Two-Dimensional Loading in Vehicle Routing Problem With Release and Due Dates

Jaikishan Soman (jaikishants@iitb.ac.in)
Shailesh J Mehta School of Management
Rahul J Patil
Shailesh J Mehta School of Management

Research Article

Keywords: Vehicle routing problem with two-dimensional loading, scatter search heuristics, strategic oscillation, vehicle routing problem with release and due dates

Posted Date: December 6th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-750430/v1

Two-dimensional loading in vehicle routing problem with release and due dates

Jaikishan T Soman[a,*], Rahul J Patil[a]

[a]Shailesh J Mehta School of Management, Indian Institute of Technology, Bombay, Mumbai - 400076, India

(jaikishants@iitb.ac.in, rahul.patil@iitb.ac.in)

*corresponding author

Declarations of interest: none

No funding was received for this work

Abstract

In this paper, we study a two-dimensional vehicle loading and routing problem, in which customer orders with deadlines become available for dispatch as per their release dates. The objective is to minimize the sum of transportation, inventory, and tardiness costs, while respecting various loading and routing constraints. This scenario allows us to study various tradeoffs that tend to arise due to temporal order aggregation across release dates. We thereby propose an integrated mathematical formulation to simultaneously model both the routing and loading requirements of the problem at hand. Specifically, as the problem is NP-hard, we propose a scatter search based heuristic approach to solve large-size instances. Further, its performance is enhanced using problem-specific procedures and strategic oscillation approaches. Additionally, the numerical experiments illustrate the influence of cost structures on both the optimal loading configurations along with the optimal routes. Importantly, our experiments also suggest that the proposed scatter search method can produce good quality solutions in less time.

Keywords: Vehicle routing problem with two-dimensional loading; scatter search heuristics; strategic oscillation; vehicle routing problem with release and due dates

1 Introduction

In 2003, the average payload transported using heavy goods vehicles in the UK was only 17.9 tons, which in fact, was significantly lower than the maximum average load of 29 tons on a 38 tons plated vehicle. Furthermore, the vehicles were only fully laden (by weight alone) on approximately 22% of the traveled distance (McKinnon, 2005). Notably, the transported payload could impact both the economic and environmental aspects of freight transport (Ligterink et al., 2012). Generally, traditional approaches assess the payload estimates of road transport, based upon truck size and weight limits. However, in practice, many physical or scheduling constraints, restrain vehicle loading to its full weight limits (McKinnon, 2005). Moreover, vehicle utilization depends how the loaded items are actually arranged in the vehicle. This is the reason as to why an optimal arrangement of the vehicle payload, along with the design of vehicle routes is important, because it not only improves vehicle utilization, but also reduces the distance traveled.

Extant literature covering the capacitated vehicle routing problem (CVRP) with loading constraints recognized the importance of the loading activity, and thereby has incorporated various loading constraints such as two (three) dimensional loading, load distribution, unloading patterns, while designing vehicle routes (Bortfeldt & Yi, 2020; Gendreau, Amendola, et al., 2008). Pollaris et al. (2015) provided a detailed literature review, studying various loading constraints that are ingrained in vehicle routing problems (VRPs). Vehicle routing problem with two-dimensional loading (2L-CVRP) is an important variant of VRP that studies the challenges faced by freight distribution companies that handle and transport non-stackable goods. Notably, this variant excludes the dimension of height while loading, because vertical stacking can potentially cause damage, especially to fragile items; it may also create other issues, especially while unloading heavy and bulky items (Iori et al., 2007; Gendreau et al., 2008).

From extant literature, we gather that one of the major assumptions in 2L-CVRP is that all the items for delivery are simultaneously available for dispatch at the beginning of the time

horizon. Nonetheless, in many freight transportation environments, these dispatches are often scheduled, based on a master production plan, which in turn is prepared by considering manufacturing constraints, such as bottleneck machine capacity and set up times. Herein, the production plan details out the release dates of the orders spread over a finite planning horizon, whereby the release date is the earliest date an order can be made available for its delivery, while planning the vehicle routes.

The concept of release dates has recently been introduced in vehicle routing, whereby it is classified under CVRP with release dates (VRPRD) (Archetti et al., 2015b, a). Release dates has been used in solving meal delivery problems, in the form of ready times for the food items (Ulmer et al., 2021; Reyes et al., 2018a). Same day delivery problem (SDDP) is another domain where the release date has been considered. Mor & Speranza (2020) reviewed many recent studies conduced in VRP with release dates.

Lateness dependent penalty costs can be significant for many manufacturing firms (Soman and Patil, 2020). As a matter of fact, literature on VRPRD has also emphasized the importance of considering due dates during route formation, and has termed it as CVRP with both release and due dates (VRPRDD) (Shelbourne et al., 2017).

We believe that including loading constraints with both release and due dates make VRPRDD more relevant practically due to the following reasons. For instance, loading configuration does provide the relative positions of items within the loading area of a vehicle. Thus, the temporal aggregation of orders across release dates could possibly contribute to a better arrangement of items. Furthermore, this aggregation could possibly contribute in generating routes with shorter distances. However, one needs to note herein that storing orders for aggregation even post their release date, may incur additional inventory holding costs. Moreover, order consolidation across release dates could also delay order delivery, which in turn, would incur tardiness costs directly proportional to the delay, hiking thereby the total cost. These tradeoffs between inventory, tardiness, and transportation costs are what make 2L- VRPRDD more interesting.

Streams of 2L-CVRP literature have studied how loading configurations can actually impact route designs, while VRPRDD literature has focused on the influence of both release and due dates in routing. Our problem addresses the gap of how release dates of orders can alter loading configurations, while changing the overall route design. The other important contribution of our paper is that it studies the various tradeoffs involved while designing routes, looking to minimize thereby the total cost of the entire supply chain, while considering both the release and due dates, along with the loading constraints.

2L-CVRP literature has proposed decomposition and approximation algorithms to solve the instances of the studied variants. This stream of research has not so far proposed a monolithic integer programming formulation that effectively integrates both the routing and loading aspects of the problem in the presence of release and due dates. A mathematical formulation is certainly useful to many firms that want to solve small to medium-sized real instances to optimality without investing time to program decomposition and approximation algorithms. Thus, we propose a novel integer programming formulation to solve the 2L-CVRP with both release and due dates, wherein we propose to use some of the principles associated with in bin packing and disjunctive formulations. Furthermore, we propose a scatter search method with strategic oscillation to solve the large-sized instances of the problem. Additionally, we

also propose problem-specific construction and improvement methods to further enhance routes and loading configurations for the vehicles.

The following section describes the detailed review of both the CVRP with loading constraints and the CVRP with release and due dates. The proposed integrated mathematical model is elaborated upon in Sect 3, while Sect 4 depicts an example, describing the salient features of our problem. In Sect 5, we introduce our Scatter Search (SS) methodology, along with its pertinent features. Numerical experiments are conducted, and results are discussed in the computational experiment in Sect 6. Finally, Sect 7 summarizes the key attributes and results of the study.

2. Literature review

In the transportation business, in order to sustain an efficient distribution network, it is essential to consider the material handling operations that includes both loading and unloading of delivery items. Freight carriers often have to handle rectangular-shaped items that cannot be stacked one above the other due to its fragility or weight (Gendreau et al., 2008). The model designed to meet such scenarios should handle both routing and two-dimensional loading requirements. Iori et al. (2007) were the first to address the need for two-dimensional loading in CVRP context. They proposed a branch and cut approach for 2L-CVRP, and solved instances up to 35 customer size with 114 items. Gendreau et al. (2008) extended their work by solving larger instances up-to 255 customers and 786 items, using an effective tabu search algorithm. Fuellerer et al. (2009) considered ant colony optimization for solving 2L-CVRP. Moura & Oliveira (2009) proposed an integrated approach to the vehicle routing problem with time windows and container loading problem, whereby they considered a monolithic formulation for the problem. The larger instances were solved using GRASP heuristic. Over the years, many effective heuristics were proposed for 2L-CVRP; they include memetic algorithm (Khebbache-Hadji et al., 2013), simulated annealing (Leung et al., 2013), Iterated local search (Pollaris, 2018), Variable neighborhood search (Felipe & Ortuño, 2012) and guided Tabu search (Zachariadis et al., 2013). Mathematical formulations for 2L-CVRP were proposed by Martı and Amaya (2012), Dominguez et al. (2014), and Pollaris et al. (2014). There are many variants of 2L-CVRP problems; for instance, 2LCVRP with time windows (Moura and Oliveira, 2009; Martı and Amaya, 2012; Khebbache-Hadji et al., 2013), 2LCVRP with heterogeneous fleet (Leung et al., 2013) etc. Another important aspect in loading is the unrestricted and sequence-based loading; while the latter ensures that the items are arranged such that, they won't disturb the unloading operations (Pollaris et al., 2014), the former, provides effective utilization of the loading area available (Martı and Amaya, 2012; Khebbache-Hadji et al., 2013; Dominguez et al., 2014). Furthermore, some variants of 2L-CVRP allow the orientation of the items to be changed, by rotating them (Fuellerer et al., 2009; Dominguez et al., 2014). A comprehensive review of vehicle routing problems with loading constraints has been provided in Pollaris et al. (2015) study.

Notably, all the variants proposed under 2L-CVRP literature assume that all the delivery items are generally available simultaneously at the beginning of the time horizon. In many of the integrated manufacturing and distribution models, the delivery operations can only start after the items are made available at the docking stations for dispatch. In such environments, the earliest dispatch date is effectively termed as the release date of an item, or of an order comprising a set of items. Recently CVRP literature has considered the implication of release

dates on routes. Arda et al. (2014) for instance, considered stochastic release dates in a multi-period vehicle loading problem, and looked into the implication of delaying shipments from logistic cost and lateness in delivery perspectives. Archetti et al. (2015a) discussed the complexities involving release dates in routing problems. Cattaruzza et al. (2016) studied an effective distribution system with last-mile delivery of merchandizes arriving on different days at the city distribution center. Many other variants of routing problems have also addressed the need for release dates in their applications. Interestingly, the concept of release dates has also been used in meal delivery problems, in the form of ready times for the food items (Ulmer et al., 2021; Reyes et al., 2018a). On the other hand, same day delivery problem (SDDP) is a recent area where the release date has been considered. Voccia et al. (2017) for example, modeled SDDP as a dynamic pickup and delivery problem. Both Klapp et al. (2016) and Klapp et al. (2018) discussed the influence of temporal aggregation within the ambit of dynamic dispatch problems, wherein a depot receives orders dynamically throughout a given period. The decision herein is effectively to plan the dispatch of the orders using the vehicles that are available, while also considering the penalty associated with the unfulfilled orders by the end of a service day. Archetti et al. (2018) considered a traveling salesman problem with release date and completion time minimization. Herein, the problem aimed at minimizing the total travel time for incapacitated vehicles to serve customers, along with the waiting time at the depot for releasing customer demands. Campelo et al. (2018) discussed the VRP with time windows and release dates, where service level agreements involving driver consistency were studied. In the e-commerce business, all the customer orders placed may not be available for delivery all together. Liu et al. (2017) considered a capacitated vehicle routing problem (CVRP) with order available time in the context of e-commerce. The objective of their model was to minimize the total distribution time. In fact, Moons et al. (2017) extended their work by integrating order picking problem with vehicle routing problem with release dates.

Furthermore, delivery due date is an important aspect of distribution problems. The implications of due dates in vehicle routing problem with release dates (VRPRD) context has been studied under the label of vehicle routing problem with release and due dates (VRPRDD). The study by Reyes et al. (2018b) on VRPRDD showed that when the customers were located on a half-line, VRPRDD could be solved to optimality in polynomial time. Archetti et al. (2015b) discussed a multi-period vehicle routing problem with due dates, whereby they developed mathematical formulations and valid inequalities to solve the problem effectively. Shelbourne et al. (2017) used path relinking and iterated local search algorithms to solve VRPRDD problems.

Though the above variants of VRPRDD have made significant contributions, they have not considered the feasibility of loading configurations in deciding the delivery load for the vehicles. Similarly, literature streams in 2L-CVRP have not discussed the relevance of both release and due dates. In fact, under the 2L-CVRP problem, all the delivery items are generally available at the start of the planning horizon. Herein, adding the release dates could trigger various options and tradeoffs. Thus, in order to build an effective loading configuration, some of the items released prior to a given date, may be held back at the depot. However, this temporal aggregation strategy does increase the cost of inventory holding. Moreover, it could also delay some items, incurring lateness dependent tardiness costs

thereof. Based on these lacunae identified from extant literature, we believe that it would be interesting to investigate the influence of these tradeoffs on optimal routes.

Furthermore, none of the above-mentioned streams of literature considered limited transportation capacity scenarios, wherein the loading capacity per day is limited, while planning the delivery operations. Importantly, the restriction on the number of vehicles loaded per day actuates real-life situations, where for example, the complete workforce isn't available, and/or other resources, such as crane capacity are limited, which in turn would restrict the loading capability of a given day. Thus, we introduce hereby a vehicle routing problem with both release and due dates considering inventory holding and tardiness costs, while adhering to two-dimensional loading of delivery items in the presence of finite loading capacity.

3. Problem description

We consider a VRPRDD problem with two-dimensional unrestricted loading situation, whereby we assume the 'oriented loading', as it is a common practice in freight transport business, handling heavy good or pallet loads (Strodl et al., 2010). Additionally, it also ensures a fixed orientation of goods while loading. Specifically, in oriented loading, the width (length) of the item is parallel to the width (length) of the vehicle loading area.

We introduce a graph $G = (N, E)$, in which N defines the set of $n + 2$ vertices corresponding to the depot (vertex 0), to the customer orders $(1, \dots, n)$, and dummy depot $(n + 1)$. Herein, E constitutes a set of arcs connecting the vertices in N. For every arc $(i, j) \in E$ there is an associated transportation cost, C_{ij} and transportation time, τ_{ij}. Customer orders form a set $O = \{1, \dots, n\}$. Each customer order, $i \in O$ comprises of m_i number of items of total weight wgt_i. Moreover, the items under each order forms a set $T_i = \{1, \dots, m_i\}, \forall\, i \in O$. Each item (i, t), where $i \in O$ and $t \in T_i$ has a width w_{it} and length l_{it}. Additionally, we consider a planning horizon comprised of a certain number of release dates during which the orders get ready to be dispatched from the depot. Herein, the release date $F(i)$ of the i^{th}customer order is the earliest date when all the items $t \in T_i$ are ready to be dispatched from the depot. Notably, the release dates of all the orders form a set R, which in turn, represents the planning horizon. The due date d_i of the customer order specifies the expected delivery date, beyond which tardiness cost q_i is imposed on a per day basis. Furthermore, orders stored at the depot after their release date $F(i)$ for the purpose of consolidation incur an inventory holding cost of h_i per day. All the items in an order are delivered together in a single visit to the delivery location. Importantly, order delivery locations are identified as their respective order numbers; and at every order location, service time ser_i is considered for unloading operations.

A fleet of homogeneous vehicles has been used for delivery operations. Hereby, we consider an environment in which the number of vehicles available each day of the planning horizon $r \in R$ is limited. Due to factors, such as limited workforce and limited handling equipment, there is a cap on the number of vehicles to be loaded each day. Further, we consider a set of V_r vehicles to be loaded each day $r \in R$, with each vehicle having a loading area of $\bar{L} \times \bar{W}$, where $\bar{L}$ represents the length, and $\bar{W}$ the width of the loading area. Additionally, we consider the Cartesian coordinates $x -$ axis in the direction of the length and $y -$ axis in the direction

of width. The weight capacity of each vehicle is Q, while every vehicle is entitled to only one route.

We propose a monolithic integrated flow-based formulation, considering both the routing and loading constraints together. Notably, the loading constraints in the formulations are in turn influenced from the two-dimensional bin packing problem of Pisinger & Sigurd (2005). The parameters U' and U are large positive numbers. U represents the penalty imposed when an order is not served, and back-ordered in a given planning horizon. However, in case of limited capacity scenario, as the number of vehicles that can be loaded during the planning horizon may not be enough to serve all the orders, some of the orders may be back-ordered to the next horizon.

Decision variables

$$x_{rkij} = \begin{cases} 1 \; if \; arc \; (i,j) \in E \; is \; covered \; by \; k^{th} \; vehicle \; started \; on \; r^{th} day, k \in V_r, r \in R \\ 0 \quad otherwise \end{cases}$$

$$p_{rki} = \begin{cases} 1 \; if \; i^{th} order \; is \; served \; by \; the \; k^{th} vehicle \; started \; on \; r^{th} day, k \in V_r, r \in R \\ 0 \; \; otherwise \end{cases}$$

$$z_i = \begin{cases} 1 \; if \; order \; i \in O \; is \; back \; ordered \; and \; not \; served \; in \; the \; given \; planning \; horizon \\ 0 \; \; otherwise \end{cases}$$

s_{rki} =Time required to reach the i^{th} order/depot location for the k^{th} vehicle started on r^{th} day, $i \in N, r \in R$

b_i =Lateness incurred for the i^{th} customer order after delivery, $i \in O$

e_i =Holding time of the i^{th} customer order at the depot, $i \in O$

$$L_{tt'} = \begin{cases} 1 \; \; if \; item \; t \; is \; left \; of \; item \; t', \; \forall t \in T_i, \; t' \in T_j, \; i \in O, j \in O \\ 0 \; \; otherwise \end{cases}$$

$$B_{tt'} = \begin{cases} 1 \; \; if \; item \; t \; is \; below \; item \; t', \forall t \in T_i, t' \in T_j, i \in O, j \in O \\ 0 \; \; otherwise \end{cases}$$

$$X_t = x \; coordinate \; of \; bottom \; left \; corner \; of \; item \; t \in T_i, \; i \in O \; inside \; the \; loading \; area \; of \; the \; vehicle$$

$$Y_t = y \; coordinate \; of \; bottom \; left \; corner \; of \; item \; t \in T_i, \; i \in O \; inside \; the \; loading \; area \; of \; the \; vehicle$$

Minimize

$$\sum_{r\in R} \sum_{k\in V_r} \sum_{i=0}^{n} \sum_{j=1}^{n+1} C_{ij} x_{rkij} + \sum_{i\in O} h_i e_i + \sum_{i\in O} q_i b_i + \sum_{i\in O} U z_i \qquad (1)$$

Subject to

$$\sum_{r\in R} \sum_{k\in V_r} p_{rki} + z_i = 1 \quad \forall i \in O \qquad (2)$$

$$\sum_{j=1}^{n+1} x_{rk0j} \leq 1 \quad \forall r \in R, k \in V_r \qquad (3)$$

$$\sum_{i=0}^{n} x_{rkig} - \sum_{j=1}^{n+1} x_{rkgj} = 0 \quad \forall g \in O, r \in R, k \in V_r \qquad (4)$$

$$\sum_{i\in O} wgt_i \times p_{rki} \leq Q \quad \forall r \in R, k \in V_r \tag{5}$$

$$s_{rki} + \tau_{ij} - U'(1 - x_{rkij}) + ser_i \leq s_{rkj} \quad \forall i \in N\backslash n+1, j \in N\backslash 0, r \in R, k \in V_r \tag{6}$$

$$\sum_{j=1}^{n+1} x_{rkij} = p_{rki} \quad \forall\, i \in O, r \in R, k \in V_r \tag{7}$$

$$F(i) \times p_{rki} \leq r \quad \forall i \in O, r \in R, k \in V_r \tag{8}$$

$$\sum_{r\in R: r\geq F(i)} \sum_{k\in V_r} (r - F(i)) \times p_{rki} \leq e_i \quad \forall i \in O \tag{9}$$

$$s_{rki} - b_i + p_{rki} \times r \leq d_i \quad \forall i \in O, r \in R, k \in V_r \tag{10}$$

$$X_t - X_{t'} + \bar{L} \times L_{tt'} \leq (\bar{L} - l_{it}) \;\forall\, t \in T_i, t' \in T_j : i = j \;\rightarrow t \neq t', i \in O, j \in O \tag{11}$$

$$Y_t - Y_{t'} + \bar{W} \times B_{tt'} \leq (\bar{W} - w_{it}) \;\forall\, t \in T_i, t' \in T_j : i = j \;\rightarrow t \neq t', i \in O, j \in O \tag{12}$$

$$L_{tt'} + L_{t't} + B_{tt'} + B_{t't} + (2 - p_{rki} - p_{rkj}) \geq 1$$
$$\forall\, t \in T_i, t' \in T_j : i = j \;\rightarrow t \neq t', i \in O, j \in O, r \in R, k \in V_r \tag{13}$$

$$\sum_{i\in O} p_{rki} \times \sum_{t\in T_i} (l_{it} \times w_{it}) \leq \bar{L} \times \bar{W} \quad \forall\, t \in T_i, i \in O, k \in V_r, r \in R \tag{14}$$

$$0 \leq X_t \leq \bar{L} - l_{it} \quad \forall\, t \in T_i, i \in O \tag{15}$$

$$0 \leq Y_t \leq \bar{W} - w_{it} \quad \forall\, t \in T_i, i \in O \tag{16}$$

$$x_{rkij} \in \{1,0\} \quad \forall i \in N, j \in N, r \in R, k \in V_r \tag{17}$$

$$p_{rki}, z_i \in \{1,0\} \quad \forall i \in O, r \in R, k \in V_r \tag{18}$$

$$L_{tt'}, B_{tt'} \in \{1,0\} \quad \forall t, t' \in T_i, i \in O \tag{19}$$

$$s_{rki} \in \mathbb{R}^+ \quad \forall\, i \in N, k \in V_r, r \in R \tag{20}$$

$$b_i,\ e_i \in \mathbb{R}^+ \quad \forall\, i \in O \tag{21}$$

$$X_t, Y_t \in \mathbb{R}^+ \quad \forall t \in T_i, i \in O \tag{22}$$

The expression (1) represents the objective function comprising of transportation cost ($\sum_{r\in R} \sum_{k\in V_r} \sum_{i=0}^{n} \sum_{j=1}^{n+1} C_{ij} x_{rkij}$), inventory holding cost ($\sum_{i\in O} h_i e_i$), tardiness cost ($\sum_{i\in O} q_i b_i$), and the penalty cost for orders not been served ($\sum_{i\in O} U z_i$). Constraints (2) ensures that every order is either served once or is back-ordered. Constraints (3) are the depot constraints ensuring all the vehicles start from the depot. Constraints (4) include the flow balance constraints, while constraints (5) encompasses the capacity constraints, which in turn, ensures the total weight of the orders served by a given vehicle is less than its weight capacity Q. Constraints (6) determine the time to reach each location, while the following constraints (7) relate the assignment variable p_{rki} with the decision variable x_{rkij}. Constraints (8) include the ones related to release dates, which in turn, ensures that the release dates of the customer orders, assigned to a vehicle, $k \in V_r$ are earlier than or equal to the day, $r \in R$ on which the vehicle actually leaves the depot. Both inequalities (9) and (10) include the inventory and delay time constraints, which determine the inventory holding time and delay in delivery for every order, respectively. Constraints (11) to (13) represent the loading constraints, whereby these constraints define the relative positions of the items contained in a

vehicle and in the process, avoid their overlapping. Additionally, for any two orders i and j that are served in the same vehicle, the variables p_{rki} and p_{rkj} become one. In constraints (13), when both p_{rki} and p_{rkj} become one, they force at least one of the load decision variables ($L_{tt'}$, $B_{tt'}$, $L_{t't}$, $B_{t't}$) to become one for every pair of items of the orders. Moreover, load decision variables specify the relative position of each item with respect to other items in the vehicle. For example, $L_{tt'} = 1$ defines the position of item t to the left of item t', similarly $B_{tt'} = 1$ defines item t below item t'. Both constraints (11) and (12) ensure that the pair of items doesn't overlap each other. In fact, for constraints (11) to (13), condition $i = j \rightarrow t \neq t'$ ensures that an item of a given order is not compared with itself; while constraints (14) ensure that the total area of items that are loaded in a vehicle should not exceed the loading area. Finally, constraints (15) ensure that the items are placed inside the loading area of the vehicle.

Lemma 1

The 2L-VRPRDD model is equivalent to 2L-CVRP when for every customer order $i \in O$, release date $F(i)$ is set as the starting day of the time horizon, due date, $d_i = U$ and $h_i = 0, q_i = 0$.

Proof

When we set release dates of all the customer orders equal to the starting day of the time horizon, the orders are available simultaneously at the beginning of the dispatch activity, similar to the case of 2L-CVRP. On the other hand, when the due dates are set to a very large number U, there's no delay in delivery, due to which, tardiness costs are zero. Further, both the inventory holding and tardiness costs per day for the orders are equated to zero, eliminating thereby both the tardiness and inventory holding costs. Ensuring the number of vehicles as same as in 2L-CVRP case also eliminate the back ordering cost, since sufficient number of vehicles are available to deliver. Thus, our proposed 2L-VRPRDD mathematical formulation does reduce to the modified 2L-CVRP, and thereby may be used to solve the instances of the 2L-CVRP.

Lemma 2

The constraints (13) ensures all the items of an order get served by only the vehicle to which the order is assigned.

Proof

Further, constraints (13) defines the relative positions of items contained in a pair of orders served in the same vehicle. When items in the same orders are considered the constraint becomes

$$L_{tt'} + L_{t't} + B_{tt'} + B_{t't} + (2 - p_{rki} - p_{rki}) \geq 1$$

$$\Longrightarrow L_{tt'} + L_{t't} + B_{tt'} + B_{t't} + 2 \times (1 - p_{rki}) \geq 1 \qquad (23)$$

Notably, for every item pair $(t, t') \in T_i$, contained in the customer order $\in O$, the variable $p_{rki} = 1$ for a given vehicle k started on r^{th} day. When $p_{rki} = 1$ the expression (23) ensures that at least one of the load decision variables become one, defining thereby the relative

positions of item pair $(t, t') \in T_i$ inside the vehicle. Thus the expression (23) forces all the items of an order to be served in the same vehicle. Therefore the proof is verified.

4 Illustration

We consider a sample of ten customer order problem with 18 items to demonstrate our model. Hereby, the planning horizon is of 3 days, wherein, each day there are some orders released from the depot, and every order has a set of items for delivery. Importantly, all the items contained in an order are released together on its release date. For example, on day 1, all the items in the orders 1, 2, 8, and 10 are released. On the following day, orders 3, 4, 5, and 7 are released, and on day 3, orders, 6 and 9 are released. We consider a homogenous vehicle fleet with a capacity of 50 units. For each day in the planning horizon, we assume that only one vehicle is available for delivery. Moreover, the vehicle that is available on a particular day cannot be carried forwarded to subsequent days. Additionally, the loading area of each vehicle is determined by length ($\bar{L}$) and width ($\bar{W}$) of the vehicle's loading compartment. We consider $\bar{L} = 35$ and $\bar{W} = 20$ for each vehicle. The routing parameters of the problem are given in Table 1, while Table 2 shows the loading parameters.

Table 1 Routing parameters of the sample problem

$i \in N$ \ $j \in N$	0	1	2	3	4	5	6	7	8	9	10	$n+1$	$F(i)$	d_i	wgt_i	h_i	q_i	ser_i
0	0	4	8	6	5	7	7	8	4	6	9	0	-	-	-	-	-	-
1	4	0	6	2	2	6	8	9	3	6	5	4	1	5	10	1	1	1
2	8	6	0	5	4	1	4	6	5	4	4	8	1	10	30	1	1	1
3	6	2	5	0	2	5	8	9	4	6	4	6	2	9	10	1	2	1
4	5	2	4	2	0	4	6	8	3	5	3	5	2	12	10	2	1	1
5	7	6	1	5	4	0	3	5	4	3	4	7	2	9	10	2	1	1
6	7	8	4	8	6	3	0	2	5	2	7	7	3	10	20	2	2	1
7	8	9	6	9	8	5	2	0	6	3	9	8	2	12	20	1	1	1
8	4	3	5	4	3	4	5	6	0	3	5	4	1	13	15	2	1	1
9	6	6	4	6	5	3	2	3	3	0	6	6	3	11	10	1	2	1
10	9	5	4	4	3	4	7	9	5	6	0	9	1	11	10	1	1	1

Table 2 Loading parameters of the sample problem

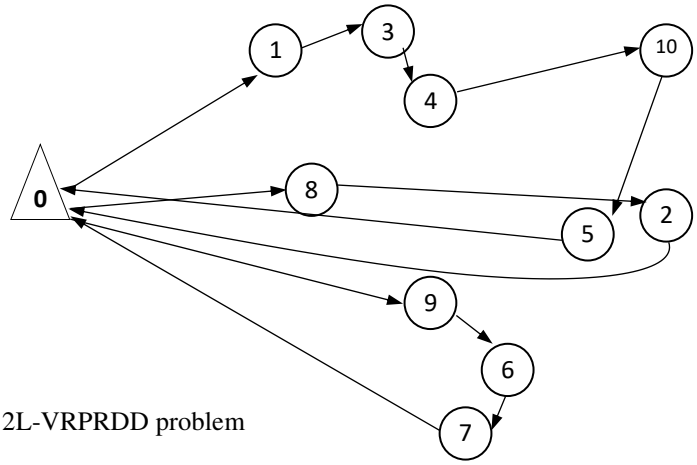
Customer orders ($i \in O$)	Number of items	Items ($t \in T_i$)	Dimensions	
			Length (l_{it})	Width (w_{it})
1	2	1	12	7
		2	29	3
2	2	1	7	9

		2	24	3
3	2	1	14	5
		2	26	3
4	2	1	29	3
		2	5	8
5	2	1	19	3
		2	9	4
6	2	1	7	15
		2	30	4
7	2	1	5	10
		2	16	8
8	2	1	19	9
		2	20	6
9	1	1	13	6
10	1	1	8	8

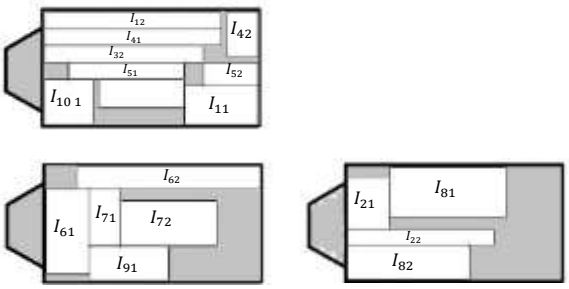
Fig 1(a) illustrates the optimal routes, while Fig 1(b) shows the loading configuration for the 2L-VRPRDD sample problem. Fig 1(c) on the other hand, provides a time-space graph for the routes, which clearly demonstrate the influence of both release and due dates in routing. Importantly, we do observe order consolidation across days in routes 0-9-6-7-0 and 0-1-3-4-10-5-0. Interestingly, even though the order 7 is released on day 2, it is dispatched only on day 3 with the other orders 6 and 9 forming the route 0-9-6-7-0. The order 7 on the other hand, is geographically closer to other orders in the route 0-9-6-7-0, making it economical thereby to store the order while incurring an inventory holding cost and saving on transportation cost than dispatching it separately, increasing thereby the transportation cost substantially. Similar cost savings can be observed in consolidating order 1 with other orders in route 0-1-3-4-10-5-0.

The role of due dates and the corresponding tardiness cost in determining the sequence of delivery in a route is crucial. The route 0-1-3-4-10-5-0 has a transportation cost of 22, tardiness cost of 18, and inventory holding cost of 2 units. Reversing the sequence of delivery as 0-5-10-4-3-1-0 would have the same transportation cost as 22 units, and inventory holding cost as 2 units; nevertheless, the tardiness cost would increase to 52 units, making this route unfavorable thereof. Further, the delay of all the orders, except order 5 in the route caused an increase in the tardiness cost. Unlike in the 2L-CVRP case, where the routes were reversible without any cost implications, the cost of routes in 2L-VRPRDD class depends on the sequence of delivery. In Fig 1(c), the red strip shows the delay in the delivery of orders, and the blue strip shows the storage duration of orders at the depot due to temporal aggregation.

The loading configuration of a given route is determined by the coordinates (X_t, Y_t) of the containing items. For route 0-8-2-0, the variables p_{112} and p_{118} are equal to 1, since both the orders are assigned to the same vehicle starting on day 1. The load decision variables from constraints (13) determine the relative positions of the items. $L_{I_{21}I_{81}} = 1$, means the first item in the second customer order is placed left of the first item of the eighth customer order. For the given route, we get $L_{I_{21}I_{81}}, B_{I_{22}I_{21}}, B_{I_{22}I_{81}}, B_{I_{82}I_{21}}, B_{I_{82}I_{22}}$, and $B_{I_{82}I_{81}}$ equal to one.



(a) Routes of sample 2L-VRPRDD problem

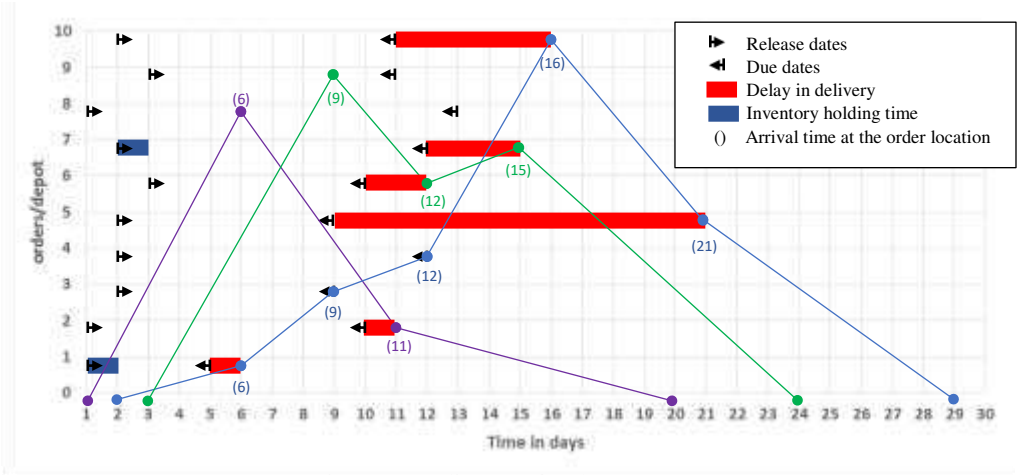


$I_{it} - t^{th}$ item in i^{th} customer order

(b) Loading configuration of the vehicles

Routes	Cost			SD
	Tran	inv	tard	
0-1-3-4-10-5-0	22	2	18	2
0-9-6-7-0	18	1	7	3
0-8-2-0	17	0	1	1
Total		86		

Tran- transportation cost, inv- inventory holding cost, tard- tardiness cost, SD- starting day of the route



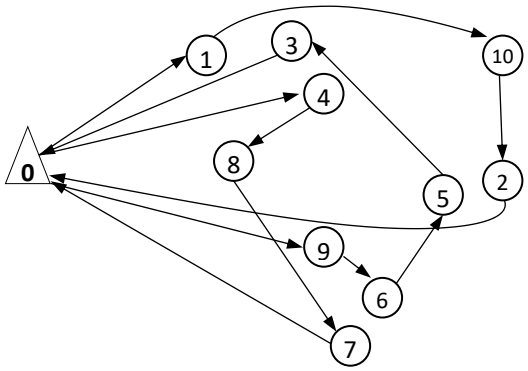
(c) Time-space diagram for the routes

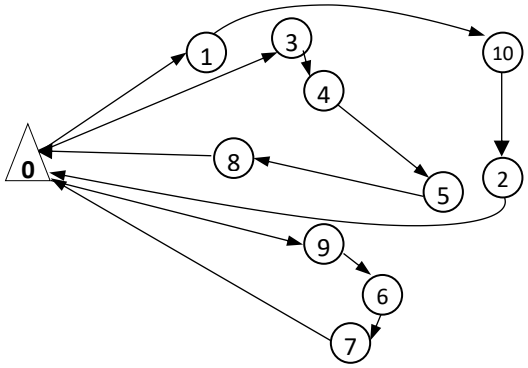
Fig 1 Routes and loading configuration for 2L-VRPRDD

4.1 Influence of release dates in loading configuration

The release dates of orders significantly affect the loading configuration of the routes. In the sample problem as above, the released date of order 8 is on day 1. As we can see from Fig 1(a), order 8 gets served in route 0-8-2-0, starting on day 1. If the release date of order 8 is pushed to the day 2 instead of day 1, then the entire route design changes, as shown in Fig 2(a). Further, the overall cost increases to 117 instead of 86, as may be seen from Fig 1(a).

When order 8 is released on day 2, it can be either be routed on day 2 or on day 3. In fact, serving order 8 in the route starting day 2 would force the other orders, apart from order 4 from route 0-1-3-4-10-5-0 (Fig 1(a)), to be reassigned to other routes, as shown in Fig 2(a). One of the reasons for the reassignment is that, the order 8 comprises of larger size items, which makes it difficult to fit in, with the majority of other orders in route 0-1-3-4-10-5-0. The change in route structure on the other hand, has increased both transportation and tardiness costs, which considerably has raised the overall cost. Notably, both orders 3 and 2 get further delayed, as shown in Fig 2(a) rather than in Fig 1(a).





Routes	Cost			SD
	Tran	inv	tard	
0-1-10-2-0	21	0	6	1
0-4-8-7-0	22	0	6	2
0-9-6-5-3-0	22	3	37	3
Total	117			

(a) Routes with loading constraints

Routes	Cost			SD
	Tran	inv	tard	
0-1-10-2-0	21	0	6	1
0-3-4-5-8-0	20	0	15	2
0-9-6-7-0	18	1	7	3
Total	88			

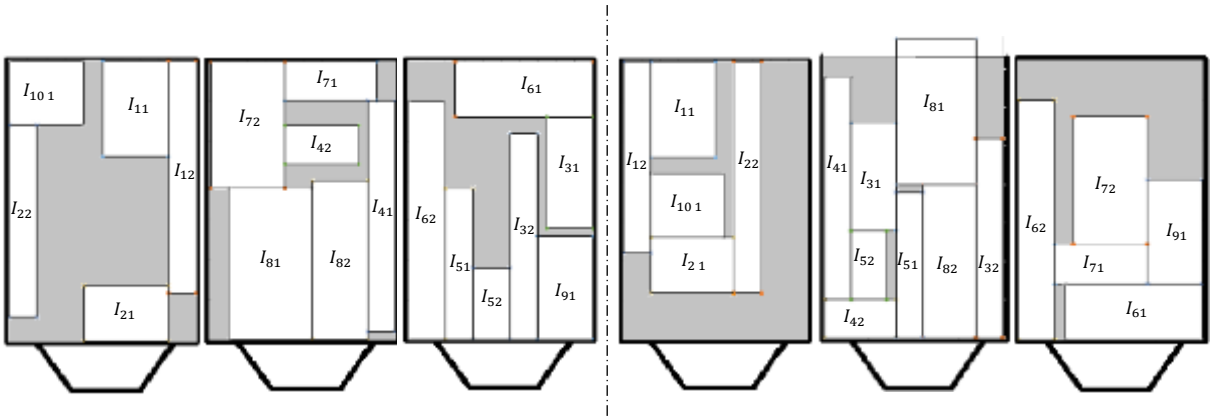
(b) Routes without loading constraints

Tran- transportation cost, inv- inventory holding cost, tard- tardiness cost, SD- starting day of the route

Fig 2 routes for 2LVRPRDD after the change in release date for order 8

Fig 2(b) illustrates the routes under the same conditions as in Fig 2(a), but without considering 2-dimensional loading. With the same change of release date for order 8, we observe a substantial amount change in the overall cost between routes, while considering two-dimensional loading (Fig 2(a)) and without loading (Fig 2(b)). In Fig 2(b), ignoring the loading constraints, the problem relates to VRPRDD. Let's note that order 8, is effectively served on route 0-3-4-5-8-0 started on day 2; but, while considering item loadings, this route becomes infeasible, and the model has to modify the routes, as shown in Fig 2(a). The order 8 comprises of large size items, making it difficult to be loaded along with other orders in route 0-3-4-5-8-0. Thus, due to the loading restrictions, both orders 3 and 5 get pushed to a later route 0-9-6-5-3-0 starting on day 3 (Fig 2(a)). Notably, the major cause for the rise in overall cost while considering two-dimensional loading is due to an increase in the tardiness cost for order 3 in route 0-9-6-5-3-0. Earlier without loading restrictions in route 0-3-4-5-8-0, order 3 was not delayed, but in route 0-9-6-5-3-0, order 3 gets delayed for 13 days. The change of release date for order 8 completely alters the route design and loading configuration, as may be seen from Fig 1(a). Further, the change in release date for a given problem instance can make its loading configuration infeasible. On the other hand, adjusting the route structure to generate a new feasible loading configuration may lead to an increase in the overall cost. Both Fig 1(a) and Fig 2(a) illustrate the impact of release dates altering route design to get

feasible loading configurations. Moreover, Fig 2(a) and Fig 2(b) show the impact of loading constraints, while routing orders with release and due dates.



(a) With loading constraints (b) Without loading constraints

Fig 3 Loading configuration for 2LVRPRDD after the change in release date for order 8

The loading configuration of the problem without loading constraints, as shown in Fig 3(b) reflects that the first item of order 8 (I_{81}) in route 0-3-4-8-5-0 cannot be accommodated in the given loading area. Thus, to accommodate this item, the route structure has to be changed, as shown in Fig 2(a). Two routes on both day 2 and day 3 have been modified to accommodate order 8. Notably, the loading configuration while considering 2-dimensional loading is shown in Fig 3(a); while both Fig 3(a) and Fig 3(b) show the impact of release dates in loading configuration. Without loading restrictions, the change of release date of order 8, actually resulted in a loading configuration, as in Fig 3 (b) with a total cost of 88 units. While considering loading constraints, the loading configurations of routes on both day 2 and day 3 had to be modified, as shown in Fig 3(a). In fact, making the routes load feasible effectively increased the overall cost to 117 units.

5. Solution methodology

Pollaris et al. (2014) showed that CVRP with loading is NP-hard; a few years later, Shelbourne et al. (2017) proved VRPRDD to be NP-hard also. Thus, our problem integrates the loading aspects to a known NP-hard problem, and thereby is NP-hard in itself, due to which, it is difficult to find good quality feasible solutions for large-sized instances. Thus, we propose a Scatter Search (SS)-based solution approach for larger instances. SS is a widely used as an evolutionary algorithm to efficiently solve diverse problems of optimization, including vehicle routing variants (Chu et al., 2006; Russell & Chiang, 2006). In fact, SS has been discussed in detail in literature (Glover, 1999; Glover et al., 2003). The performance of SS is enhanced using Strategic Oscillation (SO) principles in this study. We do so by relaxing the capacity restriction, and the loading length (L') of a given vehicle. Herein, it may be noted that Gendreau et al. (2008) had implemented similar capacity relaxation and relaxation of the loading length (L') in their proposed tabu search framework to solve 2L-CVRP. We allow controlled infeasibility in terms of both capacity and loading length violation in the routes of the solutions, which in turn facilitates better diversification. Further, we note that loading length violation effectively allows the items to be loaded beyond the loading area in the direction of length axis. SS aims to generate better solutions by systematically utilizing

good aspects of the trial solutions that are generated through an iterated cycle of controlled diversification and intensification. These 'good aspects' of the trial solutions aid in developing better offspring with enhanced solution quality. A typical solution comprises of a set of routes starting on different days, whereby every route comprises a subset of orders to be delivered, and each order in turn, constitutes a given set of items to be loaded in the vehicle. Thus, a feasible solution serves customer orders through various routes, without actually exceeding the available vehicle capacity and loading area. Like in many prior 2L-CVRP streams of literature, we also employ heuristics as well as exact algorithms to find feasible packing (Fuellerer et al., 2009; Gendreau et al., 2008). Therefore, as an important addition to SS we include a set of loading heuristics and a 'truncated branch and bound' method to search for feasible loading configurations. Notably, the loading heuristics are employed every time when the cost of a solution is computed inside the SS framework. Furthermore, the loading heuristics return a penalty corresponding to the loading length violation in the solution, which is later added with the overall cost of the solution.

Algorithm 1 describes the basic procedure employed in our scatter search. The penalty factors for loading (α) and capacity (β) decides the permissible amount of respective violations in the solutions. Lower α values can permit greater loading violations, allowing the loading length (L') thereby to go beyond the length of the loading area $\bar{L}$. Notably, a loading length is the length up to which the items are loaded in the vehicle, and is measured as the highest coordinate of the loaded items along the length axis. Similarly, lower β values allow orders to be accommodated more than the weight capacity of the route. Initially, both α and β values are kept to higher values to ensure feasible solutions (line 1). We generate b_1 number of good quality, coupled with sufficiently diverse solutions through the first diversification method based on an effective insertion based heuristics (Campbell and Savelsbergh, 2004), and store them in set $refSet_1$ (lines 2 and 3). We update the value of β and α as per line 4. Next, we generate b_2 number of highly diverse solutions through a second diversification method based on the random insertion of customer orders, and store them in set $refSet_2$. Importantly, both $refSet_1$ and $refSet_2$ form a combined reference set $refSet$. Further, we identify the common arcs among the solutions in $refSet$, and rank them according to their frequency of occurrence, along with the quality of parent solutions to which they belong. Additionally, we also consider the directions of arcs while identifying the common arcs, from which, matching arcs are used to form complete route or partial route fragments. The arcs are then prioritized for matching based on their rankings. Moreover, the complete and partial routes form partial solutions through the solution building process, which in turn allocate routes for different vehicles that are available on different days of the planning horizon. A solution repair method modifies the partial solutions by removing duplicate orders, while inserting missing ones to make them complete. Further, these complete orders may not be feasible in terms of loading and capacity restrictions, since they are relaxed through SO. The solution quality of these complete solutions is enhanced through two improvement methods. The first improvement method is inspired from the composite neighborhood structure of Shelbourne et al. (2017). The second improvement method is based on the geographical proximity of the order delivery locations (Soman and Patil, 2020). After the improvement process, the enhanced solutions with lower costs replace the higher cost elite solutions in $refset_1$. Later, the solutions in $refSet_1$ is evaluated for their loading and capacity feasibility. Based on the respective proportion of feasible solutions, the penalty factors for loading (α) and capacity violations (β) are updated.

Expression (24) shows how the value of β changes with the proportion of solutions having capacity violation (δ) in $refSet_1.\pi$ is the limit of proportion of allowable capacity infeasible solutions in the $refSet_1$. μ_1 is the revision factor for β.

$$\beta = \begin{cases} \beta \times \mu_1 & if\ \delta > \pi \\ \beta & if\ \delta = \pi \\ \frac{\beta}{\mu_1} & if\ \delta < \pi \end{cases} \quad (24)$$

Similar to β, the penalty for loading violation α gets updated depending on the proportion of infeasible solutions violating loading constraints. Expression (25) demonstrates how α value is increased when the proportion of solutions violating the length of loading area (λ) beyond the limit κ. Similarly, α value reduces when λ is less than the limit κ, which increases the loading infeasibility in the next SS iteration. μ_2 is the revision factor for α.

$$\alpha = \begin{cases} \alpha \times \mu_2 & if\ \lambda > \kappa \\ \alpha & if\ \lambda = \kappa \\ \frac{\alpha}{\mu_2} & if\ \lambda < \kappa \end{cases} \quad (25)$$

We update both β and α values when the costs of the offspring solutions do not change between iterations. The amount of penalty added to the total cost of the route subjected to capacity violation is equal to the product of β and the sum of the excess amount of order weights beyond the capacity of the vehicle. Similarly, the penalty for loading violation is the product of α and len, where len is the excess of loading length (L') above the length of the loading area ($\bar{L}$). Notably, both the penalties for loading length violation and capacity violation are computed, and added to the total cost of the routes whenever the cost of the solution is computed within SS. Further, through strategic oscillation, we allow infeasible solutions to be considered in our SS method. Note that the infeasible solutions considered in SS only violate capacity constraints and the length of loading, but respect all the other constraints in our model.

SS moves to the next iteration after updating $refSet_2$. A finite time limit is set as the termination criteria. The least-cost solution found so far through the iterations complying to all the feasibility conditions is then selected as the best solution. The detailed description of SS can be found in Soman and Patil (2020) study.

Algorithm 1 (Scatter Search with Strategic Oscillation)

1: Set β and α to high value β' and α'respectively.
2: Generate solutions using the first diversification method and store them in solution pool P.
3: Select b_1 low-cost solutions from P and add to $refSet_1$
4: Update the value of β as the average cost per unit demand of each route of the solutions in $refSet_1$ and $\alpha = 1$
5: Generate more solutions using first and second diversification method and store them in P
6: Select b_2 number of solutions from P, having the maximum diversity compared with solutions in $refSet_1$ and add them to $refSet_2$
7: Generate various subsets of solutions in $refSet$ using the subset generation method.
8: Compute the common arcs from various solution subsets and store them in set A.
9: Screen the common arcs in set A based on the arc scores/ranking and the

common arc selection threshold, γ.
10: Build partial and complete routes from the screened common arcs using route building method and store them in set RR.
11: Build solutions from the partial and complete routes in the set RR
12: If needed, repair the generated solutions using the solution repair method
13: Improve the generated solutions using the improvement methods
14: Update the $refSet_1$, if any better solutions are found from the generated solutions.
15: **if** solutions in $refSet_1$ are stagnated between iterations **then**
16: update β and α as per equation (24) and (25), respectively.
17: **if** the termination time limit is reached **then**
18: The least-cost solution found so far complying with all feasibility conditions, including the capacity and loading restriction is reported as the best solution.
19: **else** go to step 5.

5.1 Two-dimensional loading

In SS algorithm, we control the load feasibility through a penalty factor α. To determine the successful loading configurations, we employ a set of loading heuristics and a truncated branch and bound method. Specifically, we use six heuristics that are well known to check the loading feasibility of the route−Bottom Left Fill-W axis, Bottom Left Fill-L axis, Max Touching Perimeter heuristics, No wall Max Touching Perimeter heuristics, Min Area heuristics and Maximal open space heuristics. These six heuristics go on to form a loading heuristic bundle($heur\_bundle$). Importantly, such a multiple heuristics approach has been adopted even in extant literature (Zachariadis et al., 2009; Leung et al., 2013). If the loading heuristic bundle is not able to generate a feasible loading pattern, a hybrid of max perimeter and bottom left heuristics with local search (hybrid heuristics) is employed to estimate the minimum loading length(L'_{min}). Notably, if L'_{min} is more than $\bar{L}$, the length of loading area, then the amount of length violation (len) is computed. The length violation (len) is the excess of length over the length of loading area, ($len = L'_{min} - \bar{L}$). len is multiplied by the penalty factor α to compute the load penalty, which is then added to the cost of the route. We observe that many of the routes tend to repeat under different solutions across the iterations in SS. Further, some of the routes do have the same set of orders, but different sequence of visiting them. Unrestricted loading thereby aims to find a feasible loading configuration without actually considering the sequence of visit. Due to the route repetitions, an indexing strategy is adopted to store the load feasibility status of each route. The indexed packing strategy of Strodl et al. (2010) using STL maps has effectively inspired our indexing strategy. Herein, routes were labeled packable, un-packable, and unknown. The method of indexing is elaborated upon in section 5.2. Further, we note that the branch and bound method consumes more time as compared to the heuristics. Therefore we employ the same specifically to routes that get repeated for n' number of iterations, indexed with an 'unknown' status. Notably, the branch and bound method can consume significant computational time, so we limit the CPU time for the method to 10 sec, similar to the exact loading method of Strodl et al. (2010).

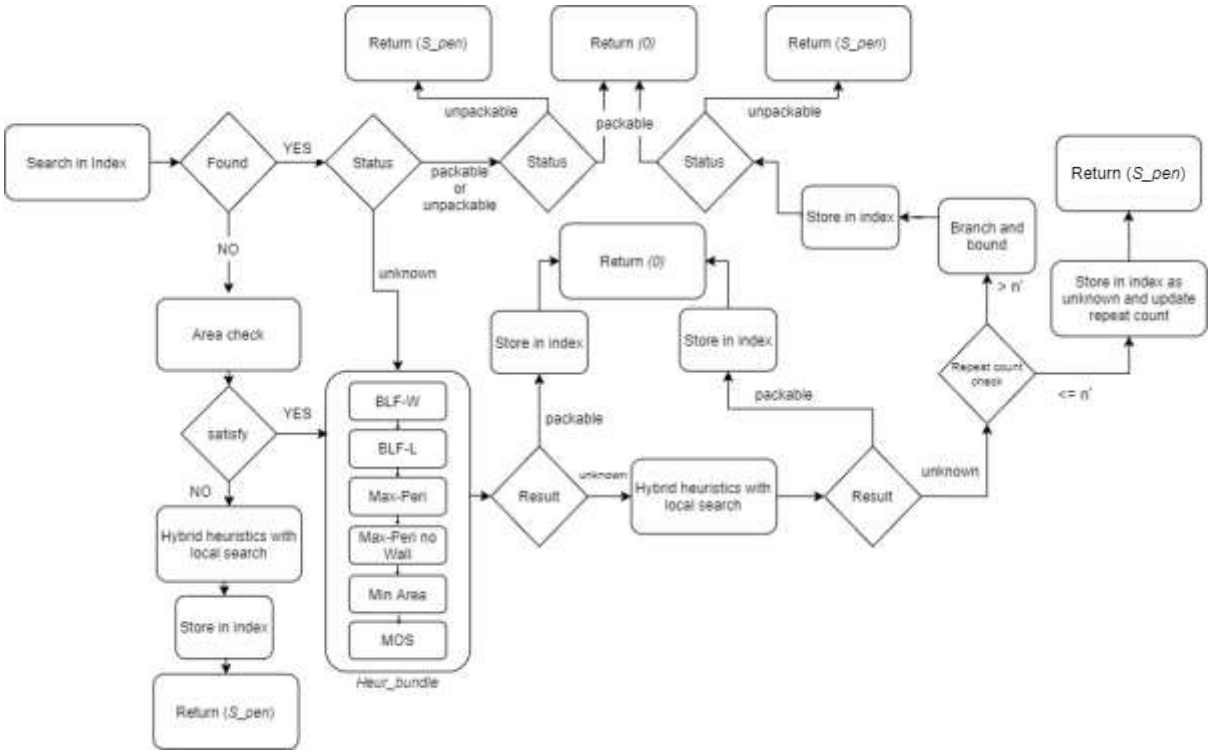


Fig 4 Flow diagram for the $load\_check$ function

A typical solution comprises of several routes. We determine the loading configuration for each route, and also use a function $load\_check$ to check for the load feasibility of a route. Fig 4 provides the flow diagram for $load\_check$ function. If the route is found infeasible in terms of loading, then $load\_check$ returns a load penalty, which is then added on to the total solution cost. Algorithm 2 describes the basic steps in $load\_check$.

Routes having the same orders, but different sequence of delivery can also have the same loading configurations. Moreover, routes are stored in the index library based on the orders they serve, and not their sequences of deliveries. Through the iterations of SS, there may be repetitions of routes having the same orders. Additionally, for every loading configuration, we store the loading penalty as S_pen. S_pen is the least penalty found for a given loading configuration. Whenever routes with $unknown$ status are repeated, each time loading penalty L_pen is calculated and compared with S_pen. If L_pen is less than S_pen, then S_pen is updated. $unknown$ status represents that no feasible loading configuration has been found so far for the given route. Furthermore, when the route is repeated for more than n' number of times without finding a feasible loading configuration, the truncated branch and bound method is used. However, even after using this method, we do not find any feasible configuration, then the status of the route is changed to $unpackable$, and the load penalty is fixed to S_pen. Importantly, $unpackable$ status categorises the route as infeasible in terms of loading. Feasible loading configuration returns zero as loading penalty and stores $S\_pen = 0$; once the feasible loading configuration is found, the status of the route is changed to $packable$

Algorithm 2 ($load\_check$)

INPUT : $route$ — S_pen: Least load penalty found for given route
OUTPUT: load penalty — L_pen: Load penalty found for given route

1: Search the index library for matching route
2: **if** route is found in index library **then**
3: for $packable$ or $unpackable$ status **return** S_pen as the loading penalty
4: **if** route is not found in index library or the status of route is $unknown$ **then**
5: **if** sum of area of all the items in route ($\sum_{t\in T_i, i\in route} l_{it} \times w_{it}$) > Loading area ($\bar{L} \times \bar{W}$) **then**
6: Run hybrid heuristics and compute loading penalty, $L\_pen = len \times \alpha$
7: Add the $route$ in the index library with $unpackable$ status and store $S\_pen = L\_pen$
8: **return** (S_pen)
9: **if** sum of area of all the items in route less than the loading area **then** do steps 10 to 30.
10: Use loading heuristics from $heur\_bundle$ in the sequence shown in Fig 4 until a feasible configuration is found for the $route$ or until all the heuristics are exhausted
11: **if** feasible loading configuration is found **then**
12: Add or modify the status of the $route$ as $packable$ in the index library, and store $S\_pen = 0$
13: **return**(S_pen)
14: **if** no feasible loading configuration is found from $heur\_bundle$ **then**
15: run hybrid heuristics with local search
16: **if** feasible loading configuration is found **then**
17: Add or modify the status of the $route$ as $packable$ in the index library and store $S\_pen = 0$, then **return** S_pen as the loading penalty
18: **if** no feasible loading configuration found **then**
19: **if** Number of times $route$ has repeated within SS iterations ($count$) $\leq n'$ **then**
20: Compute loading penalty, $L\_pen = len \times \alpha$, where len is the excess loading length from hybrid heuristics
21: **if** $L\_pen < S\_pen$ **then**
22: $S\_pen = L\_pen$
23: $count = count + 1$
24: Add or modify the status of the $route$ as $unknown$ and update S_pen in the index library, then **return** S_pen as the loading penalty.
25: **if** Number of times $route$ has repeated within SS iterations ($count$) $> n'$ **then**
26: run branch and bound method
27: **if** feasible loading configuration found **then**
28: update status of $route$ to $packable$, store $S\_pen = 0$ and **return** S_pen
29: **if** no feasible loading configuration found **then**
30: update status of $route$ to $unpackable$ and **return** last updated S_pen

5.2 Storage and updation of index library

An index library stores the details of the routes tested by the $load\_check$. Further, a route sent to $load\_check$ is cross-checked in the index library to find a matching route, which may be the same route searched by the $load\_check$ before, or a route containing the same set of orders as the given route, but having different delivery sequence. Every route is stored with a status of loading configuration, and the least loading penalty S_pen. There are three statuses−*packable, un-packable* and *unknown.* The *Packable* status is given for routes for

which a feasible loading configuration is found. *Un-packable* status is given for routes for which no feasible loading configuration was found. The routes for which the total area of items exceeds the loading compartment area, and routes for which no feasible loading configuration is found after employing truncated branch and bound method are also assigned as *Un-packable* status. *Unknown* status is given to routes for which feasible loading was not found at all, even after using the $heur\_bundle$, and hybrid heuristics with local search. We noted that the routes with $unknown$ status that repeated for more than n' times are sent to branch and bound method, which in turn, effectively assigns the *packable* or *un-packable status.* Importantly, for the $packable$ status, the least loading penalty S_pen is stored as 0. For $unknown$ status, S_pen is the least loading penalty found so far for the given route is stored. Additionally, in case of *un-packable* status, the last known S_pen is stored as the loading penalty. For new routes with *un-packable* status, the load penalty returned by the hybrid heuristics with local search is stored as S_pen. The idea behind storing S_pen, is that while routes get repeated each time the loading penalty L_pen is calculated, may increase or decrease, since it's a heuristic method. Storing the least loading penalty aid in penalizing the route based on the best-found loading configuration having the least loading length violation.

We use a STL map for storing the routes, along with their status. The advantage with the STL map is its binary tree storage with fast searching and retrieval mechanism. We use the approach of Strodl et al. (2010) and encode the routes as integers, corresponding to the binary representation of the orders served within the route. This encoded integer further acts as the key for the route in the STL map. Notably, the status and S_pen of the route is stored under this key in the map.

5.3 Loading heuristics

Before sending a route to any of the loading heuristics, an initial sequence of items to be loaded is computed. Importantly, every route has a subset of customer orders that need to be served, with each order containing multiple items. Initially, all these items are arranged in the decreasing order of their width in which ties are broken using the decreasing order of length (Gendreau et al., 2008). The items are loaded as per this sequence for all the heuristics in $heur\_bundle$. However, in hybrid heuristics, after the first iteration, this initial sequence may be altered by local search. In all the loading heuristics, the items are placed over a set of corner points, which in effect is a coordinate in the two-dimensional loading area constituted by both the length and width axis. An item's position is fixed at a corner point based on the desired criteria of individual heuristics. Furthermore, items are placed with its bottom left corner at the corner point, and both its length and width dimensions parallel to the length and width of the loading compartment, respectively. Nevertheless, placing the items at all the corner points may not be feasible due to violations of certain packing conditions. For instance, items are not allowed to overlap each other, and/or cross the dimensions of the loading compartment. Thus, all the items of the order should be loaded in the same loading compartment, wherein the items are not allowed to change their orientation while loading.

The $heur\_bundle$ comprise of six heuristics. Bottom left fill-W axis heuristics (BLF-W) places an item at a feasible corner point with the minimum width coordinate and breaking the ties with minimum length coordinate. Similarly, Bottom left fill-L axis heuristics (BLF-L) places an item at a feasible corner point with minimum length coordinate, and breaks the ties with minimum width coordinate. Maximum touching perimeter heuristics (Max_peri) places

an item at a feasible corner point, which has the maximum touching perimeter. Touching perimeter represents the perimeter that an item shares with its neighboring items, and the dimensions of the loading compartment. In the case of a tie corner point, the minimum value of loading length (L') is considered. Notably, no wall maximum touching perimeter heuristics (Max_peri no wall) is same as the Max_peri heuristics without considering the dimensions of the loading compartment for computing the touching perimeter. Initially, in the Max_peri heuristics, the items are placed at the sides of the loading compartment, since touching perimeter is nearer to the edges, and gets filled towards the center at a later point. In Max_peri No wall heuristics, items are first filled at the inner parts of the compartment and later to the sides. Minimum area heuristics (Min_area) considers the rectangular surface areas at each feasible corner point, and places an item at a corner point having the minimum area. These rectangular areas are the free spaces within the loading compartment without any items placed in it. For detailed description of the above well know heuristics, the reader may refer to Zachariadis et al. (2009) study. Maximal Open Space (MOS) heuristics of Wei et al. (2018) considered packing of items to the Maximal Open Space (MOS) with minimal fitness measure. Open space is the free space inside the loading area of the vehicle, which shares an edge with the rear end of the vehicle. MOS is an open space that is not enclosed by any other open spaces. Placing an item in a MOS would generate new open spaces, and some of them would include MOS. Fitness measure is the number of new MOS generated by placing an item at a given MOS. Notably, the item is finally placed at the MOS with minimal fitness measure.

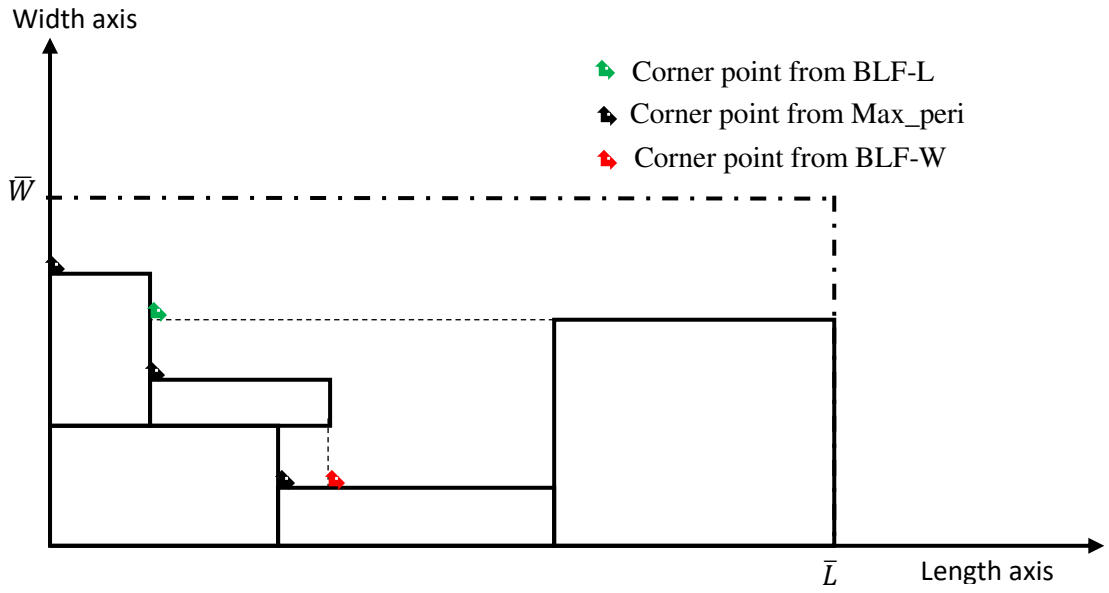


Fig 5 corner points generated in hybrid heuristics

Hybrid heuristics with local search combines maximum touching perimeter heuristics and bottom left fill heuristics. When all the heuristics in the $heur\_bundle$ fail to generate a feasible loading configuration, hybrid heuristics with local search is used. The output from hybrid heuristics is len, representing the excess of loading length. If the minimum loading length, L'_{min} is less than or equal to $\bar{L}$ (length of loading area), a feasible loading configuration is found ($len = 0$) otherwise, the excess of loading length, $len = L'_{min} - \bar{L}$ is computed. Furthermore, hybrid heuristics incorporate both the corner points generated from bottom left fill heuristics, as well as the maximum perimeter heuristics. Additionally, corner

points with minimal width coordinate (from BLF-W) and minimal length coordinates (from BLF-L) are included. The placement of items depends on the maximum touching perimeter criteria. Fig 5 shows the corner points generated in hybrid heuristics.

Algorithm 3 (Hybrid heuristics)

INPUT: $route$
OUTPUT: len, the excess of loading length over the length of loading compartment $\bar{L}$
Iteration counter : $count \rightarrow 1$
L'_{min} : Minimum loading length found over the iterations
L' : Loading length
Initialize $L'_{min} = \bar{L}$
1: Sort all the items in the route in the decreasing order of width and breaking ties with decreasing order of their length and form loading sequence
2: Keep the first item at the bottom left corner of the loading compartment
3: find corner points and store in the set $corner\_pnt$
4: **for** each item in the loading sequence except the first item **do**
5: **for** each corner point in set $corner\_pnt$ **do**
6: **if** loading feasibility conditions are satisfied at the given corner point **then**
7: Estimate the touching perimeter at the given corner point for the selected item
8: Fix the position of the selected item at the corner point with maximum touching perimeter and break the ties with minimum value of loading length, L'.
9: Update the $corner\_pnt$ set by adding new corner points and removing the absolute ones.
10: **if** loading length, $L' \leq$ length of the loading compartment, $\bar{L}$ **then**
11: **return** $(len = 0)$
12: **else**
14: **if** $count \leq hyb\_limit$ **then**
15: if $L' < L'_{min}$ then
16: $L'_{min} = L'$
15: Initiate local search to generate alternate loading sequence
16: $count++$
16: **goto** step 2
17: **else**
18: $len = L'_{min} - \bar{L}$
19: **return** (len)

The initial sequence of items to be loaded is determined in the decreasing order of item width and breaking ties with decreasing order of length. Hybrid heuristics work in iterations. For the first iteration, the initial sequence of items is used. Once all the items are positioned based on the maximum touching perimeter criteria, the loading length, L' is computed; if $L' \leq \bar{L}$ feasible loading configuration is found and len is returned as zero. If no feasible loading configuration is found ($L' > \bar{L}$), a local search is implemented, altering the sequence of

loading thereof. 2-opt insertion and swap moves are used in local search to modify the loading sequence. Hybrid heuristics move to the next iteration using the new sequence of loading. Iterations are interrupted when a feasible loading pattern is found; otherwise, it would continue for a predetermined limit of iterations (hyb_limit), and the least value of len found so far across the iteration is returned. Algorithm 3 describes the basic steps in hybrid heuristics.

Moreover, in hybrid heuristics, we employ a local search to generate a better loading sequence, which in turn, minimizes the loading length L'. Changing the loading sequence within a route does not alter the cost of the route. The composite neighborhood structure employed in the first improvement method uses neighborhood moves in-between routes of a solution. These in-between route moves can effectively alter the loading sequences in individual routes. At the same time, it can change the cost of the solution, since the delivery sequences of routes are changed. In order to generate better loading configuration, both the neighborhood moves that impart position change for items within the route (local search of hybrid heuristics) and the moves, which exchange items in- between routes (moves in first improvement method) are necessary. Zachariadis et al. (2009) had used a set of neighborhoods involving customer relocation, route exchange, and route interchange with the aim of better loading configuration. These neighborhood structures are similar to the composite neighborhood employed in our first improvement method.

The branch and bound method for loading is used when route with *unknown* status is repeated for more than n'number of iterations. The loading part of the mathematical model in section 3 (constraints (11) to (16)) is exclusively implemented for the subset of items served by the route. If a feasible solution is indeed obtained, then the route is *packable*, else *un-packable*. Depending on the loading status of the route $load\_check$ function imposes load penalty. $load\_check$ returns zero load penalty for $packable$ routes. For $unpackable$ and $unknown$ status routes S_pen is returned. Once a route is labelled $unpackable$, it is not considered again for computing load configuration. $load\_check$ returns the last known S_pen as load penalty for $unpackable$ routes.

6 Results and Discussion

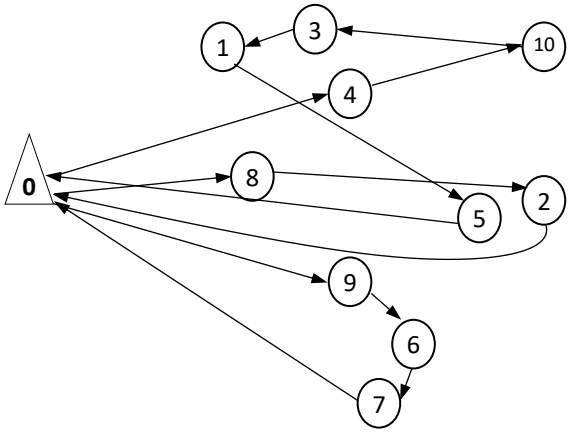
The mathematical model and the solution heuristics were programmed in C++. CPLEX 12.7 was employed to solve the mathematical model and the truncated branch and bound method for loading. The numerical experiments were run on 20 cores server with each core of Intel® Xeon® E5-2670 v2 processor @ 2.5 GHz. RAM of 128 Gigabytes in a sharing mode. The mathematical model in CPLEX was run with four threads in parallel. 3600 seconds of CPU time was set as the termination time limit for the heuristics in the case of larger datasets (order size more than 50), and 1800 seconds was kept for the smaller datasets (order sizes less than 50). Additionally, CPLEX was run for 24 hours to solve the mathematical model. It reported either an optimal solution (for smaller instances) or the best integer solution with the optimality gap at the time of termination.

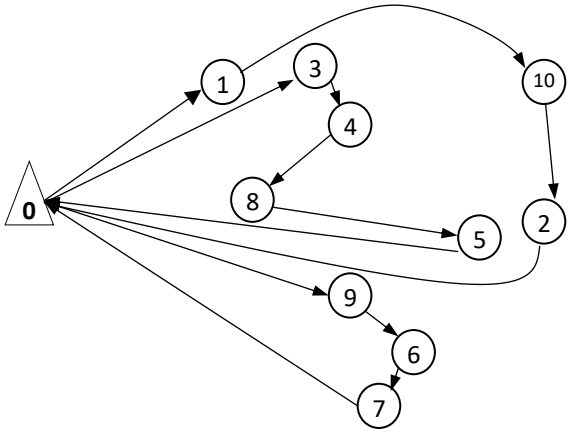
6.1 Managerial Implications

To investigate the trade-offs inherent in our problem, we consider three scenarios by altering various parameters of the example problem as described in section 4. We compare the results

of SS both with and without loading restrictions to demonstrate the impact of loading configurations in routing in the presence of release and due dates.

In the first scenario, we increase the tardiness penalty of customer orders four and ten to 10 and 11 units, respectively, to understand the role of tardiness costs. Fig 6(a) and Fig 6(b) illustrate the resulting routes both with loading restrictions and without loading restrictions. The higher tardiness penalties of orders 4 and 10 results in serving them earlier in the route to avert delayed delivery. In the route 0-1-3-4-10-5-0 from Fig 1, order 10 was delayed by 5 days, but as the tardiness cost is increased, the order was served earlier (routes 0-1-10-2-0 in Fig 6(b) and 0-4-10-3-1-5-0 in Fig 6(a)). The effect of the loading is demonstrated in Fig 6(a). The route 0-3-4-8-5-0 from Fig 6(b) becomes infeasible under the loading constraints, and the orders are served by other load feasible routes in Fig 6(a). In Fig 3(b) of section 4, the infeasibility in accommodating the first item of the order 8 (I_{81}) for the route 0-3-4-8-5-0 was shown. To accommodate this item, the route structure has to be changed, as shown in Fig 6(a). Unlike in 2L-CVRP, we cannot accommodate customer orders in any route/vehicle with space available. Importantly, in 2L-VRPRDD, we should ensure that the orders are positioned in those vehicles respecting the release dates of the orders.





Routes	Cost			SD
	Tran	inv	tard	
0-8-2-0	17	0	1	1
0-4-10-3-1-5-0	27	2	45	2
0-9-6-7-0	18	1	7	3
Total	118			

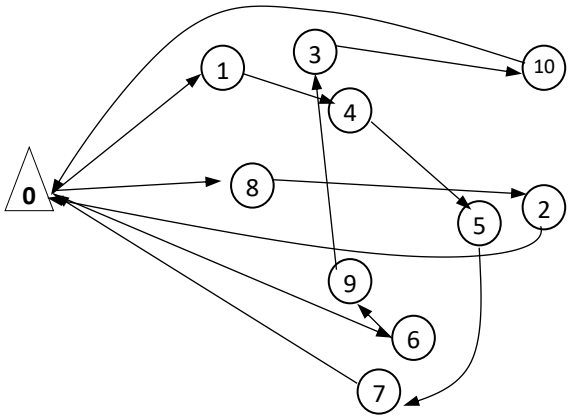
Routes	Cost			SD
	Tran	inv	tard	
0-1-10-2-0	21	0	6	1
0-3-4-8-5-0	20	0	15	2
0-9-6-7-0	18	1	7	3
Total	88			

Tran- transportation cost, inv- inventory holding cost, tard- tardiness cost, SD- starting day of the route

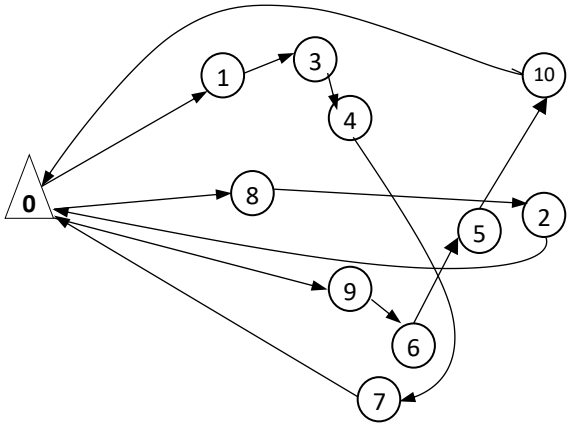
(a) VRPRDD with loading restriction (b) VRPRDD without loading restriction

Fig 6 Scenario 1 for VRPRDD with higher tardiness cost

In the second scenario, we alter the inventory holding cost per day of orders 7 and 8 to 45 and 20 units. Due to the high holding cost, the routes are redesigned in such a way that orders 1 and 7 are dispatched on the same day of their release. Fig 7 shows the resulting routes with and without loading restrictions. Similar to scenario 1, due to the loading limitations, the routes in Fig 7(b) are altered as given in Fig 7(a) in the expense of additional cost.



Routes	Cost			SD
	Tran	inv	tard	
0-8-2-0	17	0	1	1
0-1-4-5-7-0	23	1	14	2
0-6-9-3-10-0	27	3	40	3
Total	126			



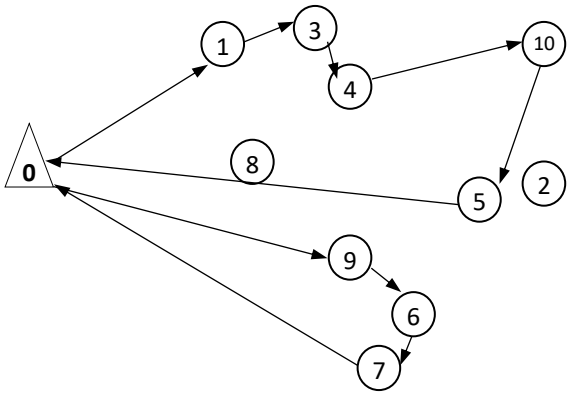
Routes	Cost			SD
	Tran	inv	tard	
0-8-2-0	17	0	1	1
0-1-3-4-7-0	24	1	10	2
0-9-6-5-10-0	23	4	21	3
Total	101			

Tran- transportation cost; inv- inventory holding cost; tard- tardiness cost; SD- starting day of the route

(a)VRPRDD with loading restriction (b) VRPRDD without loading restriction

Fig 7 Scenario 2 for VRPRDD with higher inventory holding cost

Another observation is the higher transportation cost in route 0-6-9-3-10-0, as shown in Fig 7(a). A better route in terms of transportation cost is 0-9-6-10-3-0, which has 25 units of transportation cost, and 3 units of holding cost. However, it has a tardiness cost of 45 units, increasing thereby the total cost to 73, which is higher compared to the 70 unit total cost of 0-6-9-3-10-0.



Routes	Cost			SD
	Tran	inv	tard	
0-1-3-4-10-5-0	22	2	18	2
0-9-6-7-0	18	1	7	3
Total	$2U + 68$			

Tran- transportation cost; inv- inventory holding cost; tard- tardiness cost; SD- starting day of the route

Fig 8 Scenario 3 for VRPRDD with limited transportation capacity

The third scenario considers a limited logistics capacity, where the model has to decide upon the orders that can be served in the current planning horizon. This scenario has practical relevance in environments with limited loading capabilities. Herein, it is assumed that only one vehicle can be loaded on day 2 and day 3, respectively, while no loading happens on day 1. The resulting route is shown in Fig 8. Moreover, the capacity of both the routes is exhausted, while both orders 8 and 2 are back-ordered. Notably, order 2 has the highest

weight, and is located far off from the depot, making it difficult to serve thereof. Order 8 on the other hand, has the highest area, which makes it difficult to be loaded with other orders, given that only two vehicles are available for delivery. Geographically, order 8 is closer to the depot, and has moderate order weight, but still larger area holds it up from loading.

The influence of tardiness and inventory holding costs on designing the routes and the load configuration is apparent from the results above. The temporal aggregation and lateness-based penalty provide diverse tradeoff opportunities. The firm thereby has to decide upon all orders that can effectively be consolidated, and has to also take a call on the ones that can be delayed in order to ensure that the overall cost of delivery of all orders can be minimized. Generating feasible loading configurations to accommodate the items in the vehicles poses an additional challenge. Our model can aid decision-makers in choosing effective routing solutions under various preferences with feasible loading configurations. In the limited loading capacity case, it becomes even more difficult, as the firm has to consciously decide the orders that need to be unfulfilled. The traditional vehicle routing approach that involves only the transportation cost, may not be sufficient to address the above tradeoffs involved with both release and due dates of orders. So, it is important to include the additional costs along with transportation costs to optimize the delivery routes.

In the subsequent sections, we study the computational performance of the proposed algorithm by creating test instances that represent our problem.

6.2 Dataset construction

As 2L-VRPRDD is a new variant, we modify the 2L-CVRP datasets from extant literature (Zachariadis et al., 2009; Wei et al., 2018). The release and due dates, along with the associated cost parameters are incorporated in the selected instances from 2L-CVRP to generate 2L-VRPRDD datasets. The customers in 2L-CVRP are termed as customer orders in 2L-VRPRDD, and the customer demands denote the order weights. The 2L-CVRP data instances are available at http://or.dei.unibo.it/library.

Manufacturing firms often tend to plan their distribution and transportation schedules in finite intervals of days of a week. We observed that a capsule manufacturing unit prepares a transportation schedule with dispatches twice a week from Monday to Wednesday and Thursday to Saturday, considering Sunday as a holiday. For each dataset, we randomly select the span of the planning horizon between three to six days. The release dates are uniform, and randomly generated for the orders within the span of the planning horizon. For generating the due dates for the orders, we follow the approach of Shelbourne et al. (2017). Notably, the due date of an order relates to its release date $F(i)$, an estimate of average travel time (τ'), and a uniformly generated random factor $Z(0,2uv)$. The variations between due dates of orders are determined by looseness value (u) and the range of looseness (v). The random factor, $Z(0,2uv)$ on the other hand, is a uniform distributed continuous number between 0 and $2uv$. The estimate of average travel time is computed as $\tau' = \frac{\sum_{i\epsilon O}\sum_{j\epsilon N'}\tau_{ij}}{|O|*\left\|\left[y^{\frac{1}{2}}\right]\right\|}$, $y = |N|$ is the number of nodes in the graph, O is the set of orders and τ_{ij} is the time between two nodes i

and j. $|[a]|$ will around a to the nearest integer. N' represents a set of $\left|\left[y^{\frac{1}{2}}\right]\right|$ number of nodes having the least time of transport from $i \in O$.

The due date d_i $i \in O$, is determined using the equation (26).

$$d_i = F(i) + |[(u + Z(0,2uv) - uv) \times \tau']| \quad (26)$$

We consider the travel time equitant to transportation cost $(\tau_{ij} = c_{ij}, \forall (i,j) \in E)$. The estimate of average travel time computed can thus be considered as an estimate for the average travel cost. The inventory holding cost per day $(h_i, i \in O)$ for each order is generated uniform randomly as $U(\frac{\tau'}{8}, \frac{\tau'}{16})$. Similarly, tardiness cost per day $(q_i, i \in O)$ for each order is generated as $U(\frac{\tau'}{2}, \frac{\tau'}{8})$. The looseness value, u for the due dates vary from 4 to 12 and v = 0.25. At every customer order location, a service time of one day is allotted. In order to aid order consolidation across days, we assume that the maximum number of routes that can be planned (the maximum number of vehicles that can be loaded) in the planning horizon are higher than the number of vehicles provided in the respective 2L-CVRP data instance. Notably, a similar approach of including additional vehicles was considered in Shelbourne et al. (2017) study, whereby the maximum number of vehicles that could be loaded per day were randomly allocated to days in the planning horizon. For our study, we consider the proportion of extra routes, e_v uniform randomly generated between 0 and 0.5. The total number of vehicles needed is computed as $\lceil NV \times (1 + e_v) \rceil$, where NV represents the number of vehicles available in the corresponding 2L-CVRP data instance. Further, it is assumed that the third-party logistics service provider does have sufficient supply to provide for the required vehicles each day.

6.3 Validation of mathematical model and solution methodology

We modify the benchmark 2L-CVRP instances as equivalent 2L-VRPRDD instances in order to validate our mathematical model and solution methodology. According to lemma 1, through suitable modifications, the 2L-CVRP data instances may be solved by using our 2L-VRPRDD formulation. As a result, solving the equivalent 2L-VRPRDD instances and comparing the results with the benchmark 2L-CVRP instances can validate our mathematical model and the SS heuristic. We compare the results of our mathematical model and heuristic with the 2L-CVRP solutions with that of Strodl et al. (2010). The solutions for the benchmark 2L-CVRP instances are available at http://prolog.univie.ac.at/research/VRPandBPP/. Table 3 shows the computational results of the 2L-CVRP instances.

Each data instance has five classes; class 1 is the CVRP equivalent with unit load dimensions. Classes 2 to 5 alter both the number and dimensions of the items under each customer order. For most of the benchmark instances, our heuristics were able to provide the best-known solutions, which effectively validate our solution methodology. The last column, 'Diff' shows the percentage difference between the SS solution and the benchmark solutions. Additionally, CPLEX was able to provide optimal solutions to smaller instances, using our mathematical formulation. The optimal solution of our formulation, along with the benchmark solution of the CVRP matched, and thereby validated our mathematical model. Moreover, CPLEX

provided considerably good solutions to medium-sized instances, but the optimality gap (opt) seems to be large. Similarly, the proposed scatter search method provided optimal solutions for the small-sized problems and validated the correctness of the algorithm.

Table 3 Computational result for benchmark 2L-CVRP instances

		Our Formulation						Our Formulation			
Dataset	Bench	Cost	Opt (%)	SS	Diff	Dataset	Bench	`Cost	Opt (%)	SS	Diff
0101	278.726	278.73*	-	278.726	0%	0601	495.848	498.16	27.56	495.848	0%
0102	278.726	278.73*	-	278.726	0%	0602	495.848	501.05	26.88	495.848	0%
0103	284.519	284.519	1.77	284.519	0%	0603	498.157	499.07	27.15	498.157	0%
0104	282.947	282.95*	-	282.947	0%	0604	498.539	556.19	46.74	498.539	0%
0105	278.726	278.73*	-	278.726	0%	0605	495.848	501.04	29.91	495.848	0%
0201	334.96	334.96*	-	334.96	0%	0701	568.563	568.56*	-	568.562	0%
0202	334.96	334.96*	-	334.96	0%	0702	725.46	779.50	40.94	725.459	0%
0203	352.159	352.159*	-	352.159	0%	0703	678.749	713.93	35.24	685.383	1%
0204	334.96	334.964*	-	334.96	0%	0704	700.718	703.84	33.97	703.85	0%
0205	334.96	334.964*	-	334.96	0%	0705	658.641	674.22	30.94	661.223	0%
0301	358.402	358.402*	-	358.402	0%	0801	568.562	568.56*	-	568.562	0%
0302	387.704	387.704	20.6	387.704	0%	0802	674.551	725.56	25.72	676.842	0%
0303	390.547	417.81	35.28	390.547	0%	0803	738.433	773.72	37.14	741.116	0%
0304	364.454	368.56	14.54	364.454	0%	0804	692.475	758.68	37.4	700.983	1%
0305	358.402	358.402	16.25	358.402	0%	0805	621.846	679.68	23.89	642.222	3%
0401	430.885	430.885*	-	430.885	0%	0901	607.651	607.65	37.89	607.651	0%
0402	430.885	430.88	5.4	430.885	0%	0902	607.651	607.65	39.72	607.651	0%
0403	430.887	440.94	14.58	430.885	0%	0903	607.65	607.65	36.52	607.651	0%
0404	447.373	447.37	20.42	447.373	0%	0904	625.096	655.2	48.53	625.096	0%
0405	430.885	430.88	17.03	430.885	0%	0905	607.65	625.83	46.11	607.651	0%
0501	375.28	375.28	4.78	375.28	0%	1001	535.797	543.25	36.1	535.797	0%
0502	375.282	375.28	6.54	375.282	0%	1002	689.684	785.05	60.76	698.859	1%
0503	381.69	408.08	26.84	381.689	0%	1003	615.679	688.79	56.1	643.75	4%
0504	383.875	383.875	13.31	383.875	0%	1004	710.87	782.36	60.28	755.81	6%
0505	375.28	375.28	13.75	375.28	0%	1005	692.373	761.58	59.19	701.417	1%

Bench- Benchmark solutions from Strodl et al. (2010); * Optimal solution; opt- optimality gap; SS- Scatter Search solution; Diff- the percentage difference between the SS solution and Benchmark solution.

6.4 Parameter configuration

The performance of metaheuristics depends upon the appropriate setting of parameter values. The main parameters of our SS heuristics are b_1 (size of $refSet_1$), b_2 (size of $refSet_2$), γ (common arc selection threshold), μ_1 (revision factor for β), π (limit of the proportion of capacity infeasible solutions(δ) in $refSet_1$), μ_2 (revision factor for α), and κ(limit of proportion of load infeasible solutions (λ) in $refSet_1$). As suggested in Guo and Tang (2015) and Nepomuceno et al. (2015) studies, we consider 10 solutions as the size of $refSet$, whereby both b_1 and b_2 are taken as pairs with three levels (3,7), (5,5) and (7,3) (Guo and Tang, 2015). For γ, three sets of values $(0.1, 0.5, 0.8)$ are considered. For π and κ, two levels are considered $(0.3, 0.5)$. Notably, the levels for γ, π and κ are decided through a set of pilot experiments. The values for some parameters are found robust throughout the pilot test, and it

helped us in fixing those values to the parameters like $\mu_1 = 1.2$ and $\mu_2 = 1.5$. Further, we employ a full factorial test with four parameters, wherein we consider ten representative data instances (four 2L_CVRP and six 2L_VRPRDD) for the experimentation. We consider both 2L_CVRP and 2L_VRPRDD datasets to ensure sufficient diversity in the experiments. A total of 360 $(3 \times 3 \times 2 \times 2 \times 10)$ tests has been conducted for the full factorial design. We used ANOVA to identify the main effects. Importantly, our analysis suggests that levels (3,7) for (b_1, b_2), 0.1 for γ and 0.5 for π and κ are statistically significant in finding low-cost solutions.

6.4.1 Hybrid loading heuristics using local search and random permutations.

Hybrid loading heuristics work iteratively. It returns the minimum excess loading length found after a specific number of iterations. For each iteration, there is a sequence in which items are loaded. Initially, the sequence of items to be loaded is determined based on decreasing width criteria. If the feasible loading configuration is not found, we alter the sequence of items. Importantly, the sequence of the items may be altered by random permutations or through sequential local search moves. We conduct a series of experiments using random permutations and local search through random swap moves (2opt) and insertion moves. For every iteration in the hybrid loading heuristics, we accept the sequences, which help in reducing the excess loading length, and discard the non-promising ones. Once completed, we compare the results of both approaches, as shown in Table 4.

Table 4 Result of SS with hybrid loading heuristics using permutations and local search

Dataset	With permutations	With local search	Improvement with local search	Dataset	With permutations	With local search	Improvement with local search
3301	2527.04	1716.1	32.09%	3501	1927.59	1222.36	36.59%
3302	4484.12	3669.37	18.17%	3502	2241.84	2225.14	0.74%
3303	4490.11	3796.17	15.45%	3503	2382.14	2355.42	1.12%
3304	4514.67	3673.44	18.63%	3504	2445.68	2411.87	1.38%
3305	3846.34	3183.22	17.24%	3505	2207.03	2115.83	4.13%
3401	3360.98	1065.11	68.31%	3601	2424.15	1591.82	34.33%
3402	2002.25	1712.69	14.46%	3602	2879.31	1962.17	31.85%
3403	1987.81	1947.05	2.05%	3603	3086.61	2982.32	3.38%
3404	2023.61	1987.59	1.78%	3604	2843.99	2784.12	2.11%
3405	1914	1577.32	17.59%	3605	3014.78	2865.58	4.95%

From Table 4, it may further be observed that for most of the datasets, the local search approach provided, actually improved the solutions as compared with the random permutation approach. It is because, in the local search approach, we store the best loading sequence, which provides the minimum excess loading length in the iteration. In contrast, in the permutation approach, the sequence of the items generated in each iteration, is completely different from the prior best sequence. Furthermore, in the local search approach, both the insert and swap moves find an improved neighborhood loading sequence. In contrast, changing the entire sequence in random permutation approach we end up moving far away from the current best loading sequence. Resultantly, the local search approach finds better loading sequences than the random permutation approach. Additionally, due to the better

performance of the local search approach, we employ it in the hybrid loading heuristics in our proposed SS method.

6.4.2 Time limit for branch and bound loading method

The branch and bound method for finding a loading configuration can become time-consuming, and can affect the overall computation time of the SS method. Prior literature employing this method to find loading configuration had put a time limit, truncating it therefore (Gendreau et al., 2008; Strodl et al., 2010). Strodl et al. (2010) used 10 seconds as the time limit in their branch and bound loading method. To fix the time limit for the branch and bound loading method, we compare the performance of our SS for three cutoff time limits (10 sec, 30 sec, 60 sec) as shown in Table 5. Interestingly, the cut off time of 10 seconds provided better solutions as compared to the other time limits.

Table 5 Comparison of results of SS with various cutoff time for the branch and bound loading method

	Cut off time limit				Cut off time limit		
Dataset	10 Sec	30 sec	60 sec	Dataset	10 Sec	30 sec	60 sec
3301	1716.1	2152.91	2302.23	3501	1222.36	1629.66	1698.01
3302	3669.37	3693.23	3687.52	3502	2225.14	2288.89	2260.92
3303	3796.17	3805.97	3839.68	3503	2355.42	2399.25	2411.05
3304	3673.44	3868.81	3695.97	3504	2411.87	2433.5	2462.06
3305	3183.22	3271.6	3400.25	3505	2115.83	2127.45	2148.39
3401	1065.11	1265.47	1333.31	3601	1591.82	1884.25	1962.6
3402	1712.69	2005.11	2008.72	3602	1962.17	2928.76	2897.69
3403	1947.05	1989.55	1964.31	3603	2982.32	3043.34	3038.22
3404	1987.59	2056.7	2028.62	3604	2784.12	2902.16	2800.61
3405	1577.32	1953.19	1918.89	3605	2865.58	2876.72	3143.51

6.5 Computational time performance of the SS on 2L-VRPRDD datasets

Table 6 shows the results of the computational experiments done on the generated datasets using SS. For datasets with order size up to 50, CPU time is of 1800 seconds, and for order sizes above 50, CPU time is of 3600 seconds. A similar time frame was also used in Wei et al. (2018) study covering 2L-CVRP instances.

Table 6 Computational experiment on 2L-VRPRDD datasets

Instance	Class 1	Class 2	Class 3	Class 4	Class 5
1	312.304	320.654	313.279	315.631	317.088
2	360.431	352.944	370.08	351.778	358.228
3	448.372	441.532	421.146	396.341	401.716
4	466.237	473.25	464.156	488.239	475.46
5	424.098	455.385	435.766	449.337	435.718
6	518.847	537.705	552.989	556.724	529.668
7	848.794	835.096	833.478	766.308	793.266
8	771.273	865.522	1046.37	775.397	796.086

9	655.691	668.706	696.435	680.532	651.09
10	954.215	1496.22	1007.26	1325.23	914.823
11	1047.81	1029.93	950.739	1964.75	1019.42
12	659.237	676.676	687.872	684.081	662.865
13	2288.84	3104.67	2925.15	3251.99	2739.79
14	1244.16	1348.66	1366.19	1321.88	1165.38
15	1007.64	1264.87	1301.84	1436.03	1439.95
16	774.085	760.329	777.113	769.171	748.562
17	929.548	936.64	927.97	893.754	951.439
18	922.987	1344.34	1380.02	1513.97	1279.84
19	875.378	964.384	976.294	982.324	778.636
20	384.934	687.465	878.277	858.361	674.361
21	847.587	1441.79	1696.04	1430.36	1184.81
22	909.288	1417.67	1599.73	1608.11	1274.66
23	1079.59	1353.88	1537.54	1501.83	1159.9
24	1231.49	1427.2	1475.49	1335.56	1397.37
25	1027.98	2179.46	2282.57	2349.81	1607.96
26	976.667	1981.2	2116.23	2233.69	1723.18
27	1277.87	1957.45	1919.45	1966.37	1551.94
28	3858.15	5476.7	5914.57	5029.72	4680.99
29	2270.02	4005.57	4730.17	4188.17	3698.89
30	1334.57	3250.89	3367.22	2685.31	2598.74
31	1866.23	3529.94	3563.31	3787.38	3076.84
32	1735.99	3479.51	3603.43	3495	3381.11
33	1716.1	3669.37	3796.17	3673.44	3183.22
34	1065.11	1712.69	1947.05	1987.59	1577.32
35	1222.36	2225.14	2355.42	2411.87	2115.83
36	1591.82	1962.17	2982.32	2784.12	2865.58

2L-VRPRDD instances are classified into five classes, similar to the 2L-CVRP instances. Class 1 contains loads of unit dimensions, making the instance equitant to VRPRDD. We consider the instances up to 255 customer order size and 786 items. Our SS method was able to generate good quality solutions for all the instances. In comparison to the respective 2L-CVRP instances, the cost of solutions for 2L-VRPRDD datasets is high. The additional cost of inventory storage associated with order consolidation and tardiness cost of delayed orders increases the overall cost. The release dates of orders forbid them to be routed in vehicles starting on earlier dates, unlike in 2L-CVRP cases, where any vehicle is able to serve any orders on any day. This inherent nature of our problem makes it challenging to solve, as compared to the 2L-CVRP variant. Moreover, the delayed delivery and order aggregation options can provide various tradeoffs that can help in cost savings, and build better loading configurations for the delivery vehicles thereby.

6.6 Multi-start Iterated Local Search

We compare the performance of the proposed SS methodology with a multi-start iterated local search heuristic (ILS), adopted from Shelbourne et al. (2017) study, with minor modifications to suit our problem context. When there is no improvement in the solution

quality for a specific number of iterations, we initiate the multi-start operation. The diversification module of our scatter search (SS) method is used to generate initial solutions for every multi-start operation. Notably, a composite neighborhood comprises of insertion, swap, inter-route, two opt, and exchange operators, which are all used in the intensification stage. For diversification, we use a kick operator, where the customer orders are randomly swapped within the solution. Table 7 compares the ILS approach results, and our proposed SS method on selected large size datasets. Importantly, similar to SS, a one-hour time limit was kept as the time limit for the ILS approach.

Table 7 Relative performance of SS over ILS

Instance	SS	ILS	improvement	Instance	SS	ILS	improvement
2201	909.288	1003.07	9.35%	2404	1335.56	1780.84	25.00%
2202	1417.67	1766.42	19.74%	2405	1397.37	1838.75	24.00%
2203	1599.73	2005.54	20.23%	2501	1027.98	1313.08	21.71%
2204	1608.11	2105.62	23.63%	2502	2179.46	2477.58	12.03%
2205	1274.66	1721.33	25.95%	2503	2282.57	2705.75	15.64%
2301	1079.59	1326.17	18.59%	2504	2349.81	2625.85	10.51%
2302	1353.88	1835.12	26.22%	2505	1607.96	2320.66	30.71%
2303	1537.54	2091.75	26.50%	2601	976.667	1284.22	23.95%
2304	1501.83	2140.12	29.82%	2602	1981.2	2789.65	28.98%
2305	1159.9	1705	31.97%	2603	2116.23	2798.38	24.38%
2401	1231.49	1487.99	17.24%	2604	2233.69	2784.49	19.78%
2402	1427.2	1996.73	28.52%	2605	1723.18	2479.71	30.51%
2403	1475.49	1926.66	23.42%				

The results from Table 7 show that our SS method outperforms the ILS approach. In fact, for most of the data instances, we noted an improvement of over 10%, when we employed SS. It is because SS systematically processes a pool of elite (low cost) and diverse solutions (as discussed in detail in the proposed method). On the contrary, ILS uses a single solution pass consisting of the phases of intensification and diversification within an iteration. Moreover, the advantage of the pool-based mechanism of SS over ILS is its ability to store characteristics of multiple good solutions instead of a single solution. When multiple good solutions are considered, SS identified the common 'good' traits in those solutions with the help of adaptive memory, which subsequently generated higher quality offspring solutions.

6.7 Performance of Strategic oscillation (SO)

We employ strategic oscillation (SO) principles in SS, allowing for infeasible solutions that violate both loading length and vehicle capacity in the search process. In order to control the infeasibility in the loading length, we chose to impose a penalty, which is a function of the loading length violation. Effectively, SO periodically updates the penalizing factor α in accordance with the amount of loading length violation in the solutions of each iteration. The infeasible solutions obtained through SS may contain potential traits that can lead to promising solutions in future iterations. Notably, the solutions easily accessible from both the feasible and infeasible regions differ significantly. Oscillating between the feasibility boundaries may provide an enriched set of options to carry out the search. Thus, SO

principles have been used in our algorithm as an effective means of diversification (Gallego et al., 2013; Martí et al., 2018).

Additionally, to investigate whether SO contributes to generating high-quality solutions in the search process, we compare the performance of our SS both with and without strategic oscillation. Three variants of SS are proposed. Variant 1, is the SS without any SO components, variant 2 is the SS with SO relaxing only the capacity of the routes, and variant 3 is the SS with SO relaxing both capacity and loading length constraints. Variant 3 in effect, provides higher quality solutions as compared to other variants. The incremental improvement in solution quality from the second variant to the third variant shows the influence of controlled penalization in loading length violation. The results thereby suggest that SO mechanisms do contribute to the generation of high-quality solutions.

Table 8 Performance of proposed SS with and without strategic oscillations

Dataset	Basic SS	SS_SO	SS_full	Dataset	Basic SS	SS_SO	SS_full
3301	1721.63	1717.28	1716.1	3501	1257.36	1252.74	1222.36
3302	4499.25	4071.37	3669.37	3502	2840.82	2592.31	2225.14
3303	4421.33	4055.51	3796.17	3503	2962.99	2916.62	2355.42
3304	4203.16	4064.24	3673.44	3504	3193.59	2986.66	2411.87
3305	3817.09	3387.45	3183.22	3505	2478.11	2389.74	2115.83
3401	1078.13	1071.87	1065.11	3601	1621.26	1633.1	1591.82
3402	2215.87	2226.83	1712.69	3602	3656.85	3620.96	1962.17
3403	2271.37	2283.97	1947.05	3603	3931.85	3791.37	2982.32
3404	2408.13	2348.36	1987.59	3604	3613.24	3583.16	2784.12
3405	2143.65	2228.77	1577.32	3605	3520	3415.22	2865.58

Basic SS – SS without any SO; SS_SO – SS with only SO in terms of the capacity violation; SS_full – SS with all SO components

6.8 Performance of loading heuristics

We have proposed a bundle of loading heuristics to find a feasible loading configuration for the routes. We conducted a series of experiments to compare the performance of individual loading heuristics and the bundle of heuristics (all loading heuristics used together). Table 9 suggests that SS with the loading heuristics bundle provided better quality solutions than all the individual loading heuristics. The comparison of the individual loading heuristics shows that Bottom Left Fill-W axis heuristic and Max touching perimeter heuristic performed better compared to the other loading heuristics.

Table 9 Performance of loading heuristics

Dataset	Bottom_L	Bottom_W	Hybrid	Max_peri	Max_peri no_wall	Min_area	MOS	Combined heuristic bundle
3401	1318.65	1379.65	1361.65	1347.93	1325.93	1348.52	1348.52	1065.11
3402	2096.3	1993.64	2042.91	2047.5	2068.6	2069.45	2025.12	1712.69
3403	2003.69	1969.72	1992.66	2009.02	2067.74	2007.4	2015.33	1947.05
3404	2028.8	2107.34	2024.26	2045.09	2100.92	2047.78	2051.57	1987.59

3405	1991.86	1923.14	1893.87	1981.33	1959.25	1955.69	1953.75	1577.32
3501	2058.44	1638.14	2009.68	1625.81	1628.14	1562.16	1562.16	1222.36
3502	2326.94	2293.07	2282.27	2241.37	2438.44	2313.49	2298.26	2225.14
3503	2464.41	2405.17	2395.06	2382.1	2539.4	2486.24	2443.99	2355.42
3504	2627.62	2489.02	2488.62	2486.12	2616.3	2504.98	2521.31	2411.87
3505	2186.22	2133.78	2154.26	2152.84	2188.66	2255.39	2155.08	2115.83
3601	1876.72	1918.92	2352	1853.27	1886.37	2024.98	2024.98	1591.82
3602	3034.24	2963.13	3030.72	3007.31	3041.74	2993.34	3009.3	1962.17
3603	3145.3	3077.67	3170.28	3041	3219.71	3167.79	3219.33	2982.32
3604	2996.39	2819.41	2885.64	2897.27	2985.29	2905.7	3026.85	2784.12
3605	3621.93	2954.04	2895.84	2885.62	3017.73	3014.17	3168.05	2865.58

Bottom_L – Bottom left fill- L axis heuristics; Bottom_W – Bottom left fill- W axis; Max_peri – Max touching perimeter heuristics; Max_peri_no_wall – No wall Max Touching Perimeter heuristics; Min_area – Min Area heuristics; MOS – Maximal open space heuristics;

7 Conclusion

In this paper, we introduced a new variant of vehicle routing problem with release and due dates considering temporal aggregation and lateness dependant tardiness cost while adhering to the restrictions of two-dimensional loading. We proposed a mixed-integer linear mathematical formulation with the objective of minimizing the total cost inclusive of transportation cost, inventory holding cost, and tardiness cost. We demonstrated the impact of aggregation of orders and delayed delivery while generating feasible 2-dimensional loading configurations. Our model was able to address many managerial issues involved with delivering orders with release and due dates. We considered several tradeoffs in our model. The storage of orders for consolidation generated better loading configurations, higher utilization of the vehicle and reduce the transportation cost while increasing the associated inventory holding and tardiness costs. So our model assists in selecting orders for aggregation or for delayed delivery that can generate feasible two-dimensional loading configuration and minimize the overall delivery cost. As our model functions in finite logistics capacity environments, in case of insufficient transportation capacity across the planning horizon, it finds the sub-set of orders that have to be back-ordered to the next horizon.

To solve realistic sizable data instances, we developed a scatter search method with strategic oscillation. Strategic oscillation was proposed to increase the effectiveness of SS through systematic diversification. Imposing controlled infeasibility in the solutions has helped in escaping local optima. To generate feasible loading configurations, a collection of well know loading heuristics found to be useful. The effective indexing strategy and storage of packing status of routes enabled swift computation. In the numerical experiments, the proposed SS method was found highly effective in generating good quality solutions for the 2L-VRPRDD problem variant.

Compliance with Ethical Standards

- Disclosure of potential conflicts of interest

The authors declare that they have no conflict of interest.

- Research involving Human Participants and/or Animals

This manuscript does not contain any studies with human participants or animals performed by any of the authors.

- Informed consent

Informed consent was obtained from all individual participants included in the study

Conflicts of interest

- The authors have no relevant financial or non-financial interests to disclose.

Declarations

- **Funding** Not applicable

- **Conflicts of interest/Competing interests** There are no conflicts of interest

- **Availability of data and material** The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.
- **Code availability** Code developed for the model will be made available from the corresponding author on reasonable request.

Data availability statement

- The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Authors' contributions

- Conceptualization: Jaikishan T Soman and Rahul J Patil; Methodology: Jaikishan T Soman; Formal analysis and investigation: Jaikishan T Soman and Rahul J Patil; Writing - original draft preparation: Jaikishan T Soman; Writing - review and editing: Rahul J Patil; Funding acquisition: Not Applicable; Resources: Rahul J Patil; Supervision: Rahul J Patil

References

Archetti, C., Feillet, D., Mor, A., & Speranza, M. G. (2018). An iterated local search for the Traveling Salesman Problem with release dates and completion time minimization. *Computers and Operations Research, 98,24-37*. https://doi.org/10.1016/j.cor.2018.05.001

Archetti, C., Feillet, D., & Speranza, M. G. (2015a). Complexity of routing problems with release dates. *European Journal of Operational Research*, *247*(3), 797–803.

Archetti, C., Jabali, O., & Speranza, M. G. (2015b). Multi-period Vehicle Routing Problem with Due dates. *Computers and Operations Research*, *61*, 122–134. https://doi.org/10.1016/j.cor.2015.03.014

Arda, Y., Crama, Y., Kronus, D., Pironet, T., & Van Hentenryck, P. (2014). Multi-period vehicle loading with stochastic release dates. *EURO Journal on Transportation and Logistics*, *3*(2), 93–119. https://doi.org/10.1007/s13676-013-0035-z

Attanasio, A., Fuduli, A., Ghiani, G., & Triki, C. (2007). Integrated Shipment Dispatching and Packing Problems : a Case Study. *J Math Model Algor, 6*, 77–85. https://doi.org/10.1007/s10852-006-9050-5

Bortfeldt, A., & Yi, J. (2020). The Split Delivery Vehicle Routing Problem with three-dimensional loading constraints. *European Journal of Operational Research*, *282*(2), 545–558.

Campbell, A. M., & Savelsbergh, M. (2004). Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. *Transportation Science*, *38*(3), 369–378. https://doi.org/10.1287/trsc.1030.0046

Campelo, P., Neves-Moreira, F., Amorim, P., & Almada-Lobo, B. (2018). Consistent vehicle routing problem with service level agreements: A case study in the pharmaceutical distribution sector. *European Journal of Operational Research*, *273*(1), 131–145. https://doi.org/10.1016/j.ejor.2018.07.030

Cattaruzza, D., Absi, N., & Feillet, D. (2016). The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, *50*(2), 676–693. https://doi.org/10.1287/trsc.2015.0608

Chu, F., Labadi, N., & Prins, C. (2006). A Scatter Search for the periodic capacitated arc routing problem. *European Journal of Operational Research*, *169*(2), 586–605. https://doi.org/10.1016/j.ejor.2004.08.017

Dominguez, O., Juan, A. A., & Faulin, J. (2014). A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without item rotations. *International Transactions in Operational Research, 21(3)*, 375–398. https://doi.org/10.1111/itor.12070

Felipe, A., & Ortuño, M. T. (2012). An adapted heuristic approach for a clustered traveling salesman problem with loading constraints. *4OR-Q J Oper Res, 10(3),* 245–265. https://doi.org/10.1007/s10288-012-0207-y

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2009). Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers and Operations Research*, *36*(3), 655–673. https://doi.org/10.1016/j.cor.2007.10.021

Gallego, M., Laguna, M., Martí, R., & Duarte, A. (2013). Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society*, *64*(5), 724–734. https://doi.org/10.1057/jors.2012.66

Gendreau, M., Iori, M., Laporte, G., & Martello, S. (2008). A Tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks: An International Journal*, *51*(1), 4–18.

Glover, F., (1999). Scatter search and path relinking. In: Corne, D., Dorigo,M., Glover,F., (Eds.). New Ideas in Optimization (pp 297-316). New York: McGraw Hill.

Glover, F., Laguna, M., & Marti, R. (2003). Scatter Search. *Advances in Evolutionary Computing SE - 20*, 519–537. https://doi.org/10.1007/978-3-642-18965-4_20

Guo, Q., & Tang, L. (2015). An improved scatter search algorithm for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. *Applied Soft Computing Journal*, *29*, 184–195. https://doi.org/10.1016/j.asoc.2014.12.030

Iori, M., Salazar-González, J. J., & Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, *41*(2), 253–264. https://doi.org/10.1287/trsc.1060.0165

Khebbache-Hadji, S., Prins, C., Yalaoui, A., & Reghioui, M. (2013). Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows. *Central European Journal of Operations Research*, *21*(2), 307–336. https://doi.org/10.1007/s10100-011-0204-9

Klapp, M. A., Erera, A. L., & Toriello, A. (2018). The dynamic dispatch waves problem for same-day delivery. *European Journal of Operational Research, 271*(2), 519-534. https://doi.org/10.1016/j.ejor.2018.05.032

Klapp, M. A., Erera, A. L., Toriello, A., & Stewart, M. (2016). The one-dimensional dynamic dispatch waves problem. *Transportation Science*, *52*(2), 402–415. https://doi.org/10.1287/trsc.2016.0682

Leung, S. C. H., Zhang, Z., Zhang, D., Hua, X., & Lim, M. K. (2013). Discrete Optimization A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. *European Journal of Operational Research*, *225*(2), 199–210. https://doi.org/10.1016/j.ejor.2012.09.023

Ligterink, N. E., Tavasszy, L. A., & Lange, R. De. (2012). A velocity and payload dependent emission model for heavy-duty road freight transportation. *Transportation Research Part D: Transport and Environment*, *17*(6), 487-491. https://doi.org/10.1016/j.trd.2012.05.009

Liu, L., Li, K., & Liu, Z. (2017). A capacitated vehicle routing problem with order available time in e-commerce industry. *Engineering Optimization*, *49*(3), 449–465. https://doi.org/10.1080/0305215X.2016.1188092

Martí, R., Campos, V., Hoff, A., & Peiró, J. (2018). Heuristics for the min–max arc crossing problem in graphs. *Expert Systems with Applications*, *109*, 100–113. https://doi.org/10.1016/j.eswa.2018.05.008

Martı, L., & Ã, C. A. (2012). A vehicle routing problem with multi-trips and time windows for circular items. *Journal of the Operational Research Society*, *64*(11), 1630-1643.

https://doi.org/10.1057/jors.2012.128

McKinnon, A. C. (2005). The economic and environmental benefits of increasing maximum truck weight: The British experience. *Transportation Research Part D: Transport and Environment*, *10*(1), 77–95. https://doi.org/10.1016/j.trd.2004.09.006

Moons, S., Ramaekers, K., Caris, A., & Arda, Y. (2017). Integration of order picking and vehicle routing in a B2C e-commerce context. *Flexible Services and Manufacturing Journal*, *30*(4), 813-843. https://doi.org/10.1007/s10696-017-9287-5

Mor, A., & Speranza, M. G. (2020). Vehicle routing problems over time: a survey. *4OR-Q J Oper Res*, *18*(2), 129–149. https://doi.org/10.1007/s10288-020-00433-2

Moura, A., & Oliveira, J. F. (2009). An integrated approach to the vehicle routing and container loading problems. *OR Spectrum*, *31*(4), 775–800. https://doi.org/10.1007/s00291-008-0129-4

Nepomuceno, J. A., Troncoso, A., & Aguilar-Ruiz, J. S. (2015). Scatter search-based identification of local patterns with positive and negative correlations in gene expression data. *Applied Soft Computing Journal*, *35*, 637–651. https://doi.org/10.1016/j.asoc.2015.06.019

Pisinger, D., & Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, *2*(2), 154–167. https://doi.org/10.1016/j.disopt.2005.01.002

Pollaris, H. (2018). Loading constraints in vehicle routing problems: a focus on axle weight limits. *4OR-Q J Oper Res* 16**,** 105–106 (2018). https://doi.org/10.1007/s10288-017-0352-4

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., & Limbourg, S. (2014). Capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints. *EURO Journal on Transportation and Logistics*, *5*(2), 231-255 https://doi.org/10.1007/s13676-014-0064-2

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., & Limbourg, S. (2015). Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, *37*(2), 297–330. https://doi.org/10.1007/s00291-014-0386-3

Reyes, Damián, Erera, A. L., & Savelsbergh, M. W. P. P. (2018). Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research*, *266*(1), 29–34. https://doi.org/10.1016/j.ejor.2017.09.020

Reyes, Damian, Erera, A., Savelsbergh, M., Sahasrabudhe, S., & O'Neil, R. (2018). The Meal Delivery Routing Problem. *Optimization Online*.

Russell, R. A., & Chiang, W. C. (2006). Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research*, *169*(2), 606–622. https://doi.org/10.1016/j.ejor.2004.08.018

Shelbourne, B. C., Battarra, M., & Potts, C. N. (2017). The vehicle routing problem with release and due dates. *INFORMS Journal on Computing*, *29*(4), 705–723. https://doi.org/10.1287/ijoc.2017.0756

Shelbourne, C., Battarra, M., & Chris, N. (2017). The Vehicle Routing Problem with Release and Due Dates. *INFORMS Journal on Computing*, *29*(4), 705–723.

https://doi.org/10.1287/ijoc.2017.0756

Soman, J. T., & Patil, R. J. (2020). A Scatter Search Method for Heterogeneous Fleet Vehicle Routing Problem with Release dates under Lateness Dependent Tardiness Costs. *Expert Systems with Applications*, 150, 113302.

Strodl, J., Doerner, K. F., Tricoire, F., & Hartl, R. F. (2010). On Index Structures in Hybrid Metaheuristics for Routing Problems with Hard Feasibility Checks : An Application to the 2-Dimensional Loading Vehicle Routing Problem. *In: Blesa M.J., Blum C., Raidl G., Roli A., Sampels M. (eds) Hybrid Metaheuristics. HM 2010. Lecture Notes in Computer Science, vol 6373,160-173. Springer, Berlin, Heidelberg.*

Ulmer, M. W., Thomas, B. W., Campbell, A. M., & Woyak, N. (2021). The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, *55*(1), 75-100

Voccia, S. A., Melissa Campbell, A., & Thomas, B. W. (2017). The same-day delivery problem for online purchases. *Transportation Science, 53*(1), 167-184.

Wei, L., Zhang, Z., Zhang, D., & Leung, S. C. H. (2018). A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, *265*(3), 843–859. https://doi.org/10.1016/j.ejor.2017.08.035

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints. *European Journal of Operational Research*, *195*(3), 729–743. https://doi.org/10.1016/j.ejor.2007.05.058

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2013). Integrated distribution and loading planning via a compact metaheuristic algorithm. *European Journal of Operational Research*, *228*(1), 56–71. https://doi.org/10.1016/j.ejor.2013.01.040