



The mobile production vehicle routing problem Using 3D printing in last mile distribution

Wang, Yu; Ropke, Stefan; Wen, Min; Bergh, Simon

Published in:
European Journal of Operational Research

Link to article, DOI:
[10.1016/j.ejor.2022.06.038](https://doi.org/10.1016/j.ejor.2022.06.038)

Publication date:
2022

Document Version
Peer reviewed version

[Link back to DTU Orbit](#)

Citation (APA):
Wang, Y., Ropke, S., Wen, M., & Bergh, S. (2022). The mobile production vehicle routing problem: Using 3D printing in last mile distribution. *European Journal of Operational Research*, 305(3), 1407-1423.
<https://doi.org/10.1016/j.ejor.2022.06.038>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Journal Pre-proof

The Mobile Production Vehicle Routing Problem: Using 3D Printing in Last Mile Distribution

Yu Wang, Stefan Ropke, Min Wen, Simon Bergh

PII: S0377-2217(22)00512-4
DOI: <https://doi.org/10.1016/j.ejor.2022.06.038>
Reference: EOR 17956



To appear in: *European Journal of Operational Research*

Received date: 4 June 2021
Accepted date: 17 June 2022

Please cite this article as: Yu Wang, Stefan Ropke, Min Wen, Simon Bergh, The Mobile Production Vehicle Routing Problem: Using 3D Printing in Last Mile Distribution, *European Journal of Operational Research* (2022), doi: <https://doi.org/10.1016/j.ejor.2022.06.038>

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2022 Published by Elsevier B.V.

Highlights

- We introduce a new variant of the Vehicle Routing Problem, the Mobile Production Vehicle Routing Problem (MoP-VRP).
- An efficient Adaptive Large Neighborhood Search Heuristic as well as acceleration strategies are proposed.
- We propose realistic MoP-VRP instances and make them available to the scientific community.
- We show that the mobile production and delivery mode is flexible and can lead to faster delivery times.
- We show that the cost of production and delivery using the proposed logistics mode is reasonable.

Journal Pre-proof

The Mobile Production Vehicle Routing Problem: Using 3D Printing in Last Mile Distribution

Yu Wang^a, Stefan Ropke^b, Min Wen^{a,*}, Simon Bergh^c

^aDepartment of Applied Mathematics, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu 215123, China

^bDTU Management, Technical University of Denmark, Akademivej Building 358, 2800 Kgs. Lyngby, Denmark

^c3D Printhuset A/S - Skudehavnsvej 17A, 2150 Nordhavn

Abstract

We study a new variant of the vehicle routing problem, called the Mobile Production Vehicle Routing Problem (MoP-VRP). In this problem, vehicles are equipped with 3D printers, and production takes place on the way to the customer. The objective is to minimize the weighted cost incurred by travel and delay of service. We formulate a Mixed Integer Programming (MIP) model and develop an Adaptive Large Neighbourhood Search (ALNS) heuristic for this problem. To show the advantage of mobile production, we compare the problem with the Central Production Vehicle Routing Problem (CP-VRP), where production takes place in a central depot. We also propose an efficient ALNS for the CP-VRP. We generate benchmark instances based on Vehicle Routing Problem with Time Windows (VRPTW) benchmark instances, and realistic instances based on real-life data provided by the Danish Company 3D Printhuset. Overall, the proposed ALNS for both problems are efficient, and we solve instances up to 200 customers within a short computational time. We test different scenarios with varying numbers of machines in each vehicle, as well as different production time. The results show that these are the key factors that influence travel and delay costs. The key advantage of mobile production is flexibility: it can shorten the time span from the start of production to the delivery of products, and at the same time lower delivery costs. Moreover, long-term cost estimations show that this technology has low operation costs and thus is feasible in real life practice.

Keywords: Metaheuristics, Transportation, Mobile Production, Vehicle Routing Problem, 3D Printing

1. Introduction

In the modern supply chain system, balancing the interests of manufacturers, distributors, and customers has always been a difficult problem. [Manufacturers](#) and Distributors want to minimize their operation costs. Customers, meanwhile want to receive a good quality of service. Among the various costly factors behind the operations, inventory management is a long-time problem that has complicated the production and circulation of commodities. Annual reports at Amazon, for example, have listed the inventory problem as a risk factor within the operational process for two consecutive years (Amazon, 2017; Amazon, 2018). In 2017,

*Corresponding authors

Email addresses: yu.wang1803@student.xjtlu.edu.cn (Yu Wang), ropke@dtu.dk (Stefan Ropke), min.wen@xjtlu.edu.cn (Min Wen), skb@3dprinthuset.dk (Simon Bergh)

Amazon reported that a 1 percent inventory increase would cost an additional 180 million USD (Amazon, 2017). With the rapid development of industrial manufacturing, an effective solution called make-to-order (MTO) was proposed to reduce such inventory costs.

In MTO mode, when the central warehouse receives an order, production begins there directly. The number of goods to be produced is the same as the number of orders received (i.e., no excess good is made). When production is completed, trucks deliver the assigned goods to the customers, as shown in Figure 1. To minimize the total cost, the practitioners want to find the optimal production scheduling and transportation scheme. This need has raised an optimization problem, which we have named *the Central Production-Vehicle Routing Problem (CP-VRP)*.

Although the on-demand central production in MTO mode can eliminate inventory costs, the potential drawback is that it is not flexible enough because waiting for the completion of production may delay delivery to the customers. This disadvantage does not conform to customers' wish for a high-quality service, which may ultimately lead to a decline in both service quality and brand reputation.

To improve and innovate the logistics model, Amazon applied for a patent (US Patent 9684919) in 2015 (Amazon Technologies, 2015; Boyle, 2018) that combines 3D printing technology and MTO mode. With this patent, a car-mounted 3D printer produces the products on the way to the customer, as shown in Figure 2. By applying this technique, production and transportation are synchronized and delivery times can be largely reduced.

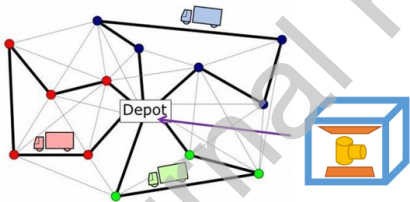


Figure 1: The Central Production

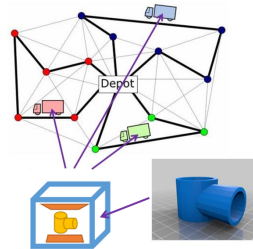


Figure 2: The Mobile Production

This zero-inventory, synchronous production, and transportation technology can meet people's demand for customization and prompt delivery, all while reducing the cost of logistics. Thus, applying this technology can be a good strategy to balance the interests of manufacturers, distributors, and customers in the modern supply chain system. With the further upgrading of industrial manufacturing and the continuous maturation of 3D printing technology, the era of intelligent manufacturing and personalized customization is coming. Thus, this new technology will have great potential for large-scale use in the future.

The integration of the car-mounted 3D-printing factory and the vehicle routing problem is denoted as the *Mobile Production Vehicle Routing Problem (MoP-VRP)*. Unlike the CP-VRP, in which delivery starts only after production is completed at the depot, production and distribution in the MoP-VRP take place synchronously. This setting makes the MoP-VRP more complex than the CP-VRP, as production and delivery

are more closely connected. This close connection also makes the existing methods proposed for the CP-VRP inapplicable to the MoP-VRP, thus we need to find a new suitable algorithm to solve the MoP-VRP.

The proposed MoP-VRP considers a limited homogeneous vehicle fleet. A certain number of machines are installed in each vehicle. Production starts at the time the vehicle departs from the depot. Each customer has a strict starting time window and a soft ending time window. The vehicle should return to the depot before a certain time. The objective is to minimize the weighted sum of travel and delay cost, where the travel cost is based on the total travel length and the delay cost is based on the delayed time.

In this paper, we introduce the MoP-VRP and analyse its properties. We present mixed integer programming models for the MoP-VRP and the CP-VRP. We develop an Adaptive Large Neighbourhood Search (ALNS) algorithm for both problems. By analysing the computational results for the two problems, we illustrate the advantages of mobile production.

Most of the MoP-VRP's applications will be in the future, when 3D printing technology has advanced to a state where popular consumer goods can be 3D printed on demand. However, an application that, in principle, could be rolled out at the time of writing is that of producing spare parts needed for repairs. Most of the machinery and equipment that we rely on in everyday life contains plastic parts that can break down. Examples of such equipment include everything from coffee machines, printers, and photocopiers to plumbing parts or even cars, buses, and planes. When a plastic part breaks down, the user may be unable to use the machine or equipment and have to wait for a repair. That repair may have to wait until spare parts are shipped from a distant warehouse, and for older equipment the spare parts may not even be available. If the spare part can be printed by a 3D printer mounted in a vehicle, the customer may get the spare part on the day she makes her request or even within hours if the MoP-VRP is solved in a dynamic setting (the present paper only considers the static MoP-VRP; the dynamic version is left for future work).

3D printers have already been used on board ships to print spare parts for repairs while at sea (Krassenstein, 2014). One may be concerned about the quality of printing as the production environment is likely to be unstable both at sea and on the road. To solve this potential risk, Phillips et al. (2020) used passive stabilization to help on board 3D printers provide equally good quality products as the traditional land-based laboratory. Based on this, we find it likely that 3D printing on board a moving van is feasible using appropriate passive stabilization. It should be noted, that the technology that we envisage to be used in a spare parts delivery scenario is fused filament fabrication (FFF), which from the beginning is more robust to movement compared to the stereolithography (SLA) technology studied by Phillips et al. (2020).

The computational results in section 5.5 show that the cost for each delivery can be kept at a reasonable level if a decent amount of daily customers can be attracted to the service. The biggest challenge, in rolling out the envisioned MoP-VRP spare-parts service may be in building a vast library of 3D models that correspond to possible spare parts and in obtaining the rights to print spare parts for many different manufacturers.

The contribution of this work is five-fold: 1) We introduce the MoP-VRP, a new, non-trivial variant of the vehicle routing problem; 2) we study properties of the MoP-VRP and CP-VRP that allows us to accelerate

heuristics for the two problems; 3) we design and implement Adaptive Large Neighborhood heuristics for both problems; 4) we propose testing instances for the two problems including a set of benchmark instances and a set of realistic instances in cooperation with 3D Printhuset; 5) we perform an extensive computational experiment to show advantages of the two forms of production and routing. [This work is an extension of the master thesis \(Bergh, 2018\).](#)

The rest of the paper is organized as follows: a literature review on related works is presented in section 2. The problem setting is described in section 3; based on this, two integer programming models, necessary assumptions, definitions, and properties for the defined problem are presented. We introduce the ALNS algorithm in section 4 and present computational results along with findings in section 5. Finally, concluding remarks are summarized in section 6.

2. Literature Review

This section presents a review of 1) the Central Production Vehicle Routing Problem and 2) mobile production technology.

2.1. The Central Production Vehicle Routing Problem

In order to effectively use resources, the optimization of production scheduling and transportation schemes in the classical MTO model have been widely studied. Different works may deal with unique problem settings, and the name given to the solved problem can vary from work to work. In the review by Moons et al. (2017), this name is generalised as *the Production Scheduling and Vehicle Routing Problem (PS-VRP)*. The most commonly seen names in the literature are the integrated/joint production-distribution scheduling problem (Belo-Filho et al., 2015; Guo et al., 2017; Mohammadi et al., 2020; Yağmur and Kesen, 2020) and the integrated production scheduling and vehicle routing problem (Chen and Vairaktarakis, 2005; Zou et al., 2018). In this work, we name it as *the Central Production Vehicle Routing Problem (CP-VRP)*.

With respect to optimization strategy, many enterprises apply a decomposition method in practice to consider production scheduling and transportation schemes independently and optimize them separately (Scholz-Reiter et al., 2011; Moons et al., 2017). While the decomposition method reduces the difficulty of solving the problem, it is hard to obtain the global best solution due to the split of problem, and the obtained solution can be quite different from the optimal one (Moons et al., 2017). To use resources more efficiently and improve service quality, more and more companies are beginning to study the integrated optimization strategy, which is to combine central production and transportation decisions into a single optimization problem (Moons et al., 2017; Chang and Lee, 2004).

In recent years, in-depth research has been conducted on the CP-VRP at various levels, and has theoretically proved that the zero-inventory MTO model can effectively help companies eliminate risks in inventory expenses and cash flow. In the literature, a large number of variants of the CP-VRP have been studied, and the corresponding mathematical models or theoretical derivations have been proposed (Amorim et al.,

2013; Jamili et al., 2016). The variants of the problem can be classified according to the production mode and further classified as single machine production (Tamannaee and Rasti-Barzoki, 2019; Jamili et al., 2016; Low et al., 2014; Li et al., 2016; Zou et al., 2018), parallel machine configuration (Amorim et al., 2013; Dayarian and Desaulniers, 2019; Kesen and Bektaş, 2019; Belo-Filho et al., 2015; Ullrich, 2013; Guo et al., 2017), production in the job shop (Mohammadi et al., 2020), and production in the flow shop (Yağmur and Kesen, 2020; Wang et al., 2020). Some of these studies have added the classic time window constraints for vehicle routing problems (Amorim et al., 2013; Guo et al., 2017; Mohammadi et al., 2020). The CP-VRP and MoP-VRP solved in this work are most relevant to the variant studied by Kesen and Bektaş (2019), where parallel machine configuration and a limited homogeneous vehicle fleet are considered. We extend this model a little by adding the duration of routes.

The mathematical model and valid inequality for the CP-VRP have been widely studied in the literature. Chang and Lee (2004) used mathematical derivation to prove that the CP-VRP problem, with only one production machine, is an NP-hard problem. Geismar et al. (2008) proved that the CP-VRP problem is an NP problem and gave the lower bound of the solved problem by mathematical proof. Armstrong et al. (2008) proved that the use of the basic properties of the tackled problem can accelerate the branch-and-bound search. Chen and Vairaktarakis (2005) defined CP-VRP with different scenarios and gave mathematical proof of the complexity of some of the defined scenarios.

Exact algorithms for CP-VRP have also been investigated in the literature. Amorim et al. (2013) verified the importance of introducing lot-sizing by solving the mixed integer programming model. Tamannaee and Rasti-Barzoki (2019) dealt with a problem with production due date and used the branch-and-bound algorithm to solve the problem. In this work, the “earliest due date first production” scheduling scheme is proposed and is proven to effectively accelerate the search for the lower bound. Dayarian and Desaulniers (2019) designed an effective branch-and-cut-and-price algorithm and introduced new branching rules to efficiently obtain the lower bound of each branch.

Most of the literature uses heuristic or meta-heuristic algorithms to solve CP-VRP related problems. The most used are the neighbourhood search algorithm (Belo-Filho et al., 2015; Jamili et al., 2016; Wang et al., 2020), genetic algorithm (Tamannaee and Rasti-Barzoki, 2019; Kesen and Bektaş, 2019; Yağmur and Kesen, 2020; Ullrich, 2013; Low et al., 2014; Li et al., 2016; Guo et al., 2017; Zou et al., 2018), and decomposition method (Dayarian and Desaulniers, 2019; Ullrich, 2013; Zou et al., 2018). Among them, Jamili et al. (2016) and Li et al. (2016) **proposed a bi-objective model** to provide a trade-off between transportation costs and service quality. In our study, we apply the Adaptive Large Neighbourhood Search (ALNS) algorithm to solve both the MoP-VRP and CP-VRP. Moreover, we propose some strategies to accelerate the ALNS for the CP-VRP.

Another existing optimization problem groups together the production and routing is the *Production Routing Problem (PRP)* (Adulyasak et al., 2015; Qiu et al., 2018), which is an integrated optimization problem that jointly optimizes production, inventory, and routing decisions. The PRP usually deals with

a multiple-period planning horizon, and in each period the number of products to be made, the number of products to be stored in the depot (i.e., the inventory), and the delivery plan must be decided. Different from the PRP, Both the CP-VRP and the MoP-VRP in this work will not consider overproduction (i.e., no inventory cost), and we only deal with a one-day planning horizon.

2.2. The Mobile Production Technology

To the best of our knowledge, only two works introduce the concept of mobile production technology in the literature; different works define a different kind of mobile production.

Malladi et al. (2020) introduced a mobile production mode in which production is conducted by a movable production unit called a “module”. The number of modules determines the capacity of production, and the target is to optimize the allocation of modules, production, and inventory planning to serve uncertain demand within a certain planning horizon.

In Pasha et al. (2020), the mobile production is named the “factory in box”. First, trucks pick up raw materials from the supplier and semi-produced products from the factory. Then, production takes place at the factory or customer’s location. Although the vehicle carries the “factory” and moves during the delivery process, production still takes place at a fixed point, not en route. The mobility of production in this work is reflected in the flexible location of the factory.

The problem studied by Malladi et al. (2020) and Pasha et al. (2020) differs from the proposed MoP-VRP. Production in these two works takes place at a stationary point, whereas production in the MoP-VRP takes place en route.

3. Problem Description and Mathematical Model

This section will describe the setting of the MoP-VRP and the CP-VRP. Then, the mathematical model for both problems will be given. After that, some properties of the MoP-VRP and CP-VRP, as well as corresponding proof are provided.

3.1. Problem Description

This section describes the setting of the MoP-VRP and the CP-VRP. We consider both problems within the context of urban delivery.

Both the MoP-VRP and CP-VRP are defined on a graph as follows: $G = (V, E)$ with $V = \{0\} \cup C$ as the set of nodes and E as the set of edges defined between connected nodes. Node 0 is the depot and $C = \{1, \dots, n\}$ is the set of customers. There exists a homogeneous set of vehicles $K = \{1, \dots, \kappa\}$, each with capacity Q . We consider a fixed number of vehicles in this work and multiple trips are not allowed. The vehicle will start from the depot and visit a sequence of customers, then go back to the depot no later than a certain time. The vehicle can depart at time 0 and should go back to the depot no later than D . Each customer $i \in C$ has demand d_i and production time p_i . Each customer also has a pre-specified time window

$[a_i, b_i]$. We consider a strict starting time window and a soft ending time window for both problems. The service time for each customer i is denoted as e_i . The distance from i to j is denoted as c_{ij} , and the traveling time from i to j is denoted as t_{ij} . We denote $M = \{1, \dots, m\}$ as the set of machines on each vehicle that produce the goods for the customers. For the CP-VRP, we define $M' = \{1, \dots, m'\}$ as the set of machines in the depot. In this work, we assume that each vehicle has the same number of machines, so $m' = m * \kappa$. In practice, there is a setup/clean up time associated with each job, but this can be included in the production time and is therefore not modelled explicitly.

For the MoP-VRP, a number of machines are installed in each vehicle. Production and delivery start simultaneously at the beginning of the planning horizon. The service start time at each location is the largest value among the arrival time, the production completion time, and the starting time window for that location. With the current technology, there is a need for human intervention every time a new printing job is started. This means that the driver will have to stop the vehicle and spend a minute or two starting the new job, which could be problematic while driving on the highway. This is an issue that is not modelled currently, but is left for future work. For the CP-VRP, the machines are installed in the depot, and each vehicle leaves the depot only when all the products assigned to the vehicle are ready.

The aim for both problems is to determine an optimal integrated production and delivery schedule such that the weighted sum of the total travel distance and the total delay is minimized. [Alternative objective functions could be considered. It could, for example, make sense to also include route duration in the objective function.](#)

Figure 3 shows an example of an MoP-VRP and a CP-VRP solution. In the example, we have two machines in total and two available vehicles. In the CP-VRP, the two machines are placed in the depot; in the MoP-VRP we have one machine on each vehicle. Each vehicle has capacity 16 and the latest time to return the depot is 34. The demand, production time, time windows, and coordinates for each customer are shown in Figure 3. The service time for each customer is 0. The graph and the distance matrix are also shown at the top of the figure. The weight for both the travel cost and delay cost is set as 1.

Clusters A and B in Figure 3 show the optimal solution for the CP-VRP and the MoP-VRP, where production starts at 0 in both problems. In the MoP-VRP, the vehicles leave the depot at 0 and production takes place on the way to the customer. However, in the CP-VRP solution, production takes place at the depot, vehicles have to wait until the production is finished before they can leave the depot. It is found that in the CP-VRP vehicle 1 will leave the depot as soon as the assigned products (1, 2) are complete, while vehicle 2 needs to wait until product 4 is made. The optimal solution for both problems has the same travel distance of 32 ($5 + 6 + 5 + 5 + 6 + 5$), the MoP-VRP just has a delay of 15 ($2 + 6 + 1 + 6$), and the CP-MoP has a delay of 54 ($4 + 15 + 12 + 23$). Therefore, as we set the weight for both terms to 1, the objective value for the MoP-VRP is 47 ($32 + 15$), which is much lower than that for the CP-VRP ($86 = 32 + 54$).

One way to reduce the delay to the customer could be to allow the machines at the depot to start

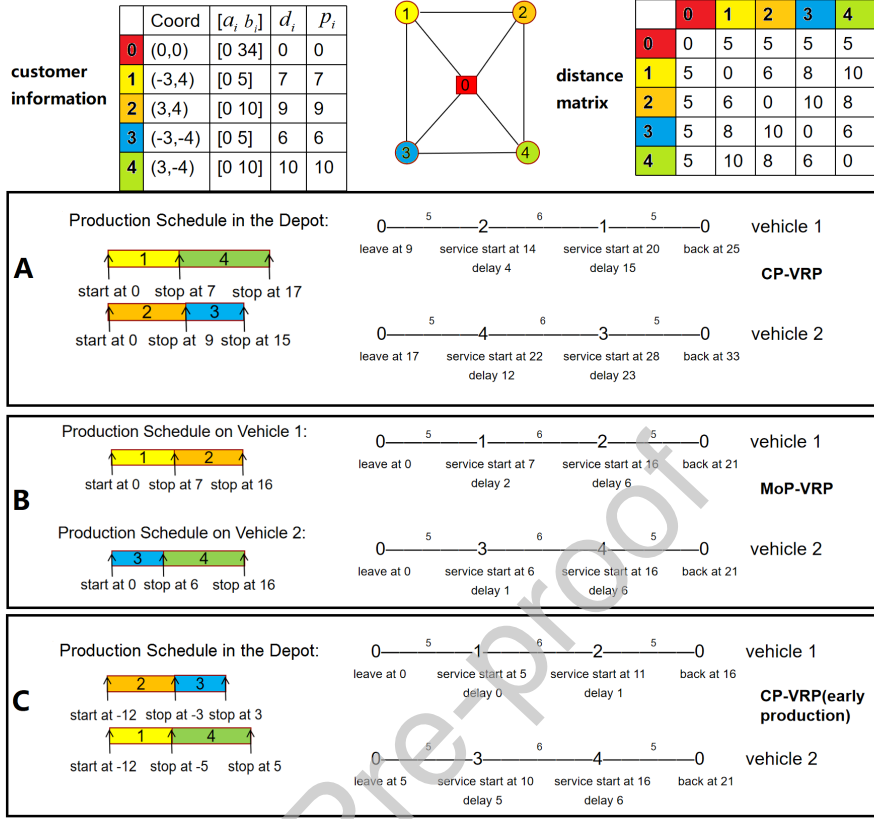


Figure 3: An Example to compare the MoP-VRP and the CP-VRP (Coord: the coordinate; $[a_i, b_i]$: the time window; d_i : the demand; p_i : the production time)

production at an earlier time, say H time units. Therefore, we introduce early production to the CP-VRP to help reduce delay costs. Note that the vehicle should depart the depot no earlier than 0, even if all the assigned products are finished before 0.

The cluster C in Figure 3 shows the optimal solution for the CP-VRP, with early production considered, in which production begins 12 time units before 0 (i.e., $H = 12$). It is found that the total delay can be largely reduced to 12 ($0 + 1 + 5 + 6$) and is lower than that in the optimal solution for the MoP-VRP. However, the drawback of early production is that: the entire operation now spans a longer time period. We return to the consequences of this drawback in Section 5.

3.2. Mathematical Model

In this section, we present mixed integer programming (MIP) models for both the MoP-VRP and the CP-VRP.

In addition to the parameters introduced in Section 3.1, we define $\delta^+(i)$ and $\delta^-(i)$ as the set of nodes that can be reached from $i \in V$ and the set of nodes that can reach $i \in V$, respectively. We define a dummy start node o and dummy end node d at the depot 0.

The binary variable x_{ij}^k equals 1 if vehicle $k \in K$ travels from i to j , and 0 otherwise. Variable y_i denotes the delay time for customer $i \in C$. The non-negative variable s_i^k is the time vehicle $k \in K$ which starts delivery at node $i \in V$. In the MoP-VRP model, the non-negative variable v_{il}^k is the production starting time for node $i \in V$ on machine $l \in M$ in vehicle k (if i is served by k) and the binary variable w_{ijl}^k equals 1 if customer j 's order is produced immediately after customer i 's on machine $l \in M$ in vehicle $k \in K$.

The mathematical model for the MoP-VRP is set as:

$$\min \quad \omega_1 \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ij}^k + \omega_2 \sum_{i \in C} y_i \quad (1)$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k = 1 \quad \forall i \in C \quad (2)$$

$$\sum_{j \in \delta^+(o)} x_{oj}^k = \sum_{j \in \delta^-(d)} x_{jd}^k = 1 \quad \forall k \in K \quad (3)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^k = \sum_{j \in \delta^-(i)} x_{ji}^k \quad \forall i \in C, k \in K \quad (4)$$

$$\sum_{i \in C} \sum_{j \in \delta^+(i)} d_i x_{ij}^k \leq Q \quad \forall k \in K \quad (5)$$

$$\sum_{l \in M} \sum_{j \in \delta^+(i)} w_{ijl}^k = \sum_{j \in \delta^+(i)} x_{ij}^k \quad \forall i \in C, k \in K \quad (6)$$

$$\sum_{j \in \delta^+(0)} w_{0jl}^k = 1 \quad \forall l \in M, k \in K \quad (7)$$

$$\sum_{j \in \delta^+(i)} w_{ijl}^k = \sum_{j \in \delta^-(i)} w_{jil}^k \quad \forall i \in C, l \in M, k \in K \quad (8)$$

$$v_{ijl}^k \geq v_{il}^k + p_i - \tilde{M}(1 - w_{ijl}^k) \quad \forall (i, j) \in E, l \in M, k \in K \quad (9)$$

$$s_i^k \geq v_{il}^k + p_i \quad \forall i \in C, l \in M, k \in K \quad (10)$$

$$s_j^k \geq s_i^k + t_{ij} + e_i - \tilde{M}(1 - x_{ij}^k) \quad \forall (i, j) \in E, k \in K \quad (11)$$

$$s_i^k \geq a_i \quad \forall i \in C, k \in K \quad (12)$$

$$s_d^k \leq D \quad \forall k \in K \quad (13)$$

$$y_i \geq s_i^k - b_i \quad \forall i \in C, k \in K \quad (14)$$

$$x_{ij}^k, w_{ijl}^k \in \{0, 1\} \quad \forall (i, j) \in E, l \in M, k \in K \quad (15)$$

$$y_i \geq 0 \quad \forall i \in C \quad (16)$$

$$v_{il}^k, s_i^k \geq 0 \quad \forall i \in V, l \in M, k \in K \quad (17)$$

The objective (1) is to minimize the weighted sum of travel and delay cost, where travel cost is based on the total travel length and the delay cost is based on the delayed time. In this work, we set the weight for both terms as 1. Constraints (2) ensure that each customer is visited exactly once by one vehicle. Constraints (3) guarantee that each vehicle starts and ends at its starting and ending depots. Constraints

(4) are the flow conservation constraints. Constraints (5) ensure that the capacity is not exceeded for each vehicle. Constraints (6) make sure that if a customer i is visited by vehicle k , then this order should be produced on one of the machines on k . Constraints (7)-(8) are the production flow conservation constraints for each machine. Constraints (9) determine the start production time for each order. If order j is produced immediately after order i by machine l in vehicle k (i.e., $x_{ij}^k = 1$), then the start production time of order j should be greater than or equal to the start production time of order i plus the production time of i , which is $v_{jl}^k \geq v_{il}^k + p_i$. Constraints (10) and (11) ensure that the delivery can start only if the vehicle has travelled to the customer and the production of that order is finished. We suggest that \tilde{M} should be a value no smaller than $D + \max\{\max_{i \in C}\{p_i\}, \max_{(i,j) \in E}\{t_{ij} + e_i\}\}$. Constraints (13) set the maximum travel duration for each vehicle. Constraints (14) determine the delay time for each customer. Constraints (15)-(17) are the non-negativity constraints.

In the CP-VRP, w_{ijl}^k and v_{il}^k changes to w_{ijl} and v_{il} because production does not take place on the vehicle. v_{il} can have a negative value because early production is allowed in this problem.

$$\min \quad \omega_1 \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ij}^k + \omega_2 \sum_{i \in C} y_i \quad (18)$$

$$\text{s.t.} \quad (2) - (5)$$

$$\sum_{l \in M'} \sum_{j \in C} w_{ijl} = 1 \quad \forall i \in C \quad (19)$$

$$\sum_{j \in \delta^+(0)} w_{0jl} = 1 \quad \forall l \in M' \quad (20)$$

$$\sum_{j \in \delta^+(i)} w_{ijl} = \sum_{j \in \delta^-(i)} w_{jil} \quad \forall i \in C, l \in M' \quad (21)$$

$$v_{jl} \geq v_{il} + p_i - \tilde{M}(1 - w_{ijl}) \quad \forall (i,j) \in E, l \in M' \quad (22)$$

$$s_o^k \geq v_{il} + p_i - \tilde{M}(1 - \sum_{j \in \delta^+(i)} x_{ij}^k) \quad \forall i \in C, l \in M', k \in K \quad (23)$$

$$v_{il} \geq -H \quad \forall i \in C, l \in M' \quad (24)$$

$$(11) - (17)$$

In the CP-VRP, constraints (20)-(22) should be interpreted like (7)-(9) in the MoP-VRP. Constraints (19) ensure that each customer's demand should be produced by some machine. Constraints (23) guarantee that each vehicle can only start when all the assigned products are finished. Constraints (24) set the early production.

3.3. Properties of the MoP-VRP and CP-VRP

It is easy to see that MoP-VRP is an NP-Hard problem: we consider an instance for the MoP-VRP, where the production time for each customer is 0. It is sufficient to show that to solve this special instance of an MoP-VRP is equivalent to solving a VRP.

Next we turn to a simple property of the MoP-VRP that allows us to discard a part of the feasible solution space when designing heuristics for the problem. To explain this property, it is convenient to start with a small example for an MoP-VRP with two machines per vehicle. Consider the vehicle route $[1\ 2\ 3\ 4\ 5\ 6]$, where each element is the ID of a customer. A possible machine schedule for this route is Machine 1: $[1\ 2\ 5]$ and Machine 2: $[4\ 3\ 6]$, where the elements of the vectors indicate which customers the machine is producing for. We propose that the schedule for machine 1 is *in line with* the route since the machine completes jobs in the same order as deliveries are being made to the customers. Machine 2's schedule, meanwhile, is not *in line with* the route because the jobs for customer 4 and 3 are produced in the opposite order of their appearance in the route.

It is obvious that changing the order of customers 3 and 4 in the schedule for machine 2 can never make the solution worse: when product 4 is produced before 3, neither of the two customers can be served before the production of both orders is finished. When product 3 is produced before 4, customer 3 can be served once product 3 has been finished while customer 4 cannot be served before both products are done; this implies that with production schedule $[3\ 4\ 6]$, the vehicle will serve customer 3 and 4 (or the customers thereafter) at the same time or earlier than production schedule $[4\ 3\ 6]$. This is formalized into the following proposition:

Proposition 1. *There exists an optimal solution for the MoP-VRP, such that the schedule of each machine is in line with the route of the truck carrying the machine.*

Proof Suppose that s^* is the unique optimal solution, where customer 2 is served after customer 1, but his product is produced before customer 1 on the same machine, and another solution \hat{s} where the sequence of production and delivery are the same for two customers. We define $f(s)$ as the objective value for solution s . As the route is fixed, the service for customer 1 in \hat{s} will start no later than that in s^* , and the service starting time for customer 2 is the same in both solutions. Therefore, $f(\hat{s}) \leq f(s^*)$; thus, s^* is not the unique optimal solution. This is contradictory to the assumption. \square

The following proposition states that the CP-VRP can be decomposed into a production planning problem and a routing problem when the CP-VRP only involves a single vehicle

Proposition 2. *When there is one vehicle and m machines in the CP-VRP, the optimal solution can be found by first minimizing the production completion time and then solving a TSP.*

Proof As multiple trips are not allowed in this work and the vehicle cannot leave before the completion of all the assigned items, and since the production order does not influence the route of vehicle, the problem can be split in two as indicated in this proposition. \square

We next investigate the structure of the schedule for a single machine in the CP-VRP. The goal of this analysis is to allow us to rule out certain machine schedules from the set of schedules to consider. This will allow us to design faster heuristics in section 4. We note that the machine schedules influence the overall solution through the times when products are available. All products to be delivered by a single route must

be available before the vehicle can depart; poor machine scheduling will therefore lead to unnecessary delay costs.

The key insight is that if the schedule for a particular machine contains two or more products that are to be delivered on the same route, then these products can be produced in direct succession. For example, consider a solution where the production schedule for a machine in the CP-VRP is $[1^{(2)} \ 2^{(3)} \ 3^{(4)} \ 4^{(3)} \ 5^{(4)} \ 6^{(2)}]$, where each element in the vector has the form $i^{(r)}$, in which i is the ID of the customer who will receive the product and r is the route that this customer is assigned to. Starting from the back of the schedule, we can gather the two products that are to be delivered by route 2. This gives the schedule $[2^{(3)} \ 3^{(4)} \ 4^{(3)} \ 5^{(4)} \ 1^{(2)} \ 6^{(2)}]$, which is at least as good as the starting schedule because the products for route 2 are finished at the same time as in the original schedule while the products for route 3 and 4 are finished earlier. We now continue backwards through the schedule to the next “scattered” products which are the products to be delivered on route 4. We gather these to obtain the schedule $[2^{(3)} \ 4^{(3)} \ 3^{(4)} \ 5^{(4)} \ 1^{(2)} \ 6^{(2)}]$. In this schedule, the products for route 2 and 4 are finished at the same time as in the previous schedule while the products to be delivered by route 3 are finished earlier. We now have a schedule where products that are to be delivered by the same route are in direct succession.

Lemma 1 below formalizes the process of gathering two products destined for the same route. In this Lemma, we let $r(i)$ denote the route that serves customer i . The proof of the Lemma follows directly from the arguments above.

Lemma 1. *Consider a production schedule $[i_1 \ i_2 \ \dots \ i_\theta]$, where θ is the length of the schedule and i_j indicates the customer ID that the machine is producing for. If there exists α, β, γ such that $1 \leq \alpha < \beta < \gamma \leq \theta$ and such that $r(i_\alpha) = r(i_\gamma)$ and $r(i_\alpha) \neq r(i_\beta)$, then job i_α can be moved to position $\gamma - 1$ and the jobs that previously occupied position $\alpha + 1$ to $\gamma - 1$ will now occupy position α to $\gamma - 2$. This change will not delay departure for any routes and may reduce the departure time for some routes.*

From Lemma 1 we derive the following proposition:

Proposition 3. *There exists an optimal solution for the CP-VRP, where the production schedule for each machine finishes all jobs delivered by one route before switching to producing jobs that are to be delivered by a different route.*

Proof Consider any optimal solution to the CP-VRP instance. If the optimal solution does not satisfy the conditions of proposition 3, then we process each machine schedule that does not satisfy the conditions by repeatedly applying Lemma 1. We go through the machine schedule from the back and gather jobs destined for the same route by repeatedly applying Lemma 1. After having applied Lemma 1 η times, we are sure that the η last jobs are grouped according to their delivery route and we therefore at most need to apply Lemma 1 as many times as there are jobs in the schedule. Applying Lemma 1 does not make the cost of the solution any worse so we will therefore eventually end up with an optimal solution in which jobs are grouped according to their delivery route for each schedule. This shows that an optimal solution with the desired properties exists. \square

The following proposition compares the objective value of the MoP-VRP and the CP-VRP when there is exactly one vehicle available.

Proposition 4. *In the case when there is one vehicle and m machines available, the objective value of the optimal MoP-VRP solution is always better than or equal to the objective value of the optimal solution to the corresponding CP-VRP instance, assuming that the early production parameter H is set to 0.*

Proof If the route and production schedule from an optimal CP-VRP solution is copied to an MoP-VRP solution, the production completion time for the customers is the same for the two problems. Since the vehicle in the CP-VRP will start from the depot after the completion of all the products, the service starting time for each customer in the MoP-VRP cannot be later than that in the CP-VRP. Thus, with the same solution, the MoP-VRP will never obtain a higher total cost than the CP-VRP because the travel cost is the same and the delay cost is at least as good as that in the CP-VRP. \square

When the number of vehicles is greater than 1, the optimal solution to the CP-VRP can be better than the optimal solution to the MoP-VRP, even when the early production parameter H is set to 0. To illustrate, consider 4 customers and a depot distributed on a line as shown on Figure 4. Numbers on top of the line indicates coordinates; letters below indicate customers. The depot is located at coordinate 0. Assume that travel times and distances are equal to the Euclidean distances, and that service times are 0. Further, assume that the production time for customers A and B is 20, while it is 1 for customers C and D, and that the ending time windows are 31 for customers A and D and 30 for customers B and C.

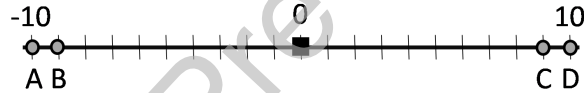


Figure 4: Example where the CP-VRP solution is better than the MoP-VRP solution.

Assuming that we have two vehicles available, each equipped with 1 machine in the MoP-VRP, then an optimal solution is to place A and B on route 1 and C and D on route 2. The total traveled distance is 40, and there will be a delay of 9 time units for customer A (total objective: 49). For the CP-VRP we have two machines at the depot. These machines finish producing all products at time 21. Reusing the two routes from the MoP-VRP leads to zero delays in the CP-VRP case, so the total objective is 40.

4. Adaptive Large Neighbourhood Search

We developed an Adaptive Large Neighbourhood Search (ALNS) heuristic to solve the large-size problems efficiently. The ALNS (Ropke and Pisinger, 2006) searches in large neighborhoods defined implicitly by a destroy method and a repair method. A destroy method disrupts part of the current solution while a repair method rebuilds the destroyed solution. In the ALNS, a set of destroy and repair methods are developed for selection. Each is assigned a weight that is adaptively adjusted based on their performance in previous iterations. In each iteration, a destroy and repair method is selected and applied to the current solution. The resultant solution is accepted based on a user-defined acceptance criterion, and the heuristic stops when the stopping criterion is met.

The pseudo-code of our ALNS is presented in Alg.1. In line 2, an initial solution is constructed by a greedy insertion heuristic. In lines 6-8, we choose whether to use the noise or not based on the previous performance;

after that we choose the destroy and repair operator. In the solution searching phase, we apply the threshold acceptance method proposed by Dueck and Scheuer (1990) which has been shown as a good criterion for the ALNS (Santini et al.,2018). We always accept the better solution but also accept a deteriorated solution if it is within a given threshold T of the best solution found so far. The parameter T is initialized in the beginning of the search and decreased linearly to 0 at the end of the search. We will tune the initial threshold $T_{initial}$ in our final test to ensure the best performance of the algorithm. The ALNS stops when a certain number of predefined iterations, N_{max} , is reached.

To enable the comparison between MoP-VRP and CP-VRP, we also apply ALNS to the CP-VRP. The pseudo-code of Alg.1 also applies to the CP-VRP.

Algorithm 1 Adaptive Large Neighbourhood Search

```

1: Function: ALNS(instance)
2:  $s = \text{greedyinsertion}(\textit{instance})$  ;
3:  $s^* = s$ ;
4: repeat
5:    $s' = s$ ;
6:   decide whether to use the noise or not;
7:   select a destroy operator and update  $s'$ ;
8:   select a repair operator and update  $s'$ ;
9:   if  $\frac{f(s') - f(s^*)}{f(s^*)} < T$  then
10:      $s = s'$ ;
11:     if  $f(s') < f(s^*)$  then
12:        $s^* = s'$ ;
13:     end if
14:   end if
15:   if  $N_{max} \% 100 = 0$  then
16:     Update weights for the destroy, repair operators and noise;
17:   end if
18:    $T = T - (T_{initial}/N_{max})$ ;
19: until  $N_{max}$  is met

```

4.1. The ALNS for the MoP-VRP

This section will describe the strategy to construct the solution for the MoP-VRP and the removal and insertion operators used in the ALNS.

4.1.1. Construction of Solution

We used a parallel greedy insertion heuristic to construct the initial solution. The mechanism is to find the best position for the customer among all the routes in each iteration (Laporte and Semet, 2002). We start with a set of empty routes. In each iteration, we evaluate the cost of inserting each unassigned customer to each possible route position as well as each possible production position in the assigned route. We then identify the minimum insertion cost for each unassigned customer, select the one with the smallest minimum insertion cost, and insert it to the best position. The procedure is repeated until all the customers are included in the routes.

Different from the traditional VRP, in the MoP-VRP, not only does the routing position need to be decided for each customer but also the machine production schedule for the corresponding customer demand. For each inserted routing position, one needs to examine all the possible production schedules for each machine. However, as proven in Proposition 1, there exists an optimal solution such that the production sequence on the same machine is consistent with the route sequence. Following Proposition 1, we only consider the production whose sequence is *in line with* that of route in our implementation. This helps to reduce computational time significantly. The pseudo-code for the insertion strategy is shown in Alg. 2.

Algorithm 2 The insertion strategy for the MoP-VRP

```

1: for each non-inserted Customer  $i$  do
2:   for each Route  $r$  do
3:     for each position  $\rho \in$  Route  $r$  do
4:       calculate travel increment on position  $\rho$ 
5:       for each Machine  $l$  on Route  $r$  do
6:         Find the last customer whose delivery position in route  $r$  is before  $\rho$ 
7:         Compute the delay increment in route  $r$  for the customers after position  $\rho$ 
8:          $(i^*, \rho^*, l^*, r^*) = (i, \rho, l, r)$  if the total cost increment is the least
9:       end for
10:    end for
11:  end for
12: end for
13: Return  $(i^*, \rho^*, l^*, r^*)$ 

```

We apply the worst-case analysis to calculate the computational complexity for Alg. 2. It is found that the computational complexity is $O(n^2 m \frac{n}{\kappa}) = O(\frac{mn^3}{\kappa})$, where n is the number of customers, m is the number of machines in each vehicle, and κ is the number of vehicles in the instance. The n^2 comes from the truth that the loops in lines 1-3 result in n^2 repetitions, the m comes from line 5, which represents the number of machines in each vehicle, and $\frac{n}{\kappa}$ is the maximum number of repetitions resultant from line 7. Under the assumption that $\kappa = O(n)$, the computational complexity can be simplified to $O(mn^2)$.

4.1.2. Destroy and Repair Operators

In this work, we propose six destroy operators and four repair operators. The MoP-VRP and CP-VRP share the same destroy and repair operators introduced in this section.

Common to all destroy methods is that we first randomly pick a value $\Phi \in [[\lambda_1 n], [\lambda_2 n]]$ as the number of customers to remove. For the interval $[[\lambda_1 n], [\lambda_2 n]]$, n is the total number of customers and λ_1 and λ_2 are the minimum and maximum ratios, respectively. We will tune the lower bound and upper bound (i.e., λ_1 and λ_2) in our final test.

The random and worst removals use the same strategy as that introduced by Ropke and Pisinger (2006). The **random removal** picks Φ customers at random to remove. The **worst removal** removes Φ customers with the highest cost savings ξ_i . Inspired by the worst removal, **Worst-Delay Removal** removes Φ customers with the highest delay cost savings, and **Worst-Dist Removal** removes Φ customers with the highest travel distance savings. Inspired by the existing Shaw removal (Shaw,1997; Shaw,1998), we also propose two similar destroy operators, **Geo Removal** and **Demand Removal**, which first remove a random seed customer and then $\Phi - 1$ customers that are close to the seed customer in terms of distance and demand, respectively. Details of the destroy operators are presented in the Appendix.

For the repair operators, the proposed ALNS applies the **regret-k method**. The idea is to choose the insertion that we will regret most if we do not perform it. Let r_{ih} indicate the route where customer i has the h th lowest insertion cost and $\Delta_{i,r_{ih}}$ be the inserting cost of customer i on route r_{ih} . The regret value g_i^* equals to $\sum_{h=1}^k \{\Delta_{i,r_{ih}} - \Delta_{i,r_{i1}}\}$. The regret-k method will choose to insert customer i that maximizes: $\arg \max\{g_i^*\}$ and will put it into the position with the lowest cost increment. Ties are broken by selecting the request with the best insertion cost. In this work, we implement the regret-1/-2/-3/-4 method, regret-1 method is the greedy insertion introduced in section 5.1 due to the tie-breaking rule.

We also introduce noise to give randomness to the objective value every time we calculate the insertion cost of a request. We calculate the noise by applying the same strategy used by Ropke and Pisinger (2006).

Each removal and insertion heuristics is given a weight that can be adaptive adjusted to measure how well each operator performs on each instance. We also give the automatically adjusted weight to the noise so that the algorithm itself can decide whether to apply the noise or not in each iteration. A higher weight indicates a more effective operator, and an operator with a higher weight is more likely to be chosen in each iteration. The mechanism and parameters to adjust the weight are exactly the same as Ropke and Pisinger (2006).

4.2. The ALNS for the CP-VRP

The ALNS for the CP-VRP shares the same framework as the ALNS for the MoP-VRP, and the destroy/repair operators for the MoP-VRP also apply to the CP-VRP. We introduce some new strategies to accelerate the ALNS for the CP-VRP.

4.2.1. Construction of Solution

We also apply a parallel **greedy** insertion heuristic to construct the initial solution for the CP-VRP. We start with a set of empty routes. In each iteration, we evaluate the cost of inserting each unassigned customer to all the possible route positions and fix the route position with the smallest cost increment. We then identify the smallest total cost increment by trying all the possible positions in the production sequences at the depot. We select the customer with the smallest minimum insertion cost and add it to the route and machine schedule in the solution. The procedure is repeated until all the customers are included in the solution.

4.2.2. The Potential Production Sequence for the CP-VRP

When we identify the best production position for a customer whose routing position is already determined, we should examine all the possible positions in the production sequence for each machine to ensure that the total cost increment is the global minimum. However, testing all the possible positions can make the computation quite time-consuming. From Proposition 3, we know that grouping production for customers who are assigned to the same vehicle to a shared machine can lead to an optimal solution. We can use this property to eliminate the cost computation of a large number of unnecessary production positions, thereby reducing the computational time significantly.

Then, we provide an example to show how we narrow the search for potential production positions. Suppose that a new customer has already been selected to be added to route r , and we now need to consider when the corresponding production should take place for each machine. Assume the machine currently produces products for five customers. Let's use a vector to indicate the route index of each customer in the corresponding production sequence, **in the same way as we did in Section 3.** For example, $[1^{(4)} \ 2^{(4)} \ 3^{(2)} \ 4^{(2)} \ 5^{(1)}]$ means that the first two customers (**i.e., customer 1 and 2**) for whom the machine produces are delivered to by vehicle 4, the next two customers (**i.e., customer 3 and 4**) by vehicle 2, and the last one (**i.e., customer 5**) by vehicle 1. Then, we may meet two cases:

Case 1: If the new customer is served by a vehicle that is not in the vector, e.g., $r = 3$, it would make sense to just try the positions marked with *: $[* \ 1^{(4)} \ 2^{(4)} \ * \ 3^{(2)} \ 4^{(2)} \ * \ 5^{(1)} \ *]$. There is no need to try positions like: $A = [1^{(4)} \ * \ 2^{(4)} \ 3^{(2)} \ 4^{(2)} \ 5^{(1)}]$, as they can never lead to a better solution than this position: $B = [* \ 1^{(4)} \ 2^{(4)} \ 3^{(2)} \ 4^{(2)} \ 5^{(1)}]$.

The reason is that for both A and B , the ready time for routes 1, 2, 4 is the same. A , however, will lead to a higher ready time for route 3.

Case 2: If the new customer (**e.g., customer 6**) is served by one of the vehicles that is already in the vector, e.g., $r = 4$, we could try these potential positions (marked with both * and \circ): $[* \ 1^{(4)} \ 2^{(4)} \ 3^{(2)} \ 4^{(2)} \ \circ \ 5^{(1)} \ \circ]$. When we try positions marked with \circ , such as: $[1^{(4)} \ 2^{(4)} \ 3^{(2)} \ 4^{(2)} \ \circ \ 5^{(1)}]$, we should also compute the cost of $V = [3^{(2)} \ 4^{(2)} \ 1^{(4)} \ 2^{(4)} \ 6^{(4)} \ 5^{(1)}]$ because this schedule is at least as good as $W = [1^{(4)} \ 2^{(4)} \ 3^{(2)} \ 4^{(2)} \ 6^{(4)} \ 5^{(1)}]$.

This is because for both V and W , the ready time for route 4 is the same but V can lead to an earlier departure time for route 2, which is cost saving.

4.2.3. The Piece-wise Linear Function for the CP-VRP

An insertion or removal of one product in the production schedule may affect the departure time for several vehicles. If the departure time is changed, the delay cost for the route may change. Further, it is time-consuming to go through all the positions in the affected route to calculate the change in the delay cost.

When the route is fixed, we find that the total delay of the route can be expressed as a non-decreasing piece-wise linear function ($Delay(\psi)$) of the route departure time, where input ψ is the departure time and $Delay(\psi)$ is the corresponding delay cost. The function has a typical shape, as given in Figure 6. With the help of $Delay(\psi)$, we can store the information of how the delay cost changes with the departure time. Then, each time we obtain a new departure time, we can easily obtain the new delay cost and thus the computational time can be largely reduced.

To introduce how the $Delay(\psi)$ helps save computational time, we present an example of a route in Figure 5, and its corresponding piece-wise linear function in Figure 6. In Figure 5, the two boxes illustrate the start and end of the route, and the circles illustrate the customers visited. The numbers in square brackets above the customers are the time windows. The numbers under the edges are travel times, and the numbers in parentheses are service times. This route would not be feasible in an ordinary VRPTW setting because it would be impossible to visit customer 2 after having visited customer 1 with the given time windows. However, in both the CP-VRP and MoP-VRP, such a solution is feasible, because delay is allowed. In Figure 6, we see that the slope is different in each interval; this represents the different number of customers whose delay costs change with the departure time. The numbers shown in the x-axis, 15 25 65, are the thresholds that mark the change of increment rate. Here, 65 is the maximum departure time to exceed that will cause an infeasible solution. Delay can be calculated as follows: if the departure time becomes 8, $Delay(8) = 15$, which is equal to $Delay(0)$; if the departure time becomes 20, $Delay(20) = 20$.

When analysing the computational complexity of the piece-wise linear function, it takes $O(n_r)$ to construct the piece-wise linear function. There is no need to reconstruct the function if the route does not change, and it takes $O(\log(n_r))$ to compute the delay if the departure time does change (n_r is the number of customers in the route r). Without the piece-wise linear cost function, it would take $O(n_r)$ to compute the delay once the departure time has changed. Therefore, the proposed function can help accelerate the algorithm.

4.2.4. Comparison of different insertion strategy

The most accurate way to calculate the cost increment when we insert a customer is shown in Alg. 3, by which we need to try all the positions in each route. For each delivery position, we need to try all the potential production sequence to ensure that the final customer insertion causes the smallest increment cost. We call this strategy the integrated strategy.

In line 5 in Alg. 3, we need to update the piece-wise linear function for route r because the route has

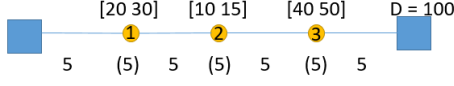


Figure 5: Example to explain the piece-wise linear function

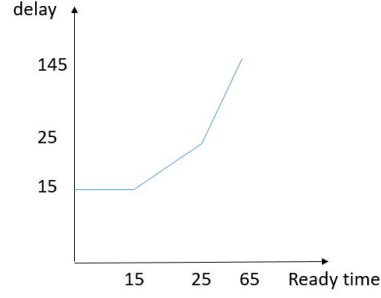


Figure 6: Piece-wise Linear Function

Algorithm 3 The integrated strategy for the CP-VRP

```

1: for each non-inserted customer  $i$  do
2:   for each Route  $r$  do
3:     for each position  $\rho$  in route  $r$  do
4:       calculate travel increment at position  $\rho$ 
5:       Update the Piece-wise Linear Function for route  $r$  temporarily
6:       for each machine  $l$  do
7:         for each position  $\nu$  in the schedule of  $l$  do
8:           Compute the delay increment for the whole solution
9:            $(i^*, r^*, \rho^*, l^*, \nu^*) = (i, r, \rho, l, \nu)$  if the total cost increment is the least
10:        end for
11:       end for
12:     end for
13:   end for
14: end for
15: Return  $(i^*, r^*, \rho^*, l^*, \nu^*)$ 

```

changed and the pre-stored cost function cannot be used anymore. We update the function temporarily, and it will turn back to the original form when we finish the computation for position ρ in route r . We also apply the worst-case analysis to estimate the computational complexity of Alg. 3. It is found that the computational complexity for Alg. 3 is $O(n^2(\frac{n}{\kappa} + m'\kappa^2\log(\frac{n}{\kappa} + 1))) = O(\frac{n^3}{\kappa} + n^2m'\kappa^2\log(\frac{n}{\kappa} + 1))$, where n is the number of customers, m' is the total number of machines in the depot, and κ is the number of vehicles in the instance. In the computational complexity, n^2 , $\frac{n}{\kappa}$, and m' come from lines 1-3, line 5, and line 6, respectively. The $\frac{n}{\kappa}$ comes from the analysis that to have the same number of customers in each route will cause the highest computational complexity. The $\kappa^2\log(\frac{n}{\kappa} + 1)$ is composed of two parts: κ and $\kappa\log(\frac{n}{\kappa} + 1)$, where κ comes from line 7 because there will be at most κ positions to consider due to the analysis from section 4.2.2, and $\kappa\log(\frac{n}{\kappa} + 1)$ comes from line 8, where the piece-wise linear cost function is applied. Under

the assumption that $\kappa = O(n)$, the computational complexity can be simplified to $O(n^2 + m'n^4)$. We can omit the $O(n^2)$ part, so the final computational complexity is $O(m'n^4)$.

To save the computational time, we apply a decomposition strategy for the CP-VRP. The pseudo-code for the proposed strategy is shown in Alg.4.

Algorithm 4 The decomposition strategy

```

1: for each non-inserted Customer  $i$  do
2:   for each Route  $r$  do
3:     Find insertion of Customer  $i$  in route  $r$  (assume that route ready time is not changed by insertion)
4:     Save insertion if total cost increment is the best
5:   end for
6:   Update the Piece-wise Linear Function for the best route
7:   for each Machine  $l$  do
8:     for each position  $\nu$  in the schedule of Machine  $l$  do
9:       Compute the delay increment for the whole solution
10:      Save the insertion if delay increment is the best
11:    end for
12:  end for
13:   $(i^*, r^*, \rho^*, l^*, \nu^*) = (i, r, \rho, l, \nu)$  if total cost increment is the best
14: end for
15: Return  $(i^*, r^*, \rho^*, l^*, \nu^*)$ 

```

It can be seen from Alg. 4 that we fix the delivery sequence at first without considering the change of ready time. We then find the best production schedule that minimizes the cost increment for all vehicles. Computational complexity is largely reduced because we separate the loop for the routes and the machines. By applying the worst-case analysis, the computational complexity for Alg. 4 is $O(\frac{n^3}{\kappa} + nm'\kappa^2 \log(\frac{n}{\kappa} + 1))$, where $\frac{n^3}{\kappa}$ comes from lines 1-5 and $nm'\kappa^2 \log(\frac{n}{\kappa} + 1)$ comes from lines 7-12. Under the assumption that $\kappa = O(n)$, the computational complexity can be simplified to $O(n^2 + mn^3)$. We can omit the $O(n^2)$ part, so the final computational complexity is $O(m'n^3)$. By applying this strategy, we can reduce the computational complexity from $O(m'n^4)$ to $O(m'n^3)$.

We need to redefine the approach to find k best routes in regret- k insertion because it is different from that in the integrated method. As shown in Alg. 5, for each customer, we choose the k routes with the lowest travel increment first. Then, we update the cost increment for the picked k routes by searching for the best production schedule. After that, we sort the k routes in ascending order of the updated cost increment.

Algorithm 5 The best k routes for each customer in the decomposition strategy

```

1: for each Route  $r$  do
2:   Find insertion of  $i$  in route  $r$  (assume that route ready time is not changed by insertion)
3:   Save insertion if travel increment is best
4: end for
5: Find the  $k$  routes with lowest travel increment
6: for each route  $r \in$  the  $k$  Best Routes do
7:   for each Machine  $l$  do
8:     for each position in the schedule of Machine  $l$  do
9:       Compute the delay increment for the whole solution
10:      Save the insertion if delay increment is the best
11:     end for
12:   end for
13:   Save the cost increment for route  $r'$ 
14: end for
15: Sort the chosen  $k$  routes in ascending order of cost increment

```

5. Computational Results

To evaluate the performance of the proposed mathematical model and ALNS, a series of experiments have been conducted on two different instance sets. The codes have been written in a workstation by using C++ with Windows 10, Intel Core i9-7940X, 3.10GHz, 32 GB RAM. CPLEX 12.8 is used to solve the MIP models.

5.1. Instances

We generate two sets of instances. One set is based on the well-known VRPTW benchmark Solomon Instance (Solomon, 1987) and the Gehring and Homberger's extended VRPTW Instance (Gehring and Homberger, 1999), where we use the same vehicle capacity, customer locations, duration, time windows, service times, and demands as in the benchmark instances. The other set is constructed to mimic a real-life scenario where mobile production is employed.

We create different scenarios by changing the production time and number of machines per vehicle. The number of machines per vehicle is either 1, 2, or 4. The production time p_i for customer i is determined by the formula $p_i = \mu d_i$, where μ is the production time for each demand unit (we name it production time coefficient) and is set as either 1, 3, or 5, and d_i is the demand for customer i .

To set the maximum number of vehicles in each instance, we run a greedy sequential insertion algorithm that inserts all customers in a solution. The number of routes this algorithm terminates with is the maximum number of vehicles of the instance.

For the realistic instances, we consider a delivery scenario in Copenhagen, Denmark as the setting. We generate data using the spare-parts-for-repairs case, as described in the introduction.



Figure 7: Real Life Instance

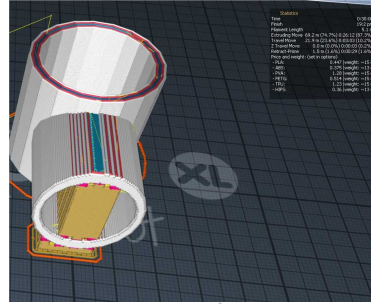


Figure 8: Production Example

As shown in Figure 7, we randomly generate 100 coordinates in a 20 by 30 km rectangle in the greater Copenhagen area. One of them is chosen as the depot and the rest as customers. The distances between the positions are generated by using Google Maps API, and the travel times are calculated using the distances divided by an average speed of 50 km/h. We assume that the drivers, working hours are from 11 am to 9 pm (i.e. 600 minutes) per day.

We also generate 25 and 50 customer instances, as the company may not receive many orders at the beginning stage of this new business mode due to immature technology or the small market. For the 25 and 50 customer scenarios, we randomly pick 25 and 50 customers, respectively, from the 99 customers. The service time for each customer is a random number within 1-5 minutes.

We create six scenarios in total, each differing in terms of the type of production time and time windows. We have three types of production time: small production time (abbreviated S), which ranges from 20 to 30 minutes; medium production time (30-40 mins, abbreviated M); and high production time (30-60 mins, abbreviated H). The three types of production times are estimated by supplying different 3D models of various spare parts to the Craftbot slicing program (Craftware, see Figure 8). The program allows us to compute the estimated printing time for any given 3D model.

We create two types of time windows: wide time windows (abbreviated W) and tight time windows (abbreviated T). The length of the time window is between 30 and 60 minutes for the wide time window, and is within 30 minutes for the tight time window.

Based on the above setting, we named the six scenarios as follows: short production time with wide time window (S-W), medium production time with wide time window (M-W), high production time with wide time window (H-W), short production time with tight time window (S-T), medium production time with tight time window (M-T) and high production time with tight time window (H-T). For each scenario, we generate five random instances with random production times and time windows.

The instances are available at <https://zenodo.org/record/4892330#.YLczY4Xitdg>.

During the implementation, we define the early production time as $H = \frac{\epsilon P}{m\kappa}$, where $P = \sum_{i=1}^n p_i$ is the total production time and ϵ is the early production coefficient. We set $\epsilon = 0.75$ and will explain why this value is chosen in Section 5.4.

5.2. Parameter Tuning

In this section, we report results from parameter tuning on the initial threshold and the number of customers to remove, which from our experience, are very important to the algorithm. We randomly pick five instances for each size (25, 50, 100, 200) (i.e., 20 instances in total) to conduct the parameter tuning test, and we let the ALNS run 10 times for each instance for each parameter setting. The values shown from Table 1 to Table 6 represent the gap between the average objective value \bar{z} over 10 runs for each parameter and the best objective value z^* found in the 10 runs among all the parameters, calculated as $\frac{\bar{z}-z^*}{z^*} \times 100\%$.

During the tuning for the initial threshold, the range to remove customers is set as (25%-50%). Table 1 and Table 2 show the influence of the initial threshold $T_{initial}$. When $T_{initial}$ is too low or too high, the heuristic has a higher chance of being trapped in one suboptimal area of the search space. The results show that $T_{initial} = 10\%$ and $T_{initial} = 17.5\%$ are the best choices for MoP-VRP and CP-VRP respectively.

Initial Threshold/%	1	2.5	5	7.5	10	12.5	15	17.5	20	30	40	50
Avg Gap/%	2.18	1.96	1.16	1.33	0.83	0.98	0.99	1.04	1.39	1.35	1.70	2.03

Table 1: Tuning of Initial Threshold (MoP-VRP)

Initial Threshold/%	1	2.5	5	7.5	10	12.5	15	17.5	20	30	40	50
Avg Gap/%	5.09	4.32	3.87	3.08	3.16	2.99	3.05	2.87	2.99	3.17	3.22	3.34

Table 2: Tuning of Initial Threshold (CP-VRP)

When we tuned the number of customers to remove, we first test the Solomon 25, 50, and 100 instances, then pick the best five ranges to continue the tuning for 200 customer instance. We tune in this way to save computational time because it is very time-consuming to test 200 customer instances. For the MoP-VRP, as shown in Table 3, the best five ranges for the instances with up to 100 customers are (20%-50%), (10%-40%), (10%-45%), (25%-45%), and (25%-35%). So they are picked to continue the tuning. Table 4 shows the average results for these five ranges on 25, 50, 100, and 200 customer instances. We find that with the increase of λ_1 , the average gap widens; it is also not good to set λ_2 as a too large (50%) or too small (35%) value. Among these ranges, (10%-40%) obtains the smallest gap and therefore is selected as the range of removing customers for the MoP-VRP.

For the CP-VRP, based on the results shown in Table 5, we pick the best five ranges as follows: (10%-35%), (10%-40%), (10%-45%), (10%-50%), and (5%-50%) to continue tuning the 200 customer instances. Table 6 shows the average results for all 25, 50, 100, and 200 customer instances. We can see that (5%-50%) is the best range for removing customers for the CP-VRP.

$\lambda_1/\%$ \backslash $\lambda_2/\%$	50	45	40	35	30
5	0.82%	0.71%	1.00%	1.07%	1.10%
10	0.87%	0.55%	0.52%	0.81%	1.13%
15	0.76%	0.64%	0.66%	0.74%	0.94%
20	0.40%	0.77%	0.81%	0.87%	1.01%
25	0.80%	0.56%	0.65%	0.63%	0.97%

Table 3: Tuning of range of customers to remove (Solomon 25, 50, 100) (MoP-VRP)

Range	(20%, 50%)	(10%, 40%)	(10%, 45%)	(25%, 45%)	(25%, 35%)
Avg Gap/%	1.36	0.65	0.7	2.07	2.01

Table 4: Tuning of range of customers to remove (25, 50, 100, 200) (MoP-VRP)

5.3. Comparison between the ALNS and the CPLEX on Small Instances

We test the MIP models for both problems on the same data set, and we give a time limit of 3600s (1h) for CPLEX to solve each instance. The instance is set with three machines in each vehicle and a production coefficient $\mu = 2$. We generate the small-size instances by randomly picking 10, 15, and 20 customers from the Solomon 25 instances.

Table 7 - 8 compare the results obtained by CPLEX and ALNS. The first three columns present the number of customers, the instance names and numbers of the different instances. For the CPLEX, we present the average upper bound (UB), the average lower bound (LB), the average computational time, the LB GAP (calculated as $\frac{UB-LB}{LB} \times 100\%$), the number of instances solved to optimality, and the number of instances where feasible solutions are found. For the ALNS, we present the average results, best results, and the average computational times. The last column shows the gap between the average results by ALNS and the UB by CPLEX, calculated as $\frac{Avg-UB}{UB} \times 100\%$. As we present the results with one digit decimal, some cells with small values in the columns LB GAP and GAP appear as 0.0%.

As shown in Table 7, the proposed ALNS for the MoP-VRP finds the same or better quality solutions than CPLEX in most instances within a much shorter computational time. There is only 1 out of 168 instances where ALNS obtains a worse average result than that of CPLEX. The gap between the results by CPLEX and ALNS grows as the instance size increases. The C instances are easier to solve compared to the R and RC instances; it becomes increasingly difficult for the CPLEX to find a feasible solution within 1 hour as the instance size increases.

The proposed ALNS for the CP-VRP also outperforms CPLEX as can be seen from Table 8. Only 6 of 168 instances in total are worse than the CPLEX. CPLEX could not find any good solutions for the CP-VRP when there are 20 customers.

$\lambda_1/\%$ \ $\lambda_2/\%$	30	35	40	45	50
5	2.29%	2.26%	2.10%	2.14%	2.01%
10	2.35%	2.06%	2.05%	2.06%	2.03%
15	2.34%	2.27%	2.18%	2.15%	2.28%
20	2.32%	2.40%	2.34%	2.13%	2.35%
25	2.34%	2.15%	2.31%	2.26%	2.11%

Table 5: Tuning of range of customers to remove (Solomon 25, 50, 100) (CP-VRP)

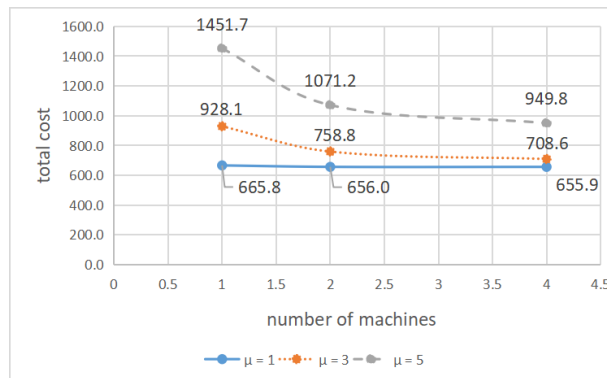
Range	(10%-35%)	(10%-40%)	(10%-45%)	(10%-50%)	(5%-50%)
Avg Gap/%	3.17	2.98	2.78	2.71	2.57

Table 6: Tuning of range of customers to remove (25, 50, 100, 200) (CP-VRP)

5.4. Comparison between MoP-VRP and CP-VRP on Large Benchmark Instances

This section presents the results of the ALNS for the MoP-VRP and the CP-VRP on the Solomon instances and Gehring and Homberger's 200 customer instances. We analyse the characteristics of the solution and compare the two problems.

First, we need to determine a suitable value for the early production coefficient ϵ for the CP-VRP. Values: 0, 0.25, 0.5, 0.75, and 1 are selected and tested on the Solomon 25 and 50 instances. In the CP-VRP, vehicles need to first wait for the production then start delivery. To avoid infeasibility caused by the waiting at the depot, we set the vehicle duration 10 times of that in the MoP-VRP. As can be seen from the table, the travel costs in the MoP-VRP and the CP-VRP are similar, but the delay costs are quite different. Comparing the delay in the CP-VRP without early production (i.e., $\epsilon = 0$) and in the MoP-VRP, we can see that mobile production helps reduce the total delay by more than 95%. As ϵ increases, the delay in the CP-VRP decreases dramatically. When ϵ reaches 1, all the products are nearly ready for delivery at time 0, and the corresponding CP-VRP is close to a VRPTW. When $\epsilon = 0.75$, the delay in the CP-VRP is comparable to that in the MoP-VRP, we therefore set ϵ to 0.75 in the rest of the tests.

Figure 9: Total cost v.s. m/κ and μ

# customer	Name	# Instance	CPLEX						ALNS			GAP
			UB	LB	Time(s)	LB GAP	# To Opt	# Feasible	Avg	Best	Avg Time(s)	
10	C	17	106.6	106.6	0.5	0.0%	17	17	106.6	106.6	0.5	0.0%
	R	23	221.3	221.3	2.7	0.0%	23	23	221.4	221.4	0.4	0.0%
	RC	16	203.3	198.4	448.6	3.8%	15	16	203.3	203.3	0.4	0.0%
15	C	17	166.0	166.0	220.4	0.0%	17	17	166.0	166.0	1.0	0.0%
	R	23	315.8	295.7	1724.7	6.2%	14	23	314.3	314.3	1.0	-0.5%
	RC	16	288.3	177.8	2422.9	96.7%	4	12	270.0	269.5	1.1	-6.3%
20	C	17	334.2	224.3	1710.8	43.5%	9	17	305.6	305.6	2.2	-8.6%
	R	23	357.0	333.3	1270.7	8.5%	10	12	340.4	340.4	2.8	-4.7%
	RC	16	573.8	257.0	3600.0	130.1%	0	8	379.8	379.2	3.2	-33.8%

Table 7: CPLEX v.s. ALNS for MoP-VRP

# customer	Name	# Instance	CPLEX						ALNS			GAP
			UB	LB	Time(s)	LB GAP	# To Opt	# Feasible	Avg	Best	Avg Time(s)	
10	C	17	129.9	129.9	186.8	0.0%	17	17	129.9	129.9	0.6	0.0%
	R	23	215.8	215.7	3.4	0.0%	23	23	215.9	215.9	0.7	0.0%
	RC	16	174.7	174.6	38.0	0.0%	16	16	174.7	174.7	0.8	0.0%
15	C	17	166.2	166.0	261.5	0.1%	16	17	166.2	166.2	1.2	0.0%
	R	23	304.4	302.8	704.1	0.5%	20	23	303.8	303.4	1.4	-0.2%
	RC	16	233.9	194.9	2386.5	29.1%	6	16	230.3	230.0	1.6	-1.5%
20	C	17	356.7	227.2	2543.5	39.1%	5	17	319.0	319.0	2.5	-10.6%
	R	23	412.4	323.9	2031.0	27.6%	11	23	359.4	359.1	3.2	-12.8%
	RC	16	418.5	244.4	3600.0	89.9%	0	15	382.5	382.2	4.0	-8.6%

Table 8: CPLEX v.s. ALNS for CP-VRP

We run the ALNS for the MoP-VRP and the CP-VRP on the Solomon instances and Gehring and Homberger's 200 customer instances and find that the key factors that influence the objective value are the production time and the number of machines set in each vehicle. Figure 9 summarizes how the total cost changes with the number of machines and production time for the MoP-VRP. As can be seen from the figure, when the machine number is fixed, the higher production time coefficient will lead to a higher total cost; when the production time coefficient (μ) is fixed, the higher number of machines in each vehicle helps reduce the total cost. This trend becomes increasingly obvious as μ becomes larger and larger.

When the production time coefficient is 1, the influence of the number of machines on the total cost is not obvious. This shows that when the production time is small, a few machines are enough, the number of machines is not the bottleneck to obtaining a good distribution plan for the MoP-VRP. When the production time coefficient increases (i.e., 3, 5), the total cost decreases significantly, typically when the number of machines in each vehicle increases from 1 to 2. The total cost decreases by a comparatively small amount when the number of machines in each vehicle increases from 2 to 4.

5.5. Results on Realistic Instances

This section analyses the travel and delay costs for both the MoP-VRP and CP-VRP. The long term costs are then estimated, and advice on operation strategy is given.

n	25			50			
	Name	Avg travel	Avg delay	Avg cost	Avg travel	Avg delay	Avg cost
$\epsilon = 0$	C	240.3	1513.3	1753.6	447.5	1945.1	2392.6
	R	415.5	1507.7	1923.2	744.4	2270.0	3014.4
	RC	394.2	4634.4	5028.6	764.1	6082.5	6846.6
$\epsilon = 0.25$	C	239.1	855.8	1094.9	428.8	747.7	1176.4
	R	416.9	695.5	1112.4	762.3	731.9	1494.2
	RC	411.2	2471.4	2882.6	814.1	2423.8	3237.9
$\epsilon = 0.5$	C	238.5	400.3	638.9	404.8	202.5	607.2
	R	438.6	174.4	612.9	744.6	143.1	887.7
	RC	384.9	915.8	1300.7	807.8	550.3	1358.1
$\epsilon = 0.75$	C	229.0	116.5	345.5	382.3	39.9	422.2
	R	433.6	35.1	468.7	701.9	22.2	724.1
	RC	368.0	126.6	494.6	699.5	96.7	796.2
$\epsilon = 1$	C	221.3	11.2	232.5	380.8	0.6	381.4
	R	434.6	23.8	458.4	694.2	17.8	712.0
	RC	376.6	17.2	393.8	657.1	40.2	697.3
MoP-VRP	C	231.9	35.8	267.7	394.8	102.9	497.7
	R	457.5	50.8	508.4	776.4	94.9	871.3
	RC	529.0	90.3	619.3	972.7	136.0	1108.7

Table 9: The Average Cost with Different Early Production Coefficient

5.5.1. The delivery cost

Table 10 shows the average results for the MoP-VRP and CP-VRP on the realistic instances with 99 customers. The first three columns show scenario names, the number of machines in each vehicle, and the average number of vehicles. For both the MoP-VRP and the CP-VRP, we show the average traveled distance (in miles), the average delay (in minutes), the average total cost, and the average computational time for the MoP-VRP. In addition, we also show the average early production time for the CP-VRP. Each row in the table presents the average results of five random instances over ten random runs.

On a high level, the average cost for the MoP-VRP and CP-VRP is similar. Certain instance types are more attractive for either MoP-VRP or CP-VRP. At first, this may seem a bit disappointing from an MoP-VRP point of view, and the reader may ask what the advantage of MoP-VRP is. Here it is important to re-iterate that the early production coefficient ϵ was chosen to make the two problems perform similarly in terms of objective value and that the early production provides a huge advantage for the CP-VRP in terms of reducing delays (see the impact of the ϵ parameter in Table 9)

The key advantage of mobile production is that it makes distribution more flexible. We find that in the CP-VRP, production has to start at least 2.5 hours (i.e., 145 mins) before the vehicles depart from the depot, and in some cases up to 6.5 (i.e. 400 mins) hours before the vehicles depart. Since the vehicles depart at 11 am, this means that almost no orders from the day of operations will be included in the CP-VRP. For the MoP-VRP, production starts when the vehicles leave the depot, which means it can accept orders until 11 am. Meanwhile, mobile production can keep an acceptable delay for customers.

Regarding the influence of the number of machines in each vehicle, the overall finding is that setting one machine on each vehicle obtains the lowest total cost in the ‘‘S_W’’ scenario. Putting more than one machine in each vehicle lowers the total cost in most scenarios in the MoP-VRP. We only consider up to four machines in each vehicle because this is what we estimate will be feasible to fit into the vans that we used to estimate

		MoP-VRP				CP-VRP					
m	κ	avg travel	avg delay	avg cost	avg time(s)	avg ahead	avg travel	avg delay	avg cost	avg time(s)	
S_W	1	5	319.9	1.6	321.5	47.9	366.3	321.0	0.8	321.8	101.4
	2	3.2	326.8	1.8	328.6	162.5	290.2	343.4	3.2	346.6	109.7
	3	3.2	324.7	1.8	326.5	244.7	193.5	338.3	2.4	340.7	133.5
	4	3.2	324.4	1.7	326.1	315.6	145.1	337.7	2.0	339.7	141.1
M_W	1	6.8	359.3	9.6	368.9	31.8	382.2	318.5	0.7	319.2	215.1
	2	4	326.4	4.1	330.5	120.0	323.8	325.8	2.2	328.1	113.6
	3	3	338.5	3.9	342.5	247.9	287.8	342.9	3.1	345.9	114.4
	4	3	335.4	4.5	339.8	329.8	215.9	342.3	3.4	345.8	136.9
H_W	1	8.2	456.2	143.5	599.7	23.7	403.3	317.4	1.7	319.1	290.1
	2	4.4	394.2	157.2	551.4	89.5	378.8	320.7	2.2	322.9	119.2
	3	3	369.4	17.0	386.5	196.0	366.9	345.6	4.1	349.7	118.3
	4	3	339.9	12.0	351.9	322.1	275.2	341.4	3.8	345.2	125.8
S_T	1	5	373.1	111.6	484.6	40.3	367.9	335.9	2.2	338.1	100.2
	2	3	367.5	18.8	386.2	159.3	306.6	371.7	35.7	407.3	105.4
	3	2.6	367.1	163.0	530.0	304.8	245.6	363.7	268.1	631.8	132.4
	4	2.6	366.7	166.1	532.7	390.6	184.2	366.2	183.2	549.3	134.6
M_T	1	7	383.1	32.8	415.8	30.2	368.5	331.8	1.5	333.3	189.8
	2	4	357.2	11.3	368.4	111.2	322.4	344.1	2.5	346.6	110.5
	3	3	373.8	16.4	390.2	236.6	286.6	376.1	13.5	389.6	114.8
	4	2.8	372.4	71.1	443.5	351.8	236.5	367.5	216.8	584.3	137.0
H_T	1	8.4	432.2	196.0	628.2	23.4	397.8	330.3	1.6	331.9	287.7
	2	4.4	400.0	162.9	562.9	86.3	382.7	343.4	3.7	347.1	119.6
	3	3.2	387.0	161.5	548.6	189.3	352.0	372.7	20.4	393.1	110.3
	4	3	371.4	51.3	422.7	304.1	277.8	376.1	10.2	386.3	114.6

Table 10: Realistic instances - 99 customers

long term costs.

5.5.2. The long-term operation cost estimation

To put mobile production into practical use, not only should we estimate the optimal number of machines in each vehicle for good distribution but also we need to investigate how many machines in each vehicle will reduce long-term costs the most. Our long-term cost estimation is shown in Table 12, and we summarize the results for long-term costs for 25, 50, and 99 customer instances in Table 13. Considering that central production already exists in real-life practice, we only estimate the long-term cost of mobile production to check the feasibility of the new logistics mode.

The prices we estimate in this work are based on a Danish setting, and costs related to 3D printing equipment are suggested by 3D Printhuset. All the estimated costs can be found in Table 11.

Regarding the wage for drivers, we assume that each worker works for 10 hour per day, five hours per week and 50 weeks per year. The average hourly wage is set to be €25.6, which is the hourly wage for an educated driver in Denmark plus a 30 percent addition that covers insurance, holiday pay, and indirect costs. This means that the cost per driver a year is €63,908. If workers without taking driver education are used for the job, the wage costs will be lower.

We consider a 10-year planning horizon and the total cost for planning horizon equals the investment cost plus ten times the yearly cost. From the 10-year total cost, we can calculate the average cost per year and the average cost per order given the total number of orders per year.

Driver		Printer	
Hourly wage	€25.6	Cost of purchasing printer	€2300
Work hours per day	10	Yearly maintenance cost	€345
Work days per week	5	Renewal cycle	2yr
Work weeks per year	50	Printer scrap value	€1000
Vehicle		Renewal cost	€1300
Cost of purchasing vehicle	€54000	Renewal cost per year	€650
Maintenance cost per year	€5400	Yearly cost	€995
Fuel		Planning Horizon	10yr
Consumption	8.08mile/L		
Price	€1.1/L		

Table 11: Cost List

Table 12 shows how we calculate long-term cost. The first four columns present the scenario names, setting, and average travel distance. The next two columns are the number of vehicles and the number of machines the company should purchase. The columns under “Investment Cost” show the vehicle cost, machine cost, and the total cost. The columns under “Yearly costs” show the maintenance costs of vehicles, maintenance costs of printers, wages of drivers, fuel costs, and total cost each year. Column 15 shows the total cost over the 10-year planning horizon. Column 16 shows the ratio of the total costs to that when there is only one vehicle in the same scenario. The last three columns show the average cost per year, the number of orders per year, and the cost per order.

Table 13 summarizes the proportion of each cost component in the total cost for the instances in different sizes and with different numbers of machines per vehicle. The third column presents the ratio of the total costs to that when there is only one vehicle for each data size. We can see that the drivers’ salary takes up the largest part of the total cost (more than 60%) and that adding more machines to each vehicle can reduce the overall cost.

Figure 10 shows how the average cost per order changes with the number of vehicles at each stage. Figure 10 illustrates that when the number of machines in each vehicle is fixed, the average cost per order gradually decreases even as the total number of customers increases. When the number of customers is fixed, the average cost per order drops dramatically when the number of machines increases from 1 to 2. Based on these findings, it is estimated that the cost for mobile production and distribution will continue to decrease as the business mode grows in popularity and technology matures.

Regarding the long-term costs associated with the central production mode, we would need workers in the central warehouse to handle the production, leading to higher wage costs. We would also need to rent or buy space for the production facility. Both issues increase the long-term costs. On the other hand, we would be able to handle deliveries using smaller and cheaper vehicles as the vehicles do not need to fit the printers. The average hourly wage for the drivers would potentially be lower as the drivers require fewer skills in this

setup. A precise comparison between the long-term cost of the two modes is left as future work.

Scenario	m	κ	avg travel	Vehicles		Investment cost			Yearly costs (Maintenance & Wage & Fuel)				total cost over 10-year	compared to 1 machine(%)	Cost per year	Orders per year	Cost per order	
				to buy	to buy	vehicle	printer	total	vehicle	printer	drivers	Fuel						total
S.W	1	5	319.9	6	6	324000	13800	337800	32400	5970	319540	10911	368821	4026008	100.0	4026001	24750	16.3
	2	3.2	326.8	5	10	270000	23000	293000	27000	9950	204505	11149	252604	2819042	70.0	281904	24750	11.4
	3	3.2	324.7	5	15	270000	34500	304500	27000	14925	204505	11076	257507	2879567	71.5	287957	24750	11.6
	4	3.2	324.4	5	20	270000	46000	316000	27000	19900	204505	11067	262472	2940723	73.0	294072	24750	11.9
M.W	1	6.8	359.3	8	8	432000	18400	450400	43200	7960	434574	12257	497991	5430306	100.0	543031	24750	21.9
	2	4	326.4	5	10	270000	23000	293000	27000	9950	255632	11133	303715	3330150	61.3	333015	24750	13.5
	3	3	338.5	4	12	216000	27600	243600	21600	11940	191724	11547	236811	2611708	48.1	261171	24750	10.6
	4	3	335.4	4	16	216000	36800	252800	21600	15920	191724	11440	240683	2659634	49.0	265963	24750	10.7
H.W	1	8.2	456.2	10	10	540000	23000	563000	54000	9950	524045	15563	603558	6598576	100.0	659858	24750	26.7
	2	4.4	394.2	6	12	324000	27600	351600	32400	11940	281195	13446	338980	3741404	56.7	374140	24750	15.1
	3	3	369.4	4	12	216000	27600	243600	21600	11940	191724	12602	237866	2622257	44.8	262226	24750	10.6
	4	3	339.9	4	16	216000	36800	252800	21600	15920	191724	11594	240838	2661180	40.3	266118	24750	10.8
S.T	1	5	373.1	6	6	324000	13800	337800	32400	5970	319540	12725	370635	4044147	100.0	404415	24750	16.3
	2	3	367.5	4	8	216000	18400	234400	21600	7960	191724	12534	233818	2572577	63.6	257258	24750	10.4
	3	2.6	367.1	4	12	216000	27600	243600	21600	11940	166161	12521	212222	2365818	58.5	236582	24750	9.6
	4	2.6	366.7	4	16	216000	36800	252800	21600	15920	166161	12507	216188	2414678	59.7	241468	24750	9.8
M.T	1	7	383.1	8	8	432000	18400	450400	43200	7960	447356	13066	511582	5566218	100.0	556622	24750	22.5
	2	4	357.2	5	10	270000	23000	293000	27000	9950	255632	12183	304765	3340645	60.0	334065	24750	13.5
	3	3	373.8	4	12	216000	27600	243600	21600	11940	191724	12749	238013	2623725	47.1	262373	24750	10.6
	4	2.8	372.4	4	16	216000	36800	252800	21600	15920	178942	12704	229166	2544457	45.7	254446	24750	10.3
H.T	1	8.4	432.2	10	10	540000	23000	563000	54000	9950	536827	14743	615520	6718195	100.0	671820	24750	27.1
	2	4.4	400.0	6	12	324000	27600	351600	32400	11940	281195	13645	339180	3743398	55.7	374340	24750	15.1
	3	3.2	387.0	5	15	270000	34500	304500	27000	14925	204505	13202	259632	2900823	43.2	290082	24750	11.7
	4	3	371.4	4	16	216000	36800	252800	21600	15920	191724	12667	241911	2671911	39.8	267191	24750	10.8

Table 12: Realistic instances - 99 customers - long term cost

n	m	Compared to 1 machine (%)	Vehicle cost	Printer cost	Driver cost	Fuel cost
25	2	67.0	23.64%	5.36%	66.46%	4.54%
25	3	66.8	23.75%	8.07%	63.64%	4.54%
25	4	68.6	23.12%	10.48%	61.97%	4.43%
50	1	100.0	17.37%	1.97%	78.15%	2.51%
50	2	66.8	19.48%	4.41%	72.18%	3.93%
50	3	64.1	19.52%	6.63%	69.77%	4.08%
50	4	65.5	19.09%	8.65%	68.27%	3.98%
99	1	100.0	16.00%	1.81%	79.68%	2.50%
99	2	61.2	17.16%	3.89%	75.10%	3.84%
99	3	52.2	17.52%	5.96%	71.89%	4.63%
99	4	51.3	16.98%	7.70%	70.76%	4.56%

Table 13: Summary for long term cost

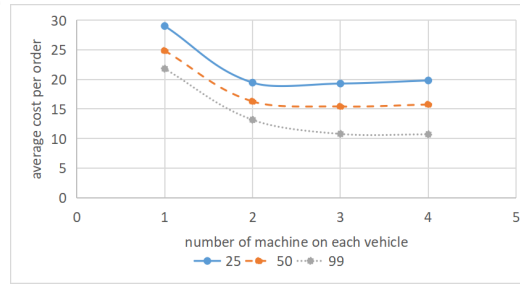


Figure 10: Cost Per Order

6. Conclusion

This paper introduces a new variant of the vehicle routing problem, called the Mobile Production Vehicle Routing Problem (MoP-VRP), where the production takes place on the way to the customer. The MoP-VRP is highly complex, as it considers multiple products, multiple machines, soft time windows, and production schedules together within the distribution plan. For this complex problem, we propose a mixed integer programming (MIP) model as well as an adaptive large neighbourhood search (ALNS) heuristic. The computational results show that the proposed ALNS is highly efficient and can find the optimal solutions for most of the small instances within a short computational time. We also apply ALNS for the Central Production Vehicle Routing Problem (CP-VRP) and devise smart strategies to accelerate the solution process. One

strategy is to avoid calculating redundant production sequences. The other is a piece-wise linear function, which speeds up the computation of delay costs when the production plan changes. The computational results show that the proposed algorithm for the CP-VRP is also efficient and solves most of the small instances to optimality within a short time. To investigate the advantage of the MoP-VRP, we generate realistic instances to compare with the CP-VRP. The instances are generated based on a Danish setting, and the values we use are estimated by 3D Printhuset. The experiments show that the key advantage of the MoP-VRP is flexibility: the MoP-VRP does not require early production and at the same time can keep a lower delivery cost than the CP-VRP. Regarding the realistic instances, we find that the MoP-VRP is feasible in practice due to its low operations costs from a long term estimation. The cost for each delivery will gradually decrease as the customer base increases.

We propose several directions for future research. First, more constraints can be added to the basic model introduced in this paper to help improve the efficiency of this new logistics mode. For example, the concept of delayed differentiation can be introduced, as Su et al. (2010) show that it can result in shorter waiting times in MTO mode. Second, exact algorithms can be developed for the MoP-VRP. The CPLEX can only solve the problem with up to 15 customers. It would be interesting to develop an exact algorithm that can provide the optimal solution for larger size instances within a reasonable time. Last but not least, we recommend investigating the dynamic version of the MoP-VRP. It would be interesting to study how incoming requests might affect the final solution in the dynamic MoP-VRP, and how different waiting strategies can be used to improve the quality of the solution.

Acknowledgement

This work was supported by the National Science Foundation of China [11601436] and the Research Development Fund of Xi'an Jiaotong-Liverpool University [RDF-16-02-50].

References

- [1] Adulyasak, Y., Cordeau, J.-F., and Jans, R. (2015). The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, 55:141–152.
- [2] Amazon (2017). Annual report 2017. <https://ir.aboutamazon.com/static-files/917130c5-e6bf-4790-a7bc-cc43ac7fb30a>(accessed February 29, 2020).
- [3] Amazon (2018). Annual report 2018. <https://ir.aboutamazon.com/static-files/0f9e36b1-7e1e-4b52-be17-145dc9d8b5ec>(accessed February 29, 2020).
- [4] Amazon Technologies, I. (2015). Item delivery using 3-d manufacturing on demand. US Patent 20150052023 A1.

- [5] Amorim, P., Belo-Filho, M., Toledo, F., Almeder, C., and Almada-Lobo, B. (2013). Lot sizing versus batching in the production and distribution planning of perishable goods. *International Journal of Production Economics*, 146(1):208–218.
- [6] Armstrong, R., Gao, S., and Lei, L. (2008). A zero-inventory production and distribution problem with a fixed customer sequence. *Annals of Operations Research*, 159(1):395–414.
- [7] Belo-Filho, M., Amorim, P., and Almada-Lobo, B. (2015). An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. *International Journal of Production Research*, 53(20):6040–6058.
- [8] Bergh, S. K. (2018). Optimization of mobile production networks. Master’s thesis, Technical University of Denmark, DTU Management.
- [9] Boyle, A. (2018). Amazon finally wins a patent for 3-d printing on demand, for pickup or delivery. <https://www.geekwire.com/2018/amazon-gets-patent-3-d-printing-demand-pickup-delivery/>(accessed February 29, 2020).
- [10] Chang, Y.-C. and Lee, C.-Y. (2004). Machine scheduling with job delivery coordination. *European Journal of Operational Research*, 158(2):470–487.
- [11] Chen, Z.-L. and Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operations. *Management Science*, 51(4):614–628.
- [12] Dayarian, I. and Desaulniers, G. (2019). A branch-price-and-cut algorithm for a production-routing problem with short-life-span products. *Transportation Science*, 53(3):829–849.
- [13] Dueck, G. and Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, 90(1):161–175.
- [14] Gehring, H. and Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, volume 2, pages 57–64. Citeseer.
- [15] Geismar, H. N., Laporte, G., Lei, L., and Sriskandarajah, C. (2008). The integrated production and transportation scheduling problem for a product with a short lifespan. *INFORMS Journal on Computing*, 20(1):21–33.
- [16] Guo, Z., Shi, L., Chen, L., and Liang, Y. (2017). A harmony search-based memetic optimization model for integrated production and transportation scheduling in mto manufacturing. *Omega*, 66:327–343.
- [17] Jamili, N., Ranjbar, M., and Salari, M. (2016). A bi-objective model for integrated scheduling of production and distribution in a supply chain with order release date restrictions. *Journal of Manufacturing Systems*, 40:105–118.

- [18] Kesen, S. E. and Bektaş, T. (2019). Integrated production scheduling and distribution planning with time windows. In *Lean and Green Supply Chain Management*, pages 231–252. Springer.
- [19] Krassenstein, B. (2014). Denmark shipping company, maersk, using 3d printing to fabricate spare parts on ships. <https://3dprint.com/9021/maersk-ships-3d-printers/>(accessed September 20, 2021).
- [20] Laporte, G. and Semet, F. (2002). Classical heuristics for the capacitated vrp. In *The vehicle routing problem*, pages 109–128. SIAM.
- [21] Li, K., Zhou, C., Leung, J. Y., and Ma, Y. (2016). Integrated production and delivery with single machine and multiple vehicles. *Expert Systems with Applications*, 57:12–20.
- [22] Low, C., Chang, C.-M., Li, R.-K., and Huang, C.-L. (2014). Coordination of production scheduling and delivery problems with heterogeneous fleet. *International Journal of Production Economics*, 153:139–148.
- [23] Malladi, S. S., Erera, A. L., and White III, C. C. (2020). A dynamic mobile production capacity and inventory control problem. *IIEE Transactions*, 52(8):926–943.
- [24] Mohammadi, S., Al-e Hashem, S. M., and Rekik, Y. (2020). An integrated production scheduling and delivery route planning with multi-purpose machines: A case study from a furniture manufacturing company. *International Journal of Production Economics*, 219:347–359.
- [25] Moons, S., Ramaekers, K., Caris, A., and Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level: a review and discussion. *Computers & Industrial Engineering*, 104:224–245.
- [26] Pasha, J., Dulebenets, M. A., Kavooosi, M., Abioye, O. F., Wang, H., and Guo, W. (2020). An optimization model and solution algorithms for the vehicle routing problem with a “factory-in-a-box”. *IEEE Access*, 8:134743–134763.
- [27] Phillips, B. T., Alder, J., Bolan, G., Nagle, R. S., Redington, A., Hellebrekers, T., Borden, J., Pawlenko, N., and Licht, S. (2020). Additive manufacturing aboard a moving vessel at sea using passively stabilized stereolithography (sla) 3d printing. *Additive Manufacturing*, 31:100969.
- [28] Qiu, Y., Wang, L., Xu, X., Fang, X., and Pardalos, P. M. (2018). A variable neighborhood search heuristic algorithm for production routing problems. *Applied Soft Computing*, 66:311–318.
- [29] Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.
- [30] Santini, A., Ropke, S., and Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24(5):783–815.

- [31] Scholz-Reiter, B., Makuschewitz, T., Novaes, A. G., Frazzon, E. M., and Lima Jr, O. F. (2011). An approach for the sustainable integration of production and transportation scheduling. *International Journal of Logistics Systems and Management*, 10(2):158–179.
- [32] Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 46.
- [33] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer.
- [34] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.
- [35] Su, J. C., Chang, Y.-L., Ferguson, M., and Ho, J. C. (2010). The impact of delayed differentiation in make-to-order environments. *International Journal of Production Research*, 48(19):5809–5829.
- [36] Tamannaie, M. and Rasti-Barzoki, M. (2019). Mathematical programming and solution approaches for minimizing tardiness and transportation costs in the supply chain scheduling problem. *Computers & Industrial Engineering*, 127:643–656.
- [37] Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research*, 227(1):152–165.
- [38] Wang, S., Wu, R., Chu, F., and Yu, J. (2020). Variable neighborhood search-based methods for integrated hybrid flow shop scheduling with distribution. *Soft Computing*, 24(12):8917–8936.
- [39] Yağmur, E. and Kesen, S. E. (2020). A memetic algorithm for joint production and distribution scheduling with due dates. *Computers & Industrial Engineering*, 142:106342.
- [40] Zou, X., Liu, L., Li, K., and Li, W. (2018). A coordinated algorithm for integrated production scheduling and vehicle routing problem. *International Journal of Production Research*, 56(15):5005–5024.

Appendix

For both the **Geo Removal** and **Demand Removal**, we firstly remove a random seed customer. Then, the **Geo Removal** will remove $\Phi - 1$ customers that are closest to that seed customer. The **Demand Removal** will remove $\Phi - 1$ customers whose demands are closest to the seed customer. Alg. 6 shows how the Geo Removal works. The Demand Removal shares the same framework. The only difference is that for the Geo Removal, c_{qi} in line 7 represents the distance between customers q and i , whereas in Demand Removal we transfer c_{qi} to $|d_q - d_i|$, which is the absolute value of the difference in demand between customers q and i .

Algorithm 6 Geo Removal

```

1: Function Geo Removal ( $s, u \in \mathbb{R}_+$ )
2: request  $q = a$  randomly selected request from  $s$ ;
3: Set of requests:  $R = \{q\}$ ;
4: while  $|R| < \Phi$  do
5:   Array:  $L =$  an array containing all request from  $s$  not in  $R$ ;
6:   Sort  $L$  such that
7:    $i < j \rightarrow c_{qL[i]} < c_{qL[j]}$ ;
8:   choose a random number  $\sigma$  from the interval  $[0,1)$ ;
9:    $R = R \cup \{L[\sigma^u|L|]\}$ ;
10: end while
11: remove the requests in  $R$  from  $s$ ;

```

Like Ropke and Pisinger (2006), a random parameter σ^u (appears in Alg. 6 line 9 and Alg. 7 line 6) is used to introduce some randomness to the worst removal and the four newly proposed removal operators, such that we can avoid removing the same customer over and over again. Parameter σ is a random number between 0 and 1 and u is a deterministic parameter.

The **worst removal** removes Φ customers with the highest cost savings ξ_i , where $\xi_i = f(s) - f(\tilde{s})$, and $f(s)$ and $f(\tilde{s})$ are the cost of the solutions with customer i and without i , respectively. The $f(s)$ and $f(\tilde{s})$ will be updated after we remove one customer from the solution, and the pseudo-code can be seen in Alg. 7. **Worst-Delay Removal** is to remove Φ customers with the highest delay cost savings and **Worst-Dist Removal** is to remove Φ customers with the highest travel distance savings, where the savings are calculated in the similar way as the **Worst Removal**.

Algorithm 7 Worst Removal

```

1: Function Worst Removal ( $s, u \in \mathbb{R}_+$ )
2:  $count = \Phi$ ;
3: while  $count > 0$  do
4:   Array:  $L =$  all planned requests  $i$ , sorted by descending  $\xi_i$ ;
5:   choose a random number  $\sigma$  from the interval  $[0,1)$ ;
6:   request:  $q = L[\sigma^u|L|]$ ;
7:   remove the request  $q$  from  $s$ ;
8:    $count = count - 1$ ;
9: end while

```

Table 14 shows important notations for the MoP-VRP and the CP-VRP.

Set		p_i	Production time of i
G	The Graph	$T, T_{initial}$	Temperature threshold, initial temperature
V	The Node: $\{0 \dots n\}$	N_{max}	Max ALNS iteration
E	The Edge	Φ	The number of customers to be removed
C	The Customer: $\{1 \dots n\}$	λ_1, λ_2	The lower bound and upper bound of removal ratio
K	The Vehicle: $\{1 \dots \kappa\}$	u	A deterministic parameter
M	The Machine in each vehicle: $\{1 \dots m\}$	σ	A random number $\in [0,1]$
M'	The Machine in the depot: $\{1 \dots m'\}$	Variable	
δ_i^-	The Node can reach i	x_{ij}^k	1 if travel on (i, j) by vehicle k , 0 otherwise
δ_i^+	The Node i can reach	y_i	Delay of customer i
R	The set of requests	s_i^k	Service time of i by vehicle k
Parameter		v_{il}^k	Production finishing time of job i by machine l in vehicle k
Q	Capacity	w_{ijl}^k	1 if job j is after job i by machine l on vehicle k , 0 otherwise
D	Close time of depot	s, s^*, s'	Current solution, best found solution, temporary solution
d_i	Demand of customer i	$f(s)$	cost of solution s
μ	Production time for each demand unit	ξ_i	The saving cost of customer i
ϵ	Early production ratio	\bar{s}	cost of solution without some customer
H	Early production time	r_{ih}	the route where customer i has the h th lowest cost
$[a_i, b_i]$	Time windows of i	g_i^*	the regret value for customer i
c_{ij}	Travel distance on (i, j)	$\Delta_{i,r_{ih}}$	the inserting cost of customer i on route r_{ih}
t_{ij}	Travel time on (i, j)	$Delay(\Psi)$	the delay cost of a fixed route with respect to departure time Ψ
e_i	Service time of i		

Table 14: Important Notations