;
    end for
    if All MTVRPTW subproblems are feasible **then**
      Update *sol*;
      Apply local search on *sol*;
      if *sol* is better than *best_sol* **then**
        set $best\_sol \leftarrow sol$;
        set $no\_improvement \leftarrow 0$;
      else
        set $no\_improvement \leftarrow no\_improvement+1$;
      end if
      Update approximations of distribution costs and times;
      Add local branching inequality to cut off the current solution;
    else
      for all $v \in \mathcal{V}$ **do**
        if subproblem v is infeasible **then**
          Add local branching inequality to cut off the current infeasible solution;
        end if
      end for
      set $no\_improvement \leftarrow no\_improvement+1$;
    end if
  until $no\_improvement \geq max\_no\_improvement$
  Apply restart mechanism;
  $restart \leftarrow restart+1$;
until $restart \geq max\_restart$
return *best_sol*

$$c_{ir} = \min \left\{ c_{0i} + c_{i(n+1)}, \min_{\substack{j,k \in \mathcal{N} \\ j \neq k}} (c_{ji} + c_{ik}) \right\}, \qquad i \in \bar{\mathcal{C}}, r = 1, \ldots, R. \tag{34}$$

Similarly, we set c_r as the cost of visiting the two nearest customers to the depot multiplied by a factor $\beta > 0$:

$$c_r = \beta \left\{ \min_{\substack{j,k \in \bar{\mathcal{C}} \\ j \neq k}} (c_{0j} + c_{k(n+1)}) \right\}, \qquad r = 1, \ldots, R. \tag{35}$$

Note that this is an optimistic estimate, since it assumes that each route always starts and ends with the two nearest customers to the depot, respectively.

Solving model LSDS without these costs would result in a solution completely driven by production costs for which it may be very difficult to find a feasible routing plan. Therefore, our idea is to take transportation costs into account approximately to drive the model to find solutions with a better trade-off between production and transportation costs. Thus, although model LSDS does not explicitly model routing decisions, it is aware that routing decisions play an important role in terms of the total cost [1, 10, 19].

We also notice that in spite of the fact that the original model (1)-(33) does not consider a fixed cost per route, in our approach this cost plays a relevant role as it forces the LSDS model to find solutions with fewer routes, which reduces the size of the subproblems to be solved in the second phase.

Finally, we introduce an estimate of the time required to serve a customer i in route r as the minimum value between the time of a round-trip from the depot and the time to visit customer i arriving from and departing to the two nearest nodes to customer i, respectively, plus twice the service time at customer i. We consider twice the service time in order to include both the time spent at the customer location and the time devoted to load its cargo at the depot:

$$\bar{\tau}_{ir} = \min \left\{ \tau_{0i} + \tau_{i(n+1)}, \min_{\substack{j,k \in \mathcal{N} \\ j \neq k}} (\tau_{ji} + \tau_{ik}) \right\} + 2s_i, \quad i \in \bar{\mathcal{C}}, r = 1, \ldots, R. \tag{36}$$

Similarly to (34), we use (36) as a way to approximately consider the time required to serve a given customer. Notice that the actual time is unknown for model LSDS since it does not consider routing decisions. However, having an idea of how long it would take to serve a customer helps us to reduce the risk of the MTVRPTW subproblems being infeasible.

4.2. First Phase: LSDS Model

As mentioned above, model LSDS decides when to produce, how much to produce and in what order to produce the items. It also assigns customers to routes, allocates routes to vehicles and decides the periods in which each route should leave from and return to the depot.

We consider the following sets of decision variables:

ε_{ir} : binary variable equal to 1 if customer i is served by route r, 0 otherwise;
γ_r : binary variable equal to 1 if route r is used, 0 otherwise.

The LSSD problem (Algorithm 1, line 6) can be formulated as follows:

$$\min \sum_{a\in\mathcal{C}}\sum_{t\in\mathcal{T}} h_a I_{at} + \sum_{a\in\mathcal{C}}\sum_{\substack{b\in\mathcal{C}\\ b\neq a}}\sum_{t\in\mathcal{T}} \hat{c}_{ab} z_{abt} + \sum_{i\in\bar{\mathcal{C}}}\sum_{r=1}^{R} c_{ir}\varepsilon_{ir} + \sum_{r=1}^{R} c_r\gamma_r \qquad (37)$$

subject to

- *Inventory constraints* (2)-(4),
- *Production capacity constraints* (5),
- *Setup constraints* (6)-(7),
- *Line configuration constraints* (8),
- *Sequencing constraints* (9),
- *Shipped quantity constraints* (15),
- *Timing constraints* (24),
- *Vehicle capacity constraints* (25),

$$\sum_{t\in\mathcal{T}} Q_{prt} = \sum_{i\in\bar{\mathcal{C}}} d_{pi}\varepsilon_{ir}, \qquad p\in\mathcal{P}, r=1,\ldots,R \tag{38}$$

$$\sum_{r=1}^{R} \varepsilon_{ir} = 1, \qquad i\in\bar{\mathcal{C}} \tag{39}$$

$$\sum_{t\in\mathcal{T}} \phi_{0rt} = \gamma_r, \qquad r=1,\ldots,R \tag{40}$$

$$\sum_{t\in\mathcal{T}} \phi_{(n+1)rt} = \gamma_r, \qquad r=1,\ldots,R \tag{41}$$

$$\sum_{v\in\mathcal{V}} \alpha_{rv} = \gamma_r, \qquad r=1,\ldots,R \tag{42}$$

$$\gamma_r \leq \sum_{i\in\bar{\mathcal{C}}} \varepsilon_{ir} \leq n\gamma_r, \qquad r=1,\ldots,R \tag{43}$$

$$\sum_{v\in\mathcal{V}} \alpha_{rv} \geq \sum_{v\in\mathcal{V}} \alpha_{(r+1)v}, \qquad r=1,\ldots,R-1 \tag{44}$$

$$\mu_{(n+1)r} - \mu_{0r} \geq \sum_{i\in\bar{\mathcal{C}}} \bar{\tau}_{ir}\varepsilon_{ir}, \qquad r=1,\ldots,R \tag{45}$$

$$\sum_{t\in\mathcal{T}} \delta_{it}\phi_{irt} \leq \mu_{ir} \leq \sum_{t\in\mathcal{T}} \bar{\delta}_{it}\phi_{irt}, \qquad i\in\{0,n+1\}, r=1,\ldots,R. \tag{46}$$

Constraints (38) guarantee that route r carries the amount of products required to satisfy the demand of all the customers in the route. Equalities (39) state that each customer must be visited once. Constraints (40)–(41) state that if route r is used, then it must leave from and return to the depot in exactly one period (not necessarily the same). Equations (42) guarantee that each selected route r is allocated to one vehicle. Conversely, if route r is not used, then it cannot be allocated to any vehicle. Constraints (43) force at least one customer to be assigned to each selected route. Similarly, these constraints avoid assigning customers to non-selected routes. Observe that constraints (40)–(43) altogether guarantee a coherent use of routes. Whenever a route is used, these constraints ensure that such a route starts and finishes in some period, is allocated to only one vehicle and, finally, must visit at least one customer. Inequalities (44) are a new set of symmetry breaking constraints. They ensure that route $r+1$ cannot be allocated to any vehicle if route r is not used (i.e., allocated to one vehicle). We use these constraints to replace (14). Constraints (45) impose a lower bound on the duration of each route, which depends on the customers assigned to the

route. Thus, whenever a customer i is assigned to route r, we add $\bar{\tau}_{ir}$ time units to the duration of the route. Notice that (45) only gives an estimation of the duration of a route, as the real duration depends on the sequence of visits, the customer time windows and the deadlines, which are not taken into account by the LSSD model. Finally, inequalities (46) denote the time window constraints at the depot. In the second phase, we allow the values of μ_{0r} and $\mu_{(n+1)r}$ to be recalculated, along with the remaining routing decisions (i.e., sequence and time of the visits). Although the LSDS model does not consider routing decisions, it is still difficult to solve as it is a multi-product lot-sizing and scheduling problem with distribution decisions.

4.3. Second Phase: MTVRPTW Model

A solution of model LSDS corresponds to a detailed production plan (production quantities, inventory level, production sequences, etc.), an assignment of vehicles and customers to routes, and the corresponding periods when each route starts and finishes. In the second phase, we construct a routing plan based on the decisions made in the first phase. Because the vehicles are independent of each other, we can obtain the routes by decomposing the routing problem into up to $|\mathcal{V}|$ subproblems and solving them independently (Algorithm 1, line 7–9).

Let N_R be the number of routes used in phase one and consider the following sets and parameters:

$\bar{C}_r$: set of customers served by route r, $r = 1, \ldots, N_R$;
S_r : set of all nodes served by route r, $r = 1, \ldots, N_R$. $S_r = \bar{C}_r \cup \{0, n+1\}$;
R_v : set of routes allocated to vehicle v, $v \in \mathcal{V}$;
t'_r : period in which route r starts, $r = 1, \ldots, N_R$;
t''_r : period in which route r finishes, $r = 1, \ldots, N_R$.

For each vehicle v, $v \in \mathcal{V}$, we proceed to solve the following formulation:

$$\min \sum_{r \in R_v} \sum_{i \in S_r} \sum_{j \in S_r} c_{ij} w_{ijr} \tag{47}$$

$$\sum_{j \in \bar{C}_r} w_{0jr} = 1, \qquad r \in R_v \tag{48}$$

$$\sum_{i \in \bar{C}_r} w_{i(n+1)r} = 1, \qquad r \in R_v \tag{49}$$

$$\sum_{\substack{i \in \bar{C}_r \cup \{0\} \\ i \neq j}} w_{ijr} = 1, \qquad r \in R_v, j \in \bar{C}_r \tag{50}$$

$$\sum_{\substack{i \in \bar{C}_r \cup \{0\} \\ i \neq j}} w_{ijr} = \sum_{\substack{i \in \bar{C}_r \cup \{n+1\} \\ i \neq j}} w_{jir}, \qquad r \in R_v, j \in \bar{C}_r \tag{51}$$

$$\delta_{0t'_r} \leq \mu_{0r} \leq \bar{\delta}_{0t'_r}, \qquad r \in R_v \tag{52}$$

$$\delta_{(n+1)t''_r} \leq \mu_{(n+1)r} \leq \bar{\delta}_{(n+1)t''_r}, \qquad r \in R_v \tag{53}$$

$$\sum_{t=t'_r}^{t''_r} \delta_{it} \phi_{irt} \leq \mu_{ir} \leq \sum_{t=t'_r}^{t''_r} \bar{\delta}_{it} \phi_{irt}, \qquad r \in R_v, i \in \bar{C}_r \tag{54}$$

$$\sum_{t=t'_r}^{t''_r} \phi_{irt} = 1, \qquad r \in R_v, i \in \bar{C}_r \tag{55}$$

$$\mu_{ir} \leq \Delta_i, \qquad r \in R_v, i \in \bar{C}_r \tag{56}$$

$$\mu_{0s} \geq \mu_{(n+1)r}, \qquad r \in R_v, s \in R_v : r < s \tag{57}$$

$$\mu_{jr} \geq \mu_{0r} + \lambda \left(\sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} \varphi_p Q_{prt} \right) + \tau_{0j} - M_{0j} \left(1 - w_{0jr} \right), \; r \in R_v, j \in \bar{C}_r \tag{58}$$

$$\mu_{jr} \geq \mu_{ir} + s_i + \tau_{ij} - M_{ij} \left(1 - w_{ijr} \right), \quad \begin{array}{l} r \in R_v, i \in \bar{C}_r, \\ j \in \bar{C}_r \cup \{n+1\}, j \neq i. \end{array} \tag{59}$$

Objective function (47) seeks to minimize the routing costs. Constraints (48)-(49) establish that each route must leave and return to the depot, respectively. Equalities (50) ensure that each customer is visited once, and constraints (51) guarantee the flow conservation. Inequalities (52) and (53) ensure that each route satisfies the time window requirements at the depot. From phase one, we already know that route r leaves node 0 in period t'_r and arrives to node $n+1$ in period t''_r, so it suffices to impose time window requirements only in those periods. In this way, we keep fixed the periods t'_r and t''_r where route r starts and finishes, respectively, but allow changes in the variables μ_{0r} and $\mu_{(n+1)r}$, which define the exact moment when route r leaves from and returns to the depot. Time window constraints at the customers are given by expression (54). Constraints (55) force each customer to

be served in one period. Notice that the period t in which a given customer is served must satisfy $t'_r \leq t \leq t''_r$. Constraints (56) and (57) establish that each customer must be served before its deadline and avoid that routes overlap in time, respectively. Finally, constraints (58) and (59) impose lower bounds on the starting time at which each node is served.

4.4. Local Search Operator

A feasible solution for the original problem is available anytime CPLEX finds feasible routes for all the subproblems (47)-(59). To improve this feasible solution, a simple local search operator, named *Reallocate*, which removes a customer i from its current route r_1 and reinserts it in a different route r_2, is implemented (Algorithm 1, line 12). This kind of movement aims to reduce the routing costs by implicitly changing the current assignment of customers to routes. All the feasible moves in terms of routing constraints (i.e., the ones that do not violate vehicle capacities, time windows and deadlines) are evaluated and the one leading to the greatest reduction in the routing cost is chosen.

Notice, however, that changes in the routes may entail changes in the current production plan and, therefore, production decisions should be re-optimized. Consequently, the best move is only accepted if, after re-optimizing the production decisions, the overall cost has been reduced.

In order to re-optimize the production decisions, we fix all the variables related to the distribution decisions in the model LSDS (i.e., ε_{ir}, α_{rv}, γ_r, Q_{prt}, ϕ_{irt}, μ_{ir}), and run CPLEX to determine the values of the remaining variables (i.e., x_{at}, I_{at}, y_{at}, z_{abt}, π_{at}). This re-optimization step may turn out to be infeasible, since the local search operator does not take into account the impact of a given move on the production decisions. Whenever this happens, we discard the best move and make a new attempt with the second best, the third one, and so on, until either a feasible production plan is achieved or the κ best moves have been evaluated.

We also implemented a second local search operator in which customer i_1 from route r_1 and customer i_2 from route r_2, $r_1 \neq r_2$, are exchanged. However, computational tests showed that this operator is computationally more expensive and does not lead to significant improvements in solution quality.

4.5. Updating Distribution Times and Costs

Let $\mathcal{R}$ be the set of routes found in phase two, after applying the local search. For each route $r \in \mathcal{R}$, D_r denotes the duration of route r and, for each $i \in S_r$, i^- and i^+ represent the predecessor and successor of customer i in route r, respectively. For each customer i belonging to route r, $i \in S_r$, D_{ri}^- denotes the duration of route r if customer i were removed from the route. Similarly, for each $i \notin S_r$, Ω_{ir} is the cheapest cost of inserting customer i in route r and D_{ri}^+ expresses the duration of route r if customer i were inserted in the cheapest position of the route. Algorithm 2, based on Absi et al. [1], shows how to update both c_{ir} and $\bar{\tau}_{ir}$, respectively (Algorithm 1, line 19).

Note that in Absi et al. [1] only the cost needs to be updated, whereas in our problem we need to update the approximation for both the distribution cost and time.

Algorithm 2 Update approximations of distribution costs and times

1: **for all** $r \in \mathcal{R}$ **do**
2: **for all** $i \in \bar{\mathcal{C}}$ **do**
3: **if** $i \in S_r$ **then**
4: $c_{ir} \leftarrow c_{i^-i} + c_{ii^+} - c_{i^-i^+}$
5: $\bar{\tau}_{ir} \leftarrow D_r - D_{ri}^-$
6: **else**
7: $c_{ir} \leftarrow \Omega_{ir}$
8: $\bar{\tau}_{ir} \leftarrow D_{ri}^+ - D_r$
9: **end if**
10: **end for**
11: **end for**

The main idea of Algorithm 2 is to update model LSDS with information about the clustering of customers. If customer i is served by route r, then $c_{i^-i} + c_{ii^+} - c_{i^-i^+}$ represents how much would be saved if i were removed from r. In this case, if c_{ir} is large, customer i has a higher probability to be assigned to a different route in the next iteration of phase one. On the other hand, if customer i is not served by route r, then Ω_{ir} represents how much the cost would increase if i were inserted in route r. In this case, if Ω_{ir} is small, customer i has a higher probability to be assigned to route r in the next iteration of phase one.

We use the same logic to update $\bar{\tau}_{ir}$. When customer i belongs to route r, $D_r - D_{ri}^-$ represents how much the duration of route r would be reduced

if customer i were removed from it. Conversely, when customer i does not belong to route r, $D_{ri}^{+} - D_r$ denotes how much the duration of route r would increase if customer i were inserted in the cheapest position of this route.

Algorithm 2 might also be seen as a mechanism to diversify the search. If the distribution costs were not updated at each iteration, the LSDS model would have no incentive to find a different solution at each iteration.

4.6. Local Branching Inequalities

During the execution of our algorithm, we iteratively add inequalities in order to (i) eliminate infeasible solutions, and (ii) force the LSDS model to produce a different solution at each iteration. These inequalities are based on the local branching approach of Fischetti and Lodi [23]. Let $\left(\bar{\gamma}_r, \bar{\varepsilon}_{ir}, \bar{\alpha}_{rv}, \bar{\phi}_{irt}\right)$ be the values of the variables $(\gamma_r, \varepsilon_{ir}, \alpha_{rv}, \phi_{irt})$ in a given iteration. The inequality we add to generate a new solution is the following (Algorithm 1, line 20):

$$\sum_{\substack{r=1:\\ \bar{\gamma}_r=1}}^{R} (1-\gamma_r) + \sum_{\substack{r=1:\\ \bar{\gamma}_r=0}}^{R} \gamma_r + \sum_{r=1}^{R} \sum_{\substack{i\in\bar{\mathcal{C}}:\\ \bar{\varepsilon}_{ir}=1}} (1-\varepsilon_{ir}) + \sum_{r=1}^{R} \sum_{\substack{i\in\bar{\mathcal{C}}:\\ \bar{\varepsilon}_{ir}=0}} \varepsilon_{ir} +$$
$$\sum_{r=1}^{R} \sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{0rt}=1}} (1-\phi_{0rt}) + \sum_{r=1}^{R} \sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{0rt}=0}} \phi_{0rt} + \sum_{r=1}^{R} \sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{(n+1)rt}=1}} \left(1-\phi_{(n+1)rt}\right) +$$
$$\sum_{r=1}^{R} \sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{(n+1)rt}=0}} \phi_{(n+1)rt} + \sum_{r=1}^{R} \sum_{\substack{v\in\mathcal{V}:\\ \bar{\alpha}_{rv}=1}} (1-\alpha_{rv}) + \sum_{r=1}^{R} \sum_{\substack{v\in\mathcal{V}:\\ \bar{\alpha}_{rv}=0}} \alpha_{rv} \geq 1. \quad (60)$$

This inequality forces at least one of the variables $(\gamma_r, \varepsilon_{ir}, \alpha_{rv}, \phi_{irt})$ to take a different value in the next iteration. Similarly, we also add an inequality for each subproblem v that appears to be infeasible in the second phase of the algorithm, as follows (Algorithm 1, line 24):

$$\sum_{r\in R_v} \sum_{\substack{i\in\bar{\mathcal{C}}:\\ \bar{\varepsilon}_{ir}=1}} (1-\varepsilon_{ir}) + \sum_{r\in R_v} \sum_{\substack{i\in\bar{\mathcal{C}}:\\ \bar{\varepsilon}_{ir}=0}} \varepsilon_{ir} + \sum_{r\in R_v} \sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{0rt}=1}} (1-\phi_{0rt}) +$$

$$\sum_{r\in R_v}\sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{0rt}=0}}\phi_{0rt}+\sum_{r\in R_v}\sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{(n+1)rt}=1}}\left(1-\phi_{(n+1)rt}\right)+\sum_{r\in R_v}\sum_{\substack{t\in\mathcal{T}:\\ \bar{\phi}_{(n+1)rt}=0}}\phi_{(n+1)rt}\geq 1. \tag{61}$$

Inequality (61) guarantees that the current schedule of vehicle v will not be assigned to this or any other vehicle in later iterations, since CPLEX has proven it is infeasible. As we take into account the vehicle capacity constraints in the LSDS model, subproblem v may only be infeasible because it cannot meet timing constraints (i.e., time windows, deadlines, etc.). Therefore, inequality (61) is valid not only for vehicle v, but for all the vehicles.

4.7. Restart Mechanism

We propose a mechanism to restart the whole procedure whenever the best solution found so far has not been improved for a given number of iterations (Algorithm 1, line 30). To this end, we implement a randomized version of the well-known *Savings* heuristic [20], in which at each iteration we randomly choose an arc instead of using the traditional criterion of choosing the arc with the best saving. In this way, each time the *Savings* heuristic is applied a different set of routes may be found. This idea has been recently explored by Caceres-Cruz et al. [16] to solve the heterogeneous fixed fleet vehicle routing problem with multi-trips.

During this procedure we suppose that we have a fleet of n homogeneous vehicles. Each vehicle may perform at most one trip over the planning horizon and its capacity is $\theta = \min_{v\in\mathcal{V}}\theta_v$. Notice that under these assumptions the constructed routes might not be feasible for the original problem. However, here our idea is not to find a set of feasible routes for the original problem, but to identify clusterings of customers in order to reset the values of c_{ir}.

Algorithm 3 shows how the randomized *Savings* heuristic works. In this algorithm, s_{ij} denotes the saving of the arc (i, j) and r_i denotes the index of the route serving customer i. An important step of this algorithm is how to choose an arc to merge two routes. In the standard *Savings* heuristic, at each iteration the algorithm always chooses the arc with the largest saving. In our implementation we randomly select an arc from the list, based on a probability assigned to each arc.

For this purpose, we use a geometric distribution during the process: every time a new arc is selected from the list $\mathcal{E}$ of available arcs (which are ordered in a decreasing way according to its savings potential), a value ϵ is

Algorithm 3 Randomized Savings Heuristic

$\mathcal{R} \leftarrow \emptyset$;
set an initially empty list of arcs $\mathcal{E} \leftarrow \emptyset$;
for all $i \in \bar{\mathcal{C}}$ **do**
  Construct a route serving only customer i, $R_i = (i)$;
  set $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_i\}$;
  set $r_i \leftarrow i$;
end for
for all $i \in \bar{\mathcal{C}}$ **do**
  for all $j \in \bar{\mathcal{C}} : j \neq i$ **do**
    $s_{ij} \leftarrow c_{i0} + c_{0j} - c_{ij}$;
    set $\mathcal{E} \leftarrow \mathcal{E} \cup \{(i, j)\}$;
  end for
end for
Sort $\mathcal{E}$ in decreasing order of s_{ij};
while $\mathcal{E} \neq \emptyset$ **do**
  set $(i, j) \leftarrow$ *ChooseArc*$(\mathcal{E})$;
  set $u_1 \leftarrow r_i$;
  set $u_2 \leftarrow r_j$;
  if all merging conditions are satisfied **then**
    MergeRoutes$(R_{u_1}, R_{u_2}, (i, j))$;
  end if
  Remove (i, j) from $\mathcal{E}$
end while
return $\mathcal{R}$

Algorithm 4 ChooseArc($\mathcal{E}$)

1: $\epsilon \leftarrow GenerateRandomValue(0.05, 0.25)$;
2: $RdmValue \leftarrow GenerateRandomValue(0, 1)$;
3: $n \leftarrow 0$;
4: $arcPr \leftarrow 0$;
5: $cumPr \leftarrow 0$;
6: **for all** $e \in \mathcal{E}$ **do**
7: $\quad arcPr \leftarrow \epsilon\,(1-\epsilon)^n$;
8: $\quad cumPr \leftarrow cumPr + arcPr$;
9: $\quad$ **if** $RdmValue < cumPr$ **then**
10: $\quad\quad$ **return** e;
11: $\quad$ **else**
12: $\quad\quad n \leftarrow n + 1$;
13: $\quad$ **end if**
14: **end for**
15: Choose randomly an arc e from $\mathcal{E}$;
16: **return** e;

drawn from a uniform distribution on $[a, b]$. The value of ϵ is the parameter of the geometric distribution that will be used to assign exponentially diminishing probabilities to each arc according to its position in the list [16, 28]. Hence, arcs with larger savings will have a higher probability of being selected than those with smaller savings. Algorithm 4 shows the procedure we use to choose an arc at each iteration.

Note that when using a geometric distribution, a very large number of arcs in the list $\mathcal{E}$ might be required for the cumulative probability to converge to one. Since the size of $\mathcal{E}$ is finite and becomes smaller at each iteration, we might reach the end of the list without finding an arc with the proper cumulative probability (Algorithm 4, line 9). In this case, we simply randomly choose an arc from the list (Algorithm 4, line 15). In our experiments, we use the interval $[0.05, 0.25]$ to draw the values of the parameter ϵ.

After selecting an arc it is necessary to verify if all the conditions for merging are met. In this step, besides the vehicle capacity constraints, we also need to take into account the time window constraints. Due to the presence of time windows, we must account for route orientation. Two partial routes serving customers i and j, respectively, have compatible orientations if i is the first (last) customer in its route, and j is the last (first) customer

Algorithm 5 MergeRoutes(R_{u_1}, R_{u_2}, (i, j))

if i is the first customer of R_{u_1} **and** j is the last customer of R_{u_2} **then**
  $R_{u_2} \leftarrow R_{u_2}||(j, i)||R_{u_1}$;
  for all $i \in R_{u_1}$ **do**
    set $r_i \leftarrow u_2$
  end for
  Delete R_{u_1} from $\mathcal{R}$;
else
  $R_{u_1} \leftarrow R_{u_1}||(i, j)||R_{u_2}$;
  for all $j \in R_{u_2}$ **do**
    set $r_j \leftarrow u_1$
  end for
  Delete R_{u_2} from $\mathcal{R}$;
end if

in its route. This means that admissible arcs are from the last customer on one route to the first customer on the other one. Algorithm 5 shows how the merging of two routes is done. The symbol || stands for the concatenation of two sequences of clients.

Let $\mathcal{R}$ be the outcome (i.e., a set of routes) of Algorithm 3 and S_r the set of customers served by a given route $r \in \mathcal{R}$. As in Algorithm 2, i^- and i^+ denote the predecessor and successor of customer i in route r, respectively, and for each $i \notin S_r$, Ω_{ir} is the cheapest cost of inserting customer i in route r. The restart mechanism resets the distribution costs c_{ir} according to Algorithm 6. This algorithm works similarly to Algorithm 2. The general idea is to reset the approximations of the distribution costs by using the set of routes $\mathcal{R}$, instead of using the routes of the current solution, as done by Algorithm 2. As the cardinality of $\mathcal{R}$ might be less than R, we reset the distribution costs of routes $r = |\mathcal{R}| + 1$ to R to their initial values (see Section 4.1).

5. Computational Experiments

In order to evaluate the efficiency of both the integrated model and our heuristic approach, we perform extensive computational experiments on a set of randomly generated instances described in Section 5.1. The model and the heuristic were coded in C++ using the IBM Concert Technology and solved by CPLEX 12.6.1. For the integrated model, we use the CPLEX de-

Algorithm 6 Reset Distribution Costs

for all $r \in \mathcal{R}$ **do**
  for all $i \in \bar{\mathcal{C}}$ **do**
    if $i \in S_r$ **then**
      $c_{ir} \leftarrow c_{i^-i} + c_{ii^+} - c_{i^-i^+}$
    else
      $c_{ir} \leftarrow \Omega_{ir}$
    end if
  end for
end for
for $r = |\mathcal{R}| + 1$ **to** R **do**
  for $i \in \bar{\mathcal{C}}$ **do**
    $c_{ir} \leftarrow \min \left\{ c_{0i} + c_{i(n+1)}, \min_{\substack{j,k \in \mathcal{N} \\ j \neq k}} (c_{ji} + c_{ik}) \right\}$
  end for
end for

fault settings and a maximum computational time limit of twenty-four hours was imposed for each instance. Experiments with different CPLEX settings suggested that the default ones provided the best average performance.

For the heuristic, at each iteration, the LSDS model is solved using CPLEX 12.6.1. To avoid long computing times, we set the CPLEX MIP emphasis to feasibility and stop the execution when the branch-and-bound tree exceeds 5,000 nodes and a feasible solution has been found. Otherwise, we let CPLEX run until a first feasible solution is found.

In the second phase, we solve each MTVRPTW subproblem by running CPLEX for up to 60 seconds. Preliminary tests pointed out that CPLEX may have difficulties to prove the infeasibility of a given subproblem. Therefore, whenever the solver is not able to find a feasible solution before reaching the preset computational time limit, we assume that the subproblem is infeasible and hence add inequality (61) to the LSDS model to cut off the current solution. Following this strategy we possibly cut off a feasible solution, but we limit the computing times. This is very important when solving instances of considerable size, where CPLEX may take a long time to prove that a given MTVRPTW subproblem is infeasible.

For the local search, we set κ equal to five, which means that at most the five best moves will be evaluated. The restart mechanism is applied when

Demand of product p at customer i	$d_{pi} \in U[10, 100]$
Units of item a per unit of product p	$\eta_{ap} \in U[0, 5]$
Width (W_a) and length (L_a) of item a	$W_a, L_a \in U[5, 10]$
Coordinates of node i	$(X_i, Y_i) \in [0, 500]$
Initial (minimum) stock of item a	$I_{a0} = I_a^{\min} = 0$
Unit processing time of item a	$\rho_a = 1$
Unit weight of product p	$\varphi_p = 1$
Production capacity in period t	$K_t = \left\lceil \frac{\sum_{a \in \mathcal{C}} \rho_a \left(\sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} d_{pi} \eta_{ap} \right)}{0.6 \cdot \lvert \mathcal{T} \rvert} \right\rceil$
Setup time between items a and b	$\varsigma_{ab} \in U[6, 10]$ if $a \neq b$, 0 otherwise
Travel time from nodes i to node j	$\tau_{ij} = \left\lceil \frac{60}{80} \left(\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \right) \right\rceil$
Capacity of vehicle v	$\theta_v = \left\lceil U[0.25, 0.50] \cdot \sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} d_{pi} \cdot \varphi_p \right\rceil$
Time window of node $i \in \{0, \bar{\mathcal{C}}\}$ in period t	$\left[\delta_{it}, \bar{\delta}_{it}\right] = [480 + 1440\,(t-1)\,, 1080 + 1440\,(t-1)]$
Time window of node $n+1$ in period t	$\left[\delta_{(n+1)t}, \bar{\delta}_{(n+1)t}\right] = [1440\,(t-1)\,, 1440t]$
Time to load/unload per unit of weight	$\lambda = 1/50$
Due date of customer $i \in \bar{\mathcal{C}}$	$\Delta_i = \bar{\delta}_{i\lvert\mathcal{T}\rvert}$
Maximum number of routes over the planning horizon	$R = n$
Unit inventory holding cost of item a	$h_a = 0.001 \cdot W_a \cdot L_a$
Setup cost of a changeover from item a to item b	$\hat{c}_{ab} = 25 \cdot \varsigma_{ab}$
Travel cost from node i to node j	$c_{ij} = \left\lceil \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \right\rceil$

Table 1: Parameters used to generate random instances.

the best solution found so far has not been improved for five consecutive iterations, and the whole procedure is stopped after the restart mechanism has been applied five times.

All the experiments were executed on computers with two Intel Xeon 3.07 GHz processors and 96 GB of memory.

5.1. Instances Generation

We generated a set of random instances following an approach similar to the one proposed by Gramani et al. [26], for a cutting stock problem, and Armentano et al. [8], for a multi-product PRP. The number of final products ($|\mathcal{P}|$), items ($|\mathcal{C}|$), customers (n) and vehicles ($|\mathcal{V}|$) are chosen in the sets {3, 5}, {3, 5}, {10, 15, 20, 30, 40, 50} and {2, 3}, respectively. The planning horizon is fixed to 8 periods for all the instances. We generate five instances for each combination of final products, items, customers and vehicles, resulting in a total of 240 instances. Table 1 presents how the remaining parameters were generated.

Observe that we defined the width and length for each item a in order to estimate inventory holding costs as a function of the size of the items. It means that the larger a given item is, the more expensive it is to hold it in stock. The overall production capacity usage is defined to be around 60%,

following the work of Amorim et al. [6]. For the time windows generation, we assume that customers may be visited at any time within regular business hours, vehicles must also leave the depot within regular working hours, but they may return at any time. Furthermore, since we considered a relatively short planning horizon (8 days), we supposed that the customer deadlines match the end of the last time windows. The time to load/unload per unit of weight was set to 1/50 min/kg, which implies that fifty kilograms are loaded/unloaded in one minute. Finally, we set the upper bound on the number of routes as the number of customers in the problem.

5.2. Numerical Results

In this section we present the computational results on the random instances described in Section 5.1. Tables 2–7 compare results achieved by the integrated model and our decomposition heuristic. All the columns are self-explanatory, except for column "Dev(%)", which reports the relative deviation from the CPLEX solutions. Each entry is calculated as:

$$\text{Dev}(\%) = \frac{UB^H - UB^{MIP}}{UB^{MIP}} \times 100,$$

where UB^{MIP} denotes the total cost of the best solution found by CPLEX and UB^H is the total cost of the best solution found by the decomposition heuristic. Negative entries correspond to instances where our heuristic found a better solution than CPLEX.

In general, after 24 hours of computing time, CPLEX is not able to solve most of the instances. Indeed, only two instances with 10 customers were solved optimally (see instances 4 and 6 in Table 2). This suggests that, even after long runtimes, proving the optimality of a given solution is difficult. The main reason for it is the weakness of the lower bounds provided by the formulation. Except for a few instances with $n = 10$ customers, the optimality gaps are large and, therefore, reaching a guarantee of optimality is unlikely. Furthermore, the weak lower bounds make it difficult to accurately measure the quality of the solutions provided by CPLEX. In some cases, CPLEX may have found an optimal solution but it is not able to prove it because of the poor quality of the lower bounds. This is the case, for example, for the sixth instance with 10 customers, whose optimal solution was found in less than one hour, but the solver took more than ten hours to prove it.

Another drawback of solving the integrated model is that CPLEX is not able to provide feasible solutions for most of the instances with 30 or more

						MIP				Heuristic			
Instance	n	$\|\mathcal{P}\|$	$\|\mathcal{C}\|$	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	UB^{MIP}	Gap(%)	CPU(s)	Nodes	UB^{H}	CPU(s)	Iterations	Dev(%)
1	10	3	3	8	2	4,013.31	12.93	86,400.00	1,631,909	**3,960.76**	457.34	32	−1.31
2	10	3	3	8	2	**3,873.99**	4.50	86,400.00	5,885,924	3,908.74	494.55	40	0.90
3	10	3	3	8	2	**3,522.77**	16.09	86,400.00	3,611,059	3,524.52	474.04	45	0.05
4	10	3	3	8	2	3,803.46	0.00	8,732.46	209,729	**3,803.46**	375.02	32	0.00
5	10	3	3	8	2	**3,754.05**	9.46	86,400.00	6,694,815	3,779.69	402.98	30	0.68
6	10	3	3	8	3	4,074.22	0.00	36,744.30	962,256	**4,074.22**	475.80	48	0.00
7	10	3	3	8	3	3,817.56	9.22	86,400.00	2,958,690	**3,817.56**	289.26	30	0.00
8	10	3	3	8	3	4,668.97	10.54	86,400.00	5,764,870	**4,668.97**	326.45	32	0.00
9	10	3	3	8	3	3,786.86	4.72	86,400.00	5,151,117	**3,786.86**	443.60	34	0.00
10	10	3	3	8	3	4,237.05	7.38	86,400.00	2,245,959	**4,237.05**	475.10	47	0.00
11	10	3	5	8	2	5,250.37	29.00	86,400.00	1,649,734	**5,192.89**	720.42	41	−1.09
12	10	3	5	8	2	**6,786.34**	34.33	86,400.00	1,492,934	7,005.79	762.86	34	3.23
13	10	3	5	8	2	**6,695.40**	31.88	86,400.00	1,714,283	6,971.00	917.48	40	4.12
14	10	3	5	8	2	6,017.04	36.66	86,400.00	1,718,036	**5,809.29**	579.36	34	−3.45
15	10	3	5	8	2	**4,229.14**	15.23	86,400.00	1,765,188	4,260.97	478.40	31	0.75
16	10	3	5	8	3	**6,676.29**	29.97	86,400.00	2,740,611	7,026.02	477.34	30	5.24
17	10	3	5	8	3	**5,214.48**	24.93	86,400.00	1,551,910	5,223.06	485.87	39	0.16
18	10	3	5	8	3	**5,460.79**	24.63	86,400.00	3,110,250	5,473.35	690.19	42	0.23
19	10	3	5	8	3	**7,411.21**	24.79	86,400.00	1,762,620	7,509.22	656.17	42	1.32
20	10	3	5	8	3	**5,261.29**	30.58	86,400.00	2,432,910	5,370.88	551.21	30	2.08
21	10	5	3	8	2	**5,425.21**	17.96	86,400.00	2,000,556	5,435.74	491.00	27	0.19
22	10	5	3	8	2	3,972.29	13.97	86,400.00	1,000,102	**3,970.91**	546.29	33	−0.03
23	10	5	3	8	2	**4,376.42**	7.81	86,400.00	1,024,896	4,414.69	966.17	32	0.87
24	10	5	3	8	2	**4,852.70**	11.00	86,400.00	1,849,466	4,852.70	548.57	27	0.00
25	10	5	3	8	2	**4,055.38**	13.52	86,400.00	3,865,027	4,111.78	1,285.51	41	1.39
26	10	5	3	8	3	**4,680.26**	3.97	86,400.00	1,429,067	4,699.06	685.11	32	0.40
27	10	5	3	8	3	**4,246.78**	16.66	86,400.00	1,620,357	4,284.70	685.81	29	0.89
28	10	5	3	8	3	**5,545.58**	8.33	86,400.00	1,077,560	5,634.18	636.94	36	1.60
29	10	5	3	8	3	**4,444.97**	5.02	86,400.00	1,294,110	4,454.68	808.92	39	0.22
30	10	5	3	8	3	**4,872.25**	15.27	86,400.00	1,667,163	4,892.68	664.71	33	0.42
31	10	5	5	8	2	**7,034.80**	31.85	86,400.00	1,808,379	7,080.40	1,204.79	36	0.65
32	10	5	5	8	2	**7,291.19**	36.26	86,400.00	1,207,982	7,309.21	1,429.54	47	0.25
33	10	5	5	8	2	**7,868.27**	34.88	86,400.00	1,148,781	8,101.68	1,189.39	34	2.97
34	10	5	5	8	2	**6,058.22**	31.28	86,400.00	2,639,269	6,943.72	1,258.91	35	14.62
35	10	5	5	8	2	**7,446.51**	31.49	86,400.00	771,880	7,833.31	1,109.13	37	5.19
36	10	5	5	8	3	7,176.16	30.85	86,400.00	1,056,992	**7,117.60**	600.34	31	−0.82
37	10	5	5	8	3	7,105.11	34.60	86,400.00	516,184	**7,103.89**	1,149.36	44	−0.02
38	10	5	5	8	3	**7,148.19**	32.14	86,400.00	531,922	7,433.28	802.37	32	3.99
39	10	5	5	8	3	**5,742.64**	30.59	86,400.00	639,393	5,849.67	899.90	38	1.86
40	10	5	5	8	3	**6,462.28**	28.89	86,400.00	769,816	6,712.22	830.60	31	3.87
Mean						**5,359.00**	19.83	83,216.92	2,074,343	5,441.01	708.17	36	1.29

Table 2: Computational results for instances with $n = 10$ customers

						MIP				Heuristic			
Instance	n	$\|\mathcal{P}\|$	$\|\mathcal{C}\|$	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	UB^{MIP}	Gap(%)	CPU(s)	Nodes	UB^{H}	CPU(s)	Iterations	Dev(%)
41	15	3	3	8	2	**3,890.65**	20.81	86,400.00	2,613,910	3,964.35	369.94	32	1.89
42	15	3	3	8	2	5,625.78	20.78	86,400.00	2,877,480	**5,602.20**	565.84	35	−0.42
43	15	3	3	8	2	**4,741.37**	17.31	86,400.00	451,270	4,863.58	367.93	33	2.58
44	15	3	3	8	2	**5,392.16**	13.97	86,400.00	459,212	5,529.34	548.16	31	2.54
45	15	3	3	8	2	5,134.40	24.82	86,400.00	426,450	**4,999.58**	916.43	41	−2.63
46	15	3	3	8	3	**5,228.28**	26.28	86,400.00	2,018,792	5,303.27	846.09	40	1.43
47	15	3	3	8	3	5,295.35	17.00	86,400.00	823,550	**5,295.35**	479.33	31	0.00
48	15	3	3	8	3	**5,768.05**	14.38	86,400.00	994,886	5,778.38	346.96	32	0.18
49	15	3	3	8	3	**5,945.70**	12.26	86,400.00	625,683	5,954.12	542.92	43	0.14
50	15	3	3	8	3	4,725.65	10.58	86,400.00	829,435	**4,720.06**	504.29	40	−0.12
51	15	3	5	8	2	**7,725.78**	41.84	86,400.00	152,537	7,903.25	1,056.88	31	2.30
52	15	3	5	8	2	5,811.95	39.00	86,400.00	1,670,815	**5,500.22**	1,049.00	37	−5.36
53	15	3	5	8	2	**4,870.16**	42.75	86,400.00	70,471	4,973.04	516.84	36	2.11
54	15	3	5	8	2	5,626.91	37.39	86,400.00	167,752	**5,582.66**	1,647.26	45	−0.79
55	15	3	5	8	2	5,946.17	38.41	86,400.00	108,339	**5,709.59**	504.53	30	−3.98
56	15	3	5	8	3	7,418.60	38.51	86,400.00	67,361	**6,955.31**	491.22	30	−6.24
57	15	3	5	8	3	7,310.39	40.86	86,400.00	183,744	**7,305.61**	960.79	33	−0.07
58	15	3	5	8	3	7,017.20	36.85	86,400.00	249,881	**6,858.64**	830.78	34	−2.26
59	15	3	5	8	3	6,436.70	43.68	86,400.00	60,725	**6,270.49**	766.88	36	−2.58
60	15	3	5	8	3	**5,931.46**	33.33	86,400.00	527,510	5,994.97	1,353.13	36	1.07
61	15	5	3	8	2	**6,021.58**	22.35	86,400.00	619,732	6,071.01	1,045.19	37	0.82
62	15	5	3	8	2	5,076.98	20.56	86,400.00	1,068,640	**5,048.38**	1,778.98	52	−0.56
63	15	5	3	8	2	**5,688.68**	23.48	86,400.00	153,032	5,709.68	1,028.89	40	0.37
64	15	5	3	8	2	**6,631.85**	17.40	86,400.00	248,257	6,894.18	745.13	33	3.96
65	15	5	3	8	2	**6,312.77**	20.72	86,400.00	1,818,213	6,361.23	1,897.46	34	0.77
66	15	5	3	8	3	**6,089.43**	26.87	86,400.00	1,350,346	6,145.43	1,433.92	45	0.92
67	15	5	3	8	3	5,769.22	17.69	86,400.00	551,105	**5,769.22**	1,257.82	30	0.00
68	15	5	3	8	3	5,657.16	27.40	86,400.00	170,008	**5,445.72**	739.88	33	−3.74
69	15	5	3	8	3	**4,115.19**	8.28	86,400.00	1,355,528	4,117.38	854.24	35	0.05
70	15	5	3	8	3	4,577.99	9.47	86,400.00	274,991	**4,577.99**	1,098.22	36	0.00
71	15	5	5	8	2	**8,270.59**	40.18	86,400.00	110,036	8,277.13	1,285.29	36	0.08
72	15	5	5	8	2	7,765.48	34.75	86,400.00	123,732	**7,758.50**	1,402.92	39	−0.09
73	15	5	5	8	2	**8,123.29**	40.63	86,400.00	111,480	8,136.01	1,298.68	36	0.16
74	15	5	5	8	2	8,157.03	34.51	86,400.00	89,412	**8,143.78**	3,009.87	46	−0.16
75	15	5	5	8	2	8,130.90	42.33	86,400.00	120,191	**7,812.74**	2,572.07	36	−3.91
76	15	5	5	8	3	**5,903.06**	35.95	86,400.00	129,108	5,919.30	2,104.34	42	0.28
77	15	5	5	8	3	8,249.39	39.05	86,400.00	42,184	**8,042.92**	3,296.21	42	−2.50
78	15	5	5	8	3	9,071.73	41.00	86,400.00	76,603	**8,367.62**	1,444.69	38	−7.76
79	15	5	5	8	3	8,323.04	38.88	86,400.00	124,170	**8,252.95**	1,034.51	37	−0.84
80	15	5	5	8	3	7,301.59	39.40	86,400.00	76,492	**7,261.40**	2,502.07	42	−0.55
Mean						6,276.99	28.79	86,400.00	599,827	**6229.41**	1,162.39	37	−0.57

Table 3: Computational results for instances with $n = 15$ customers

						MIP				Heuristic			
Instance	n	$\|\mathcal{P}\|$	$\|\mathcal{C}\|$	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	UB^{MIP}	Gap(%)	CPU(s)	Nodes	UB^{H}	CPU(s)	Iterations	Dev(%)
81	20	3	3	8	2	6,046.92	17.84	86,400.00	460,333	**6,046.92**	824.02	37	0.00
82	20	3	3	8	2	7,339.53	44.57	86,400.00	20,543	**5,668.56**	1,172.51	55	−22.77
83	20	3	3	8	2	5,489.26	36.32	86,400.00	47,901	**5,288.71**	532.51	32	−3.65
84	20	3	3	8	2	5,612.45	40.83	86,400.00	31,478	**5,230.07**	1,242.39	43	−6.81
85	20	3	3	8	2	5,914.04	36.64	86,400.00	47,425	**5,596.14**	1,348.81	40	−5.38
86	20	3	3	8	3	6,286.95	18.91	86,400.00	356,810	**6,084.90**	481.18	32	−3.21
87	20	3	3	8	3	**6,175.56**	23.73	86,400.00	442,273	6,236.99	765.52	41	0.99
88	20	3	3	8	3	5,765.13	29.65	86,400.00	62,261	**5,393.28**	478.49	30	−6.45
89	20	3	3	8	3	5,505.58	33.99	86,400.00	51,859	**5,144.40**	510.88	43	−6.56
90	20	3	3	8	3	6,174.52	29.87	86,400.00	1,195,481	**6,040.64**	821.86	42	−2.17
91	20	3	5	8	2	–	–	86,400.00	33,488	**7,528.84**	1,228.75	40	–
92	20	3	5	8	2	–	–	86,400.00	45,439	**6,844.97**	1,902.73	40	–
93	20	3	5	8	2	7,039.20	34.57	86,400.00	79,290	**7,017.57**	1,001.97	38	−0.31
94	20	3	5	8	2	–	–	86,400.00	89,997	**7,030.76**	1,443.90	34	–
95	20	3	5	8	2	8,498.11	39.30	86,400.00	59,570	**7,780.75**	1,146.46	33	−8.44
96	20	3	5	8	3	–	–	86,400.00	16,514	**8,835.16**	707.92	31	–
97	20	3	5	8	3	7,931.25	42.76	86,400.00	114,870	**7,706.05**	1,779.37	41	−2.84
98	20	3	5	8	3	9,369.82	40.45	86,400.00	34,101	**9,277.08**	1,343.53	29	−0.99
99	20	3	5	8	3	**8,323.71**	44.35	86,400.00	61,189	8,431.57	1,214.41	34	1.30
100	20	3	5	8	3	–	–	86,400.00	40,009	**7,636.18**	1,844.49	30	–
101	20	5	3	8	2	7,936.40	44.29	86,400.00	26,504	**6,386.26**	1,457.82	42	−19.53
102	20	5	3	8	2	8,137.06	35.95	86,400.00	189,386	**7,930.72**	2,950.07	50	−2.54
103	20	5	3	8	2	6,909.33	36.70	86,400.00	34,275	**6,104.78**	1,156.94	34	−11.64
104	20	5	3	8	2	7,514.08	42.34	86,400.00	16,802	**6,580.70**	2,118.12	44	−12.42
105	20	5	3	8	2	6,688.13	37.30	86,400.00	45,502	**6,252.63**	4,494.13	49	−6.51
106	20	5	3	8	3	8,094.18	39.59	86,400.00	31,883	**7,331.90**	5,446.30	41	−9.42
107	20	5	3	8	3	6,280.95	45.01	86,400.00	28,894	**5,312.95**	1,650.98	30	−15.41
108	20	5	3	8	3	7,102.14	34.82	86,400.00	36,302	**6,627.72**	1,284.90	31	−6.68
109	20	5	3	8	3	6,887.83	38.16	86,400.00	85,129	**6,520.65**	815.71	27	−5.33
110	20	5	3	8	3	7,784.97	34.05	86,400.00	203,830	**7,276.56**	1,551.97	33	−6.53
111	20	5	5	8	2	9,832.02	41.87	86,400.00	31,444	**9,330.25**	2,519.46	27	−5.10
112	20	5	5	8	2	11,489.50	37.02	86,400.00	21,925	**10,847.60**	1,882.42	39	−5.59
113	20	5	5	8	2	–	–	86,400.00	27,302	**11,770.51**	5,336.09	38	–
114	20	5	5	8	2	8,784.16	45.59	86,400.00	11,065	**8,027.25**	2,022.50	40	−8.62
115	20	5	5	8	2	–	–	86,400.00	8,125	**8,641.82**	2,344.18	45	–
116	20	5	5	8	3	11,831.50	37.87	86,400.00	39,516	**10,892.61**	2,714.87	37	−7.94
117	20	5	5	8	3	11,891.00	47.59	86,400.00	7,051	**10,432.21**	5,815.76	47	−12.27
118	20	5	5	8	3	12,391.90	39.87	86,400.00	22,503	**10,850.02**	2,964.40	39	−12.44
119	20	5	5	8	3	9,867.55	28.79	86,400.00	70,734	**9,531.67**	2,308.27	50	−3.40
120	20	5	5	8	3	13,235.70	47.13	86,400.00	15,042	**11,432.56**	3,638.71	46	−13.62
Mean						8,003.95	37.20	86,400.00	106,101	**7,572.52**	1,906.63	38	−7.04

Table 4: Computational results for instances with $n = 20$ customers

						MIP				Heuristic			
Instance	n	$\|\mathcal{P}\|$	$\|\mathcal{C}\|$	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	UB^{MIP}	Gap(%)	CPU(s)	Nodes	UB^{H}	CPU(s)	Iterations	Dev(%)
121	30	3	3	8	2	–	–	86,400.00	2,741	**6,725.48**	2,237.48	34	–
122	30	3	3	8	2	8,395.66	37.83	86,400.00	38,227	**7,816.30**	3,262.17	31	−6.90
123	30	3	3	8	2	–	–	86,400.00	3,399	**7,792.07**	2,607.45	40	–
124	30	3	3	8	2	–	–	86,400.00	2,461	**7,304.30**	2,867.76	33	–
125	30	3	3	8	2	–	–	86,400.00	4,390	**7,538.67**	2,451.80	28	–
126	30	3	3	8	3	8,571.60	54.24	86,400.00	6,044	**6,035.92**	2,828.92	33	−29.58
127	30	3	3	8	3	–	–	86,400.00	5,554	**8,491.91**	3,570.71	27	–
128	30	3	3	8	3	–	–	86,400.00	2,810	**7,825.10**	2,356.47	30	–
129	30	3	3	8	3	–	–	86,400.00	3,579	**6,670.72**	3,659.49	30	–
130	30	3	3	8	3	–	–	86,400.00	4,012	**6,297.16**	3,554.69	38	–
131	30	3	5	8	2	–	–	86,400.00	1,926	**9,691.07**	3,104.28	33	–
132	30	3	5	8	2	–	–	86,400.00	3,186	**10,830.62**	3,279.93	28	–
133	30	3	5	8	2	–	–	86,400.00	2,339	**9,424.70**	5,251.72	38	–
134	30	3	5	8	2	–	–	86,400.00	3,056	**11,346.72**	4,438.07	31	–
135	30	3	5	8	2	–	–	86,400.00	4,716	**10,639.29**	2,814.19	26	–
136	30	3	5	8	3	–	–	86,400.00	1,647	**9,522.54**	2,919.18	30	–
137	30	3	5	8	3	–	–	86,400.00	1,619	**11,829.88**	4,293.41	36	–
138	30	3	5	8	3	–	–	86,400.00	2,755	**9,992.65**	2,839.66	26	–
139	30	3	5	8	3	–	–	86,400.00	2,607	**6,669.65**	6,442.31	60	–
140	30	3	5	8	3	–	–	86,400.00	1,979	**9,650.25**	5,408.40	32	–
141	30	5	3	8	2	–	–	86,400.00	1,620	**10,434.49**	3,595.55	37	–
142	30	5	3	8	2	–	–	86,400.00	3,488	**9,843.34**	6,730.42	41	–
143	30	5	3	8	2	–	–	86,400.00	2,315	**8,516.00**	8,282.30	37	–
144	30	5	3	8	2	–	–	86,400.00	2,182	**7,563.69**	1,642.55	31	–
145	30	5	3	8	2	–	–	86,400.00	3,782	**8,725.91**	6,114.86	27	–
146	30	5	3	8	3	–	–	86,400.00	3,101	**8,102.84**	7,913.45	31	–
147	30	5	3	8	3	9,727.28	46.57	86,400.00	3,485	**8,309.28**	6,540.98	29	−14.58
148	30	5	3	8	3	–	–	86,400.00	1,769	**7,830.03**	7,151.65	32	–
149	30	5	3	8	3	–	–	86,400.00	1,932	**9,546.77**	8,911.28	33	–
150	30	5	3	8	3	–	–	86,400.00	3,442	**7,211.02**	6,233.74	36	–
151	30	5	5	8	2	–	–	86,400.00	2,234	**11,343.12**	4,529.60	33	–
152	30	5	5	8	2	–	–	86,400.00	2,285	**17,289.70**	5,961.69	26	–
153	30	5	5	8	2	–	–	86,400.00	1,906	**12,138.44**	6,254.19	34	–
154	30	5	5	8	2	–	–	86,400.00	2,737	**10,062.16**	8,178.30	30	–
155	30	5	5	8	2	–	–	86,400.00	1,859	**12,320.81**	7,239.01	30	–
156	30	5	5	8	3	–	–	86,400.00	1,638	**12,638.15**	7,236.19	28	–
157	30	5	5	8	3	–	–	86,400.00	1,843	**13,390.47**	10,432.90	32	–
158	30	5	5	8	3	–	–	86,400.00	1,700	**12,117.37**	7,584.87	43	–
159	30	5	5	8	3	–	–	86,400.00	1,866	**12,695.75**	9,736.26	34	–
160	30	5	5	8	3	–	–	86,400.00	1,831	**12,465.77**	6,504.63	46	–
Mean						8,898.18	46.21	86,400.00	3,652	**9,616.00**	5,174.06	33	−17.02

Table 5: Computational results for instances with $n = 30$ customers

						MIP				Heuristic			
Instance	n	$\|\mathcal{P}\|$	$\|\mathcal{C}\|$	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	UB^{MIP}	Gap(%)	CPU(s)	Nodes	UB^{H}	CPU(s)	Iterations	Dev(%)
161	40	3	3	8	2	–	–	86,400.00	3,096	**7,433.32**	7,163.40	31	–
162	40	3	3	8	2	–	–	86,400.00	1,168	**6,932.96**	5,013.45	55	–
163	40	3	3	8	2	9,974.20	54.55	86,400.00	2,046	**7,209.82**	7,384.26	29	−27.72
164	40	3	3	8	2	–	–	86,400.00	3,096	**8,210.88**	10,675.60	36	–
165	40	3	3	8	2	–	–	86,400.00	2,035	**7,019.21**	6,366.87	32	–
166	40	3	3	8	3	–	–	86,400.00	1,288	**7,649.14**	6,503.53	34	–
167	40	3	3	8	3	–	–	86,400.00	1,449	**8,341.77**	12,872.70	34	–
168	40	3	3	8	3	–	–	86,400.00	1,435	**8,524.84**	8,087.53	33	–
169	40	3	3	8	3	–	–	86,400.00	1,318	**7,203.91**	9,024.63	40	–
170	40	3	3	8	3	–	–	86,400.00	2,898	**7,186.48**	5,286.15	31	–
171	40	3	5	8	2	–	–	86,400.00	1,509	**11,795.54**	4,900.73	30	–
172	40	3	5	8	2	–	–	86,400.00	1,509	**12,244.95**	8,329.32	32	–
173	40	3	5	8	2	–	–	86,400.00	1,509	**11,970.99**	8,723.19	33	–
174	40	3	5	8	2	–	–	86,400.00	2,848	**11,568.07**	7,040.81	26	–
175	40	3	5	8	2	–	–	86,400.00	953	**11,825.04**	6,015.96	28	–
176	40	3	5	8	3	–	–	86,400.00	2,530	**12,521.46**	7,880.99	30	–
177	40	3	5	8	3	–	–	86,400.00	1,766	**12,948.66**	6,238.63	37	–
178	40	3	5	8	3	–	–	86,400.00	2,621	**11,008.08**	6,136.95	30	–
179	40	3	5	8	3	–	–	86,400.00	1,262	**11,397.47**	10,702.80	38	–
180	40	3	5	8	3	–	–	86,400.00	2,141	**8,900.21**	11,810.10	50	–
181	40	5	3	8	2	–	–	86,400.00	5,336	**10,599.76**	13,745.00	41	–
182	40	5	3	8	2	–	–	86,400.00	1,701	**14,379.70**	5,424.59	29	–
183	40	5	3	8	2	–	–	86,400.00	1,885	**14,756.00**	17,757.60	29	–
184	40	5	3	8	2	–	–	86,400.00	1,140	**11,431.08**	8,658.49	28	–
185	40	5	3	8	2	–	–	86,400.00	4,527	**10,766.61**	5,534.79	30	–
186	40	5	3	8	3	–	–	86,400.00	1,107	**7,035.79**	13,389.40	37	–
187	40	5	3	8	3	–	–	86,400.00	1,432	**9,267.54**	18,507.80	36	–
188	40	5	3	8	3	–	–	86,400.00	1,702	**11,448.84**	13,094.80	35	–
189	40	5	3	8	3	–	–	86,400.00	1,304	**11,696.88**	13,921.00	39	–
190	40	5	3	8	3	–	–	86,400.00	1,001	**9,185.02**	13,247.40	39	–
191	40	5	5	8	2	–	–	86,400.00	2,205	**14,656.10**	19,289.90	33	–
192	40	5	5	8	2	–	–	86,400.00	501	**14,171.30**	16,549.40	41	–
193	40	5	5	8	2	–	–	86,400.00	632	**13,393.36**	15,902.70	39	–
194	40	5	5	8	2	–	–	86,400.00	3,206	**17,826.20**	18,782.90	32	–
195	40	5	5	8	2	–	–	86,400.00	1,885	**16,980.80**	13,505.50	28	–
196	40	5	5	8	3	–	–	86,400.00	2,010	**13,883.90**	28,976.70	43	–
197	40	5	5	8	3	–	–	86,400.00	606	**16,069.40**	16,775.10	31	–
198	40	5	5	8	3	–	–	86,400.00	1,132	**12,578.44**	14,948.20	34	–
199	40	5	5	8	3	–	–	86,400.00	3,122	**15,037.40**	27,841.30	38	–
200	40	5	5	8	3	–	–	86,400.00	1,069	**17,757.10**	18,197.40	39	–
Mean						9,974.20	54.55	86,400.00	1,900	**11,370.35**	11,755.19	35	−27.72

Table 6: Computational results for instances with $n = 40$ customers

						MIP				Heuristic			
Instance	n	$\|\mathcal{P}\|$	$\|\mathcal{C}\|$	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	UB^{MIP}	Gap(%)	CPU(s)	Nodes	UB^{H}	CPU(s)	Iterations	Dev(%)
201	50	3	3	8	2	–	–	86,400.00	1,890	**10,898.45**	8,277.03	34	–
202	50	3	3	8	2	–	–	86,400.00	1,403	**9,237.78**	10,553.00	33	–
203	50	3	3	8	2	–	–	86,400.00	538	**8,967.56**	17,817.50	27	–
204	50	3	3	8	2	–	–	86,400.00	2,280	**9,814.34**	8,867.59	30	–
205	50	3	3	8	2	–	–	86,400.00	1,418	**8,871.72**	6,676.96	26	–
206	50	3	3	8	3	–	–	86,400.00	532	**8,212.02**	23,251.70	31	–
207	50	3	3	8	3	–	–	86,400.00	535	**10,637.55**	21,110.00	43	–
208	50	3	3	8	3	–	–	86,400.00	597	**8,348.17**	16,251.30	42	–
209	50	3	3	8	3	–	–	86,400.00	1,598	**8,174.64**	7,974.83	36	–
210	50	3	3	8	3	–	–	86,400.00	1,590	**9,603.85**	26,018.40	31	–
211	50	3	5	8	2	–	–	86,400.00	3,346	**14,271.50**	10,003.10	26	–
212	50	3	5	8	2	–	–	86,400.00	489	**13,115.20**	15,222.20	27	–
213	50	3	5	8	2	–	–	86,400.00	1,450	**11,783.77**	9,442.87	35	–
214	50	3	5	8	2	–	–	86,400.00	491	**10,309.82**	13,402.50	27	–
215	50	3	5	8	2	–	–	86,400.00	490	**10,145.22**	14,758.10	32	–
216	50	3	5	8	3	–	–	86,400.00	503	**12,107.37**	29,669.40	40	–
217	50	3	5	8	3	–	–	86,400.00	1,009	**15,864.90**	13,940.80	29	–
218	50	3	5	8	3	–	–	86,400.00	996	**14,254.30**	15,065.50	38	–
219	50	3	5	8	3	–	–	86,400.00	498	**12,302.42**	25,508.40	26	–
220	50	3	5	8	3	–	–	86,400.00	1,136	**12,450.73**	15,011.60	28	–
221	50	5	3	8	2	–	–	86,400.00	1,485	**12,825.05**	33,645.60	29	–
222	50	5	3	8	2	–	–	86,400.00	990	**14,168.90**	12,759.60	29	–
223	50	5	3	8	2	–	–	86,400.00	990	**10,651.21**	8,777.88	26	–
224	50	5	3	8	2	–	–	86,400.00	1,980	**11,339.20**	19,327.60	30	–
225	50	5	3	8	2	–	–	86,400.00	1,980	**12,661.37**	25,045.30	33	–
226	50	5	3	8	3	–	–	86,400.00	513	**10,910.71**	46,229.60	35	–
227	50	5	3	8	3	–	–	86,400.00	1,539	**8,690.50**	51,364.20	28	–
228	50	5	3	8	3	–	–	86,400.00	513	**13,548.20**	41,251.70	28	–
229	50	5	3	8	3	–	–	86,400.00	2,052	**10,758.28**	40,876.00	26	–
230	50	5	3	8	3	–	–	86,400.00	1,539	**9,613.43**	23,083.00	33	–
231	50	5	5	8	2	–	–	86,400.00	514	**18,487.90**	27,027.40	27	–
232	50	5	5	8	2	–	–	86,400.00	526	**20,292.90**	26,941.10	26	–
233	50	5	5	8	2	–	–	86,400.00	1,028	**14,346.60**	32,617.20	30	–
234	50	5	5	8	2	–	–	86,400.00	520	**21,666.70**	24,258.70	31	–
235	50	5	5	8	2	–	–	86,400.00	1,028	**14,724.30**	19,450.50	33	–
236	50	5	5	8	3	–	–	86,400.00	1,092	**19,271.10**	36,701.90	30	–
237	50	5	5	8	3	–	–	86,400.00	538	**17,746.30**	23,042.30	35	–
238	50	5	5	8	3	–	–	86,400.00	538	**18,290.50**	47,300.20	26	–
239	50	5	5	8	3	–	–	86,400.00	2,690	**20,743.60**	44,242.30	28	–
240	50	5	5	8	3	–	–	86,400.00	538	**16,369.70**	32,821.10	28	–
						–	–	86,400.00	1,135	**12,911.94**	23,139.65	31	–

Table 7: Computational results for instances with $n = 50$ customers

customers. The number of nodes explored in the branch-and-bound tree provides some insights into the difficulty of solving these instances. Overall, CPLEX is not able to find feasible solutions for most of the instances in which the number of nodes explored is very small after 24 hours of computing time (see Tables 5–7). Solving each node of the branch-and-bound tree becomes much harder when the size of the problem increases and, therefore, CPLEX cannot explore many nodes during the search.

Regarding our heuristic, results show that it is able to provide competitive solutions for instances with 10 customers. For these instances, the average total cost of the heuristic solutions is 5,441.01, which represents an average deviation of only 1.29% from the CPLEX solutions. However, the heuristic requires only 0.20 hours of computing time (i.e., 12 minutes), on average, in comparison with the 23.12 hours required by the solver.

For instances with 15 customers, our heuristic provides better results than CPLEX in shorter computing times. CPLEX reached the computational time limit of 24 hours for all the cases (Table 3). The heuristic, on the other hand, just required an average computing time of 0.32 hours (i.e., $\approx$19 minutes) and returned solutions with an average deviation of -0.57%. In particular, our heuristic provided better solutions than CPLEX for 19 out of 40 instances, with an average deviation of -2.35%. For the remaining 21 instances, where CPLEX obtained better solutions, the average deviation of the heuristic solutions was only 1.03%.

For instances with 20 customers the heuristic is clearly superior to CPLEX. The solver did not return any feasible solution for seven out of 40 instances (Table 4), while the heuristic provided feasible solutions for all cases. Moreover, it obtained a significant reduction of the total cost for most of the instances, with an average deviation of -7.04%, requiring approximately 32 minutes of computing time, on average.

For instances with $n \geq 30$, the performance of CPLEX deteriorates significantly, while the heuristic is still able to provide solutions within reasonable computational times. In general, the heuristic found feasible solutions for all the cases, whereas the solver only found feasible solutions for four instances. As described above, instances of these sizes are beyond the current capabilities of CPLEX, which is not able to explore many nodes in the branch-and-bound tree before reaching the computational time limit.

Note that the average computing time of the heuristic increases with the size of the problem, ranging from 0.20 hours, for ten customers, to 6.43 hours, for fifty customers. These times, besides being much shorter than the time

required by CPLEX, can be considered acceptable in practical situations, since this kind of integrated problems is usually solved on a weekly basis. Computational experiments pointed out that the main reason for the increase in computing times is the hardness of model LSSD, which needs to be solved at each iteration of the algorithm. Although this model is smaller than the full integrated model, it is still difficult to solve and the time required to reach the stopping criterion increases with the size of the problem.

Information related to average costs, computing times, number of iterations and feasible solutions found during the execution of the heuristic are presented in Table 8. We grouped the 240 instances according to the number of customers (n), items ($|\mathcal{C}|$) and vehicles, which have a direct impact on the size of the problems. Therefore, each entry of Table 8 represents the average of ten instances.

Regarding the average number of feasible solutions found during the execution of the algorithm, we observe that it tends to decrease with the size of the problem. The larger the instances the more difficult it is for the heuristic to find feasible solutions at each iteration. Nevertheless, we are still able to provide feasible solutions for the largest instances, where CPLEX cannot find a single feasible solution after one day of computing time.

Another interesting feature of our heuristic is its robustness. In general, the heuristic is able to provide consistent results over different independent runs. This can be observed not only in terms of the total cost, but also in terms of the computing time, deviation percentage, number of iterations and number of feasible solution solutions found during the execution of the algorithm. Tables 9 and 10 compare the average results of a single run of the heuristic with the average results of five other independent runs.

These results show the potential for practical application of our heuristic, which achieves good quality solutions within reasonable times for all the instances, including the ones whose size is not manageable for state-of-the-art commercial solvers.

6. Concluding Remarks and Research Directions

In this paper we proposed a decomposition heuristic to deal with a production routing problem that takes into account relevant features from make-to-order companies. The heuristic decomposes the problem into two parts that are then solved in an iterative way. Roughly speaking, the first part takes care of all the production decisions as well as the assignment of customers to

n	$\|\mathcal{C}\|$	$\|\mathcal{V}\|$	Total Cost	Inv.+Setup	Routing	CPU(h)	#Iterations	#Feas. Sol.
10	3	2	4,176.30	2,054.60	2,121.70	0.17	34	26
		3	4,455.00	2,207.30	2,247.70	0.15	36	30
	5	2	6,650.83	4,392.03	2,258.80	0.27	37	29
		3	6,481.92	4,202.52	2,279.40	0.20	36	29
Average			5,441.01	3,214.11	2,226.90	0.20	36	28
15	3	2	5,504.35	2,932.25	2,572.10	0.26	37	22
		3	5,310.69	2,800.99	2,509.70	0.23	37	26
	5	2	6,979.69	4,460.19	2,519.50	0.40	37	21
		3	7,122.92	4,628.62	2,494.30	0.41	37	28
Average			6,229.41	3,705.51	2,523.90	0.32	37	24
20	3	2	6,108.55	3,605.25	2,503.30	0.48	43	27
		3	6,197.00	3,500.50	2,696.50	0.38	35	18
	5	2	8,482.03	5,816.83	2,665.20	0.58	37	23
		3	9,502.51	6,607.61	2,894.90	0.68	38	24
Average			7,572.52	4,882.55	2,689.98	0.53	38	23
30	3	2	8,226.03	5,017.43	3,208.60	1.11	34	13
		3	7,632.08	4,380.18	3,251.90	1.46	32	14
	5	2	11,508.66	8,259.36	3,249.30	1.42	31	10
		3	11,097.25	7,772.95	3,324.30	1.76	37	18
Average			9,616.00	6,357.48	3,258.53	1.44	33	14
40	3	2	9,873.93	6,525.23	3,348.70	2.44	34	11
		3	8,754.02	5,355.32	3,398.70	3.16	36	14
	5	2	13,643.24	10,106.14	3,537.10	3.31	32	8
		3	13,210.21	9,769.71	3,440.50	4.15	37	17
Average			11,370.35	7,939.10	3,431.25	3.27	35	13
50	3	2	10,943.56	7,324.96	3,618.60	4.22	30	7
		3	9,849.74	6,234.44	3,615.30	8.26	33	10
	5	2	14,914.39	11,180.79	3,733.60	5.36	29	6
		3	15,940.09	12,163.69	3,776.40	7.87	31	10
Average			12,911.94	9,225.97	3,685.98	6.43	31	8

Table 8: Average results of the heuristics on the random instances

	10 customers		15 customers		20 customers	
	1 run	5 runs	1 run	5 runs	1 run	5 runs
Inv.+Setup Cost	3,214.11	3,192.18	3,705.51	3,695.61	4,882.55	4,879.77
Routing Cost	2,226.90	2,252.96	2,523.90	2,516.03	2,689.98	2,696.09
Total Cost	5,441.01	5,445.13	6,229.41	6,211.64	7,572.52	7,575.86
CPU(h)	0.20	0.18	0.32	0.29	0.53	0.48
Dev(%)	1.29	1.35	−0.57	−0.82	−7.04	−7.06
#Iterations	36	35	37	36	38	37
#Feas. Sol.	28	28	24	25	23	22

Table 9: Comparison of the average results of the heuristic after five independent runs (10 to 20 customers)

	30 customers		40 customers		50 customers	
	1 run	5 runs	1 run	5 runs	1 run	5 runs
Inv.+Setup Cost	6,357.48	6,345.25	7,939.10	7,912.58	9,225.97	9,215.55
Routing Cost	3,258.53	3,178.26	3,431.25	3,445.02	3,685.98	3,723.35
Total Cost	9,616.00	9,523.51	11,370.35	11,357.60	12,911.94	12,938.90
CPU(h)	1.44	1.43	3.27	2.83	6.43	6.35
Dev(%)	−17.02	−18.02	−27.72	−27.35	–	–
#Iterations	33	35	35	32	31	32
#Feas. Sol.	14	15	13	11	8	8

Table 10: Comparison of the average results of the heuristic after five independent runs (30 to 50 customers)

routes, without considering routing decisions explicitly. In the second part, the sequence and time of the visits is decided by solving a multi-trip vehicle routing problem for each vehicle used in the first phase.

Computational experiments showed that CPLEX is capable of solving instances with up to twenty customers. For instances with no more than fifteen customers, the heuristic provided solutions of similar quality to those returned by the solver. With twenty customers, it clearly outperformed CPLEX, finding solutions of better quality. In all cases, the heuristic required considerably shorter computing times. In addition, our procedure returned feasible solutions for instances with up to fifty customers, for which the solver could not find any solution within one day of computing time.

As a research perspective, it would be interesting to develop an effective heuristic procedure to solve the LSDS subproblem. This model becomes harder to solve when the size of the problem increases and, therefore, developing efficient ways to solve it is necessary in order to reduce computing times on larger instances. Moreover, it would also allow us to perform more iterations, which may help to find better solutions.

Another interesting idea is to apply Lagrangian Relaxation to the integrated model in order to compute good lower bounds, which can be used to measure the quality of heuristic solutions. Moreover, by applying specific routines to recover feasibility, new feasible solutions for the original problem may be found while solving the Lagrangian dual problem.

Acknowledgments

This work was supported by the São Paulo Research Foundation (FAPESP) under Grant number 2014/10565-8 and the Brazilian National Council for Scientific and Technological Development (CNPq) under Grants numbers 140179/2014-3 and 312569/2013-0. This support is gratefully acknowledged.

References

[1] Absi, N., Archetti, C., Dauzère-Pérès, S., Feillet, D., 2014. A two-phase iterative heuristic approach for the production routing problem. Transportation Science 49 (4), 784–795.

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

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

[4] Adulyasak, Y., Cordeau, J.-F., Jans, R., 2015. Benders decomposition for production routing under demand uncertainty. Operations Research 63 (4), 851–867.

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

[6] Amorim, P., Belo-Filho, M., Toledo, F., Almeder, C., Almada-Lobo, B., 2013. Lot sizing versus batching in the production and distribution planning of perishable goods. International Journal of Production Economics 146 (1), 208–218.

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

[8] Armentano, V., Shiguemoto, A., Løkketangen, A., 2011. Tabu search with path relinking for an integrated production–distribution problem. Computers & Operations Research 38 (8), 1199–1209.

[9] Bard, J. F., Nananukul, N., 2009. Heuristics for a multiperiod inventory routing problem with production decisions. Computers & Industrial Engineering 57 (3), 713–723.

[10] Bard, J. F., Nananukul, N., 2009. The integrated production–inventory–distribution–routing problem. Journal of Scheduling 12 (3), 257–280.

[11] Bard, J. F., Nananukul, N., 2010. A branch-and-price algorithm for an integrated production and inventory routing problem. Computers & Operations Research 37 (12), 2202–2217.

[12] Bertazzi, L., 2005. Minimizing the total cost in an integrated vendor–managed inventory system. Journal of Heuristics 11, 393–419.

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

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

[15] Brahimi, N., Aouam, T., 2016. Multi-item production routing problem with backordering: a milp approach. International Journal of Production Research 54 (4), 1076–1093.

[16] Caceres-Cruz, J., Grasas, A., Ramalhinho, H., Juan, A. A., 2014. A savings-based randomized heuristic for the heterogeneous fixed fleet vehicle routing problem with multi-trips. Journal of Applied Operational Research 6 (2), 69–81.

[17] Chandra, P., Fisher, M. L., 1994. Coordination of production and distribution planning. European Journal of Operational Research 72, 503–517.

[18] Chen, Z.-L., 2004. Integrated production and distribution operations: taxonomy, models, and review. In: Simchi-Levi, D., Wu, S., Shen, Z.-J. (Eds.), Handbook of quantitative supply chain analysis: modeling in the e-business era. Kluwer Academic Publishers, Ch. 17, pp. 711–745.

[19] Chitsaz, M., , Cordeau, J.-F., Jans, R., 2017. A unified decomposition matheuristic for assembly, production and inventory routing. Tech. rep., GERAD G–2017–03, HEC Montréal, Canada.

[20] Clarke, G., Wright, J. W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581.

[21] Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., Vigo, D., 2007. Vehicle routing. In: Barnhart, C., Laporte, G. (Eds.), Transportation. Vol. 14 of Handbooks in operations research and management science. Elsevier, Ch. 6, pp. 367–428.

[22] Díaz-Madroñero, M., Peidro, D., Mula, J., 2015. A review of tactical optimization models for integrated production and transport routing planning decisions. Computers & Industrial Engineering 88, 518–535.

[23] Fischetti, M., Lodi, A., 2003. Local branching. Mathematical Programming 98 (1), 23–47.

[24] Fumero, F., Vercellis, C., 1999. Synchronized development of production, inventory, and distribution schedules. Transportation Science 33, 330–340.

[25] Golden, B. L., Raghavan, S., Wasil, E. A. (Eds.), 2008. The vehicle routing problem: latest advances and new challenges, 1st Edition. Springer US, New York.

[26] Gramani, M., França, P., Arenales, M., 2009. A lagrangian relaxation approach to a coupled lot-sizing and cutting stock problem. International Journal of Production Economics 119 (2), 219–227.

[27] Jans, R., Degraeve, Z., 2008. Modeling industrial lot sizing problems: a review. International Journal of Production Research 46 (6), 1619–1643.

[28] Juan, A. A., Faulin, J., Jorba, J., Riera, D., Masip, D., Barrios, B., 2011. On the use of monte carlo simulation, cache and splitting techniques to improve the clarke and wright savings heuristics. Journal of the Operational Research Society 62 (6), 1085–1097.

[29] Karimi, B., Fatemi Ghomi, S., Wilson, J., 2003. The capacitated lot sizing problem: A review of models and algorithms. Omega 31 (5), 365–378.

[30] Lei, L., Liu, S., Ruszczynski, A., Park, S., 2006. On the integrated production, inventory, and distribution routing problem. IIE Transactions 38 (11), 955–970.

[31] Miranda, P. L., Morabito, R., Ferreira, D., 2016. Optimization model for a production, inventory, distribution routing problem in small furniture companies. Tech. rep., Production Engineering Department, Federal Univeristy of São Carlos, Brazil.

[32] Miranda, P. L., Morabito, R., Ferreira, D., 2017. Mixed integer formulations for a coupled lot-scheduling and vehicle routing problem in furniture settings. Tech. rep., Production Engineering Department, Federal Univeristy of São Carlos, Brazil.

[33] Pochet, Y., Wolsey, L. A., 2006. Production planning by mixed integer programming. Springer Science & Business Media.

[34] Ruokokoski, M., Solyali, O., Cordeau, J.-F., Jans, R., Süral, H., 2010. Efficient formulations and a branch-and-cut algorithm for a production-routing problem. Tech. rep., GERAD G–2010–66, HEC Montréal, Canada.

[35] Shiguemoto, A. L., Armentano, V. A., 2010. A tabu search procedure for coordinating production, inventory and distribution routing problems. International Transactions in Operational Research 17 (2), 179–195.

[36] Toth, P., Vigo, D. (Eds.), 2002. The Vehicle Routing Problem, 1st Edition. SIAM, Philadelphia, PA, USA.